Système de couleurs Windows

Article • 13/06/2023

Objectif

Les schémas de gestion des couleurs sont implémentés dans les systèmes d’exploitation Microsoft Windows pour améliorer le rendu du contenu couleur dans toutes les formes de communication numérique.

Le schéma de gestion des couleurs utilisé à partir de Windows 2000, Windows XP et Windows Server 2003 est appelé Gestion des couleurs des images (ICM) 2.0. Le schéma de gestion des couleurs utilisé à partir de Windows Vista s’appelle Windows Color System (WCS) 1.0. Le schéma de gestion des couleurs WCS 1.0 est un sur-ensemble d’API et de fonctionnalités ICM 2.0.

Le cas échéant

ICM peut être utilisé dans toutes les applications sur les systèmes d’exploitation Windows 2000 et ultérieur. WCS peut être utilisé dans toutes les applications sur Windows Vista et les systèmes d’exploitation ultérieurs.

Développeurs concernés

L’API WCS est conçue pour être utilisée par les programmeurs C/C++. Vous devez connaître l’interface utilisateur graphique de Windows, l’architecture pilotée par les messages et une connaissance pratique des concepts de gestion des couleurs.

Conditions d’exécution

Les applications qui utilisent l’API ICM nécessitent Windows 2000 ou une version ultérieure. Les applications qui utilisent l’API WCS nécessitent Windows Vista ou une version ultérieure. Pour obtenir des informations d’exécution spécifiques sur une fonction, consultez la section Exigences de la page de référence de cette fonction.

Contenu de cette section

- Schémas et algorithmes du système de couleurs Windows
- À propos de Windows Color System version 1.0
- Utilisation de WCS 1.0
- Référence
- Glossaire WCS 1.0

Rubriques connexes

Opengl

Windows Multimedia

Windows GDI

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

Considérations relatives à la sécurité : Système de couleurs Windows

Article • 13/06/2023

Ce document fournit des informations sur les considérations de sécurité liées à la gestion des couleurs des images. Ce document ne fournit pas tout ce que vous devez savoir sur les problèmes de sécurité. Au lieu de cela, utilisez-le comme point de départ et référence pour ce domaine technologique.

Les machines virtuelles non-Microsoft peuvent s'exécuter dans le contexte système

Les modules de gestion des couleurs (CMM) non-Microsoft doivent être traités comme des pilotes d'imprimante non-Microsoft, car ils peuvent s'exécuter dans un contexte système pour les opérations d'impression.

Commentaires

Concepts de base de la gestion des couleurs

Article • 13/06/2023

Pour comprendre comment Windows Color Systemn (WCS) version 1.0 est utilisé, une connaissance pratique des concepts de gestion des couleurs est requise. Cette vue d'ensemble présente suffisamment d'arrière-plan de gestion des couleurs pour fournir une définition des termes et concepts de gestion des couleurs qui apparaissent dans la référence du programmeur WCS dans la documentation du Kit de développement logiciel (SDK) de plateforme. Il n'est pas exhaustif dans sa couverture. Les rubriques de cette section couvrent les principaux domaines de la gestion des couleurs, tels que :

- Couleur dans l'imagerie
- Espaces de couleurs
- Conversion des couleurs et correspondance des couleurs
- Schémas et algorithmes du système de couleurs Windows
- Informations complémentaires

Commentaires

Couleurs dans les images

Article • 13/06/2023

La gestion des couleurs n'est pas une nouvelle rubrique. Il a été utilisé pendant presque aussi longtemps qu'il y a eu des sociétés commerciales d'impression et d'édition. Un ensemble bien développé de théorie et d'expérience a été appliqué au cours des dernières années aux systèmes informatisés d'imagerie et d'édition. Les concepts incorporés dans ces théories et pratiques sont basés sur la perception des couleurs humaines.

Il n'est pas nécessaire que les développeurs d'applications possèdent un haut niveau d'expertise dans la physiologie de l'œil et du système nerveux. Il n'est pas non plus nécessaire d'être un expert en physique de la lumière. Toutefois, une compréhension des principes fondamentaux de la vision et de l'imagerie aidera les développeurs à utiliser Windows Color System 1.0. Ces principes de base sont les suivants :

- Perception humaine des couleurs
- Couleurs primaires additives
- Couleurs primaires soustractives
- Description de la couleur

Commentaires

Perception humaine des couleurs

Article • 13/06/2023

La vision humaine est rendue possible par la présence de cellules sensibles à la lumière dans l'œil *appelées bâtonnets* et *cônes*. La recherche a montré que les tiges ne semblent pas être impliquées dans la perception de la couleur chez les êtres humains. Il existe trois types de cônes différents dans la rétine. Chaque type de cône détecte le rouge, le vert ou le bleu. Toutes les autres couleurs que les humains voient sont des mélanges de ces trois couleurs. Par instance, le blanc est perçu lorsque des quantités approximativement égales de rouge, de vert et de bleu sont observées. Le noir est visible lorsqu'aucun rouge, vert ou bleu (très peu ou pas de lumière du tout) n'est détecté par l'œil. La quantité de couleur que les humains voient dépend également de la force, de la concentration et de la position de la source de lumière. Les conditions d'éclairage peuvent avoir un effet profond sur la perception de la couleur.

Dans les systèmes d'imagerie, les couleurs peuvent être mélangées de différentes façons pour produire un résultat souhaité pour l'œil. Les méthodes de mélange les plus couramment utilisées sont basées sur les *couleurs primaires additives* et les *couleurs primaires soustractives*. Toutes les couleurs peuvent être reproduites à l'aide de l'une ou l'autre des méthodes.

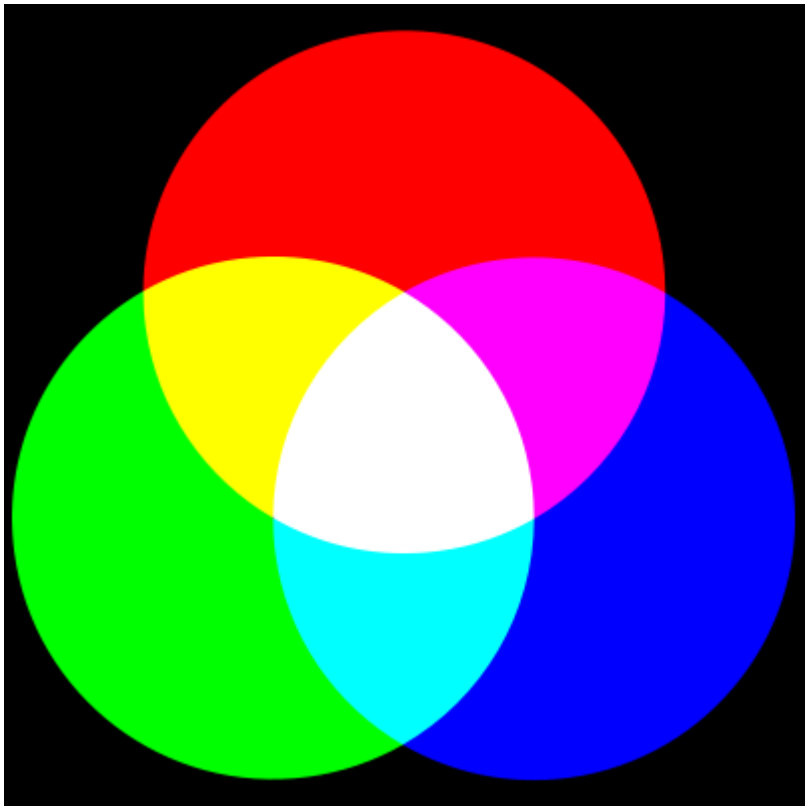
Commentaires

Couleurs primaires additives

Article • 13/06/2023

La méthode additive de mélange de couleurs est basée sur l'hypothèse que vous commencez par le noir. Autrement dit, si aucune autre couleur n'est présente (l'image est noire) et que vous ajoutez du rouge, l'image apparaît rouge. Si vous ajoutez ensuite du bleu, l'image s'affiche en magenta.

Les trois couleurs primaires additives sont le rouge, le vert et le bleu. La figure suivante illustre le mélange additif de couleurs primaires.



Commentaires

Couleurs primaires soustractives

Article • 13/06/2023

La méthode soustractive du mélange de couleurs est basée sur l'hypothèse que vous commencez par le blanc. Toutes les couleurs sont présentes en quantités égales. Si vous soustrayez le cyan et le jaune du blanc, l'image obtenue est magenta. Dans une image verte, la soustraction de cyan entraîne la modification de l'image en jaune.

Les couleurs primaires soustractives sont cyan, jaune et magenta. La figure suivante illustre le mélange soustractif de couleurs primaires.

Commentaires

Description de la couleur

Article • 13/06/2023

La couleur peut être décrite en termes de composants. Les composants les plus souvent utilisés par les professionnels de l'imagerie sont *la teinte*, *la chromatique* et *la légèreté*. Une teinte est ce que l'on considère normalement comme une couleur. La teinte d'une couleur fixe sa place dans le spectre visible de la lumière. La chroma d'une couleur est la façon dont une couleur est « pure » ou « forte ». Le gris neutre est dit avoir une saturation zéro. La légèreté d'une couleur fait référence à l'intensité de la lumière qui est réfléchie ou transmise par une image.

Les termes *teinte*, *tonalité* et *nuance* sont également bien utilisés dans la littérature sur l'imagerie des couleurs. Une teinte d'une couleur est obtenue en mélangeant sa teinte avec le blanc. Une tonalité d'une couleur est créée en mélangeant une teinte avec du gris. Une nuance d'une couleur est faite en ajoutant du noir à sa teinte.

Commentaires

Espaces de couleurs

Article • 13/06/2023

L'œil humain est souvent capable de détecter beaucoup plus de couleurs que les appareils numériques peuvent reproduire. Par instance, si vous regardez une page blanche vierge de papier, votre œil détecte probablement au moins 100 nuances distinctes de blanc. Un mur blanc peut facilement avoir 1 500 nuances de blanc.

Les appareils photo numériques, scanners et autres appareils d'acquisition d'images de haute qualité peuvent également détecter des centaines de milliers, voire des millions de couleurs. En raison de la présence de tant de couleurs détectables, les professionnels de l'imagerie ont inventé des modèles pour spécifier des couleurs. Ces modèles sont appelés *espaces de couleurs*.

La raison pour laquelle ces modèles sont appelés espaces de couleurs est que la plupart d'entre eux peuvent être mappés dans un système de coordonnées 2D, 3D ou 4D similaire à un système de coordonnées cartésien. Par conséquent, les couleurs peuvent être dites composées de coordonnées dans un espace 2D, 3D ou 4D. Les composants de couleur d'un espace de couleurs sont également appelés *canaux de couleurs*.

Certains espaces de couleurs sont conçus pour être indépendants de tout appareil utilisé pour produire des images en couleur. Certains sont très dépendants de l'appareil. Les espaces de couleurs dépendants de l'appareil et indépendants de l'appareil sont décrits dans les sections suivantes :

- Espaces de couleurs RVB
- Espaces de couleurs HSV
- Espaces de couleurs HLS
- Espaces de couleurs CMJ et CMJN
- Espaces de couleurs dépendants de l'appareil
- Espaces de couleurs indépendants de l'appareil

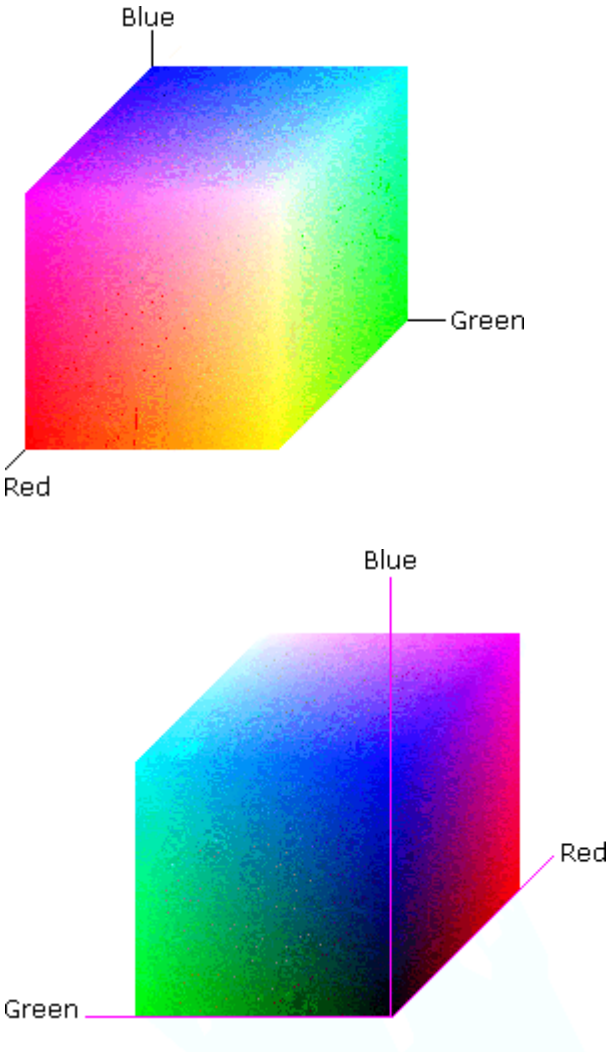
Commentaires

Obtenir de l’aide sur Microsoft Q&A

Espaces de couleurs RVB

Article • 13/06/2023

Un espace de couleurs RVB est créé en mappant les couleurs rouge, vert et bleu sur un système de coordonnées cartésien 3D. Ce résultat est un cube 3D comme ceux illustrés dans la figure suivante. Cette figure affiche le même cube RVB sous deux angles différents. Notez que l'origine du système de coordonnées est noire. C'est là que les composants de couleur rouge, vert et bleu (RVB) sont tous 0.0. Le coin diagonalement opposé du cube est blanc, où les composants de couleur RVB sont à leur valeur maximale.



Comme la plupart des espaces de couleurs, l'espace de couleur RVB est normalisé. Autrement dit, toutes les valeurs de couleur sont limitées à la plage de zéro à un inclus. Le noir est donc (0.0, 0.0, 0.0) et le blanc est (1.0, 1.0, 1.0, 1.0).

Dans l'espace de couleurs RVB, les couleurs principales sont le rouge, le vert et le bleu. Les couleurs secondaires sont cyan, jaune et magenta.

Les espaces de couleurs RVB peuvent être dépendants de l’appareil ou indépendants de l’appareil.

Commentaires

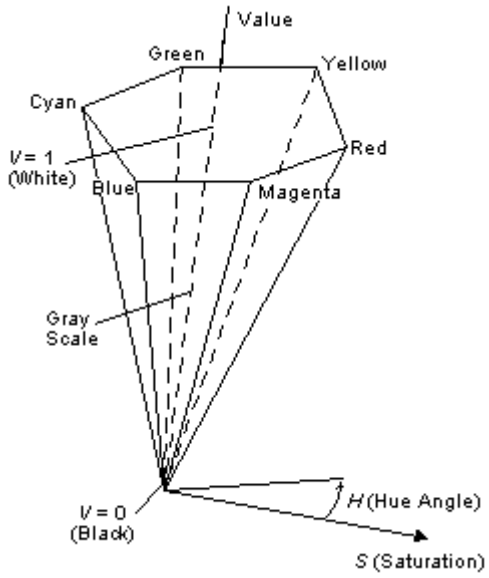
Cette page a-t-elle été utile ? Yes No

Obtenir de l’aide sur Microsoft Q&A

Espaces de couleurs HSV

Article • 13/06/2023

Les espaces de couleurs de teinte, de saturation et de valeur (HSV) sont souvent utilisés par les artistes. « Hue » est ce que nous considérons normalement comme couleur. Il s'agit de l'attribut d'une couleur par laquelle nous lui donnons un nom tel que « rouge » ou « bleu ». « Valeur » est un autre mot pour « légèreté », l'attribut d'une couleur qui le rend équivalent à une nuance de gris entre le noir et le blanc. La saturation est une mesure de la façon dont une couleur apparaît différemment d'un gris de la même légèreté. La saturation zéro indique qu'il n'y a pas de teinte, juste une échelle de gris. L'espace de couleur HSV est normalisé.



La figure précédente montre un dessin au trait de l'espace HSV sous la forme d'une hexcone. Chacune de ses sections croisées est un hexagone. Aux sommets de chaque section croisée se trouvent les couleurs rouge, jaune, vert, cyan, bleu et magenta. Une couleur dans l'espace HSV est spécifiée en indiquant un angle de teinte, le niveau de chroma et le niveau de luminosité. Un angle de teinte égal à zéro est rouge. L'angle de teinte augmente dans le sens inverse des aiguilles d'une montre. Les couleurs complémentaires sont séparées de 180.

Les espaces de couleurs HSV peuvent être dépendants de l'appareil ou indépendants de l'appareil.

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l’aide sur Microsoft Q&A

Espaces de couleurs HLS

Article • 13/06/2023

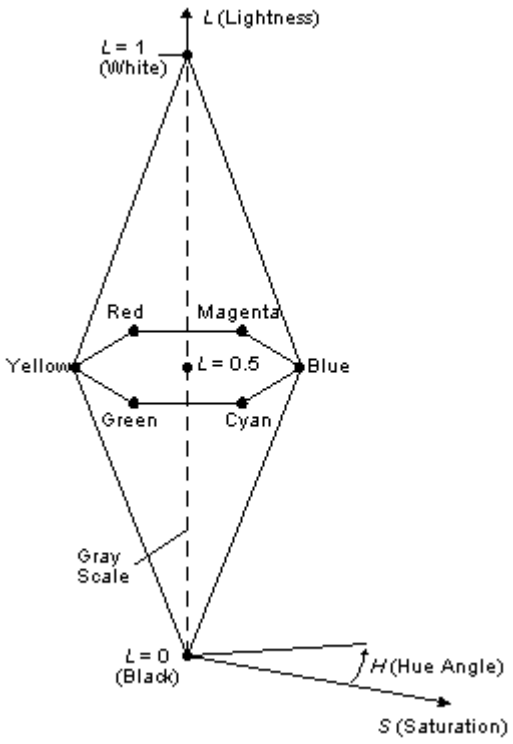
Les espaces de couleur HLS sont également largement utilisés par les artistes. Ses composants de couleur sont la teinte, la légèreté et la saturation (chroma).

Hue a la même signification que le modèle HSV, sauf qu'un angle de teinte de 0 correspond au bleu dans ce modèle. Magenta est à 60, le rouge est à 120. Comme pour le modèle HSV, les couleurs complémentaires sont séparées de 180.

La luminosité est la quantité de noir ou de blanc dans une couleur. L'augmentation de la luminosité ajoute du blanc à la teinte. La diminution de la luminosité ajoute du noir à la teinte.

La saturation dans le modèle HLS est une mesure de la « pureté » d'une teinte. À mesure que la saturation diminue, la teinte devient plus grise. Une valeur de saturation de zéro entraîne une valeur d'échelle de gris.

La figure suivante est un dessin au trait de l'espace HLS, qui est une double hexadécimal. Toute section horizontale de l'espace de couleur HLS est un hexagone. HLS est un espace de couleurs normalisé. Autrement dit, les valeurs de luminosité et de saturation sont comprises entre 0,0 et 1,0 inclus. Hue varie de 0 à 360 inclus.



Les espaces de couleurs HLS peuvent être dépendants de l'appareil ou indépendants de l'appareil.

Commentaires

Cette page a-t-elle été utile ? Yes No

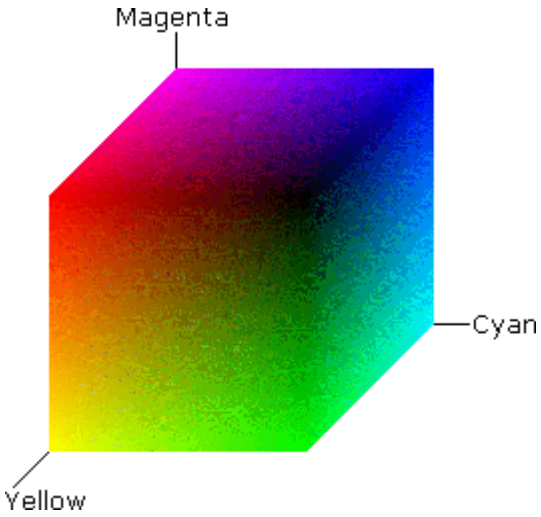
Obtenir de l'aide sur Microsoft Q&A

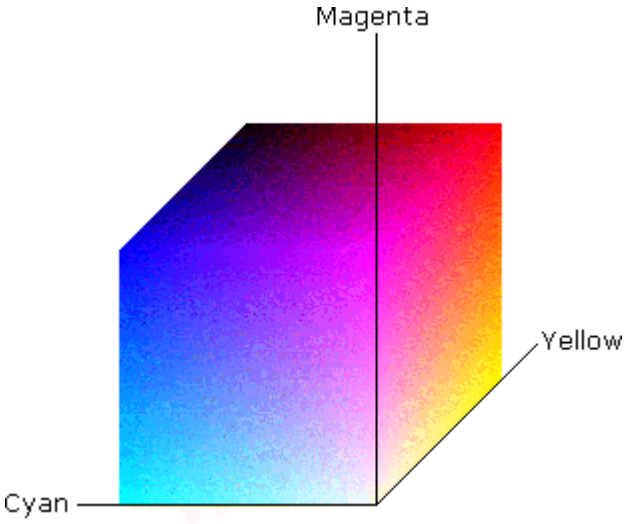
Espaces de couleurs CMJ et CMJN

Article • 13/06/2023

Les espaces de couleurs CMY et CMJN sont souvent utilisés dans l'impression couleur. Un espace de couleur CMY utilise le cyan, le magenta et le jaune (CMY) comme couleurs primaires. Le rouge, le vert et le bleu sont les couleurs secondaires.

Les illustrations suivantes sont des représentations en couleur de l'espace de couleur CMY. L'espace de couleur CMY est normalisé.





L'espace de couleur CMY est soustractif. Par conséquent, le blanc est à (0.0, 0.0, 0.0) et le noir est à (1.0, 1.0, 1.0). Si vous commencez par le blanc et ne soustrait aucune couleur, vous obtenez le blanc. Si vous commencez par le blanc et que vous soustrayez toutes les couleurs de la même façon, vous obtenez du noir.

L'espace de couleur CMJN est une variante du modèle CMY. Il ajoute du noir (Cyan, Magenta, Jaune et blacK). L'espace de couleur CMJN comble l'écart entre la théorie et la pratique. En théorie, le composant noir supplémentaire n'est pas nécessaire. Toutefois, l'expérience avec différents types d'encres et de papiers a montré que lorsque des

composants égaux de cyan, de magenta et d'encres jaunes sont mélangés, le résultat est généralement brun foncé, et non noir. L'ajout d'une entrée manuscrite noire au mélange résout ce problème.

Les espaces de couleurs CMY et CMJN peuvent être indépendants de l'appareil, mais le plus souvent ils sont utilisés en référence à un appareil spécifique.

Commentaires

Espaces de couleurs dépendants de l'appareil

Article • 13/06/2023

La plupart des espaces de couleurs dépendent de l'appareil. Même si un appareil particulier, tel qu'une imprimante, peut utiliser l'espace de couleur CMJN, les couleurs qu'il affiche pour des valeurs CMJN spécifiques sont souvent légèrement différentes de tous les autres types d'imprimantes. C'est précisément pour cette raison que le système de gestion des couleurs WCS 1.0 est si utile.

Tous les espaces de couleur ont un*point blanc*, qui est défini comme le blanc le plus blanc pouvant être produit dans cet espace de couleurs. Étant donné que les appareils peuvent différer les uns des autres, leurs points blancs peuvent également différer. Toutes les couleurs qu'un appareil peut produire sont relatives à son point blanc. Par conséquent, un système de gestion des couleurs doit être en mesure de mapper le point blanc d'un espace de couleurs dans un autre et de conserver les emplacements relatifs de toutes les couleurs. Il doit également être en mesure de mapper une couleur dans un espace de couleur à sa correspondance la plus proche dans un autre espace de couleur, quelles que soient les différences dans les points blancs. WCS 1.0 est en mesure d'accomplir ces deux tâches.

L'espace colorimétrique RVB est souvent utilisé sur les moniteurs d'ordinateur. En tant que tel, il dépend de l'appareil. Les imprimantes utilisent généralement des colorants CMJN. Chaque imprimante implémente sa propre version de l'espace de couleur CMJN. Il se peut que les images numériques ne stockent pas réellement les couleurs qu'elles contiennent. Ils peuvent stocker des numéros d'index dans une palette de couleurs. Le résultat est qu'il est très difficile pour les développeurs d'applications individuelles d'utiliser ou de fournir une méthode standardisée de déplacement d'images couleur d'un appareil à un autre. Les professionnels de l'imagerie rencontrent généralement la frustration de créer une image graphique sur un écran d'ordinateur et de la faire tourner très différemment lorsqu'elle est imprimée. WCS 1.0 répond à ces besoins en matière d'imagerie.

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l’aide sur Microsoft Q&A

Espaces de couleurs indépendants de l'appareil

Article • 13/06/2023

Reconnaissant la nécessité de mesures de couleur standard et indépendantes des appareils, la Commission internationale de l'éclairage (CIE) a créé un espace de couleurs basé sur des couleurs primaires « imaginaires ». Aucun appareil réel n'est censé produire des couleurs dans cet espace de couleurs. Il est utilisé comme un moyen de convertir des couleurs d'un espace de couleurs à un autre. Les couleurs primaires de cet espace de couleurs sont les couleurs abstraites X, Y et Z.

L'espace de couleur CIEXYZ 1931 est largement utilisé comme base pour la conversion de l'espace de couleur. Avec l'essor d'Internet, cependant, les considérations relatives à la bande passante ont rendu l'espace de couleur XYZ difficile à utiliser. L'échange d'images sur la bande passante limitée d'Internet nécessite un modèle de couleur plus compact.

Par conséquent, Hewlett-Packard et Microsoft ont proposé l'adoption d'un espace de couleurs RVB prédéfini standard appelé sRGB, afin de permettre une gestion précise des couleurs avec très peu de surcharge de données. Cette norme a été approuvée en tant que norme internationale officielle par la Commission électrotechnique internationale (IEC) sous la forme IEC 61966-2-1 : Systèmes et équipements multimédiasParamètres et gestion de la couleurPartie 2-1 : Gestion des couleursDéfiniment RVB. Il est disponible directement à partir de la IEC à l'adresse https://www.iec.ch/ . Des informations sur les problèmes techniques liés à sRGB sont disponibles sur Internet à l'adresse suivante :

Espace de couleur par défaut standard pour Internet - sRGB

Une version de fichier d'aide d'un livre blanc traitant des problèmes techniques impliqués dans sRGB, sRGB.HLP, est disponible dans le dossier \Help de la référence du programmeur WCS 1.0 dans le Kit de développement logiciel (SDK) de plateforme.

Différents formats de fichier peuvent utiliser ou ajouter un indicateur pour spécifier que l'image se trouve dans l'espace de couleur sRGB. Au format bitmap indépendant de l'appareil (DIB) Windows, la définition du membre **bV5CSType** de la structure BITMAPV5HEADER sur LCS_sRGB spécifie que les couleurs DIB se trouvent dans l'espace de couleurs sRGB.

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l’aide sur Microsoft Q&A

Conversion des couleurs et correspondance des couleurs

Article • 13/06/2023

Le processus de conversion des couleurs d'un espace de couleurs en un autre est appelé *conversion de couleurs*. Toutes les couleurs d'un espace colorimétrique sont fixes par rapport au point blanc de l'espace de couleurs. Étant donné que le point blanc d'un espace de couleurs varie d'un appareil à l'autre, une couleur convertie doit ensuite être mise en correspondance avec sa couleur visuellement la plus proche dans l'espace de couleur de destination. Ce processus est appelé *mappage de couleurs*.

Par exemple, si vous avez une image numérique que vous avez créée sur votre écran, elle peut se trouver dans un espace de couleurs RVB dépendant de l'appareil. Si vous souhaitez l'imprimer sur une imprimante, elle doit être convertie en espace de couleur de l'imprimante. Étant donné que l'imprimante utilise probablement un espace de couleur CMJN dépendant de l'appareil, toutes les valeurs RVB doivent être converties en CYMK. Il s'agit de la conversion de couleur. Une fois que les valeurs sont spécifiées en termes d'espace CYMK, elles doivent être mises en correspondance avec la couleur la plus proche que l'imprimante peut produire. C'est ce qu'on appelle le mappage des couleurs ou la correspondance des couleurs.

La conversion de couleurs et le mappage des couleurs doivent prendre en compte un certain nombre de facteurs spécifiques à l'appareil. Par instance, les blocs de construction des images d'écran sont des pixels. Chaque pixel a un nombre défini de bits pour stocker sa couleur ou sa valeur d'index de couleur. Le nombre de bits par pixel dépend du type d'affichage et de l'adaptateur graphique utilisés, ainsi que du mode de définition de l'adaptateur. La plupart des types d'imprimantes utilisent différents colorants et impriment à des résolutions particulières.

En outre, les caractéristiques de tonalité physique d'un appareil sont souvent différentes sur différents appareils. Par instance, lorsque les couleurs sont produites par des écrans d'ordinateur, elles peuvent sembler différentes de celles qu'elles auraient si elles étaient produites sur une presse à imprimer. Un facteur de correction, appelé *correction gamma*, est fréquemment utilisé pour compenser ces différences.

Le résultat de ces facteurs dépendants de l'appareil est que chaque appareil de couleur a un ensemble particulier de couleurs qu'il peut produire. Ce jeu de couleurs est connu sous le nom *de sa gamme*. Pour effectuer une conversion de couleurs et un mappage de couleurs, les couleurs d'une image doivent être converties de l'espace de couleurs et de

la gamme de l’appareil source dans l’espace de couleurs de l’appareil de destination. Ils sont ensuite mis en correspondance dans la gamme de l’appareil de destination.

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l’aide sur Microsoft Q&A

Informations complémentaires

Article • 13/06/2023

Vous trouverez une explication plus détaillée des termes et concepts abordés dans cette vue d'ensemble de la gestion des couleurs dans les sources suivantes :

- R.W.G. Hunt, *Measuring Color*, Prentice Hall 1991 (2e édition).
- James D. Foley, *Computer Graphics: Principles and Practice*, Addison-Wesley 1990 (2e édition).
- Roy Hall, *Illumination and Color in Computer Generated Imagery*, Springer Verlag 1988.
- Fred W. Billmeyer, Jr., Max Saltzmann, et Roy Berns, *Principles of Color Technology*, Wiley 1999 (3e édition).

Une autre bonne source d'information la page Web de l'International Color Consortium (ICC) (color.org). Parmi les membres de la CPI figurent des dirigeants et des organismes de normalisation de l'industrie informatique et de l'imagerie.

Commentaires

Schémas et algorithmes du système de couleurs Windows

Article • 13/06/2023

Cette section contient des descriptions étendues des schémas et algorithmes utilisés par le système de couleurs Windows.

- Schéma, gestion des versions et stratégies de localisation des types de profils communs WCS
- Schéma et algorithmes de profil de modèle d'apparence de couleur
- Schéma et algorithmes de profil de modèle d'appareil couleur
- Schéma d'étalonnage
- Schéma et algorithmes de profil de modèle gamut Map
- Algorithmes de création de transformation WCS

Commentaires

Schéma des types de profils communs du système de couleurs Windows, stratégies de gestion des versions et de localisation

Article • 13/06/2023

Schéma des types de profils communs WCS V1

Ce qui suit fournit la définition de schéma v1.0 pour les types de profils communs WCS.

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"

xmlns:wcs="http://schemas.microsoft.com/windows/2005/02/color/WcsCommonProfi
leTypes"

targetNamespace="http://schemas.microsoft.com/windows/2005/02/color/WcsCommo
nProfileTypes"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  blockDefault="#all"
  version="1.0">

  <xs:import namespace="http://www.w3.org/XML/1998/namespace" />

  <xs:annotation>
    <xs:documentation xml:lang="en">
      Common types used in WCS profile schemas.
      Copyright (C) Microsoft. All rights reserved.
    </xs:documentation>
```

```
  </xs:annotation>

  <xs:simpleType name="NonNegativeFloatType">
    <xs:restriction base="xs:float">
      <xs:minInclusive value="0"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="NonNegativeCIEXYZType">
    <xs:attribute name="X" type="wcs:NonNegativeFloatType" use="required"/>
    <xs:attribute name="Y" type="wcs:NonNegativeFloatType" use="required"/>
    <xs:attribute name="Z" type="wcs:NonNegativeFloatType" use="required"/>
  </xs:complexType>

  <xs:simpleType name="GUIDType">
    <xs:restriction base="xs:string">
      <xs:pattern value="\{[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-
fA-F0-9]{4}-[a-fA-F0-9]{12}\}"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="LocalizedTextType">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute ref="xml:lang" use="required"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name="MultiLocalizedTextType">
    <xs:sequence>
      <xs:element name="Text" type="wcs:LocalizedTextType" minOccurs="1" />
    </xs:sequence>
  </xs:complexType>

</xs:schema>
```

Seuls les encodages UTF-8 ou UTF-16 sont pris en charge. Tous les autres encodages XML sont fortement déconseillés afin de préserver une interopérabilité optimale.

Ajouts de schémas de types de profils communs WCS V2

Dans Windows 7, le schéma Types de profils communs WCS a été mis à jour pour inclure des types pour prendre en charge le schéma d'étalonnage WCS.

```
XML
```

```
  <xs:complexType name="ParameterizedCurvesType">
    <xs:sequence>
      <xs:element name="RedTRC" type="wcs:ParameterizedCurveType"/>
      <xs:element name="GreenTRC" type="wcs:ParameterizedCurveType"/>
      <xs:element name="BlueTRC" type="wcs:ParameterizedCurveType"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="ParameterizedCurveType">
    <xs:attribute name="Gamma" type="wcs:NonNegativeFloatType"
use="required"/>
    <xs:attribute name="Offset1" type="xs:float" use="optional"/>
    <xs:attribute name="Gain" type="wcs:NonNegativeFloatType"
use="optional"/>
    <xs:attribute name="Offset2" type="xs:float " use="optional"/>
    <xs:attribute name="TransitionPoint" type="wcs:NonNegativeFloatType"
use="optional"/>
    <xs:attribute name="Offset3" type="xs:float" use="optional"/>
  </xs:complexType>

  <xs:simpleType name="FloatList">
    <xs:list itemType="xs:float"/>
  </xs:simpleType>

  <xs:complexType name="OneDimensionLutType">
    <xs:sequence>
      <xs:element name="Input" type="wcs:FloatList"/>
      <xs:element name="Output" type="wcs:FloatList"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="HDRToneResponseCurvesType">
    <xs:sequence>
      <xs:element name="RedTRC" type="wcs:OneDimensionLutType"/>
      <xs:element name="GreenTRC" type="wcs:OneDimensionLutType"/>
      <xs:element name="BlueTRC" type="wcs:OneDimensionLutType"/>
    </xs:sequence>
    <xs:attribute name="TRCLength" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:int">
          <xs:minInclusive value="2" />
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
```

Contrôle de version du schéma de profil WCS

Dans cette annexe, le terme « consommateur de profil » fait référence au logiciel WCS ou à un composant logiciel tiers qui consomme des profils WCS.

Les versions plus récentes des consommateurs de profils pourront utiliser des profils WCS écrits selon des versions antérieures des schémas. Pour tirer pleinement parti des fonctionnalités définies dans la dernière version des schémas, la version la plus récente d'un consommateur de profil est généralement requise. Toutefois, les consommateurs de profils plus anciens peuvent utiliser des profils écrits selon des versions plus récentes des schémas. Le consommateur de profil peut ignorer les éléments et attributs XML qu'il ne comprend pas. Les résultats sont corrects, mais les performances ou la fidélité peuvent se dégrader par rapport aux résultats obtenus avec la dernière version du consommateur de profil.

Voici des instructions relatives au contrôle de version pour les consommateurs de profils :

- Une version donnée d'un consommateur de profil doit reconnaître tous les éléments et attributs d'un espace de noms de version, ou aucun d'entre eux.
- Si un consommateur de profil rencontre un élément ou un attribut d'un espace de noms qu'il ne comprend pas, il doit le traiter comme une condition d'erreur, sauf si le profil contient des instructions explicites sur le traitement de secours.
- La spécification Open Packaging Specification définit un URI d'espace de noms supplémentaire, l'espace de noms de compatibilité du balisage, qui définit les éléments et les attributs qui fournissent ce guide de traitement de secours.
- Toutes les versions de tous les consommateurs de profils doivent reconnaître et respecter tous les éléments et attributs de l'espace de noms de compatibilité du balisage.
- Les consommateurs de profils basés sur une nouvelle version des schémas de profil doivent prendre en charge tous les espaces de noms de version définis précédemment.

Dans la version v1.0, WCS reconnaît les éléments et les attributs des espaces de noms suivants :

- `https://schemas.microsoft.com/windows/2005/02/color/ColorDeviceModel`
- `https://schemas.microsoft.com/windows/2005/02/color/GamutMapModel`
- `https://schemas.microsoft.com/windows/2005/02/color/ColorAppearanceModel`
- `https://schemas.microsoft.com/winfx/2005/06/markup-compatibility`

Voici un exemple de compatibilité de balisage.

XML

```
<?xml version="1.0" encoding="utf-8" ?>
<ColorAppearanceModel

xmlns=http://schemas.microsoft.com/windows/2005/02/color/ColorAppearanceMode
```

```
1

xmlns:cam2=http://schemas.microsoft.com/windows/2007/08/color/ColorAppearanc
eModel
  xmlns:mc=http://schemas.microsoft.com/winfx/2005/02/markup-compatibility

xs:schemaLocation="http://schemas.microsoft.com/windows/2005/02/color/ColorA
ppearanceModel ColorAppearanceModel.xsd"
  xmlns:xs='http://www.w3.org/2001/XMLSchema-instance'
  mc:Ignorable="cam2">

  <ProfileName>Default profile for ICC viewing conditions</ProfileName>
  <mc:AlternateContent>
    <mc:Choice Requires="cam2">
      <cam2:AudioDescription source="ProfileExplanation.wav"/>
    </mc:Choice>
    <mc:Fallback>
      <Description>Appropriate for a print in a D50 viewing
booth</Description>
        </mc:Fallback>
  </mc:AlternateContent>
  <Author>Microsoft</Author>

  <ViewingConditions>
    <WhitePointName>D50</WhitePointName>
    <Background X="19.3" Y="20.0" Z="16.5" />
    <Surround>Average</Surround>
    <LuminanceOfAdaptingField>31.8</LuminanceOfAdaptingField>
    <DegreeOfAdaptation>-1</DegreeOfAdaptation>
    <cam2:Humidity>30%</cam2:Humidity>
  </ViewingConditions>

</ColorAppearanceModel>
```

Localisation de profil WCS

Configuration requise

Les profils WCS contiennent certains éléments tels que ProfileName et Description qui contiennent du texte lisible par l'homme. Ce texte est localisable.

Voici des recommandations en matière de gestion des versions pour les créateurs de profils :

- Pour les profils créés par des tiers, tels que des fabricants d'imprimantes, l'auteur du profil doit incorporer du texte descriptif dans plusieurs langues.
- Les éléments suivants doivent être localisables :

Élément	CDMP	GMMP	CAMP
ProfileName	x	x	x
Description	x	x	x
Auteur	x	x	x

Expression d'éléments localisables

Au lieu de contenir directement du texte, chaque élément localisable contient un ou plusieurs sous-éléments wcs:Text, chacun d'entre eux doit spécifier l'attribut xml:lang. Comme décrit dans la section 2.12 de la spécification XML (« Identification de la langue »), la valeur de l'attribut xml:lang doit être un identificateur de langue IETF RFC 3066 tel que « en-US », « en » ou « fr-FR ». Dans les schémas WCS, la valeur de l'attribut xml:lang ne doit pas être la chaîne vide « », même si XML l'autorise dans le cas général. Par exemple :

XML

```
<cdm:ColorDeviceModel ... />
    <cdm:Description>
        <wcs:Text xml:lang="en-US">Hello</wcs:Text>
        <wcs:Text xml:lang="fr-FR">Bonjour</wcs:Text>
    </cdm:Description>
    ...
</cdm:ColorDeviceeModel>
```

Aucun élément wcs:Text ne peut avoir le même attribut xml:lang. Chaque schéma de profil WCS doit appliquer cette contrainte séparément pour chaque élément localisable. Seuls les codages UTF-8 et UTF-16 sont pris en charge.

Prise en charge des schémas

La conception utilise les types XSD wcs:LocalizedText et wcs:MultiLocalizedType définis dans le schéma type de profil commun WCS :

XML

```
    <xs:complexType name="LocalizedText">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="xml:lang" type="xs:string"
```

```
use="required"/>
            <xs:extension/>
        <xs:simpleContent/>
    </xs:complexType/>

    <xs:complexType name="MultiLocalizedType">
        <xs:element name="Text" type="cam:LocalizedText" minOccurs="1"/>
    </xs:complexType>
```

Chaque schéma de profil WCS déclare que chaque élément localisable est de type MultiLocalizedType, par exemple :

XML

```
<xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"

xmlns:cdm="http://schemas.microsoft.com/windows/2005/02/color/ColorDeviceMod
el"

xmlns:wcs="http://schemas.microsoft.com/windows/2005/02/color/WcsCommonProfi
leTypes"
    targetNamespace=

"http://schemas.microsoft.com/windows/2005/02/color/ColorDeviceModel"
    version="1.0">
    <xs:import namespace=

"http://schemas.microsoft.com/windows/2005/02/color/WcsCommonProfileTypes"/>

    <xs:element name="cdm:Description" type="wcs:MultiLocalizableType"
minOccurs="0"/>

    <xs:unique name="uniqueDescriptionLanguage">
      <xs:selector xpath="cdm:ColorDeviceModel/cdm:Description/wcs:Text"/>
      <xs:field xpath="@xml:lang"/>
    </unique>

    ...
</xs:schema>
```

Le schéma CDMP applique la contrainte selon laquelle chaque enfant wcs:Text de l'élément cdm:Description doit avoir un attribut xml:lang unique. Il applique la même contrainte pour les autres éléments localisables. Les schémas CAMP et GMMP font de même.

Sélection de la langue

Lorsque vous décidez de la version linguistique d'un élément localisable à afficher, le code d'application doit sélectionner la « version la plus proche » de la « langue actuelle ». Ce que signifient réellement « version la plus proche » et « langue actuelle » dépend de la plateforme ; voir ci-dessous. Si le profil ne contient pas de version de langue qui correspond à la langue actuelle, l'application doit sélectionner la première version du profil (le premier élément enfant wcs:Text de l'élément localisable).

Sur Windows Vista, la sélection de la langue s'effectue comme suit : chaque thread a une liste associée de « langues d'interface utilisateur préférées », qui peut être obtenue en appelant **GetThreadPreferredUILanguages**. La liste est retournée par ordre de préférence utilisateur. Par instance, sur un système configuré pour l'anglais américain, la liste retournée par **GetThreadPreferredUILanguages** est { « en-US », « en » }. Cela signifie : 1) anglais américain s'il est présent. 2) Sinon, utilisez une variante régionale de l'anglais, telle que « en-GB » ou simplement « en ».

Pour chaque langue de cette liste, dans l'ordre de liste, le code de l'interface utilisateur recherche un élément wcs:Text dont l'attribut xml:lang est une correspondance exacte. La première version correspondante est utilisée. Si aucune version ne correspond, le premier élément wcs:Text est utilisé.

Profils intégrés

Les profils WCS standard fournis avec Windows Vista, tels que sRGB.cdmp, ont les mêmes propriétés localisables que d'autres profils.

La solution Windows standard consiste à placer les chaînes localisées dans une DLL MUI et à y faire référence comme suit :

XML

```
<cdm:Description resourceId="mscms.dll,-101">
    <wcs:Text xml:lang="en-US">Hello, world!</wcs:Text>
<cdm:Description>
```

Sur un système Windows, le texte de description est extrait de l'ID de ressource 101 dans la version localisée appropriée des ressources MUI pour mscms.dll. Les systèmes autres que Windows utilisent les sous-éléments wcs:Text comme décrit ci-dessus.

Nous introduisons un attribut d'ID facultatif et globalement unique sur l'élément racine de chaque schéma de profil WCS. Le profil standard wcsRGB.cdmp peut ressembler à ceci :

XML

```
<cdm:ColorDeviceModel
    ID=http://schemas.microsoft.com/2005/02/color/profiles/wcsRGB.cdmp
    ... >
    <cdm:Description>
        <wcs:Text xml:lang="en-US">Hello.</wcs:Text>
        <wcs:Text xml:lang="fr-FR">Bonjour.</wcs:Text>
    </cdm:Description>
    ...
</cdm:ColorDeviceModel>
```

Pour les profils de couleurs WCS fournis avec des fenêtres, le CPL Couleur affiche des informations descriptives à partir des ressources, plutôt que des éléments wcs:Text dans les profils. Cela permet à Windows d'afficher des informations localisées pour ces profils dans toutes les langues prises en charge, sans exiger que les profils contiennent des informations descriptives dans toutes les langues.

Rubriques connexes

Concepts de base de la gestion des couleurs

Schémas et algorithmes du système de couleurs Windows

Commentaires

Algorithme et schéma de profil de modèle d’apparence de couleur WCS

Article • 13/06/2023

Vue d’ensemble

Ce schéma est utilisé pour spécifier le contenu d’un profil de modèle d’apparence de couleur (CAMP). Les algorithmes de base de référence associés sont décrits dans les sections suivantes.

Le CAMP est composé de balises XML qui fournissent des valeurs paramétriques aux variables du modèle d’apparence de couleur de base CIECAM02. Des détails sur les plages de paramètres sont fournis dans la spécification du modèle d’apparence de couleur de base et la recommandation CIECAM02.

Architecture du profil de modèle d’apparence de couleur

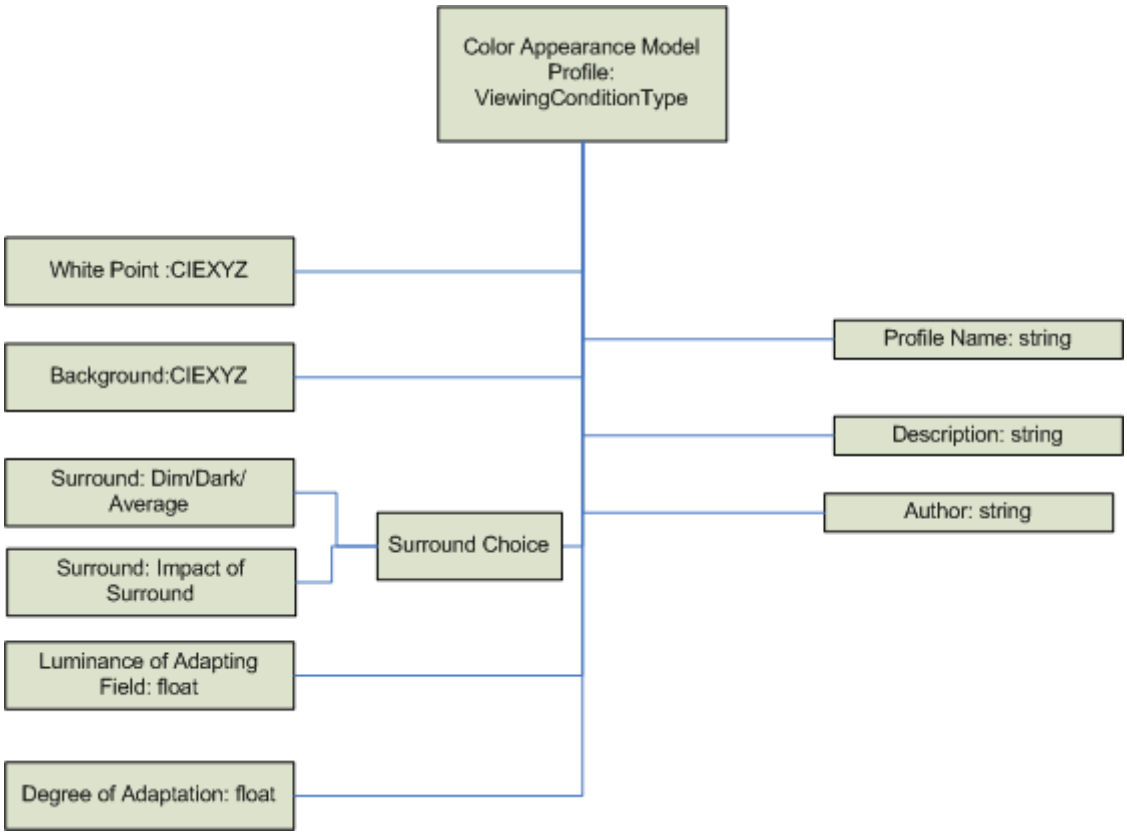


Schéma CAMP

C++

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema

xmlns:cam="http://schemas.microsoft.com/windows/2005/02/color/ColorAppearanceModel"

xmlns:wcs="http://schemas.microsoft.com/windows/2005/02/color/WcsCommonProfileTypes"

targetNamespace="http://schemas.microsoft.com/windows/2005/02/color/ColorAppearanceModel"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  blockDefault="#all"
  version="1.0">

  <xs:annotation>
    <xs:documentation>
      Color Appearance Model profile schema.
      Copyright (C) Microsoft. All rights reserved.
    </xs:documentation>
  </xs:annotation>

  <xs:import
namespace="http://schemas.microsoft.com/windows/2005/02/color/WcsCommonProfi
```

```
leTypes" />

  <xs:annotation>
    <xs:documentation>
      ColorAppearanceModel element contains viewing conditions
      parameters based on CIECAM02.
    </xs:documentation>
  </xs:annotation>
  <xs:element name="ColorAppearanceModel">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ProfileName" type="wcs:MultiLocalizedTextType"/>
        <xs:element name="Description" type="wcs:MultiLocalizedTextType"
minOccurs="0"/>
        <xs:element name="Author" type="wcs:MultiLocalizedTextType"
minOccurs="0"/>
        <xs:element name="ViewingConditions">
          <xs:complexType>
            <xs:sequence>
              <xs:choice>
                <xs:element name="WhitePoint"
type="wcs:NonNegativeCIEXYZType"/>
                <xs:element name="WhitePointName">
                  <xs:simpleType>
                    <xs:restriction base="xs:string">
                      <xs:enumeration value="D50"/>
                      <xs:enumeration value="D65"/>
                      <xs:enumeration value="A"/>
                      <xs:enumeration value="F2"/>
                    </xs:restriction>
                  </xs:simpleType>
                </xs:element>
              </xs:choice>
              <xs:element name="Background"
type="wcs:NonNegativeCIEXYZType"/>
              <xs:choice>
                <xs:element name="ImpactOfSurround" type="xs:float"/>
                <xs:element name="Surround">
                  <xs:simpleType>
                    <xs:restriction base="xs:string">
                      <xs:enumeration value="Average"/>
                      <xs:enumeration value="Dim"/>
                      <xs:enumeration value="Dark"/>
                    </xs:restriction>
                  </xs:simpleType>
                </xs:element>
              </xs:choice>
              <xs:element name="LuminanceOfAdaptingField" type="xs:float"/>
              <xs:element name="DegreeOfAdaptation" type="xs:float"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="NormalizeToMediaWhitePoint" minOccurs="0">
          <xs:simpleType>
            <xs:restriction base="xs:string">
```

```
              <xs:enumeration value="True"/>
              <xs:enumeration value="False"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="ID" type="xs:string" use="optional" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Éléments de schéma CAMP

ColorAppearanceModel

Cet élément est une séquence de :

1. Chaîne ProfileName,
2. chaîne de description facultative,
3. chaîne d'auteur facultative,
4. Élément ViewingConditions.

Conditions de validation : Chaque sous-élément est validé par son propre type. Les longueurs de chaîne sont limitées à 10 000 caractères.

Espace de noms

xmlns:cam= »http://schemas.microsoft.com/windows/2005/02/color/ColorAppearanceModel"

targetNamespace= »http://schemas.microsoft.com/windows/2005/02/color/ColorAppearanceModel"

Version

Version >0.1 ou <= « 1.0 » avec la première version de Windows Vista.

Conditions de validation : Toute valeur <de version =2.0 est également valide pour prendre en charge les modifications sans rupture du format.

Documentation

Schéma de profil du modèle d’apparence de couleur.

Conditions de validation : Chaque sous-élément est validé par son propre type.

SurroundType

Cet élément est une énumération des paramètres CIECAM02 « Moyenne », « Dim » ou « Sombre » ou les paramètres quantitatifs réels de la recommandation CIECAM02 c, impact de l’environnement.

Conditions de validation : Le paramètre c peut être comprise entre 0,525 et 0,69.

AffichageConditions

Cet élément se compose des sous-éléments suivants :

Élément	Type
Point blanc	WhitePointType
Arrière-plan	CIEXYZ
Entourent	SurroundType
LuminanceOfAdaptingField	float
DegreeOfAdaptation	float
NormalizeToMediaWhitePoint	Booléen

Conditions de validation : Les sous-éléments CIEXYZ sont validés par NonNegativeXYZType. Le LuminanceOfAdaptingField est un maximum de 10 000cd/m^2. La valeur DegreeOfAdaptation peut être comprise entre 0.0 et 1.0. La valeur NormalizeToMediaWhitePoint peut être « true » ou « false ». Si le sous-élément NormalizeToMediaWhitePoint est absent, la valeur par défaut est « true ». Consultez la section relative à l’algorithme CAMP suivante.

WhitePointType

Cet élément est une énumération de la valeur de source lumineuse CIE (« D50 », « D65 », « A » ou « F2 ») ou un sous-élément CIEXYZ.

Conditions de validation : Chaque sous-élément est validé par son propre type.

CIEXYZType

L'élément CIEXYZType est composé de trois éléments nonNegativeFloatType à virgule flottante IEEE à précision unique, nommés « X », « Y » et « Z ». Ces mesures peuvent être des valeurs réfléchissantes absolues (non relatives) CIEXYZ 1931 ou absolues (non relatives) des valeurs directes (transmissives) CIEXYZ 1931 en candelas par mètre d'unités au carré.

Conditions de validation : Cela signifie que seules les valeurs réelles sont valides et que les valeurs de mesure CIEXYZ négatives ne sont pas valides. Étant donné qu'il s'agit de valeurs absolues, les valeurs peuvent dépasser 1,0f. Une limite raisonnable pour toute valeur X, Y ou Z sera arbitrairement définie sur 10000.0f.

Algorithme CAMP

Le modèle d'apparence de couleur (CAM) est basé sur les équations du modèle d'apparence de couleur CIECAM02.

Cette classe implémente la modélisation de l'apparence des couleurs. Notez que le CAM WCS *n'est pas* remplaçable, par exemple à l'aide d'un plug-in. Il s'agit d'un objectif de conception d'avoir un seul modèle d'apparence de couleur. Le CAM est basé sur les recommandations de CIECAM02.

CIECAM02 peut être utilisé de deux façons. Dans le sens colorimétrique-apparence, il fournit un mappage de l'espace CIE XYZ à l'espace d'apparence de couleur. Dans le sens de l'apparence à la couleur, il est mappé de l'espace d'apparence des couleurs à l'espace XYZ. L'apparence de couleur met en corrélation la légèreté, J, chroma, C et teinte, h. Ces trois valeurs forment un système de coordonnées cylindrique. Souvent, il s'avère plus pratique de travailler dans un système de coordonnées rectangulaire, donc calculez a = C cos h et b = C sin h, pour donner CIECAM02 Jab.

Vous pouvez utiliser des valeurs de légèreté CAM supérieures à 100. Le comité CIE qui a formulé CIECAM02 n'a pas examiné le comportement de l'axe de luminosité pour les valeurs d'entrée avec une luminance supérieure au point blanc adopté; autrement dit, pour les valeurs Y d'entrée supérieures à la valeur Y de point blanc adoptée. L'expérimentation a montré que les équations de luminance dans CIECAM02 se comportent raisonnablement pour ces valeurs. La légèreté augmente de façon exponentielle et suit le même exposant (environ 1/3).

Les utilisateurs souhaitent parfois modifier la façon dont le degré d'adaptation (D) est calculé. La conception WCS permet aux utilisateurs de contrôler ce calcul en modifiant la valeur degreeOfadaptation dans les paramètres des conditions d'affichage.

Pour fournir une correspondance plus cohérente aux attentes des utilisateurs influencées par l'ICC, la valeur degreeOfAdaptation dans le camps par défaut est 1.0. Cela produit de meilleurs résultats dans tous les cas autres que MinCD Absolute, où l'on peut laisser WCS calculer le degré d'adaptation (via degreeOfAdaptation = -1).

Au lieu d'utiliser une valeur surround de « Moyenne », « Dim » et « Sombre », une valeur d'environnement continue, calculée à partir de la valeur c, est fournie. La valeur de c doit être un float compris entre 0,525 et 0,69.

À partir de *c*, *Nc* et *F* peuvent être calculés à l'aide de l'interpolation linéaire au niveau de la pièce entre les valeurs déjà fournies pour « Moyenne », « Dim » et « Sombre ». Cela modélise ce qui est illustré dans la figure 1 de CIE 159:2004, la spécification CIECAM02.

degreeOfAdaption	**Comportement**
-1.0	$D = F\left[1 - \frac{1}{3.6}\right]e^{\left(\frac{-(L_A+42)}{92}\right)}$ Il s'agit du comportement CIECAM02 par défaut.
0,0 <= degreeOfAdaption <= 1,0	*D* = degreeOfAdaptation (valeur fournie par l'utilisateur)

Une certaine quantité de vérification des erreurs a également été ajoutée à l'implémentation. Les nombres d'équations suivants sont ceux utilisés dans la définition CIE 159:2004 de CIECAM02.

ColorimetricToAppearanceColors

Les valeurs d'entrée sont vérifiées pour vérifier le caractère raisonnable : si X ou Z < 0.0, ou si Y < -1.0, le HRESULT est E_INVALIDARG. Si -1.0 <= Y < 0.0, J, C et h sont tous définis sur 0.0.

Certaines conditions internes peuvent produire des résultats d'erreur. Au lieu de produire de tels résultats, les résultats internes sont clippés pour produire des valeurs dans la plage. Cela se produit pour les spécifications de couleurs qui seraient sombres et impossiblement chromatiques : Dans l'équation 7.23, si A < 0, A = 0. Dans l'équation 7.26, si t < 0, t = 0.

AppearanceToColorimetricColors

Le caractère raisonnable des valeurs d'entrée est vérifié. Si C < 0 , C > 300 ou J > 500, le HRESULT est E_INVALIDARG.

R'_a, G'_a, et B'_a, (équations 8.19 - 8.21) sont coupés à la plage 399,9 .

Pour tous les profils de modèle d'apparence de couleur (CAMP), le moteur WCS examine le point blanc adopté. Si Y n'est pas 100,0, le point blanc adopté est mis à l'échelle de sorte que Y soit égal à 100,0. La même mise à l'échelle sera appliquée à la valeur d'arrière-plan. Le facteur de mise à l'échelle est 100.0/adoptedWhitePoint.Y. Le même facteur de mise à l'échelle est appliqué à chaque X, Y et Z. Si le champ NormalizeToMediaWhitePoint est défini sur « True », ou s'il est absent du CAMP, le moteur met également à l'échelle toutes les couleurs d'appareil entrées sur DeviceToColorimetric afin que la valeur Y du point blanc du média de l'appareil soit égale à 100,0. Les couleurs d'appareil provenant de ColorimetricToDevice sont mises à l'échelle selon l'inverse multiplicatif de ce facteur de mise à l'échelle. Si l'indicateur NormalizeToMediaWhitePoint est défini sur « False », les données colorimétriques ne sont pas mises à l'échelle.

Pour certaines tâches, il est judicieux de mettre à l'échelle les valeurs colorimétriques provenant de DeviceToColorimetric. Les équations de légèreté hyperbolique dans le CAM sont vraiment conçues pour une luminosité de point blanc de 100,0. Le seul endroit où une différence dans la luminance absolue (ou illuminance) entre en jeu est dans la luminosité du champ d'adaptation. Par conséquent, le CAM doit être initialisé avec un point blanc Y de 100,0. Toutefois, si le point blanc moyen du modèle d'appareil est utilisé comme point blanc adopté, toutes les couleurs provenant de l'appareil doivent être mises à l'échelle en conséquence, sinon le blanc de l'appareil ne sortira pas avec une valeur J de 100,0. Les valeurs Y doivent donc être mises à l'échelle dans les mesures. Les valeurs de mesure peuvent être mises à l'échelle avant l'initialisation du modèle d'appareil. Les résultats seraient alors déjà dans la plage appropriée. Mais cela compliquerait le test du modèle d'appareil, car les valeurs sortantes nécessiteraient une mise à l'échelle. Pour les tâches dans lesquelles le point blanc moyen de l'appareil est perçu comme un vrai blanc, la normalisation par le point blanc du média de l'appareil est souhaitable.

Le CAM est initialisé directement à partir du CAMP. Cela permet aux développeurs une certaine flexibilité dans l'initialisation de la CAM, en fonction de la tâche qu'ils souhaitent effectuer. Dans certaines tâches, les observateurs ignorent toute couleur dans les points blancs du média, car ils « savent » cognitivement que les médias source et de destination sont « blancs ». Dans ce cas, les développeurs souhaitent initialiser les cams avant et inverse avec leurs points blancs multimédias respectifs. Dans certains cas, les observateurs peuvent comparer la couleur des arrière-plans multimédias. Dans ce cas, il est recommandé d'utiliser une seule CAM pour les deux appareils, et il peut être

souhaitable de ne pas mettre à l'échelle les valeurs colorimétriques de chaque appareil en fonction du point blanc moyen de cet appareil. Ensuite, les différentes valeurs tristimulus du média entraînent des valeurs d'apparence différentes dans CIECAM02.

Rubriques connexes

Concepts de base de la gestion des couleurs

Schémas et algorithmes du système de couleurs Windows

Commentaires

Schéma d’étalonnage WCS

Article • 13/06/2023

Cette rubrique décrit le schéma d’étalonnage WCS qui développe le profil de modèle d’appareil couleur WCS.

Schéma d’étalonnage WCS

La définition de schéma suivante est utilisée pour spécifier les nouvelles définitions Windows 7 qui prennent en charge l’étalonnage du profil de modèle d’appareil de couleur WCS .

C++

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  Copyright (C) Microsoft. All rights reserved. The Windows Color System
  color profile schemas may be used according to the terms of the license
agreement
  available at
https://www.microsoft.com/whdc/device/display/color/wcs_license.mspx.
  -->
<xs:schema
  xmlns:cal="http://schemas.microsoft.com/windows/2007/11/color/Calibration"

xmlns:wcs="http://schemas.microsoft.com/windows/2005/02/color/WcsCommonProfi
leTypes"

targetNamespace="http://schemas.microsoft.com/windows/2007/11/color/Calibrat
ion"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  blockDefault="#all"
  version="1.0">

  <xs:annotation>
    <xs:documentation>
      Color Device Model Calibration profile schema.
      Copyright (C) Microsoft. All rights reserved.
    </xs:documentation>
  </xs:annotation>

  <xs:import
namespace="http://schemas.microsoft.com/windows/2005/02/color/WcsCommonProfi
leTypes" />

  <xs:complexType name="AdapterGammaConfiguration">
    <xs:choice>
```

```
      <xs:element name="ParameterizedCurves"
type="wcs:ParameterizedCurvesType"/>
      <xs:element name="HDRToneResponseCurves"
type="wcs:HDRToneResponseCurvesType"/>
    </xs:choice>
  </xs:complexType>

  <xs:complexType name="Calibration">
    <xs:sequence>
      <xs:element name="AdapterGammaConfiguration"
type="cal:AdapterGammaConfiguration" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

</xs:schema>
```

Pour la compatibilité avec Windows Vista, les profils contenant des balises d'étalonnage doivent inclure l'attribut `mc:Ignoreable="cdm_calibration"`.

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

Algorithmes et schéma de profil de modèle d’appareil de couleur WCS

Article • 13/06/2023

Cette rubrique fournit des informations sur le schéma de profil de modèle d’appareil couleur WCS et ses algorithmes associés.

Cette rubrique contient les sections suivantes :

Vue d'ensemble

Ce schéma est utilisé pour spécifier le contenu d'un profil de modèle d'appareil couleur (CDMP). Les algorithmes de base de référence associés sont décrits ci-dessous.

Le schéma DMP (Device Model Profile) de base se compose des données de mesure d'échantillonnage.

Le composant d'échantillonnage du schéma XML DMP prend en charge les cibles de mesure de couleur de base, en se concentrant sur les cibles standard courantes et les cibles optimisées pour les modèles d'appareils de base.

En outre, le profil d'appareil fournit des informations spécifiques sur le modèle d'appareil ciblé et fournit une stratégie que le modèle d'appareil de secours de base peut utiliser si le modèle ciblé n'est pas disponible. Les instances de profil peuvent inclure des extensions privées à l'aide de mécanismes d'extension XML standard.

Architecture du profil de modèle d'appareil couleur

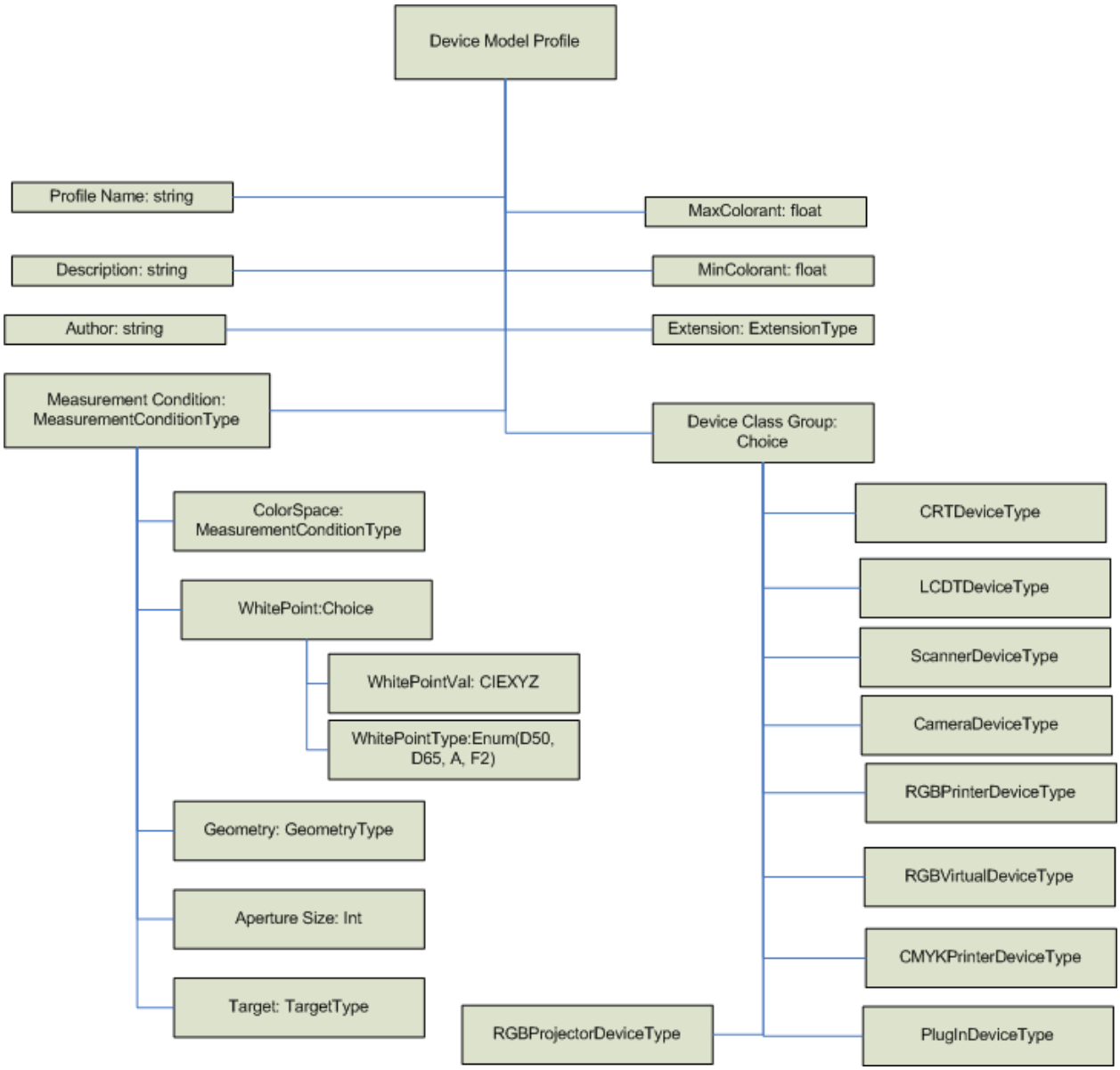


Schéma CDMP

C++

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema

xmlns:cdm="http://schemas.microsoft.com/windows/2005/02/color/ColorDeviceModel"

xmlns:wcs="http://schemas.microsoft.com/windows/2005/02/color/WcsCommonProfileTypes"

targetNamespace="http://schemas.microsoft.com/windows/2005/02/color/ColorDeviceModel"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  blockDefault="#all"
  version="1.0">
```

```
  <xs:annotation>
    <xs:documentation>
      Color Device Model profile schema.
      Copyright (C) Microsoft. All rights reserved.
    </xs:documentation>
  </xs:annotation>

  <xs:import
namespace="http://schemas.microsoft.com/windows/2005/02/color/WcsCommonProfi
leTypes" />

  <xs:complexType name="RGBType">
    <xs:attribute name="R" type="xs:float" use="required"/>
    <xs:attribute name="G" type="xs:float" use="required"/>
    <xs:attribute name="B" type="xs:float" use="required"/>
  </xs:complexType>

  <xs:complexType name="NonNegativeRGBType">
    <xs:attribute name="R" type="wcs:NonNegativeFloatType" use="required"/>
    <xs:attribute name="G" type="wcs:NonNegativeFloatType" use="required"/>
    <xs:attribute name="B" type="wcs:NonNegativeFloatType" use="required"/>
  </xs:complexType>

  <xs:complexType name="NonNegativeCMYKType">
    <xs:attribute name="C" type="wcs:NonNegativeFloatType" use="required"/>
    <xs:attribute name="M" type="wcs:NonNegativeFloatType" use="required"/>
    <xs:attribute name="Y" type="wcs:NonNegativeFloatType" use="required"/>
    <xs:attribute name="K" type="wcs:NonNegativeFloatType" use="required"/>
  </xs:complexType>

  <xs:complexType name="NonNegativeRGBSampleType">
    <xs:sequence>
      <xs:element name="RGB" type="cdm:NonNegativeRGBType"/>
      <xs:element name="CIEXYZ" type="wcs:NonNegativeCIEXYZType"/>
    </xs:sequence>
    <xs:attribute name="Tag" type="xs:string" use="optional"/>
  </xs:complexType>

  <xs:complexType name="NonNegativeCMYKSampleType">
    <xs:sequence>
      <xs:element name="CMYK" type="cdm:NonNegativeCMYKType"/>
      <xs:element name="CIEXYZ" type="wcs:NonNegativeCIEXYZType"/>
    </xs:sequence>
    <xs:attribute name="Tag" type="xs:string" use="optional"/>
  </xs:complexType>

  <xs:group name="RGBPrimariesGroup">
    <xs:sequence>
      <xs:element name="WhitePrimary" type="wcs:NonNegativeCIEXYZType"/>
      <xs:element name="RedPrimary" type="wcs:NonNegativeCIEXYZType"/>
      <xs:element name="GreenPrimary" type="wcs:NonNegativeCIEXYZType"/>
      <xs:element name="BluePrimary" type="wcs:NonNegativeCIEXYZType"/>
      <xs:element name="BlackPrimary" type="wcs:NonNegativeCIEXYZType"/>
    </xs:sequence>
```

```
  </xs:group>

  <xs:complexType name="MeasurementConditionsType">
    <xs:annotation>
      <xs:documentation>
      Optional measurement conditions.

      We only support CIEXYZ for measurement color space in this version.
      If the white point value from the measurement conditions is not
available,
      the default processing will use
        - "D50" for printer and scanners
        - "D65" for camera and displays.
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="ColorSpace" minOccurs="0">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="CIEXYZ"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:choice minOccurs="0">
        <xs:element name="WhitePoint" type="wcs:NonNegativeCIEXYZType"/>
        <xs:element name="WhitePointName">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="D50"/>
              <xs:enumeration value="D65"/>
              <xs:enumeration value="A"/>
              <xs:enumeration value="F2"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
      </xs:choice>
      <xs:element name="Geometry" minOccurs="0">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="0/45"/>
            <xs:enumeration value="0/diffuse"/>
            <xs:enumeration value="diffuse/0"/>
            <xs:enumeration value="direct"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="ApertureSize" type="xs:int" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="DisplayMeasurementType">
    <xs:sequence>
      <xs:group ref="cdm:RGBPrimariesGroup"/>
      <xs:element name="GrayRamp">
        <xs:complexType>
```

```
          <xs:sequence>
            <xs:element name="Sample" type="cdm:NonNegativeRGBSampleType"
maxOccurs="4096"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="RedRamp">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Sample" type="cdm:NonNegativeRGBSampleType"
maxOccurs="4096"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="GreenRamp">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Sample" type="cdm:NonNegativeRGBSampleType"
maxOccurs="4096"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="BlueRamp">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Sample" type="cdm:NonNegativeRGBSampleType"
maxOccurs="4096"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="TimeStamp" type="xs:dateTime"/>
  </xs:complexType>

  <xs:complexType name="RGBProjectorMeasurementType">
    <xs:sequence>
      <xs:group ref="cdm:RGBPrimariesGroup"/>
      <xs:element name="ColorSamples">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Sample" type="cdm:NonNegativeRGBSampleType"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="TimeStamp" type="xs:dateTime"/>
  </xs:complexType>

  <xs:simpleType name="OneBasedIndex">
    <xs:restriction base="xs:int">
      <xs:minInclusive value="1"/>
    </xs:restriction>
  </xs:simpleType>
```

```
  <xs:complexType name="RGBCaptureMeasurementType">
    <xs:sequence>
      <xs:element name="PrimaryIndex">
        <xs:complexType>
          <xs:all>
            <xs:element name="White" type="cdm:OneBasedIndex"/>
            <xs:element name="Black" type="cdm:OneBasedIndex"
minOccurs="0"/>
            <xs:element name="Red" type="cdm:OneBasedIndex" minOccurs="0"/>
            <xs:element name="Green" type="cdm:OneBasedIndex"
minOccurs="0"/>
            <xs:element name="Blue" type="cdm:OneBasedIndex" minOccurs="0"/>
            <xs:element name="Cyan" type="cdm:OneBasedIndex" minOccurs="0"/>
            <xs:element name="Magenta" type="cdm:OneBasedIndex"
minOccurs="0"/>
            <xs:element name="Yellow" type="cdm:OneBasedIndex"
minOccurs="0"/>
          </xs:all>
        </xs:complexType>
      </xs:element>
      <xs:element name="NeutralIndices">
        <xs:simpleType>
          <xs:list itemType="cdm:OneBasedIndex"/>
        </xs:simpleType>
      </xs:element>
      <xs:element name="ColorSamples">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Sample" type="cdm:NonNegativeRGBSampleType"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="TimeStamp" type="xs:dateTime"/>
  </xs:complexType>

  <xs:complexType name="RGBPrinterMeasurementType">
    <xs:sequence>
      <xs:element name="ColorCube">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Sample" type="cdm:NonNegativeRGBSampleType"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="TimeStamp" type="xs:dateTime"/>
  </xs:complexType>

  <xs:complexType name="CMYKPrinterMeasurementType">
    <xs:sequence>
      <xs:element name="ColorCube">
        <xs:complexType>
```

```
          <xs:sequence>
            <xs:element name="Sample" type="cdm:NonNegativeCMYKSampleType"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="TimeStamp" type="xs:dateTime"/>
  </xs:complexType>

  <xs:complexType name="GammaType">
    <xs:attribute name="value" type="wcs:NonNegativeFloatType"
use="required"/>
  </xs:complexType>

  <xs:complexType name="GammaOffsetGainType">
    <xs:attribute name="Gamma" type="wcs:NonNegativeFloatType"
use="required"/>
    <xs:attribute name="Offset" type="wcs:NonNegativeFloatType"
use="required"/>
    <xs:attribute name="Gain" type="wcs:NonNegativeFloatType"
use="required"/>
  </xs:complexType>

  <xs:complexType name="GammaOffsetGainLinearGainType">
    <xs:attribute name="Gamma" type="wcs:NonNegativeFloatType"
use="required"/>
    <xs:attribute name="Offset" type="wcs:NonNegativeFloatType"
use="required"/>
    <xs:attribute name="Gain" type="wcs:NonNegativeFloatType"
use="required"/>
    <xs:attribute name="LinearGain" type="wcs:NonNegativeFloatType"
use="required"/>
    <xs:attribute name="TransitionPoint" type="wcs:NonNegativeFloatType"
use="required"/>
  </xs:complexType>

  <xs:simpleType name="FloatList">
    <xs:list itemType="xs:float"/>
  </xs:simpleType>

  <xs:complexType name="OneDimensionLutType">
    <xs:sequence>
      <xs:element name="Input" type="cdm:FloatList"/>
      <xs:element name="Output" type="cdm:FloatList"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="HDRToneResponseCurvesType">
    <xs:sequence>
      <xs:element name="RedTRC" type="cdm:OneDimensionLutType"/>
      <xs:element name="GreenTRC" type="cdm:OneDimensionLutType"/>
      <xs:element name="BlueTRC" type="cdm:OneDimensionLutType"/>
    </xs:sequence>
    <xs:attribute name="TRCLength" use="required">
```

```
      <xs:simpleType>
        <xs:restriction base="xs:int">
          <xs:minInclusive value="0" />
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>

  <xs:complexType name="GamutBoundarySamplesType">
    <xs:sequence>
      <xs:element name="RGB" type="cdm:RGBType" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="RGBVirtualMeasurementType">
    <xs:sequence>
      <xs:element name="MaxColorantUsed" type="xs:float"/>
      <xs:element name="MinColorantUsed" type="xs:float"/>
      <xs:group ref="cdm:RGBPrimariesGroup"/>
      <xs:choice>
        <xs:element name="Gamma" type="cdm:GammaType"/>
        <xs:element name="GammaOffsetGain" type="cdm:GammaOffsetGainType"/>
        <xs:element name="GammaOffsetGainLinearGain"
type="cdm:GammaOffsetGainLinearGainType"/>
        <xs:element name="HDRToneResponseCurves"
type="cdm:HDRToneResponseCurvesType"/>
      </xs:choice>
      <xs:element name="GamutBoundarySamples"
type="cdm:GamutBoundarySamplesType" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="TimeStamp" type="xs:dateTime"/>
  </xs:complexType>

  <xs:element name="ColorDeviceModel">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ProfileName" type="wcs:MultiLocalizedTextType"/>
        <xs:element name="Description" type="wcs:MultiLocalizedTextType"
minOccurs="0"/>
        <xs:element name="Author" type="wcs:MultiLocalizedTextType"
minOccurs="0"/>
        <xs:element name="MeasurementConditions"
type="cdm:MeasurementConditionsType" minOccurs="0"/>
        <xs:element name="SelfLuminous" type="xs:boolean" />
        <xs:element name="MaxColorant" type="xs:float"/>
        <xs:element name="MinColorant" type="xs:float"/>
        <xs:choice>
          <xs:element name="CRTDevice">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="MeasurementData"
type="cdm:DisplayMeasurementType"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
```

```
          <xs:element name="LCDDevice">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="MeasurementData"
type="cdm:DisplayMeasurementType"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="RGBProjectorDevice">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="MeasurementData"
type="cdm:RGBProjectorMeasurementType"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="ScannerDevice">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="MeasurementData"
type="cdm:RGBCaptureMeasurementType"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="CameraDevice">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="MeasurementData"
type="cdm:RGBCaptureMeasurementType"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="RGBPrinterDevice">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="MeasurementData"
type="cdm:RGBPrinterMeasurementType"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="CMYKPrinterDevice">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="MeasurementData"
type="cdm:CMYKPrinterMeasurementType"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="RGBVirtualDevice">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="MeasurementData"
type="cdm:RGBVirtualMeasurementType"/>
              </xs:sequence>
            </xs:complexType>
```

```
          </xs:element>
        </xs:choice>
        <xs:element name="PlugInDevice" minOccurs="0">
          <xs:complexType>
            <xs:sequence>
              <xs:any namespace="##other" processContents="skip"
                minOccurs="0" maxOccurs="unbounded" />
            </xs:sequence>
            <xs:attribute name="GUID" type="wcs:GUIDType" use="required"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="ID" type="xs:string" use="optional" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Ajout d'étalonnage WCS CDMP v2.0

L'élément **ColorDeviceModel** du schéma CDMP a été mis à jour dans Windows 7 pour inclure le nouvel élément d'étalonnage. Voici la modification apportée au schéma CDMP.

C++

```
  ...
  <xs:element name="ColorDeviceModel">
    <xs:complexType>
      <xs:sequence>
        ...
        <xs:element name="PlugInDevice" minOccurs="0">
             ...
        </xs:element>
        <xs:element name="Calibration" type="cal:Calibration"
minOccurs="0"/>
        ...
      <xs:sequence>
    ...
    <xs:complexType>
  ...
```

Éléments de schéma CDMP

ⓘ Notes

Les primaires sont des échantillons principaux de rouge, vert, bleu, noir et blanc. Une rampe primaire est une rampe tonale allant de la plus petite luminance à la valeur primaire complète. Le nombre maximal d'entrées dans une rampe de tonalité est de 4 096.

ⓘ **Notes**

Les DPM doivent avoir des données de mesure.

ColorDeviceModelProfile

Cet élément est de type ColorDeviceModel.

Conditions de validation : Chaque sous-élément est validé par son propre type.

ColorDeviceModel

Cet élément est une séquence des sous-éléments suivants

1. Chaîne ProfileName,
2. chaîne de description facultative,
3. chaîne d'auteur facultative,
4. facultatif MeasurementConditions MeasurementConditionsType,
5. Self-Luminous Boolean,
6. MaxColorant float,
7. MinColorant float,
8. Choix des éléments
    a. CRTDevice,
    b. LCDDevice,
    c. RGBProjectorDevice,
    d. ScannerDevice,
    e. CameraDevice,
    f. RGBPrinterDevice,
    g. CMYKPrinterDevice,
    h. RGBVirtualDevice,
9. PlugInDevice,
10. ExtensionType facultative

Conditions de validation : Chaque sous-élément est validé par son propre type. Les sous-éléments de chaîne ont un maximum de 10 000 caractères. Le sous-élément

MaxColorant doit être supérieur ou égal à zéro (0) et supérieur au sous-élément MinColorant. MinColorant peut être négatif.

NamespaceVersion

xmlns:cdm= »http://schemas.microsoft.com/windows/2005/02/color/ColorDeviceModel
"

targetNamespace= »http://schemas.microsoft.com/windows/2005/02/color/ColorDevice
Model"

Version

Version = « 1.0 » avec Windows Vista.

Conditions de validation : Toute valeur >de version 0.1 ou <=2.0 est valide pour prendre en charge les modifications non cassants du format.

Documentation

Schéma du profil de modèle d'appareil.

Élément CRTDevice

Cet élément est une séquence de sous-éléments d'un MeasurementData DisplayMeasurementType.

Conditions de validation : Chaque sous-élément est validé par son propre type.

Élément LCDDevice

Cet élément est une séquence de sous-éléments d'un MeasurementData DisplayMeasurementType.

Conditions de validation : Chaque sous-élément est validé par son propre type.

Élément ProjecteurDevice

Cet élément est une séquence de sous-éléments d'un MeasurementData RGBProjectorMeasurementType.

Conditions de validation : Chaque sous-élément est validé par son propre type.

Élément ScannerDevice

Cet élément est une séquence de sous-éléments d'un MeasurementData RGBCaptureMeasurementType

Conditions de validation : Chaque sous-élément est validé par son propre type.

Élément CameraDevice

Cet élément est une séquence de sous-éléments d'un MeasurementData RGBCaptureMeasurementType

Conditions de validation : Chaque sous-élément est validé par son propre type.

Élément RGBPrinterDevice

Cet élément est une séquence de sous-éléments d'un MeasurementData RGBPrinterMeasurementType.

Conditions de validation : Chaque sous-élément est validé par son propre type.

Élément CMYKPrinterDevice

Cet élément est une séquence de sous-éléments d'un MeasurementData CMYKPrinterMeasurementType.

Conditions de validation : Chaque sous-élément est validé par son propre type.

Élément RGBVirtualDevice

Cet élément est une séquence de sous-éléments d'un RGBVirtualMeasurementDataType.

Conditions de validation : Chaque sous-élément est validé par son propre type.

PlugInDeviceType

Cet élément est une séquence d'un GUID GUIDType et de tous les sous-éléments.

Conditions de validation : Le GUID est utilisé pour faire correspondre le GUID de dll PlugIn DM. Il existe un maximum de 100 000 sous-éléments personnalisés.

RGBVirtualMeasurementType

Cet élément est une séquence composée de

1. RVBPrimariesGroup group
2. Un choix de
3. - Gamma
   - GammaOffsetGain
   - GammaOffsetGainLinearGam
   - Éléments ToneResponseCurves
4. facultatif GamutBoundarySamples GamutBoundarySamplesType
5. TimeStamp dateTime

Conditions de validation : Chaque sous-élément de ces types a ses propres conditions de validation.

GammaType

Cet élément est un type complexe constitué de l'attribut

- Gamma NonNegativeFloatType

Conditions de validation : À déterminer à partir des commentaires de l'industrie.

GammaOffsetGainType

Cet élément est un type complexe composé des attributs

- Gamma NonNegativeFloatType
- Offset NonNegativeFloatType
- Gain NonNegativeFloatType

Conditions de validation : À déterminer à partir des commentaires de l'industrie.

GammaOffsetGainLinearGainType

Cet élément est un type complexe composé des attributs

- Gamma NonNegativeFloatType
- Offset NonNegativeFloatType

- Gain NonNegativeFloatType
- LinearGain NonNegativeFloatType
- TransitionPoint NonNegativeFloatType.

Conditions de validation : À déterminer à partir des commentaires de l'industrie.

ToneResponseCurvesType

Cet élément est une séquence de

1. RedTRC FloatPairList
2. GreenTRC FloatPairList
3. BlueTRC FloatPairList

L'élément a également un attribut TRCLength de type unsignedint.

Conditions de validation : À déterminer à partir des commentaires de l'industrie.

GamutBoundarySamplesType

Cet élément est une séquence de RVBTypes.

Conditions de validation : Les occurrences maximales actuellement non limitées, à plafonner en fonction des commentaires de l'industrie.

FloatPairList

Cet élément est un type simple de liste de paires de floats.

Conditions de validation : À déterminer à partir des commentaires de l'industrie.

CMYKPrinterMeasurementType

Cet élément est un

1. séquence d'un élément ColorCube constitué d'une séquence de Sample NonNegativeCMYKSampleType

2. Attribut dateTime TimeStamp.

Conditions de validation : À déterminer à partir des commentaires du secteur.

RGBPrinterMeasurementType

Cet élément est un

1. séquence de l'élément ColorCube comprenant une séquence de Sample NonNegativeRGBSampleType
2. Attribut dateTime TimeStamp.

Conditions de validation : À déterminer à partir des commentaires du secteur.

RGBCaptureMeasurementType

Cet élément est une séquence de

1. ComplexType PrimaryIndex de
2. a. OneBasedIndex blanc
   b. Facultatif Black OneBasedIndex
   c. Red OneBasedIndex facultatif
   d. OneBasedIndex vert facultatif
   e. Facultatif Blue OneBasedIndex
   f. Facultatif Cyan OneBasedIndex
   g. Magenta OneBasedIndex facultatif
   h. OneBasedIndex jaune facultatif
3. NeutralIndices des lignes de OneBasedIndex
4. Séquence ColorSamples de l'exemple NonNegativeRGBSampleType

L'élément a également un attribut dateTime TimeStamp.

Conditions de validation : À déterminer à partir des commentaires du secteur.

OneBasedIndex

Cet élément est un type simple de restriction base unsigned int avec minInclusive value = « 1 ».

Conditions de validation : À déterminer à partir des commentaires du secteur.

RGBProjectorMeasurementType

Cet élément est une séquence de

1. Groupe RGBPrimariesGroup
2. élément ColorSamples constitué d'une séquence de NonNegativeRGBSampleType

L'élément a également un attribut dateTime TimeStamp.

Conditions de validation : À déterminer à partir des commentaires du secteur.

DisplayMeasurementType

Cet élément est une séquence de

1. groupe RGBPrimariesGroup
2. GrayRamp de la séquence de l'exemple NonNegativeRGBType
3. RedRamp de la séquence de l'exemple NonNegativeRGBType
4. GreenRamp de la séquence de l'exemple NonNegativeRGBType
5. BlueRamp de la séquence de l'exemple NonNegativeRGBType

L'élément DisplayMeasurementType a également un attribut dateTime TimeStamp.

Conditions de validation : À déterminer à partir des commentaires du secteur.

MeasurementConditionsType

MeasurementConditionsType est une séquence de sous-éléments qui contient :

1. Valeur d'énumération de chaîne restreinte ColorSpace de « CIEXYZ »
2. choix facultatif de l'énumération de chaîne WhitePoint NonNegativeXYZType ou WhitePointName des valeurs D50, D65, A ou F2
3. Geometry GeometryType
4. ApertureSize entier en millimètres

Les valeurs par défaut sont les suivantes :

1. Imprimantes RVB et CMJN :
    a. CIEXYZ MeasurementSpaceType
    b. D50 WhitePointValue
    c. 0/45 GeometryType
    d. ApertureSize 10mm
2. Scanners:
    a. CIEXYZ MeasurementSpaceType
    b. D50 WhitePointValue
    c. 0/45 GeometryType
    d. ApertureSize 10mm
3. Affichages et appareil virtuel RVB :
    a. CIEXYZ MeasurementSpaceType
    b. D65 WhitePointValue

c. 0/45 GeometryType
d. ApertureSize 10mm

4. Caméras:
   a. CIEXYZ MeasurementSpaceType
   b. D65 WhitePointValue
   c. Direct GeometryType
   d. ApertureSize 10mm

Conditions de validation : La validation de chaque sous-élément est déterminée par les conditions de validation de ces sous-éléments. Si un sous-élément est manquant, la valeur par défaut du type de modèle d'appareil est utilisée.

GeometryType

String

Valeurs d'énumération :

- "0/45"
- « 0/diffuse »
- « diffuse/0 »
- « Direct »

Conditions de validation : Toute valeur à l'exception des valeurs enumberation répertoriées n'est pas valide. Ces informations ne modifient pas le comportement de traitement de la base de référence.

RGBPrimariesGroup

Cet élément est une séquence de

1. WhitePrimary NonNegativeXYZType
2. RedPrimary NonNegativeXYZType
3. GreenPrimary NonNegativeXYZType
4. BluePrimary NonNegativeXYZTYpe
5. BlackPrimary NonNegativeXYZType

Conditions de validation : À déterminer à partir des commentaires du secteur.

NonNegativeCMYKSampleType

Cet élément est une séquence de

1. CMYK NonNegativeCMYKType
2. CIEXYZ NonNegativeXYZType

L'élément a également une chaîne de balise d'attribut facultative

Conditions de validation : À déterminer à partir des commentaires du secteur.

NonNegativeRGBSampleType

Cet élément est une séquence de

1. RGB NonNegativeRGBType
2. CIEXYZ NonNegativeXYZType

L'élément a également une chaîne de balise d'attribut facultative

Conditions de validation : À déterminer à partir des commentaires du secteur.

NonNegativeCMYKType

Cet élément constitué d'attributs

1. C float
2. M float
3. Y float
4. K float

Conditions de validation : À déterminer à partir des commentaires du secteur.

NonNegativeRGBType

Cet élément constitué d'attributs

1. R float
2. G float
3. B float

Conditions de validation : À déterminer à partir des commentaires du secteur.

ExtensionType

L'élément ExtensionType est une séquence de n'importe quel type de sous-élément et est utilisé pour les informations propriétaires provenant d'applications non-Microsoft.

Conditions de validation : Cet élément est facultatif. Il peut y avoir un maximum de 1 000 sous-éléments d'extension.

NonNegativeXYZType

L'élément NonNegativeXYZType est composé de trois éléments à virgule flottante IEEE à précision unique nommés « X », « Y » et « Z ». Ces valeurs sont limitées aux valeurs de mesure des profils DMP. Ces mesures peuvent être des valeurs réfléchissantes absolues (non relatives) CIEXYZ 1931 ou absolues (non relatives) des valeurs directes (transmissives) CIEXYZ 1931 en candelas par mètre d'unités au carré.

Conditions de validation : Seules les valeurs réelles sont valides et les valeurs de mesure CIEXYZ négatives ne sont pas valides. Comme il s'agit de valeurs absolues, les valeurs peuvent être supérieures à 1,0f. Une limite raisonnable pour toute valeur « X », « Y » ou « Z . » est arbitrairement définie sur 10000.0f.

XYZType

L'élément XYZType est composé de trois valeurs à virgule flottante IEEE simple précision : « X », « Y » et « Z ».

Algorithmes de base CDMP

Base de référence du modèle d'appareil CRT

Pour comprendre ce modèle, vous devez prendre en compte à la fois le processus de caractérisation et la modélisation de l'appareil. Dans le processus de caractérisation, les mesures XYZ sont d'abord effectuées sur les couleurs obtenues en remplissant la mémoire tampon d'affichage d'un dispositif d'affichage CRT. Les exemples de valeurs suivants génèrent des données correctes pour le modèle d'appareil CRT de base :

Rouge : R = 15, 30, 45, 60, 75, 90, 105, 120, 135, 150, 165, 180, 195, 210, 225, 240, 255, G = B = 0

Vert : G = 15, 30, 45, 60, 75, 90, 105, 120, 135, 150, 165, 180, 195, 210, 225, 240, 255, R = B = 0

Bleu : B = 15, 30, 45, 60, 75, 90, 105, 120, 135, 150, 165, 180, 195, 210, 225, 240, 255, R = G = 0

Neutres : R = G= B = 0, 8, 16, 32, 64, 128, 192, 255

Les incréments autres que 15 et les incréments non linéaires peuvent également être utilisés. Chaque rampe rouge, verte, bleue et neutre doit contenir au moins trois échantillons, mais il est recommandé de fournir d'autres échantillons. Vous devez fournir des échantillons pour le rouge pur, le vert, le bleu, le noir et le blanc. Il n'est pas nécessaire d'espacer uniformément les échantillons.

Le processus de construction de la matrice tristimulus se compose de deux étapes. Tout d'abord, estimez la valeur xyz du point noir, ou la torche. Cette étape est largement basée sur le travail de Berns[3] avec une fonction d'objectif légèrement modifiée pour l'optimisation non linéaire. Ensuite, calculez la matrice tristimulus en fonction du résultat de l'étape 1 et également d'un calcul de moyenne sur toutes les mesures par canal, et pas seulement sur celle du nombre numérique maximal.

Chacune de ces étapes contient des procédures détaillées. Le point de départ est les rampes (17 étapes dans notre exemple) pour chacun des canaux R, G et B. Lorsque les mesures XYZ sont tracées sur le plan de chromaticité *xy* , une situation typique est illustrée à la figure 1. L'étape 1 consiste à résoudre un problème d'optimisation non linéaire pour trouver le point noir « le mieux adapté » qui minimisera la dérive de la chromaticité lorsque l'on traverse les canaux R, G et B. Sur la base de Berns[3], nous recherchons ($X_{K,YK,ZK}$) qui réduit la fonction d'objectif suivante :

$$\Phi(X_K,Y_K,Z_K)=\Psi(X_K,Y_K,Z_K;S_R)+\Psi(X_K,Y_K,Z_K;S_G)+\Psi(X_K,Y_K,Z_K;S_B),$$

où S_R, S_G et S_B sont l'ensemble de points de données correspondant aux points sur les canaux R, G et B. Pour tout ensemble *S*, définissez :

$$\Psi(X_K,Y_K,Z_K;S)=\sum_{i\in S}\left\|f(X_i,Y_i,Z_i;X_K,Y_K,Z_K)-\bar{f}(X_K,Y_K,Z_K;S)\right\|^2,$$

$$f(X,Y,Z;X_K,Y_K,Z_K)=\left(\frac{X-X_K}{X-X_K+Y-Y_K+Z-Z_K}\ ,\ \frac{Y-Y_K}{X-X_K+Y-Y_K+Z-Z_K}\right),$$

$$\bar{f}(X_K,Y_K,Z_K;S)=\frac{1}{|S|}\sum_{i\in S}f(X_i,Y_i,Z_i;X_K,Y_K,Z_K).$$

Dans le précédent, | *S* | est la cardinalité de *S*, c'est-à-dire le nombre de points dans l'ensemble *S*. $f(X,Y,Z;X_K,Y_K,Z_K)$ est les coordonnées de chromaticité du point $(X-X_K,Y-Y_K,Z-Z_K)$, $\bar{f}(X_K,Y_K,Z_K;S)$, est la moyenne, ou centre de masse, de tous les points de l'ensemble *S* dans le plan de chromaticité. Ainsi, $\Psi(X_K,Y_K,Z_K;S)$ est la somme des deuxièmes moments des points sur le centre de la masse et est une mesure de la façon dont les points sont répartis à ce sujet. Enfin, $\Phi(X_K,Y_K,Z_K)$ Est une mesure

totale de la façon dont les trois clusters de points sont répartis sur leurs centres de masse respectifs.

Dans le calcul de $f(X,Y,Z;X_K,Y_K,Z_K)$, si $X=X_K,Y=Y_K,Z=Z_K$, le calcul est ignoré et la cardinalité de *S* est ajustée en conséquence.

Malgré la complexité apparente de la fonction objective, il s'agit d'une somme des carrés de nombreuses fonctions différentiables dans $X_{K,YK\ ZK}$ (17 points 2*xy* -composants 3 canaux = 102, dans l'exemple) et, par conséquent, est accessible aux techniques de moindres carrés non linéaires standard, telles que l'algorithme Levenberg-Marquardt, qui est l'algorithme utilisé dans WCS. Notez que la fonction d'objectif précédente est différente de celle suggérée à Berns[3] en ce que cette dernière fonction mesure la variance des distances par rapport au centre de la masse, de sorte que la variance est égale à zéro lorsque les points sont équidistants par rapport au centre de la masse, même s'ils peuvent s'étendre un peu à ce sujet. Dans l'exemple, la dispersion des points est directement liée à l'aide des deuxièmes moments.

Comme pour tout algorithme itératif pour le problème des moindres carrés non linéaires, Levenberg-Marquardt nécessite une estimation initiale. Il y a deux candidats évidents. L'un est (0, 0, 0); l'autre est le point noir mesuré. Pour l'objet CTE, le point noir mesuré est d'abord utilisé comme estimation initiale. Si un maximum de 100 itérations est dépassé sans atteindre le seuil d'une distance moyenne de 0,001 de chaque point à partir de son centre de masse (ce qui correspond à une valeur de seuil de (0,001) 17 3 = 0,000051 pour la fonction objective), une autre série d'itérations avec la estimation initiale de (0, 0, 0) est effectuée. L'estimation résultante du point noir est XYZ comparée à la meilleure estimation de la série précédente d'itérations (avec le point noir mesuré comme estimation initiale). Utilisez l'estimation qui donne la plus petite valeur pour la fonction objective. Le choix de 100 itérations et la distance d'erreur de 0,001 ont été sélectionnés de manière empirique. Dans les versions ultérieures, il peut être raisonnable de paramétrer la distance d'erreur.

Le résultat de l'étape 1 est le point noir estimé ($X_{K,YK,ZK}$). L'étape 2 consiste à déterminer la matrice tristimulus en faisant la moyenne de la chromaticité des points dans les trois clusters obtenus à l'étape 1. Pour les CRT, cette opération est effectuée principalement pour réduire les effets des erreurs de mesure. Les points utilisés pour la moyenne de la chromaticité doivent être les mêmes points que ceux utilisés dans l'optimisation à l'étape 1. En d'autres termes, si le premier point (nombre numérique 15, dans l'exemple) dans chaque rampe est ignoré dans l'étape d'optimisation, la même chose doit être effectuée dans la moyenne. Si $(\bar{x}_r,\bar{y}_r)$, $(\bar{x}_g,\bar{y}_g)$, et $(\bar{x}_b,\bar{y}_b)$ Sont les coordonnées de chromaticité moyenne des canaux rouge, vert et bleu, la procédure suivante détermine la matrice tristimulus. Tout d'abord, résolvez le système linéaire 3?3 :

$$\begin{pmatrix} \bar{x}_r & \bar{x}_g & \bar{x}_b \\ \bar{y}_r & \bar{y}_g & \bar{y}_b \\ 1-\bar{x}_r-\bar{y}_r & 1-\bar{x}_g-\bar{y}_g & 1-\bar{x}_b-\bar{y}_b \end{pmatrix} \begin{pmatrix} t_r \\ t_g \\ t_b \end{pmatrix} = \begin{pmatrix} X_W \\ Y_W \\ Z_W \end{pmatrix},$$

t_r, t_g t_b

$X_{W,YW,ZW}$

$$\begin{pmatrix} t_r\bar{x}_r & t_g\bar{x}_g & t_b\bar{x}_b \\ t_r\bar{y}_r & t_g\bar{y}_g & t_b\bar{y}_b \\ t_r(1-\bar{x}_r-\bar{y}_r) & t_g(1-\bar{x}_g-\bar{y}_g) & t_b(1-\bar{x}_b-\bar{y}_b) \end{pmatrix}.$$

Une fois la matrice tristimulus déterminée, la détermination des courbes de tonalité suit l'approche standard. Pour les affichages CRT, les canaux individuels sont supposés suivre le modèle « GOG » :

$$f(x) = \left((1-k_g)+k_g x\right)^{\gamma},$$

où k_g est le « gain », 1 -k_g est le « décalage », et ? est le « gamma ». La matrice inverse de la matrice tristimulus est appliquée aux données XYZ des neutres pour obtenir les données RVB linéaires, qui sont ensuite corrélées avec les valeurs RVB numériques à l'aide de la régression non linéaire sur le modèle GOG. Ces caractéristiques ne doivent pas être les mêmes pour les canaux R, G et B, et ne sont généralement pas les mêmes.

Berns[1] : Berns, *Billmeyer and Saltzman's Principles of Color Technology*, 3 rd Ed. John Wiley & Sons (2000).

Berns[2] : Berns et Katoh, The digital to radiometric transfer function for computer controlled CRT displays, *CIE Expert Symposium '97 Colour Standards for Imaging Technology*, novembre 1997.

Berns[3]: Berns, Fernandez et Taplin, Estimateing Black-Level Emissions of Computer-Controlled Displays, *Color Research and Application*, 28: 379-383 Wiley Periodicals, Inc. (2003)

Kang[1]: Kang, *Color Technology for Electronic Imaging Devices*, SPIE (1997)

Katoh[1]: Katoh, Deguchi et Berns, Une caractérisation précise de la proposition de moniteur CRT (II) pour une extension de la méthode CIE et sa vérification, *Opt. Rev.* 8: 397-408 (2001)

Base de référence du modèle d'appareil LCD

La base de référence du modèle d'appareil LCD est similaire à la base de référence du modèle d'appareil CRT. Cette section explique en quoi la modélisation LCD diffère de la modélisation CRT.

Une différence réside dans le fait que vous ne pouvez pas supposer que les affichages LCD suivent le modèle GOG utilisé pour les CRT, et que les courbes de tonalité sont obtenues par interpolation de données mesurées. Pour cette raison, l'axe neutre de l'appareil doit être échantillonné plus fréquemment.

Voici quelques exemples de valeurs typiques qui peuvent générer de bonnes données pour la base de référence du modèle d'appareil LCD :

Rouge : R = 15, 30, 45, 60, 75, 90, 105, 120, 135, 150, 165, 180, 195, 210, 225, 240, 255, G = B = 0

Vert : G = 15, 30, 45, 60, 75, 90, 105, 120, 135, 150, 165, 180, 195, 210, 225, 240, 255, R = B = 0

Bleu : B = 15, 30, 45, 60, 75, 90, 105, 120, 135, 150, 165, 180, 195, 210, 225, 240, 255, R = G = 0

Neutres : R = G = B = 0, 15, 30, 45, 60, 75, 90, 105, 120, 135, 150, 165, 180, 195, 210, 225, 240, 255.

Le processus de moyenne des chromatiques de couleur mesurées pour obtenir les chromaticités des appareils primaires est plus critique pour les LCDs que pour les CRT. Lorsque les mesures XYZ sont tracées sur le plan de chromaticité *xy* , une situation typique est illustrée dans la figure 1. Notez comment la chromaticité dérive vers le point noir. Cela est dû au fait que tous les LCD ont une certaine quantité de fuite de lumière.

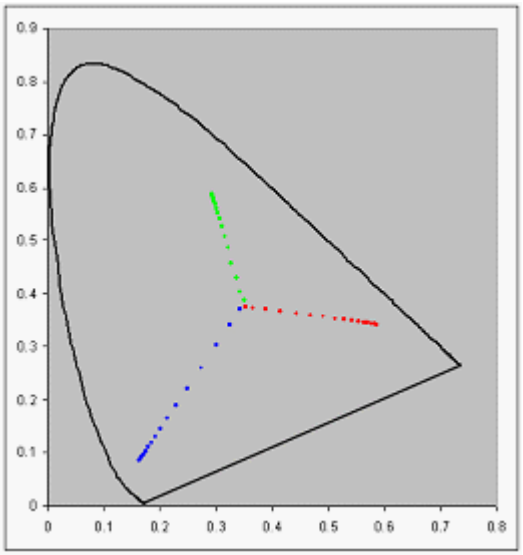


Figure 1 : Diagramme de chromaticité utilisant des données brutes sans correction

Lorsque cette valeur est soustraite des mesures XYZ brutes, une situation typique est représentée à la figure 2. Les points sont maintenant regroupés autour de trois centres, bien qu'ils ne tombent pas de la même manière sur eux. Le processus de moyenne décrit pour les CRT améliore considérablement les résultats pour les LCD.

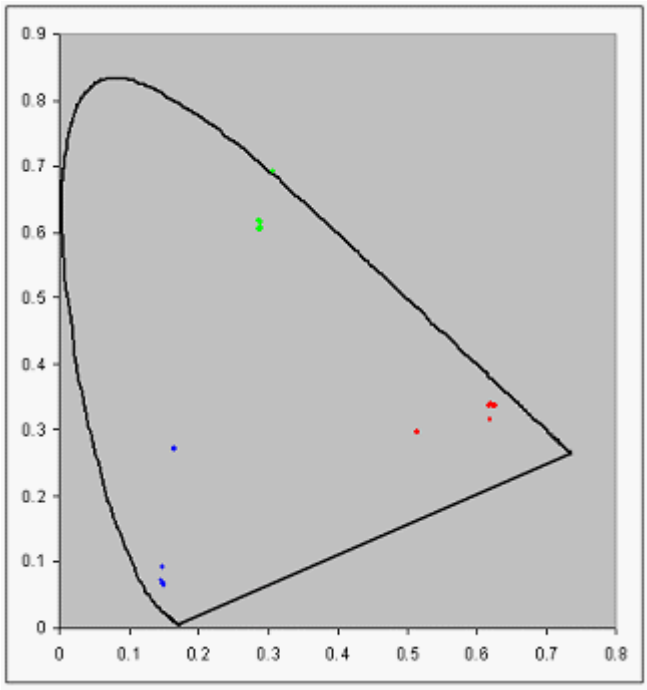


Figure 2 : Diagramme de chromaticité utilisant des données avec un point noir ajusté

Base de référence du modèle d'appareil de capture RVB

Le modèle d'appareil de capture RVB de base est une sous-classe de la classe IDeviceModel. Dans la caractérisation colorimétrique des appareils de capture de couleurs, tels que les scanneurs et les appareils photo numériques, l'approche suivante est utilisée. Une cible composée de correctifs de couleur avec des valeurs CIEXYZ connues est capturée à l'aide de l'appareil de capture. Le résultat de la capture est une image bitmap RVB dans laquelle la couleur de chaque correctif est encodée dans une valeur RVB. Ces valeurs RVB d'appareil sont spécifiques à un appareil de capture particulier. L'objectif de la caractérisation colorimétrique est d'établir une relation empirique entre les valeurs RVB de l'appareil et les valeurs CIEXYZ, ou une transformation mathématique de RVB en XYZ qui modélise aussi précisément que possible le comportement de l'appareil de capture.

Une telle transformation mathématique peut être modélisée raisonnablement par des polynômes de faible degré. Cette procédure est détaillée dans la littérature, par exemple Kang[92], Kang[97]. Dans Kang[97], une approche est signalée qui utilise un ensemble de trois polynomials avec 3, 6, 8, 9, 11, 14 ou 20 termes dans les variables R, G et B, tandis que les trois polynomials régressent respectivement dans les composants X, Y et Z de l'espace CIEXYZ. Pour le polynomial à 20 termes, la forme est la suivante :

$$X = a_1 + a_2R + a_3G + a_4B + a_5R^2 + a_6RG + a_7RB + a_8G^2 + a_9GB + a_{10}B^2$$
$$+ a_{11}R^3 + a_{12}R^2G + a_{13}R^2B + a_{14}RG^2 + a_{15}RGB + a_{16}RB^2 + a_{17}G^3 + a_{18}G^2B + a_{19}GB^2 + a_{20}B^3$$

Il existe des expressions similaires pour Y et Z. La technique mathématique permettant d'ajuster les polynômes s'inscrit dans la « régression linéaire multivariée » et est décrite dans n'importe quel texte élémentaire dans Statistiques.

Cette méthode de régression linéaire souffre de ne pas réduire la fonction objective « droite ». Par conception, la régression linéaire trouve la solution des moindres carrés, ce qui implique que les coefficients obtenus réduisent la somme totale des carrés d'erreurs dans l'espace sous-jacent, ou, équivalent, la somme des carrés des distances euclidiennes. Dans la pratique, vous souhaitez réduire la somme des carrés de ? Es, où ? E est l'une des normes acceptées telles que CIE94. La réduction de cette fonction objective est un problème de régression non linéaire.

Dans le nouveau moteur, Lab to XYZ est l'espace de couleur CIE qui est régressé dans, et le polynomial cubique à 20 termes est utilisé comme modèle pour l'appareil de capture, ou coefficients ls,as,bs de sorte que les polynômes suivants réduisent la somme des carrés de ? E_{CIE94} s.

$$\begin{aligned}
\hat{L}(R,G,B) &= \lambda_1 + \lambda_2R + \lambda_3G + \lambda_4B \\
&+ \lambda_5R^2 + \lambda_6RG + \lambda_7RB + \lambda_8G^2 + \lambda_9GB + \lambda_{10}B^2 \\
&+ \lambda_{11}R^3 + \lambda_{12}R^2G + \lambda_{13}R^2B + \lambda_{14}RG^2 + \lambda_{15}RGB \\
&+ \lambda_{16}RB^2 + \lambda_{17}G^3 + \lambda_{18}G^2B + \lambda_{19}GB^2 + \lambda_{20}B^3 \\
\hat{u}(R,G,B) &= \alpha_1 + \alpha_2R + \alpha_3G + \alpha_4B \\
&+ \alpha_5R^2 + \alpha_6RG + \alpha_7RB + \alpha_8G^2 + \alpha_9GB + \alpha_{10}B^2 \\
&+ \alpha_{11}R^3 + \alpha_{12}R^2G + \alpha_{13}R^2B + \alpha_{14}RG^2 + \alpha_{15}RGB \\
&+ \alpha_{16}RB^2 + \alpha_{17}G^3 + \alpha_{18}G^2B + \alpha_{19}GB^2 + \alpha_{20}B^3 \\
\hat{v}(R,G,B) &= \beta_1 + \beta_2R + \beta_3G + \beta_4B \\
&+ \beta_5R^2 + \beta_6RG + \beta_7RB + \beta_8G^2 + \beta_9GB + \beta_{10}B^2 \\
&+ \beta_{11}R^3 + \beta_{12}R^2G + \beta_{13}R^2B + \beta_{14}RG^2 + \beta_{15}RGB \\
&+ \beta_{16}RB^2 + \beta_{17}G^3 + \beta_{18}G^2B + \beta_{19}GB^2 + \beta_{20}B^3
\end{aligned}$$

La solution (l_i, a_i, b_i) dans l'espace numérique réel À 60 dimensions **R**203 doit être telle que l'erreur totale suivante soit réduite :

$$\Psi(\lambda_i,\alpha_i,\beta_i) = \sum_i \Delta E^2_{CIE94}(\hat{L}(R_i,G_i,B_i),\hat{u}(R_i,G_i,B_i),\hat{v}(R_i,G_i,B_i);L_i,u_i,v_i)$$

où la somme est effectuée via toutes les paires de points de données ($R_{i,Gi,Bi;}$ $L_{i,ui,vi}$) dans le jeu de données échantillonné plus des points de contrôle supplémentaires à détailler dans ce qui suit. Il s'agit d'un problème de régression non linéaire, car les paramètres *?* $_i$, a_i, * $_i$* entrent dans la fonction objective d'une manière non linéaire (non quadratique).

Parce que la fonction objective ? est une fonction non linéaire (et non quadradratique) des paramètres *?* ¡, *a*ᵢ et * ¡*, vous devez recourir à des techniques itératives pour résoudre le problème d'optimisation. Étant donné que la forme de la fonction objective est une somme de carrés, une technique d'optimisation standard appelée algorithme Levenberg-Marquardt est utilisée. Il est considéré comme la méthode de choix pour les problèmes de moindres carrés non linéaires. Pour les algorithmes itératifs tels que Levenberg-Marquardt, vous devez fournir une estimation initiale. Une bonne estimation initiale est généralement essentielle pour trouver la valeur minimale correcte. Dans ce cas, un bon candidat pour la supposition initiale est la solution du problème de régression linéaire. Tout d'abord, réduisez la somme du carré des distances euclidiennes dans l'espace lab, en définissant une fonction d'objectif quadratique :

$$\Theta(\lambda_i, \alpha_i, \beta_i) = \sum_i dist^2_{CIELUV}(\hat{L}(R_i, G_i, B_i), \hat{u}(R_i, G_i, B_i), \hat{v}(R_i, G_i, B_i); L_i, u_i, v_i)$$

$$dist^2_{CIELUV}(L, u, v, L_0, u_0, v_0) = (L - L_0)^2 + (u - u_0)^2 + (v - v_0)^2$$

La solution mathématique à ce problème de « moindres carrés linéaires » est bien connue. Parce que *?* ¡ apparaît uniquement dans la modélisation *L* , *un*ᵢ n'apparaît que dans la modélisation *u* et * ¡* n'apparaît que dans la modélisation *v* ; le problème d'optimisation peut être décomposé en trois sous-problèmes : un pour *L*, un pour *u* et un pour *v*. Considérez les équations *L* . (Les équations *uet v suivent* exactement le même argument.) Le problème de la minimisation de la somme des carrés d'erreurs en *L* peut être considéré comme la résolution de l'équation matricielle suivante dans le sens des moindres carrés :

$$\begin{pmatrix} 1 & R_1 & G_1 & B_1 & \cdots & G_1B_1^2 & B_1^3 \\ 1 & R_2 & G_2 & B_2 & \cdots & G_2B_2^2 & B_2^3 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & R_N & G_N & B_N & \cdots & G_NB_N^2 & B_N^3 \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_{20} \end{pmatrix} = \begin{pmatrix} L_1 \\ L_2 \\ \vdots \\ L_N \end{pmatrix} \text{ or } \mathbf{R\Lambda = L}$$

où *N* correspond au nombre total de points de données (points échantillonnées d'origine plus points de contrôle créés de la manière décrite ci-dessous). En règle générale, *N* étant beaucoup plus grand que 20, l'équation précédente est sur-déterminée, ce qui nécessite une solution de moindres carrés. Une solution de formulaire fermé pour **?** est disponible :

$$\mathbf{\Lambda = (R^T R)^{-1}(R^T L)}$$

Dans la pratique, l'évaluation directe à l'aide de la solution de forme fermée n'est pas utilisée, car elle présente de mauvaises propriétés numériques. Au lieu de cela, une sorte d'algorithme de factorisation matricielle est appliquée à la matrice de coefficients qui réduit le système d'équations à une forme canonique. Dans l'implémentation actuelle, la

décomposition de valeur unique (SVD) est appliquée à la matrice **R** , puis le système décomposé résultant est résolu.

La solution au problème de régression linéaire, désignée par $\tilde{\lambda}_i,\ \tilde{\alpha}_i,\ \tilde{\beta}_i,$, est utilisée comme point de départ de l'algorithme de Levenberg-Marquardt. Dans cet algorithme, une étape d'essai est calculée qui doit rapprocher le point de la solution optimale. L'étape d'essai répond à un ensemble d'équations linéaires dépendant de la valeur fonctionnelle et des valeurs des dérivés au point actuel. Pour cette raison, les dérivés de la fonction objective ? en ce qui concerne les paramètres $?_i$, i_i sont obligatoires entrées dans l'algorithme Levenberg-Marquardt. Bien qu'il existe 60 paramètres, il existe un raccourci qui vous permet de calculer beaucoup moins. Par la règle de chaîne du calcul,

$$\frac{\partial \Psi}{\partial \lambda_j} = \sum_{i=1}^{N} \left.\frac{\partial \Delta E^2_{CIE94}(L, u, v, L_i, u_i, v_i)}{\partial L}\right|_{\hat{L},\hat{u},\hat{v}} R_{ij}$$

$$\frac{\partial \Psi}{\partial \alpha_j} = \sum_{i=1}^{N} \left.\frac{\partial \Delta E^2_{CIE94}(L, u, v, L_i, u_i, v_i)}{\partial u}\right|_{\hat{L},\hat{u},\hat{v}} R_{ij}$$

$$\frac{\partial \Psi}{\partial \beta_j} = \sum_{i=1}^{N} \left.\frac{\partial \Delta E^2_{CIE94}(L, u, v, L_i, u_i, v_i)}{\partial v}\right|_{\hat{L},\hat{u},\hat{v}} R_{ij}$$

où j = 1, 2, , 20, $L_{i,ui,vi}$ sont la valeur CIELAB du i ième point d'exemple, et R_{ij} est l'entrée (i,j)th de la matrice **R** définie ci-dessus. Ainsi, au lieu de calculer des dérivés pour 60 paramètres, vous pouvez calculer des dérivés pour *L*, *a* et *b* à l'aide de la différenciation numérique vers l'avant.

Il est également nécessaire de configurer un critère d'arrêt pour les algorithmes itératifs. Dans l'implémentation actuelle, les itérations sont terminées si le carré MOYEN DECIE94 est inférieur à 1 ou si le nombre d'itérations effectuées a dépassé 10. Le nombre 10 provient de l'expérience pratique selon laquelle si les premières itérations ne réduisent pas considérablement l'erreur, d'autres itérations n'aideraient pas grand-chose autre que le déplacement du point d'une manière oscillante, c'est-à-dire que l'algorithme risque de ne pas converger. Même dans le cas où l'algorithme diverge, nous pouvons être sûrs que le DECIE94 n'est pas pire que ce que nous avons commencé, c'est-à-dire avec les paramètres obtenus à partir de la régression linéaire.

Même avec la méthode précédente de régression non linéaire, il existe plusieurs problèmes avec l'ajustement. Un problème est que les polynômes ont tendance à dépasser ou à sous-dépasser les points de données. Des maxima et des minima locaux artificiels peuvent être introduits dans le processus d'ajustement. Cela peut être contrecarré en utilisant des polynômes de faible degré, ce qui est la raison pour laquelle vous ne devez pas utiliser plus de trois degrés. Un aspect plus sérieux du dépassement ou du dépassement est que, si un polynomial peut prendre théoriquement n'importe

quelle valeur réelle, l'espace qu'il tente de modéliser comporte généralement des contraintes physiques et des contraintes pratiques. CIEXYZ doit avoir tous les éléments X, Y, Z non négatifs, ce qui est une contrainte physique. Dans la pratique, ils ne prennent que des valeurs de la magnitude de centaines, et non des milliers ou plus. De même, CIELAB ou CIELUV a ses propres contraintes physiques et pratiques. Lorsque l'espace RVB est suffisamment rempli de points d'échantillon, le problème de dépassement ou de sous-ramification n'est pas sérieux. Toutefois, les points RVB capturés de l'appareil de capture ne remplissent généralement pas uniformément l'espace RVB. Les points peuvent se remplir uniquement à l'intérieur des 80 % du cube RVB, ou pire, ils peuvent résider dans un collecteur de dimensions inférieures. Dans ce cas, les polynômes régressés peuvent faire un mauvais travail pour extrapoler les valeurs au-delà des points de données, renvoyant parfois des prédictions irréalistes. Vous souhaitez un modèle qui retourne toujours des valeurs raisonnablement réalistes. Cela nécessite une méthode qui peut contrôler efficacement le comportement des limites des polynômes de régression en imposant des coûts supplémentaires aux polynômes qui se comportent de manière erratique près de la limite du cube RVB. Une autre mesure pour s'assurer que les polynômes retournent toujours des valeurs réalistes est appliquée en détournant la sortie à l'intérieur du locus spectral CIE.

C'est à ce stade que la modélisation du scanneur et la modélisation de caméra diffèrent les unes des autres. Le modèle de caméra est censé extrapoler vers des régions au-delà des points échantillonné, y compris les « surbrillances spéculaires », la même extrapolation n'est pas requise pour le modèle scanneur. On s'attend à ce que la cible du scanneur couvre une caractérisation similaire aux matériaux imprimés à analyser. Le modèle de scanneur doit toujours être robuste dans le sens où il ne doit pas retourner des valeurs irréalistes, mais l'extrapolation bien au-delà de la cible de caractérisation n'est pas nécessaire. Pour garantir la robustesse, le polynôme L-ci-dessus est coupé à 100, c'est-à-dire que le modèle polynomial est forcé de ne pas extrapoler au-delà de « Dmin » de la cible du scanneur.

Le modèle de caméra étant censé extrapoler en surbrillances spéculaires, le découpage à 100 n'est pas souhaitable. Au lieu de cela, l'algorithme suivant est utilisé.

Des points de contrôle artificiels sont introduits pour contrôler le comportement des polynômes dans les régions où l'échantillonnage est insuffisant. En plaçant stratégiquement ces points de contrôle avec les valeurs appropriées, ils servent à « tirer » les polynômes dans la direction requise. Dans l'implémentation actuelle, huit points de contrôle correspondant aux coins du cube RVB sont utilisés. Si les valeurs de l'appareil sont normalisées en unity, ces points sont les suivants :

$R = 0, G = 0, B = 0$

R = 0, *G* = 0, *B* = 1

R = 0, *G* = 1, *B* = 0

R = 0. *G* = 1, *B* = 1

R = 1, *G* = 0, *B* = 0

R = 1, *G* = 0, *B* = 1

R = 1, *G* = 1, *B* = 0

R = 1, *G* = 1, *B* = 1

Sauf pour le *R* =*G B* = blanc = 1, qui est associé à une valeur CIELAB de *L* = 100, *u* =*v* = 0, l'algorithme d'extrapolation suivant est utilisé pour déterminer la valeur CIELAB appropriée à associer. En règle générale, pour un donné (*R,G,B*), un poids est associé à chacun des ($R_{i,Gi,Bi}$) dans le jeu de données échantillonné. Il y a deux objectifs à attribuer le poids. Tout d'abord, le poids est inversement proportionnel à la distance entre (*R,G,B*) et ($R_{i,Gi,Bi}$). Deuxièmement, vous souhaitez ignorer ou affecter un poids 0 à des points dont la teinte est différente de celle du point donné (*R,G,B*). Pour prendre en compte la teinte, considérez les points qui se trouvent à l'intérieur d'un cône dont le sommet est à (0, 0, 0), dont l'axe coïncide avec la ligne joignant (0, 0, 0) à (*R,G,B*), et dont l'angle semi-vertical ? satisfait cos ? = 0,9. Pour obtenir une illustration de ce cône, voir la figure 3.

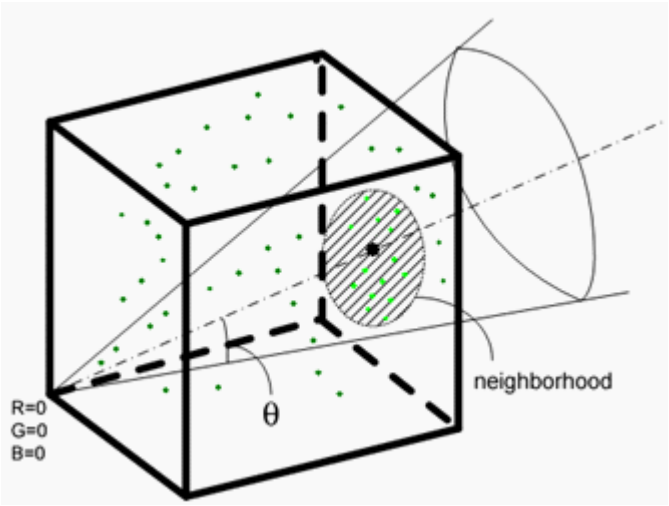


Figure 3 : Filtrage des points d'échantillon par angle et distance. La forme du quartier représenté est uniquement à des fins d'illustration. La forme réelle dépend de la distance utilisée ; il s'agit d'un quartier en forme de diamant si la norme 1 est utilisée.

Dans ce cône, un deuxième filtrage est effectué en fonction de la distance RVB, qui utilise la norme 1, définie par

$$dist(R,G,B;R_i,G_i,B_i) = |R-R_i| + |G-G_i| + |B-B_i|$$

Avec le cône actuel, la recherche initiale concerne les points qui se trouvent à une distance de 0,1 de (R,G,B). Si aucun point n'est trouvé dans ce rayon, le rayon est augmenté de 0,1 et la recherche est redémarrée. Si la ronde suivante n'a aucun point non plus, le rayon est augmenté de 0,1. Ce processus se poursuit jusqu'à ce que le rayon dépasse MaxDist/5, où MaxDist = 3, dans le cas d'une norme 1. Si aucun point n'est trouvé, le cône est agrandi en diminuant le cos ? de 0,05, c'est-à-dire augmenter l'angle ? et le redémarrage de l'ensemble du processus avec un rayon croissant. Ce processus se poursuit jusqu'à ce qu'un ensemble de points non vide soit trouvé, ou cos ? atteint 0, c'est-à-dire que le cône s'est ouvert pour devenir un plan. À ce stade, la recherche est redémarrée en augmentant le rayon, sauf que la recherche continue jusqu'à ce que le rayon atteigne MaxDist. Cela garantit que dans le pire des scénarios, un ensemble de points non vides sera trouvé. L'algorithme est résumé dans le diagramme de flux de la figure 4.

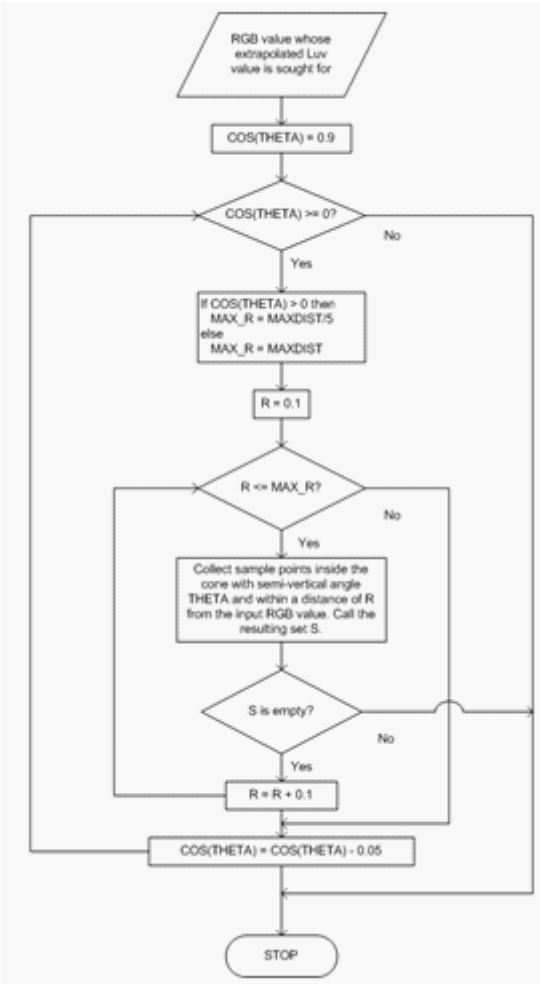


Figure 4 : Diagramme de flux pour déterminer l'ensemble S de points d'exemple utilisés dans l'extrapolation d'une valeur RVB d'entrée

En supposant que le processus précédent génère un ensemble non vide S de points ($R_{i,Gi,Bi}$) et correspondant ($L_{i,ai,bi\ i}$), alors pour chaque point de ce type, un poids w_i est attribué, donné par

$$h_i = \left(\frac{\max(\text{MaxDist} - d_i, 0)}{d_i} \right)^x \text{ where } d_i = |R - R_i| + |G - G_i| + |B - B_i|$$

$$w_i = h_i \Big/ \sum_{j \in S} h_j$$

Enfin, l'extrapolant est défini par

$$L = \sum_{i \in S} w_i L_i$$

$$u = \sum_{i \in S} w_i u_i$$

$$v = \sum_{i \in S} w_i v_i$$

Les équations précédentes constituent une instance des « méthodes pondérées à distance inverse », communément appelées méthodes Shepard. En exécutant chacun des huit points d'eq (6) à l'algorithme, huit points de contrôle sont obtenus, chacun avec les valeurs *R,G,B* et L,a,b, qui sont placés dans le pool avec les exemples de données d'origine.

Pour vous assurer que le modèle produit toujours des valeurs de couleur valides et pour assurer l'intégrité et la stabilité du système dans l'ensemble du pipeline de traitement des couleurs, vous devez effectuer un découpage final vers la sortie du modèle polynomial. La gamut visuelle CIE est décrite par le composant achromatique (*Y* ou *L*) et le composant chromatique (*xy* ou *a'b*', qui sont liés à l'espace XYZ par une transformation projective). Dans l'implémentation actuelle, la chromaticité *a'b*' est utilisée, car elle est directement liée à l'espace CIELUV. Pour toute valeur *CIELAB* , commencez par couper *L* sur une valeur non négative :

$$L \leftarrow \max(0, L)$$

Pour permettre l'extrapolation pour les surbrillances spéculaires, *L* n'est pas clippé à 100, la limite supérieure « conventionnelle » pour *L* dans l'espace lab.

Ensuite, si *L* = 0, *a* et *b* sont coupés de telle sorte que a*= b =* 0. Si *L* ? 0, calculer

$$u'_{rel} = u / 13L$$

$$v'_{rel} = v / 13L$$

Il s'agit des composants *d'un* vecteur dans le diagramme *a'b*' du point blanc (*u?*',v?') à la couleur en question. Définissez le locus spectral CIE comme la coque convexe de tous les points (*a*',*b*'), paramétrés par la longueur d'onde ?:

$$u'(\lambda) = \frac{4\bar{x}(\lambda)}{\bar{x}(\lambda) + 15\bar{y}(\lambda) + 3\bar{z}(\lambda)}$$
$$v'(\lambda) = \frac{9\bar{y}(\lambda)}{\bar{x}(\lambda) + 15\bar{y}(\lambda) + 3\bar{z}(\lambda)}$$

où $\bar{x}(\lambda)$, $\bar{y}(\lambda)$, $\bar{z}(\lambda)$ Sont les fonctions de correspondance des couleurs CIE pour l'observateur à 2 degrés. Si le vecteur se trouve en dehors du locus CIE, la couleur est coupée au point sur le locus CIE qui est l'intersection du locus et de la ligne définie par le vecteur. Voir figure 5. Si le découpage *s'est* produit, la valeur *a* et *b* est reconstruite en soustrayant *d'abord a*' et *b?*' du clippé *a*' et b', puis en multipliant par 13 *L*.

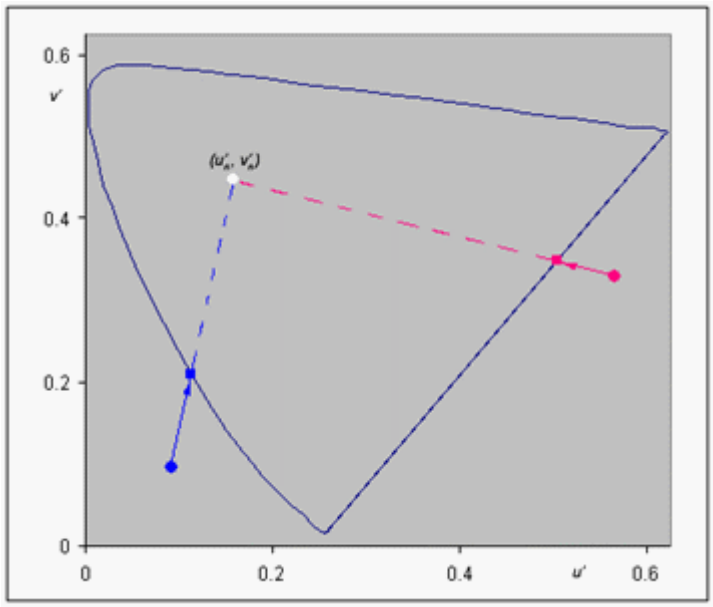


Figure 5 : algorithme de découpage pour les valeurs Lab qui se trouvent en dehors de la gamut visuelle CIE

Dans l'implémentation actuelle, le locus spectral CIE dans le plan *a'b*' est représenté par une courbe linéaire au niveau de la pièce avec 35 segments (correspondant à une longueur d'onde comprise entre 360 nm et 700 nm inclus). En triant les segments de ligne afin que leurs angles sous-attachés au point blanc soient croissants, ce qui équivaut à des longueurs d'onde décroissantes, le segment de ligne qui croise le rayon formé par le vecteur ci-dessus peut être trouvé par une simple recherche binaire.

Base de référence du modèle d'appareil d'imprimante RVB

Une caractérisation d'appareil d'une imprimante RVB consiste à construire un modèle empirique de l'appareil qui prédit la couleur CIELUV indépendante de l'appareil pour une valeur RVB donnée

Il existe deux façons de construire le modèle empirique. L'une consiste à utiliser les données d'appareil pour une imprimante RVB, et l'autre consiste à utiliser des données de paramètres analytiques. Dans la première, les données de mesure fournies par un utilisateur pour un périphérique d'imprimante RVB sont utilisées pour construire une table de recherche 3D (LUT). Les données de mesure se composent de valeurs XYZ pour les correctifs RVB échantillonnées uniformément. Les tailles d'échantillonnage standard sont de 9 ou 17 pour chaque composant. Chaque patch est mesuré avec un colorimètre ou un spectrophotomètre dans l'espace CIEXYZ. La valeur XYZ d'un correctif est ensuite convertie en valeur CIELUV, formant un LUT 3D. Dans le modèle d'appareil, la méthode d'interpolation tétraédrique de Sakamoto est appliquée au LUT 3D afin de prédire les données CIELUV pour une donnée RVB. (Conférer le 4275413 us Patent 4275413 (Sakamoto et al.), US Patent 4511989 (Sakamoto), Kang [1]. Les deux brevets mentionnés ont expiré.). Les données de paramètre analytique transmises dans la deuxième méthode sont simplement un LUT qui a été créé précédemment. En règle générale, ce LUT a été créé à l'aide de la première méthode, bien qu'il puisse être construit à la main.

Dans la gestion actuelle des couleurs, la carte source est définie comme la carte qui va de l'espace RVB à un espace de couleurs CIEXYZ indépendant de l'appareil. La carte de destination est définie comme la carte qui va de l'espace colorimétrique CIEXYZ indépendant de l'appareil à l'espace RVB. Il s'agit de l'inverse de la carte source.

Le modèle empirique est directement utilisé dans la carte source. Il mappe d'abord une donnée RVB en données CIELUV, qui sont converties en données XYZ. Dans la carte de destination, les données CIEXYZ indépendantes de l'appareil sont d'abord converties en données CIELUV. Ensuite, le modèle empirique et la méthode classique Newton-Raphson sont utilisés pour prédire les meilleures données RVB pour les données CIELUV. Les détails sur la conversion de données CieLUV en données RVB sont les suivants :

Après avoir généré un LUT 3D de RVB vers CieLUV, la carte de RVB à LUV est créée à l'aide de l'interpolation tétraédrique sur RVB. Cette carte est indiquée par les équations suivantes :

$$L = L(R, G, B)$$
$$u = u(R, G, B)$$
$$v = v(R, G, B)$$

L'inversion de la carte consiste à résoudre, pour toute couleur L^*, u^*, v^*, le système suivant d'équations non linéaires :

$$L^* = L(R, G, B)$$
$$u^* = u(R, G, B)$$
$$v^* = v(R, G, B)$$

Une équation non linéaire basée sur la méthode classique Newton-Raphson est utilisée dans le nouveau CTE. Une estimation initiale, ou *a priori* voir, $s_{précédent}$ -(R 0, G 0, B 0) est obtenue en recherchant dans une « matrice de départ » composée d'une grille 8x8x8 uniforme de paires précalcalcées (RVB,Luv). Le Luv RVB correspondant le plus proche de L*u*v* est choisi. Chaque point de la matrice initiale correspond au centre d'une cellule afin que les itérations ne commencent pas par un point sur la face limite du cube RVB. En d'autres termes, le RVB des graines est défini par : STEP = 1/8 s_{ijk} = (STEP/2 + (i-1) STEP, STEP/2+(j-1)STEP, STEP/2+(k-1)STEP) avec i,j,k = 1...8 À la *iième* étape de Newton-Raphson, l'estimation suivante $R_{i+1}, G_{i+1}, B_{i+1}$ est obtenue par la formule :

$$\begin{pmatrix} R_{i+1} \\ G_{i+1} \\ B_{i+1} \end{pmatrix} = \begin{pmatrix} R_i \\ G_i \\ B_i \end{pmatrix} + \begin{pmatrix} \partial L/\partial R & \partial L/\partial G & \partial L/\partial B \\ \partial u/\partial R & \partial u/\partial G & \partial u/\partial B \\ \partial v/\partial R & \partial v/\partial G & \partial v/\partial B \end{pmatrix}^{-1}_{R_i, G_i, B_i} \begin{pmatrix} L^* - L(R_i, G_i, B_i) \\ u^* - u(R_i, G_i, B_i) \\ v^* - v(R_i, G_i, B_i) \end{pmatrix}$$

L'itération s'arrête lorsque l'erreur (distance dans l'espace CIELUV) est inférieure à un niveau de tolérance prédéfinis (0,1 dans le CTE), ou lorsque le nombre d'itérations a dépassé le nombre maximal autorisé d'itérations (10 dans le CTE). Les valeurs de la tolérance et du nombre d'itérations ont été déterminées de manière empirique comme étant efficaces. Dans les versions ultérieures, la valeur de tolérance peut être modifiée.

La matrice jacobienne est calculée à l'aide de la différence vers l'avant, sauf à un point limite (un ou plusieurs des R, G, B est 1) où la différence vers l'arrière est utilisée à la place. Au lieu de calculer l'inverse de la matrice jacobienne, le système linéaire est résolu directement à l'aide de Gauss-Jordan'élimination avec pivotation complète.

À la fin des itérations, la convergence peut toujours ne pas être obtenue, car Newton-Raphson est un algorithme « local », c'est-à-dire qu'il ne fonctionne bien que si vous commencez par une estimation initiale qui est proche de la vraie solution. Si, à la fin des itérations Newton-Raphson, la convergence au sein de la tolérance d'erreur prédéfinie n'a pas été atteinte, les itérations sont redémarrées avec un nouvel ensemble de graines, défini comme suit.

Par exemple, la meilleure solution obtenue jusqu'à présent est (r, g, b). De cette solution, N graines a posteriori sont dérivées, où N = 4. Intuitivement, la solution est déplacée « vers le centre » dans une taille d'étape qui dépend de N. Voir la figure 6.

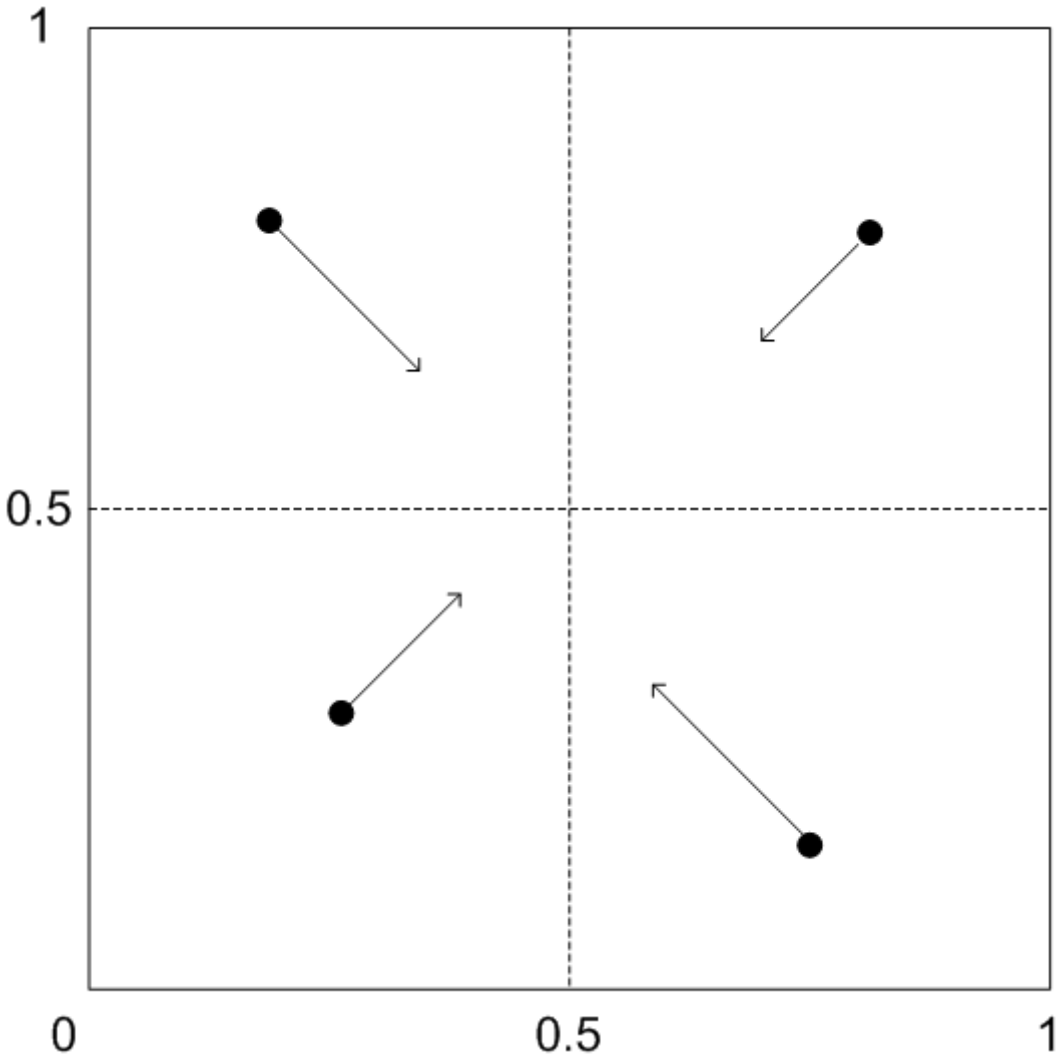


Figure 6 : Le sens de perturbation de la solution dépend de l'octant dans lequel elle se trouve.

En d'autres termes, si r > 0,5, la valeur sur le canal R est réduite, sinon la valeur est augmentée. Il existe une logique similaire pour les canaux G et B. Les définitions précises sont les suivantes :

PERTURBATION = 0,5/(N+1)

Dir(r) = -1 si r > 0,5 ; +1 sinon. De même pour Dir(g) et Dir(b)

Le jth a posteriori seed, s ????, est (r + Dir(r) *j * PERTURBATION, g + Dir(g) * j * PERTURBATION, b + Dir(b) * j * PERTURBATION)

Essayez la première ???? et s'il donne une nouvelle solution dans les limites de tolérance d'erreur, vous pouvez arrêter. Sinon, essayez la deuxième ???? et ainsi de suite jusqu'à ce que le Nth s ????.

Le schéma de l'algorithme entier est illustré à la figure 7.

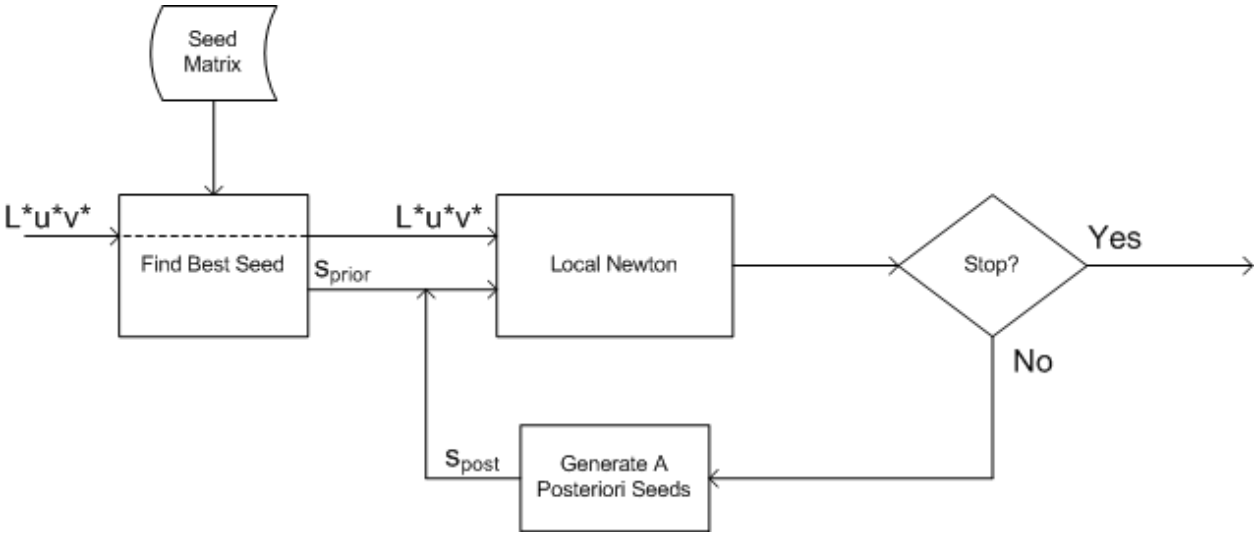


Figure 7 : Schémas de l'inversion du modèle d'appareil

Base de référence du modèle d'appareil virtuel RVB

Ce modèle d'appareil (DM) est un algorithme de reproduction de matrice/tonalité simple. La matrice est dérivée du point blanc et des primaires à l'aide d'algorithmes de science des couleurs standard. La courbe de reproduction tonalité est dérivée des paramètres de mesure selon les descriptions ICC de CurveType et ParametricType (ou inversées selon les besoins). Les détails des algorithmes internes seront fournis après une validation supplémentaire des problèmes de plage dynamique élevée.

Le modèle d'appareil virtuel RVB est une courbe de reproduction de matrice/tonalité idéalisée RVB similaire à la conception de profil basé sur trois composants de matrice ICC. Les paramètres de « mesure virtuelle » du DM incluent une valeur de point blanc (CIEXYZ absolu), des valeurs primaires RVB (CIEXYZ absolu) et une courbe de reproduction de tonalité basée sur le paramètre ICC ParametricCurveType et CurveType dans une mise en forme XML cohérente avec les schémas DMP.

L'encodage du type de fonction ICC parametricCurveType et la prise en charge correspondante dans IRGBVirtualDeviceModelBase sont indiqués dans le tableau suivant.

Type de fonction	Paramètres	Type	Notes
$Y = X^{\gamma}$	g	GammaType	Implémentation courante
$Y = (aX + b)^{\gamma} \quad (X \geq -b/a)$ $Y = 0 \quad (X < -b/a)$	ga b	GammaOffsetGainType	CIE 122-1966
$Y = (aX + b)^{\gamma} + c \quad (X \geq -b/a)$ $Y = c \quad (X < -b/a)$	ga b c	GammaOffsetGainOffsetType	IEC 61966-3

Type de fonction	Paramètres	Type	Notes
$Y = (aX + b)^{\gamma} \quad (X \geq d)$ $Y = cX \quad (X < d)$	ga b c d	GammaOffsetGainGainType	IEC 61966-2.1 (sRGB)
$Y = (aX + b)^{\gamma} + e \quad (X \geq d)$ $Y = (cX + f) \quad (X < d)$	ga b c d e f	N/A	Non pris en charge dans WCS

La courbe tonalité des appareils virtuels RVB est appliquée dans DeviceToColorimetric entre les données d'entrée, pDeviceColors et la matrice multipliée. Pour ColorimetricToDevice, une méthode doit être utilisée pour inverser la courbe de tonalité. Dans l'implémentation de la base de référence, cela s'effectue par interpolation directe dans la même courbe de tonalité que celle utilisée pour DeviceToColorimetric.

Les courbes doivent être spécifiées dans les profils sous forme de paires de nombres dans l'espace flottant. Le premier nombre représente les valeurs dans pDeviceColors. Le deuxième nombre représente la sortie de la courbe de tonalité. Toutes les valeurs de la courbe de tonalité doivent être comprises entre minColorantUsed et maxColorantUsed. Les courbes de tonalité doivent contenir au moins deux entrées : une pour minColorantUsed et une pour maxColorantUsed. Le nombre maximal d'entrées dans le ToneCurve est 2048. En général, plus il y a d'entrées dans la table, plus vous pouvez modéliser la courbure avec précision. Une interpolation linéaire au niveau de la pièce est effectuée entre les entrées.

Vous pouvez envisager d'autres méthodes d'interpolation. Si vous savez quelque chose sur le comportement sous-jacent de l'appareil, vous pouvez utiliser moins d'exemples et modéliser plus précisément avec une courbe d'ordre supérieur. Mais si vous utilisez le mauvais type de courbe, vous serez très imprécis. Sans plus d'informations, vous ne pouvez pas deviner le type de courbe. Par conséquent, utilisez l'interpolation linéaire et fournissez de nombreux points de données.

Base de référence du modèle d'appareil d'imprimante CMYK

Une caractérisation d'appareil d'une imprimante CMJN consiste à construire un modèle empirique de l'appareil qui prédit la couleur imprimée pour une valeur CMJN donnée. La caractérisation inclut également l'inversion de ce modèle afin qu'une prescription de la valeur CMJN pour une couleur donnée à imprimer puisse être donnée. Cela est généralement défini en termes de valeur CIEXYZ ou CIELAB.

En règle générale, une cible IT8.7/3 contenant des correctifs CMJK est utilisée. Les correctifs consistent à échantillonner l'espace CMJN de manière bien définie afin qu'une

grille rectangulaire (avec un espacement non uniforme en C, M, Y et K) soit formée. Chaque patch est ensuite mesuré avec un colorimètre ou un spectrophotomètre. Les mesures dans les valeurs CIEXYZ forment ensuite une table de recherche (LUT), à partir de laquelle un modèle empirique de l'imprimante est généré à l'aide de la méthode d'interpolation de Sakamoto. US Patent 4275413 (Sakamoto et al.), US Patent 4511989 (Sakamoto), Kang [1]. Les deux brevets mentionnés ont expiré.

Les exigences spécifiques pour les exemples de mesure CMJN nécessaires pour qu'un profil de modèle d'appareil soit accepté comme valide par le modèle d'appareil de base de l'imprimante CMJN sont les suivantes. (L'exemple d'ensemble est le plus clairement décrit comme un ensemble d'exemples de cubes CMY, chacun associé à un niveau K spécifique.)

- Au minimum, les cubes CMY valides doivent être fournis pour les niveaux K = 0 et K = 100.
- Les niveaux K intermédiaires peuvent être non uniformément espadés.
- Tout niveau K intermédiaire sans cube CMY valide est ignoré.
- Les cubes CMY peuvent utiliser des intervalles d'échantillonnage non uniformes (espacement de grille), mais le même ensemble d'intervalles d'échantillons doit être utilisé dans toutes les dimensions C,M et Y dans le cube CMY pour un niveau K donné.
- Chaque cube CMY de niveau K peut utiliser un nombre et un espacement différents d'intervalles d'échantillons.
- Tous les cubes CMY doivent contenir les « coins » du cube CMY, c'est-à-dire, Échantillons CMY à [0,0,0], [0,0,100], [0,100,0], [100,0,0], [0,100,100], [100,0,100], [100,100, 0], [100,100,100].
- Tous les niveaux de grille CMY intermédiaires doivent être entièrement échantillonné dans chaque canal. En d'autres termes, un exemple doit exister à chaque intersection de grille 3D dans le cube CMY.
- Pour les cubes K = 0 et K = 100 cubes CMY, 2x2x2 cubes « corners-only » sont le minimum accepté comme valide.

  [REMARQUE : pour K=0 et K=100 niveaux, un cube CMY de 3x3x3 est traité comme un cube « corners-only » 2x2x2 ; le niveau intermédiaire de l'exemple est ignoré. Les cubes 4x4x4 et les cubes plus grands auront tous les exemples sur la grille utilisés.]

- Pour les niveaux K intermédiaires, les cubes CMY 4x4x4 sont le minimum accepté comme valide.

Un profil de haute qualité utilise des grilles d'échantillonnage plus fines que le minimum requis pour la validité, généralement 9x9x9x9 ou plus. Les exemples des cibles IT8.7/3, IT8.7/4 et ECI produisent des profils de modèle d'appareil valides (DPM) pour le modèle d'appareil de base de l'imprimante CMJN. Bien que ce modèle d'appareil soit en mesure d'ignorer les exemples superflus (hors grille) dans ces cibles, il n'est pas garanti de pouvoir le faire pour d'autres cibles. Par conséquent, il est recommandé de supprimer les échantillons superflus des jeux de mesures entrant dans les profils pour ce modèle d'appareil.

L'inversion du modèle d'imprimante présente plus de difficultés. Étant donné une couleur d'entrée dans CIECAM, il est question de savoir si cette couleur se trouve dans le gamut de l'imprimante. Il existe également la question de la disposition des points dans l'espace d'apparence des couleurs. Bien que nous puissions organiser les valeurs CMJN pour qu'elles tombent sur une grille rectangulaire, comme c'est le cas dans la cible IT8.7/3, il n'en va pas de même pour les couleurs imprimées obtenues, car elles se trouvent dans l'espace d'apparence des couleurs. En général, ils sont dispersés dans l'espace d'apparence des couleurs sans motif particulier.

Il existe généralement deux approches pour résoudre le problème d'inversion des points dispersés. Une approche consiste à utiliser une subdivision géométrique de la gamme d'imprimantes à l'aide de solides 3 dimensions élémentaires, tels que les tétraèdres. Une subdivision du gamut d'imprimante dans l'espace d'apparence des couleurs peut être induite à partir de la subdivision correspondante de l'espace CMY(K), voir Hung[93], Kang[97]. Cette approche présente l'avantage de la simplicité de calcul. Dans le cas d'un tétraèdre, seuls quatre points sont utilisés dans une interpolation. D'autre part, le résultat dépend fortement de quelques points, ce qui signifie qu'une erreur de mesure aura un effet significatif sur le résultat. L'interpolant a également tendance à être moins lisse. La deuxième approche ne suppose aucune subdivision et est basée sur la technique de l'interpolation de données éparses. Un exemple classique est la technique de l'interpolation de Shepard, ou méthode pondérée inverse (voir Shepard [68]). Ici, plusieurs points entourant le point d'entrée sont choisis d'une manière ou d'une autre, chacun d'eux ayant un poids, généralement inversement proportionnel à la distance, et l'interpolant est pris comme la moyenne pondérée des points voisins. Dans cette approche, le choix des points voisins est primordial pour les performances. Bien que trop peu de points puissent rendre l'interpolant imprécis et non lisse, trop de points imposent un coût de calcul élevé, car les pondérations sont généralement des fonctions non linéaires et coûteuses à calculer.

Les deux approches décrites ci-dessus présentent des problèmes. L'approche de la subdivision dépend particulièrement du fait que les données sont relativement vides de bruit et que, généralement, l'interpolant n'est pas très lisse. L'interpolation de données éparses est plus tolérante au bruit des données et donne généralement un interpolant plus fluide, mais elle est plus coûteuse en termes de calcul.

Le nouveau CTE adopte une autre approche. L'appareil CMJK est traité comme une collection de plusieurs appareils CMY, chacun ayant une valeur spécifique de noir (K). À l'aide d'un algorithme de sélection qui prend comme paramètres légèreté et chroma, un niveau de noir est choisi. Les valeurs CMY sont obtenues par inversion de la table CMY en Luv correspondante à l'aide des méthodes Newton utilisées ailleurs par l'algorithme d'imprimante RVB.

Procédez comme suit.

1. Imprimez la cible de caractérisation, qui est soit la cible IT8.7/3, soit une cible contenant l'échantillonnage de l'espace CMJN à intervalles réguliers ou irréguliers.
2. Mesurez la cible à l'aide d'un spectrophotomètre et convertissez les mesures en espace CIELUV.
3. Construisez la carte avant de CMJN à Luv.
4. Utilisez la carte avant pour construire un ensemble de cartes CMY à Luv pour une plage de valeurs K.
5. Pour toute valeur Luv d'entrée, la valeur CMJN correspondante est obtenue en sélectionnant l'une des cartes construites à l'étape 4 ci-dessus et en inversant à l'aide de la méthode de Newton pour obtenir un jeu de colorant CMY pour accompagner la valeur K sélectionnée.

Les étapes 1 et 2, qui sont des procédures standard, sont effectuées par un programme de profilage qui ne fait pas partie du nouveau CTE. La cible IT8.7/3 contient un échantillonnage assez détaillé de toutes les valeurs CMJN à différents niveaux de C, M, Y et K. Une cible personnalisée avec échantillonnage uniforme des canaux C, M, Y et K peut également être utilisée. Une fois la cible imprimée, un spectrophotomètre ou un colorimètre peut être utilisé pour mesurer la valeur XYZ de chaque correctif, et la valeur XYZ peut être convertie en valeur Luv à l'aide du modèle CIELUV.

L'étape 3, la construction de la carte en avant de CMYK à Luv, peut être obtenue en appliquant n'importe quelle technique d'interpolation connue, telle que la méthode tétraédrale ou multiligne, sur la grille rectangulaire dans l'espace CMJN. Dans le nouveau CTE, une interpolation tétraédrique à 4 dimensions est utilisée. Étant donné que les grilles d'échantillonnage CMY sont généralement différentes à chaque niveau de K, nous utilisons une technique de super-échantillonnage, comme indiqué ci-dessous. Pour un point CMJN donné, les niveaux K de sandwich sont d'abord déterminés en

fonction de la valeur K. Ensuite, introduisez une « super-grille » à chaque niveau K qui est une union des grilles CMY sur chacun des deux niveaux K. À chaque niveau K, la valeur Luv de tout point de grille nouvellement introduit est obtenue par une interpolation tétraédrique à 3 dimensions au sein de ce niveau K. Enfin, une interpolation tétraédrique à 4 dimensions pour le point CMJK spécifique est effectuée sur cette nouvelle grille.

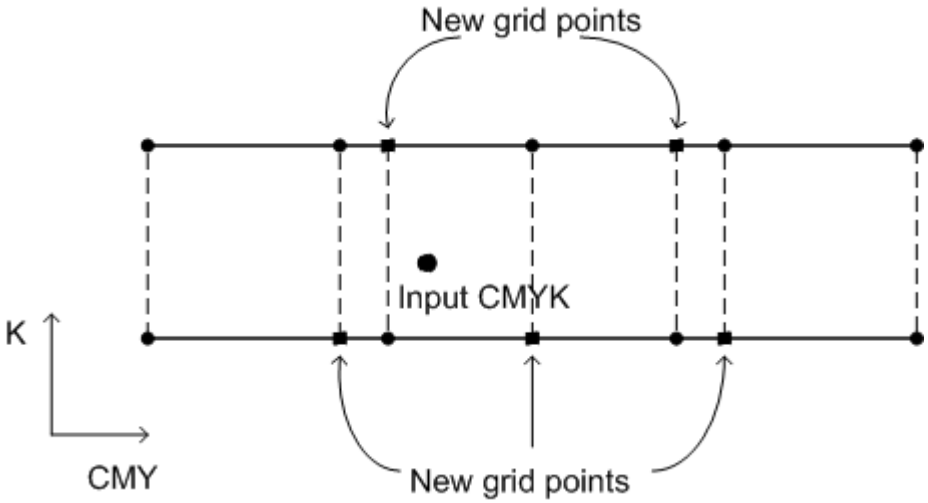


Figure 8 : Suréchantillonnage

L'étape 4 construit un ensemble de tables de recherche CMY-to-Luv (LUT). La carte avant construite à l'étape 3 est appelée à plusieurs reprises pour rééchantillonner l'espace CMJN. L'espace de couleur CMJN est échantillonné à l'aide d'un espacement homogène de K et d'un échantillonnage différent, mais toujours uniformément espacé, de CMY.

L'étape 5 est une procédure permettant d'obtenir la valeur CMJN à l'aide des luts construits à l'étape 4 pour n'importe quel point de Luv d'entrée. La valeur appropriée de K est choisie en analysant la légèreté ainsi que le degré de couleur dans le Luv demandé. Une fois la table sélectionnée, les valeurs CMY sont obtenues à partir de la table à l'aide de la méthode de Newton (comme indiqué sous le modèle d'imprimante RVB précédemment).

L'espace CIELUV est utilisé dans le modèle d'imprimante au lieu de CIEJab, car le modèle d'appareil doit être basé uniquement sur les données colorimétriques disponibles dans le DMP. Le DMP contient des données colorimétriques pour chaque correctif mesuré, y compris le point blanc du média, de sorte qu'il est possible de convertir des données CIEXYZ en données CIELUV. Toutefois, il n'est pas possible de convertir en CIECAM02 JCh ou Jab, car il n'y a pas accès aux informations de condition d'affichage dans le DMP.

Base de référence du modèle d'appareil projecteur RVB

Remarque : De nombreux projecteurs RVB ont plusieurs modes de fonctionnement. Dans un mode, qui est souvent la valeur par défaut et peut être appelé quelque chose comme « présentation », la réponse en couleur du projecteur est optimisée pour une luminosité maximale. Toutefois, dans ce mode, le projecteur perd la capacité de reproduire des couleurs légèrement chromatiques, telles que les jaunes pâles et certaines tonalités de chair. Dans un autre mode, souvent appelé « film », « vidéo » ou « sRGB », le projecteur est optimisé pour la reproduction d'images réalistes et de scènes naturelles. Dans ce mode, la luminosité maximale est échangée pour améliorer la qualité globale de la reproduction des couleurs. Pour obtenir une reproduction satisfaisante des couleurs avec des projecteurs RVB, il est nécessaire de placer le projecteur dans un mode où une gamme de couleurs lisse peut être reproduite.

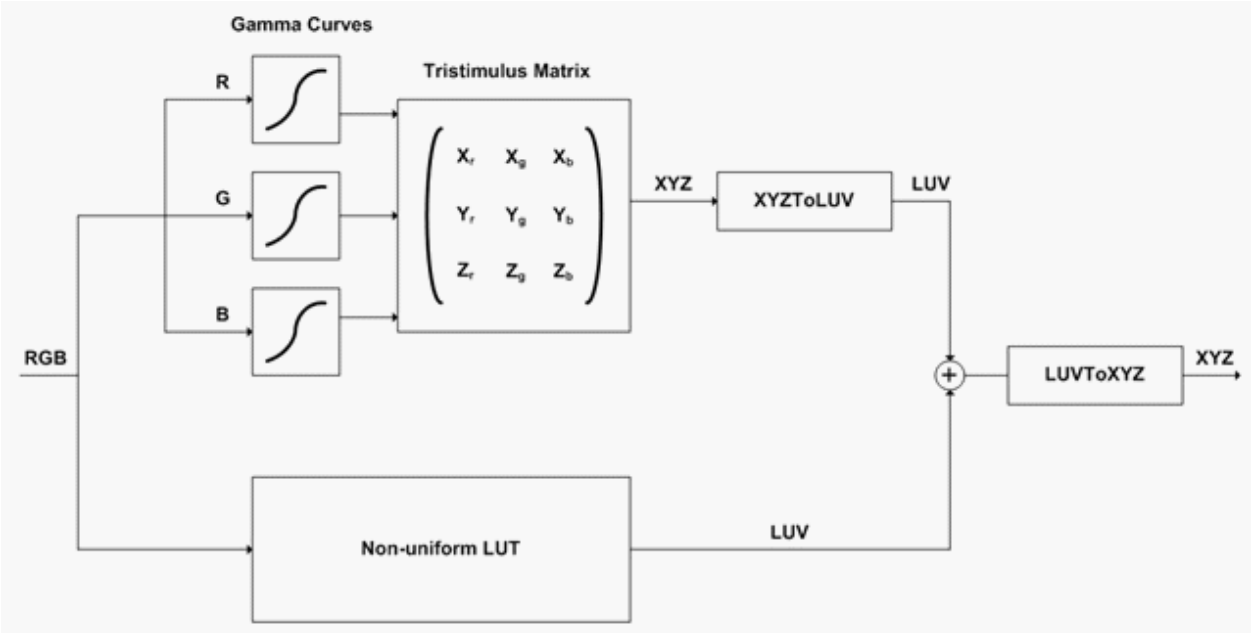


Figure 9 : modèle d'appareil DLP

Une valeur RVB entrante passe par deux voies de calcul. La première est le modèle matriciel, qui aboutit à une valeur XYZ. Cette opération est immédiatement suivie de la conversion de XYZ en Luv. La deuxième est l'interpolation LUT non uniforme utilisant l'interpolation tétraédrale. La sortie de l'interpolation est déjà dans l'espace Luv par construction. Les deux sorties sont ajoutées pour obtenir la valeur Luv prédite. Il est finalement converti en XYZ, qui est la sortie attendue du modèle colorimétrique pour l'appareil DLP.

Étant donné que les projecteurs sont des appareils d'affichage, ils prennent également en charge l'inversion du modèle, c'est-à-dire la transformation de XYZ en RVB. Étant donné que le modèle d'appareil transforme l'espace RVB en espace XYZ, qui sont tous deux tridimensionnels, l'inversion équivaut à résoudre trois équations non linéaires dans trois inconnues. Cela peut être effectué par des techniques de résolution d'équations standard, telles que Newton-Raphson. Toutefois, il est préférable de convertir d'abord

XYZ en Luv, puis d’inverser la transformation Luv en RVB, car l’espace Luv est plus linéaire perceptulement que l’espace XYZ.

Base de référence du modèle d’appareil ICC

L’interopérabilité du flux de travail ICC CITE est activée en créant un profil de modèle d’appareil de référence d’appareil ICC spécial qui stocke l’objet de profil et crée une transformation ICC à l’aide d’un profil XYZ sans opération. Cette transformation est ensuite utilisée pour traduire entre les couleurs de l’appareil et CIEXYZ.

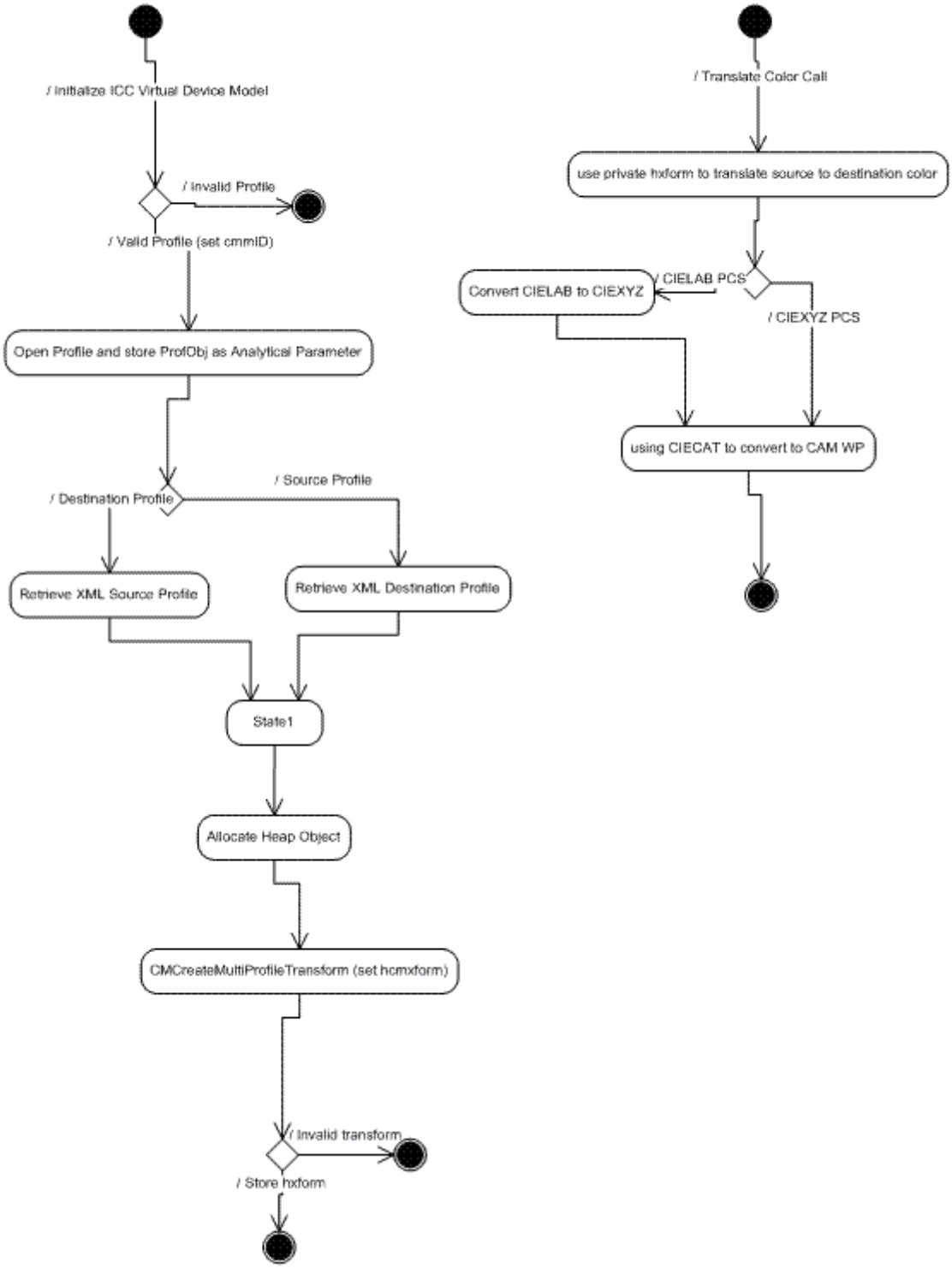


Figure 10 : Interopérabilité des flux de travail ICC CITE

Rubriques connexes

Concepts de base de la gestion des couleurs

Schémas et algorithmes du système de couleurs Windows

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l’aide sur Microsoft Q&A

Algorithmes et schéma de profil de modèle de mappage de couleur WCS

Article • 13/06/2023

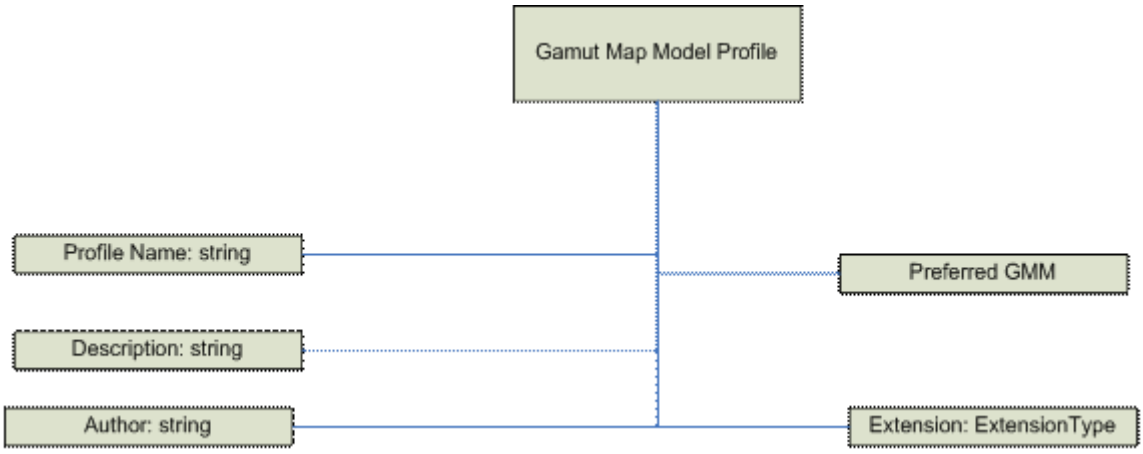
Vue d’ensemble

Ce schéma est utilisé pour spécifier le contenu d’un profil de modèle de carte gamut (GMMP). Les algorithmes de base associés sont décrits dans la rubrique suivante.

Le schéma GMMP de base se compose d’informations d’en-tête courantes, d’une référence facultative à un plug-in Gamut Map Model préféré et de balises d’extension.

En outre, le GMMP fournit des informations explicites sur le modèle de carte gamut ciblé et fournit une stratégie sur le modèle de carte gamut de référence de secours à utiliser si le modèle ciblé n’est pas disponible. Le schéma peut inclure des informations d’extension privées, mais n’inclut aucune autre information superflue.

Architecture de profil de modèle de carte gamut



L’échantillonnage de l’espace colorant du dispositif de sortie est effectué en itérant dans les colorants de 0.0 à 1.0 avec une étape fractionnaire, en accumulant toutes les combinaisons de chaque colorant à chaque étape, puis en les convertissant de l’espace colorant du périphérique en espace d’apparence de couleur à l’aide de la méthode DM::D eviceToColorimetricColors suivie de la méthode CAM::ColorimetricToAppearanceColors. Voici un exemple de la façon dont cette opération est effectuée pour RVB.

```
C++

For (red= 0.0; red <= 1.0; red += redStep) {

     For (green = 0.0; green <= 1.0; green += greenStep) {

          For (blue = 0.0; blue <= 1.0; blue += blueStep) {

               Colorants[0] = red; colorants[1] = green; colorants[2] =
```

```
blue;

                pRGBDM->DeviceToColorimetricColors(1, colorants, &amp;XYZ);

                pCAM->ColorimetricToAppearanceColors(1, &amp;XYZ, &amp;JCh);

           }

      }

}
```

Génération de la limite de gamut

Il existe trois composants à la limite de la gamme : les primaires, les échantillons neutres et les interpréteurs de commandes. Les primaires sont générées en prenant les primaires d'appareil et en appliquant la transformation DeviceToColorimetric/ColorimetricToAppearance. Les échantillons neutres sont générés en échantillonnant l'espace colorant de l'appareil dans la zone neutre et en appliquant la même transformation. Pour trois dispositifs colorants (RVB ou CMY), les échantillons neutres sont définis comme ayant tous les colorants égaux, par exemple, R == G == B. Pour CMJN, les échantillons neutres sont définis comme ayant C == M == Y == 0.

Les facteurs qui influencent les données utilisées pour créer la limite de la gamme sont les exemples de données (appareils de base uniquement) et les conditions d'affichage. La taille d'étape utilisée pour effectuer l'échantillonnage complet de l'espace colorant (pour les moniteurs et pour la coque plausible) est une constante interne et n'est pas disponible pour une manipulation extérieure. La modification des conditions d'affichage modifie le comportement du modèle d'apparence de couleur (CAM) et modifie la forme de la limite de gamut. Vous devez donc générer une limite de gamut liée au profil des conditions d'affichage. Si des exemples de données sont utilisés, comme dans le cas des imprimantes de base et des périphériques de capture, les exemples auront un impact important sur la forme de la gamme de référence et affecteront le comportement du modèle lui-même.

Pour les périphériques d'entrée, tels que les caméras et les scanneurs, différents échantillonnages sont utilisés pour générer l'interpréteur de commandes de référence et l'interpréteur de commandes plausible. L'interpréteur de commandes de référence est généré à partir des mesures utilisées pour initialiser le modèle d'appareil. L'interpréteur de commandes plausible est généré de la même façon que l'illustration précédente pour les périphériques de sortie. La différence est que lorsque vous entrez une cible classique,

vous n'obtenez pas de valeurs complètement saturées (où R, G ou B = 255). Vous devez extrapoler à l'aide du modèle d'appareil pour estimer ces valeurs.

Schéma GMMP

C++

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema

xmlns:gmm="http://schemas.microsoft.com/windows/2005/02/color/GamutMapModel"

xmlns:wcs="http://schemas.microsoft.com/windows/2005/02/color/WcsCommonProfi
leTypes"

targetNamespace="http://schemas.microsoft.com/windows/2005/02/color/GamutMap
Model"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  blockDefault="#all"
  version="1.0">

  <xs:annotation>
    <xs:documentation>
      Gamut Map Model profile schema.
      Copyright (C) Microsoft. All rights reserved.
    </xs:documentation>
  </xs:annotation>

  <xs:import
namespace="http://schemas.microsoft.com/windows/2005/02/color/WcsCommonProfi
leTypes" />

  <xs:element name="GamutMapModel">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ProfileName" type="wcs:MultiLocalizedTextType"/>
        <xs:element name="Description" type="wcs:MultiLocalizedTextType"
minOccurs="0"/>
        <xs:element name="Author" type="wcs:MultiLocalizedTextType"
minOccurs="0"/>
        <xs:element name="DefaultBaselineGamutMapModel">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="HPMinCD_Absolute"/>
              <xs:enumeration value="HPMinCD_Relative"/>
              <xs:enumeration value="SGCK"/>
              <xs:enumeration value="HueMap"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
```

```
        <xs:element name="PlugInGamutMapModel" minOccurs="0">
          <xs:complexType>
            <xs:sequence>
              <xs:any namespace="##other" processContents="skip"
                minOccurs="0" maxOccurs="unbounded" />
            </xs:sequence>
            <xs:attribute name="GUID" type="wcs:GUIDType" use="required"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="ID" type="xs:string" use="optional" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Éléments de schéma GMMP

GamutMapModel

Cet élément est une séquence des sous-éléments suivants :

1. Chaîne ProfileName,
2. DefaultBaselineGamutMapModel,
3. Chaîne de description facultative,
4. Chaîne d'auteur facultative,
5. PlugInGamutMap facultatif et
6. ExtensionType facultatif.

Conditions de validation : chaque sous-élément est validé par son propre type.

Espace de noms

xmlns:gmm= »http://schemas.microsoft.com/windows/2005/02/color/GamutMapModel
"

targetNamespace= »http://schemas.microsoft.com/windows/2005/02/color/GamutMap
Model"

Version

Version « 1.0 » avec la première version de Windows Vista.

Conditions de validation : 1.0 dans Windows Vista. Les versions <2.0 sont également valides pour prendre en charge les modifications de format sans rupture.

Documentation

Schéma de profil de modèle de carte de gamut.

Conditions de validation : chaque sous-élément est validé par son propre type.

Type DefaultBaselineGamutMapModel

Type UINT

Valeurs d'énumération :
« MinCD_Absolute » « MinCD_Relative » « SIG_KNEE » « HueMap »

Conditions de validation : les valeurs doivent correspondre à l'une des énumérations ci-dessus.

PlugInGamutMapType

Cet élément est une séquence d'un GUID GUIDType et de tous les sous-éléments.

Conditions de validation : le GUID est utilisé pour correspondre au GUID de la DLL PlugIn GMM. Il existe un maximum de 100 000 sous-éléments personnalisés.

ExtensionType

Cet élément est une séquence facultative de tous les sous-éléments.

Conditions de validation : il peut y avoir un maximum de 100 000 sous-éléments.

Algorithmes de base GMMP

Alignement des axes neutres

La plupart des algorithmes de mappage de gamut ont pour objectif de mapper l'axe neutre de l'appareil source à l'axe neutre de l'appareil de destination : autrement dit, le blanc passe au blanc, le noir au noir et le gris au gris. Cela est résolu en partie par la mise à l'échelle de la luminosité des couleurs sources pour qu'elle corresponde à la plage de luminosité de l'appareil de destination. Mais cela ne répond pas complètement au problème.

Il est une propriété physique de la plupart des appareils d'imagerie que la chromaticité du blanc de l'appareil ne correspond pas exactement à la chromaticité de l'appareil noir. Par exemple, le blanc du moniteur dépend de la somme des chromaticités et des luminosités relatives des trois primaires, tandis que le moniteur noir dépend de la réflectance de la surface d'affichage. Le blanc de l'imprimante dépend de la chromaticité du papier, tandis que le noir de l'imprimante dépend de l'encre ou du toner utilisé. Un modèle d'apparence qui mappe le blanc de l'appareil exactement à l'axe neutre de l'espace d'apparence (chroma exactement égal à zéro) ne mappe pas l'appareil en noir à l'axe neutre. Étant donné que l'œil est plus sensible aux différences chromatiques à mesure que la légèreté augmente, le gris moyen sera représenté comme encore plus chromatique que le noir d'appareil. (La figure 1 illustre la courbure des axes neutres en deux dimensions. En fait, les axes neutres forment une courbe plus complexe en trois dimensions.)

Alors que le CAM prédit que ces couleurs neutres d'appareil apparaîtront chromatiques, les observateurs réels semblent compenser cela. La plupart des gens ne considèrent pas ces valeurs neutres d'appareil comme chromatiques. Pour la plupart des modèles de mappage de gamut, par conséquent, vous devez continuer à mapper des sources neutres à des appareils neutres.

Pour mapper des sources neutres à des appareils neutres, ajustez les limites de la gamme source et de destination et chaque pixel lorsque vous appliquez l'algorithme de mappage de gamut. Commencez par ajuster chaque valeur dans la couleur source afin que l'axe neutre de l'appareil source au niveau de la luminosité de la couleur source tombe directement sur l'axe neutre de l'espace d'apparence. (Voir le côté gauche de la figure 1.) Ajustez ensuite la description de la limite de gamut de l'appareil de destination afin que chaque couleur de l'axe neutre de l'appareil de destination au niveau de la couleur de la limite de la gamme de l'appareil de destination tombe directement sur l'axe neutre de l'espace d'apparence. (Voir le côté droit de la figure 1.)

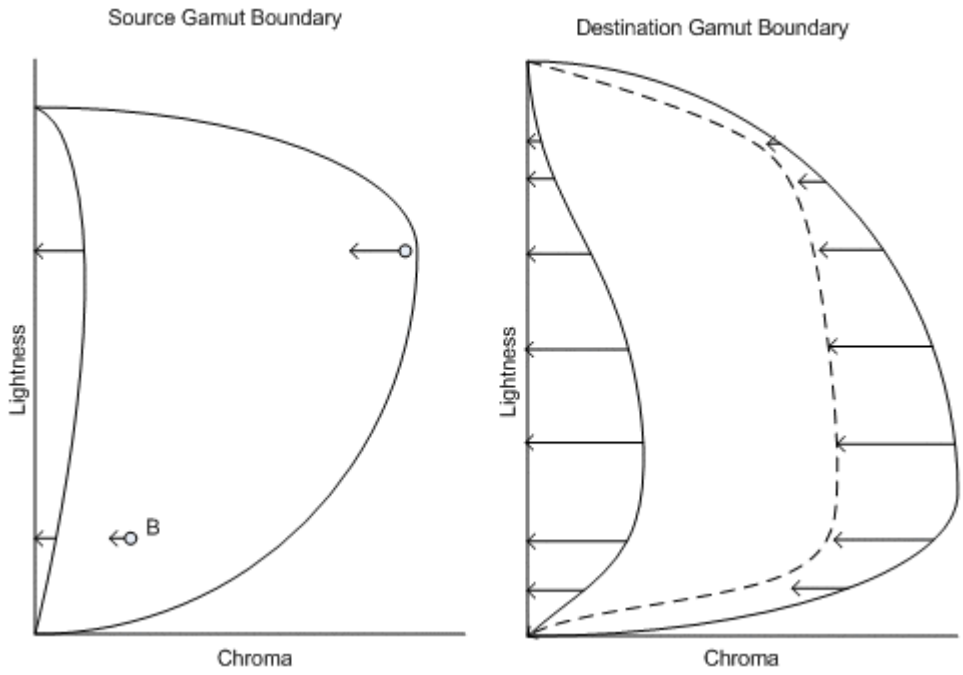


Figure 1 : Alignement des axes neutres illustrés. Gauche : ajustement des points sources par rapport à l'axe neutre de l'appareil source. Droite : ajustement de la description de la limite de gamut de destination par rapport à la description de la limite de la gamme de destination.

Notez que vous ajustez chaque valeur de pixel source par rapport à l'axe neutre à cette valeur de luminosité. Cela garantit que les appareils sources neutres tomberont sur l'axe neutre du modèle d'apparence. Vous déplacez également toutes les autres couleurs à cette légèreté du même montant, de sorte qu'il n'y ait pas de discontinuités dans la représentation de la gamme source. Vous devez décaler par des quantités différentes à différents niveaux de luminosité, car les neutres de l'appareil source ne sont pas représentés comme également chromatiques aux différents niveaux de luminosité. De toute évidence, il ne s'agit pas d'une transformation triviale.

La gestion des valeurs de l'appareil de destination est un peu plus difficile. Au départ, vous ajustez la limite de la gamme de destination entière de la même manière, mais par rapport à l'axe neutre de l'appareil de destination. Ceci est illustré dans la figure 1 sur le côté droit. Cet ajustement garantit que les valeurs grises sources seront mappées aux valeurs grises de destination. Il s'agit de l'espace dans lequel les algorithmes de mappage de gamut fonctionnent.

Toutefois, cet espace ne décrit pas avec précision le véritable comportement de l'appareil de destination. Vous devez inverser le mappage avant que les couleurs mappées par gamut soient transmises au modèle d'apparence et au modèle d'appareil de destination. Vous décalerez toutes les valeurs mappées par l'opposé du décalage appliqué précédemment à l'axe neutre de l'appareil de destination. Cela mappe l'axe neutre de destination à la valeur représentée à l'origine par le CAM. Il en est de même pour la limite de la gamme et toutes les valeurs intermédiaires.

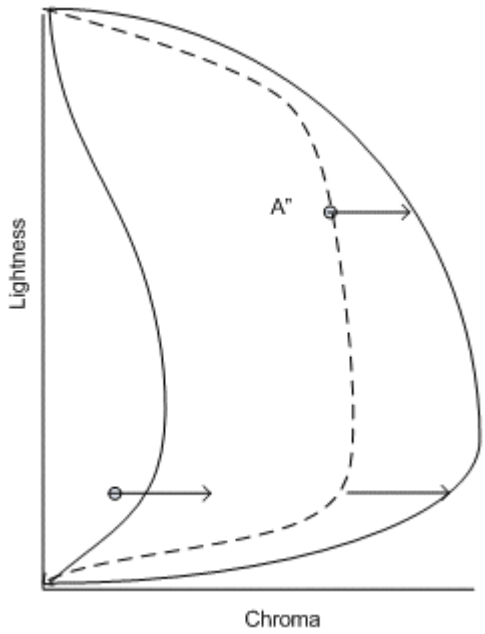


Figure 2 : Annulation de l'alignement de l'axe neutre de l'appareil de destination

Différence de couleur minimale (MinCD)

Minimum Color Difference (MinCD) Relative et Absolute versions : équivalent à l'intention colorimétrique ICC.

> ⓘ **Notes**
>
> Le GMM MinCD convient pour le mappage de graphiques et de dessins en courbes contenant des couleurs de « logo » (couleurs spot), des dégradés de couleurs de logo avec des couleurs hors gamme et pour la phase finale des transformations de vérification. Bien que le GMM MinCD puisse être utilisé pour les images photographiques qui se trouvent entièrement dans la gamme de destination, il n'est pas recommandé pour le rendu général des images photographiques. Le mappage des couleurs hors gamut aux couleurs de la surface de gamut de destination peut entraîner des artefacts indésirables, tels que des irrégularités de ton ou de chroma dans des dégradés lisses qui franchissent la limite de gamut. BasicPhoto est recommandé pour les images photographiques. Si une image photographique ou contone nécessite un mappage de gamut autre que BasicPhoto, l'alternative doit consister à créer un module GMM plug-in implémentant ce mappage, au lieu d'utiliser MinCD.

Les couleurs de la gamme restent inchangées. Pour les couleurs hors gamut, la luminosité et la couleur sont ajustées en recherchant le point dans la gamme de

destination qui a la distance de couleur minimale par rapport aux points d'entrée hors gamut. La distance de couleur est calculée dans l'espace JCh. Toutefois, vous pondez différemment la distance en légèreté (J) et la distance en chroma (C) ou en teinte (h). Une fonction de poids dépendant de la chromaie est utilisée pour la distance en légèreté, de sorte que le poids est plus petit pour la petite chromatique et plus grand pour la grande chroma jusqu'à ce qu'un seuil chromatique soit atteint, après quoi le poids reste à 1, c'est-à-dire le même poids que la distance dans la chroma ou la teinte. Cela suit l'utilisation recommandée pour CMC et CIEDE2000. Il existe deux variantes : colorimétrique relative et colorimétrique absolue.

Colorimétrique relative : Tout d'abord, alignez les axes neutres source et de destination, comme décrit précédemment. Ensuite, attachez la couleur de source ajustée à la limite de la gamme de destination ajustée. (Voir la figure 4. Mappage chromatique le long d'une légèreté constante.) Réajustez les valeurs de l'appareil de destination comme décrit précédemment. Dans le cas d'une limite de gamme de destination monochrome, le découpage chromatique signifie que la valeur chromatique (C) est définie sur zéro (0,0).

Colorimétrique absolue : Cela est similaire à la colorimétrie relative, mais sans l'alignement de l'axe source et de destination neutre. La valeur source est directement coupée sur l'axe neutre de destination. Notez que si les limites de la gamme source et de destination sont monochromes, le comportement est identique à la variante colorimétrique relative ; autrement dit, l'alignement des axes neutres est effectué, puis la chroma est coupée à zéro. Cela garantit qu'une sortie raisonnable est atteinte même si les milieux et les colorants sont sensiblement différents.

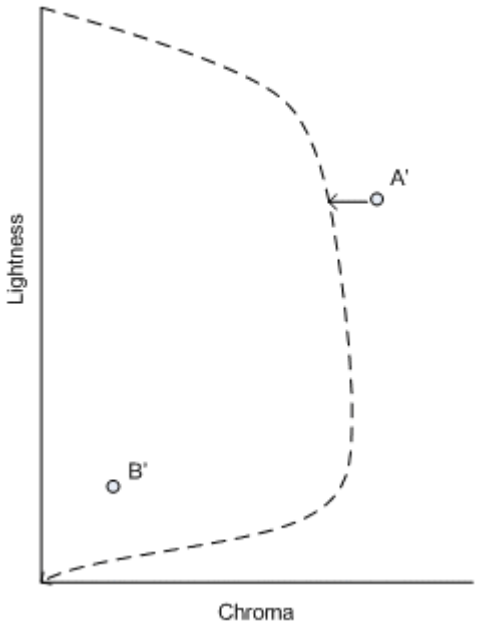


Figure 3 : Découpage minCD sur la gamme ajustée

BasicPhoto

Vue d'ensemble

BasicPhoto : équivalent à l'intention préférée, picturale ou perceptive de la CPI.

Cet algorithme est une variante du mappage de la luminosité sigmoidale et de la mise à l'échelle du genou du cusp (CUSP) qui sont décrits par CIE TC8-03 dans CIE156:2004. Cet algorithme de variante prend en charge les descripteurs de limites de gamut (GBD) avec des interpréteurs de commandes à deux gamuts ; c'est-à-dire des GBD avec un interpréteur de commandes de référence et un interpréteur de commandes plausible. L'algorithme SGCK, qui suppose à l'origine qu'un seul interpréteur de données gamut dans le GBD, est basé sur l'algorithme de SIG_KNEE (Braun, 1999), qui intègre le mappage de légèreté sigmoïdale et la mise à l'échelle de la fonction genoux proposées par Braun et Fairchild (1999), combiné avec la dépendance chromatique de GCUSP (Morovic, 1998). Il maintient la constante de teinte perçue, par exemple, l'angle de teinte dans jab corrigé de teinte, et utilise une échelle de légèreté sigmoïdale générique (indépendante de l'image) qui est appliquée de manière dépendante de la chromaie et d'une fonction de genou à 90 pour cent. La variante est mise à l'échelle le long de lignes de légèreté constante.

Cas de l'interpréteur de commandes à gamut unique

Il est utile de passer en revue l'algorithme dans le cas où les GBD source et de destination n'ont qu'un seul interpréteur de commandes gamut. Dans ce cas, l'algorithme se compose des calculs suivants.

Tout d'abord, effectuez un mappage de légèreté initiale à l'aide de la formule suivante :

$$J_R = (1 - p_C)J_O + p_C J_S \quad (1)$$

où J_o est la légèreté d'origine et J_R est la légèreté de reproduction.

$$p_C = 1 - ((C^3)/(C^3 + 5 \times 10^5))^{1/2} \quad (2)$$

Lorsque la limite de la gamme source est monochrome, la valeur chromatique est égale à zéro pour la limite monochrome en raison de l'alignement neutre de l'axe. Cela entraîne le cas dégénéré où C est égal à zéro. Dans ce cas, p_C est défini sur 1.

p_C est un facteur de pondération chroma-dépendant (Morovic, 1998) qui dépend de la chroma de la couleur d'origine, C et $_{JS}$ sont le résultat de la légèreté d'origine mappée à l'aide d'une fonction sigmoïdale.

Pour calculer $_{JS}$ (Braun et Fairchild, 1999), une table de recherche unidimensionnelle (LUT) entre les valeurs de légèreté d'origine et de reproduction est d'abord configurée sur la base d'une fonction normale cumulative discrète (S).

$$S_i = \sum_{n=0}^{n=i} \frac{1}{\sqrt{2\pi}\Sigma} e^{-\frac{\left(\frac{100n}{m} - x_0\right)^2}{2\Sigma^2}} \quad (3)$$

où x $_0$ et S sont respectivement la moyenne et l'écart-type de la distribution normale et *i* = 0,1,2... *m,m* est le nombre de points utilisés dans le LUT. S_i est la valeur de la fonction normale cumulative pour *i* /*m* pour cent. Les paramètres dépendent de la légèreté du point noir de la gamme de reproduction et peuvent être interpolés à partir du tableau 1. Pour plus d'informations sur le calcul de ces paramètres, voir Braun et Fairchild (1999, p. 391).

J $_{minOut}$

5,0

10.0

15.0

20,0

x $_0$

53.7

56.8

58.2

60.6

S

43,0

40,0

35,0

34,5

Tableau 1 : Calcul du paramètre de compression de légèreté BasicPhoto

Pour utiliser S comme LUT (S_{LUT}) de mappage de luminosité, il doit d'abord être normalisé dans la plage de luminosité de [0,100]. Les données normalisées sont ensuite mises à l'échelle dans la plage dynamique de l'appareil de destination, comme indiqué dans l'équation 4, où $J_{min\backslash Out}$ et $J_{max\backslash Out}$ sont les valeurs de légèreté du point noir et du point blanc du support de reproduction, respectivement.

$$S_{LUT} = \frac{(S_i - \min(S))}{(\max(S) - \min(S))}(J_{\max Out} - J_{\min Out}) + J_{\min Out} \quad (4)$$

À ce stade, les valeurs *JS peuvent* être obtenues à partir du $_{LUT}$ S en interpolant entre les points *m* des valeurs *J O*' et *J S* correspondantes qu'il contient, et en utilisant l'équation suivante comme entrée.

$$J_{O'} = 100(J_O - J_{\min In})/(J_{\max In} - J_{\min In}) \quad (5)$$

J_{minIn} est la valeur de légèreté du point noir du support d'origine, J_{maxIn} est la valeur de légèreté du point blanc du support d'origine et J_O est la légèreté d'origine. Pour référence ultérieure, vous pouvez désigner par *S* la fonction sigmoïdale définie de la manière qui vient d'être décrite, comme illustré dans la figure 4 suivante.

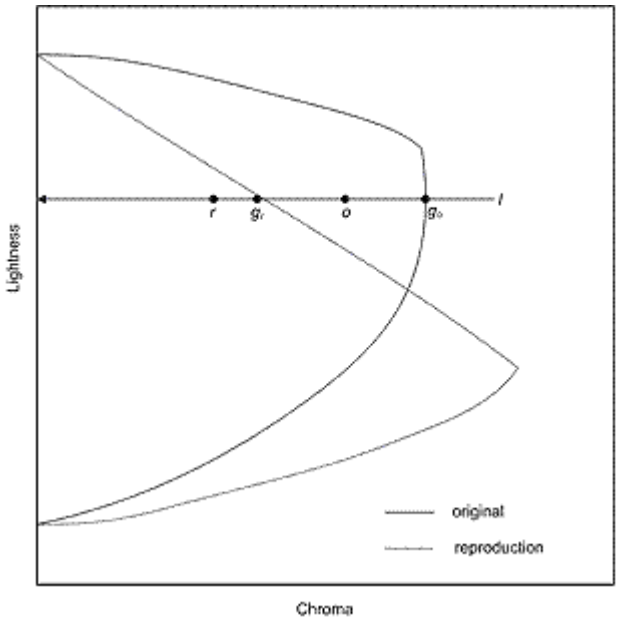


Figure 4 : Mappage chromatique le long d'une luminosité constante

Deuxièmement, si la limite de gamut de destination est chromatique, compressez la chroma le long des lignes de légèreté constante (l) et effectuez la compression comme suit.

$$d_r = \begin{cases} d_o; d_o \le 0.9d_{gr} \\ 0.9d_{gr} + (d_o - 0.9d_{gr})0.1d_{gr}/(d_{go} - 0.9d_{gr}); d_o > 0.9 \times d_{gr} \end{cases} \quad (6)$$

où *d* représente la distance par rapport à *E* sur *l*; *g* représente la limite de la gamme moyenne ; *r* représente la reproduction ; et *o* la figure 5 d'origine.

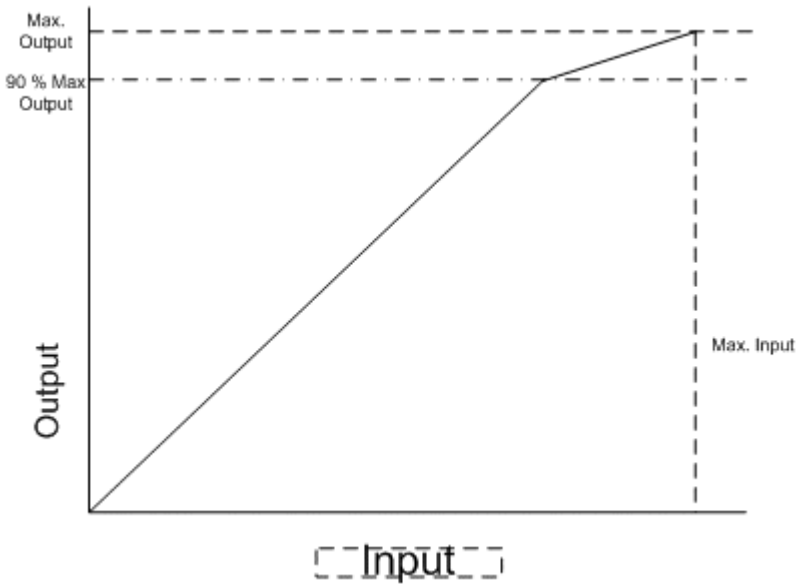


Figure 5 : Compression chromatique et légèreté dans BasicPhoto

Si la limite de gamut de destination est monochrome, la valeur de chroma est rogné sur zéro.

Troisièmement, utilisez un clip MinCD (décrit précédemment) pour éliminer toute erreur résiduelle.

Amélioration du noir

L'algorithme précédent peut être modifié pour améliorer le noir lorsque la destination est un périphérique d'imprimante. Le problème est lié au choix de J_{minOut}, qui ne correspond généralement pas à la couleur la plus foncée qu'une imprimante peut produire.

Plus précisément, la couleur avec la densité la plus élevée, obtenue en plaçant 100 % d'encres (ou la couverture maximale possible, si la limitation gcr/encre est appliquée), n'est généralement pas « neutre » dans l'espace d'apparence des couleurs. Voir la figure 6. En d'autres termes, si la légèreté minimale neutre est utilisée pour l'appareil de destination, le scaler de luminosité construit correspond à une légèreté minimale qui n'est pas la densité la plus élevée que peut atteindre l'imprimante. Considérez le cas d'usage ultérieur du moniteur à l'imprimante. Le moniteur noir, R=G=B=0, sera alors imprimé comme n'étant pas la densité la plus élevée. L'impact sur la qualité de l'image est qu'il y a un manque de profondeur et de contraste.

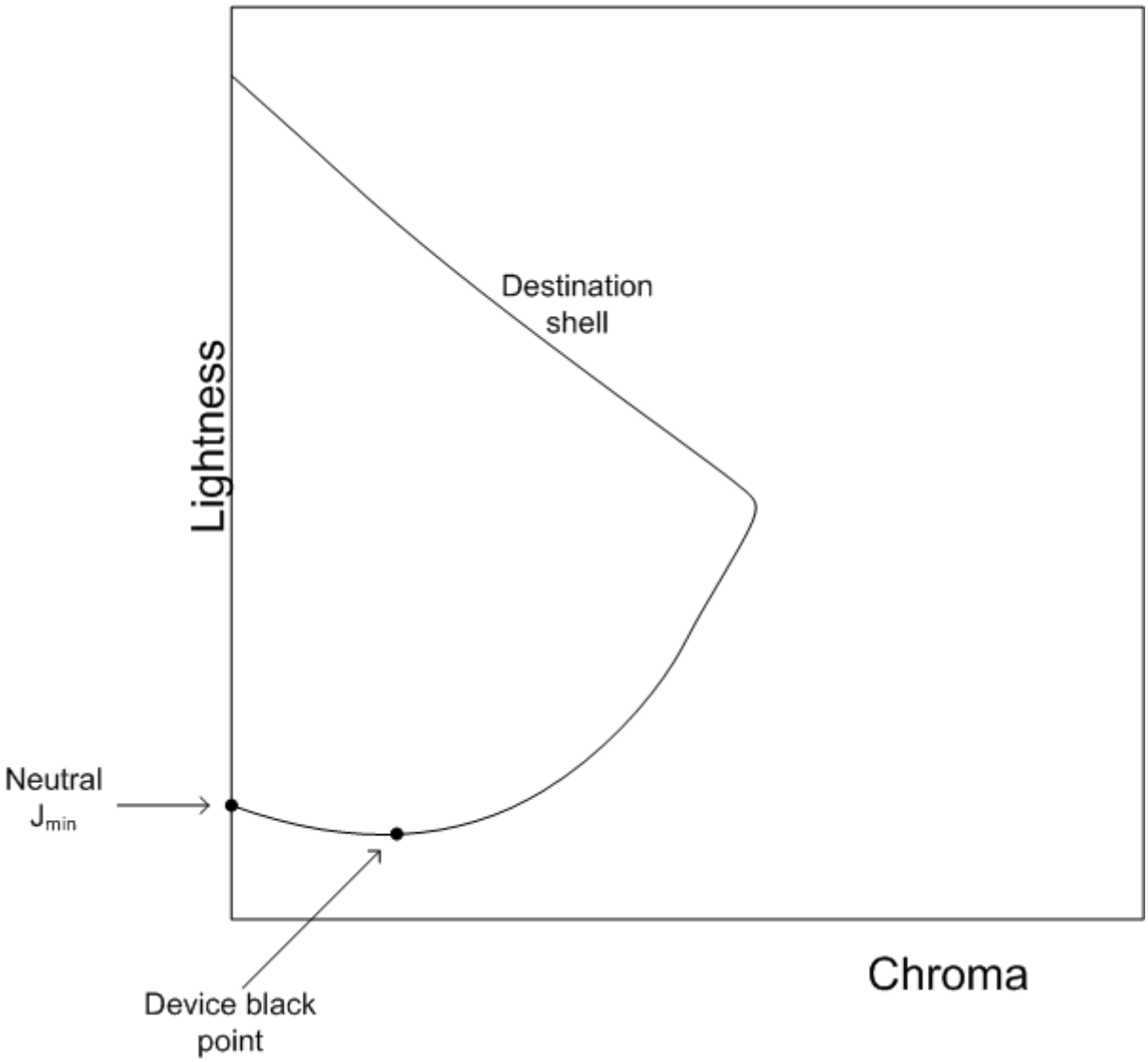


Figure 6 : Le point noir de l'appareil peut être plus foncé que la luminosité minimale neutre.

Supposons que la destination « point noir de l'appareil » est Jkakbk/JkCkh k. Si C k n'est pas zéro, le point noir de l'appareil n'est pas neutre par rapport à CAM02. Si vous utilisez J k pour la destination « légèreté minimale neutre » dans la construction du scaler de légèreté ; c'est-à-dire, définition

$J_{minOut} = Jk$

et l'appliquez à l'interpréteur de commandes de gamut source, vous obtenez la configuration illustrée dans la figure 7. Dans la figure, le plan de teinte correspond à h k.

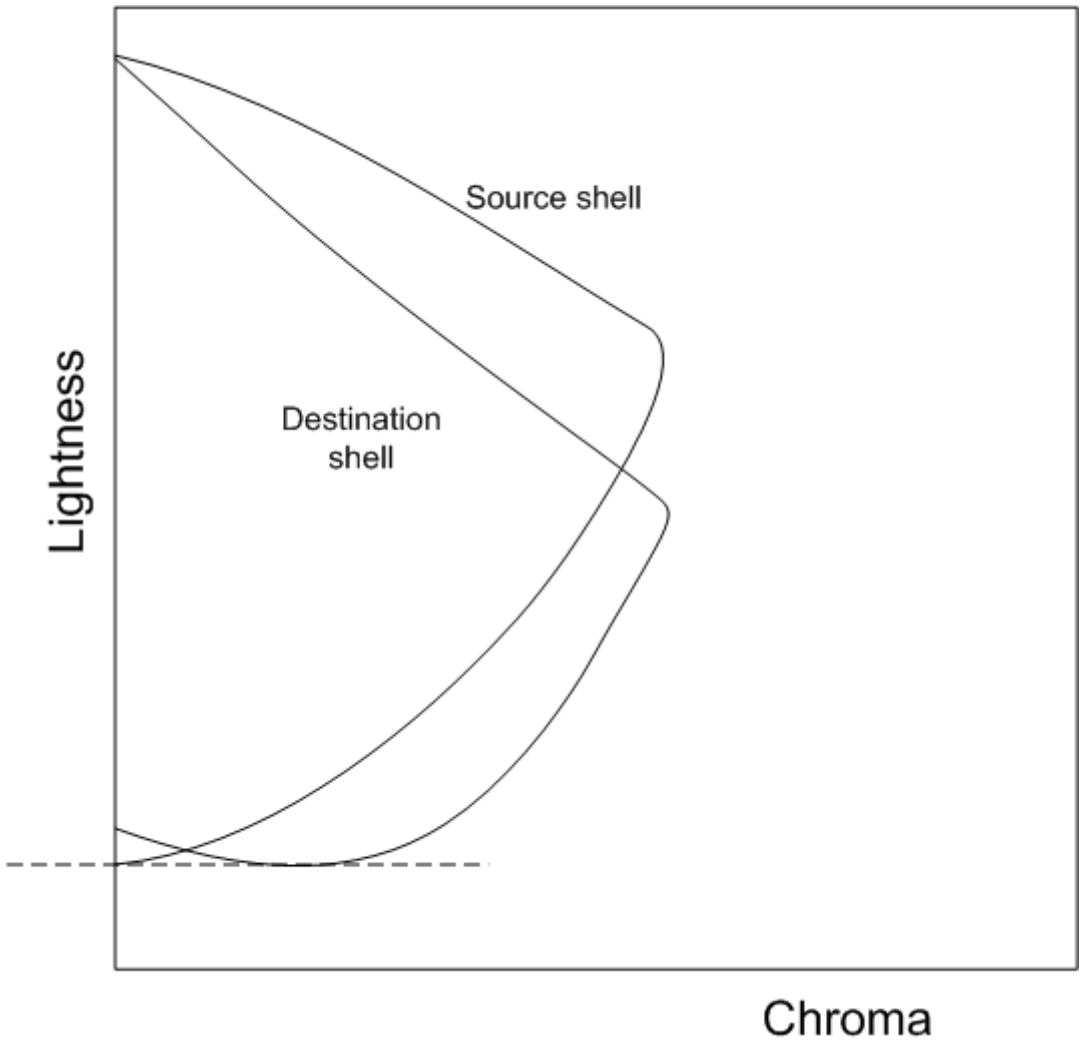


Figure 7 : Géométrie à l'aide du scaler de luminosité modifié avec point noir de l'appareil de destination

Pour permettre à l'algorithme de compression de chroma suivant de continuer, vous souhaitez aligner les luminosités maximale et minimale sur les interpréteurs de commandes source et de destination. Pour ce faire, vous pouvez ajuster l'interpréteur de commandes de destination entre $J_{\text{neutralMin}}$ et J k en déplaçant les points vers la gauche. De plus, cette transformation doit être appliquée sur l'ensemble de l'espace Jab, et pas seulement sur le plan de teinte correspondant à h k.

La transformation est

$$\chi(J,a,b) = \begin{cases} (J,a,b) & \text{if } J \geq J_{\text{neutralMin}} \\ (J, a - \dfrac{J_{\text{neutralMin}} - J}{J_{\text{neutralMin}} - J_k} a_k, b - \dfrac{J_{\text{neutralMin}} - J}{J_{\text{neutralMin}} - J_k} b_k) & \text{if } J < J_{\text{neutralMin}} \end{cases}$$

La figure 8 montre l'effet de la transformation.

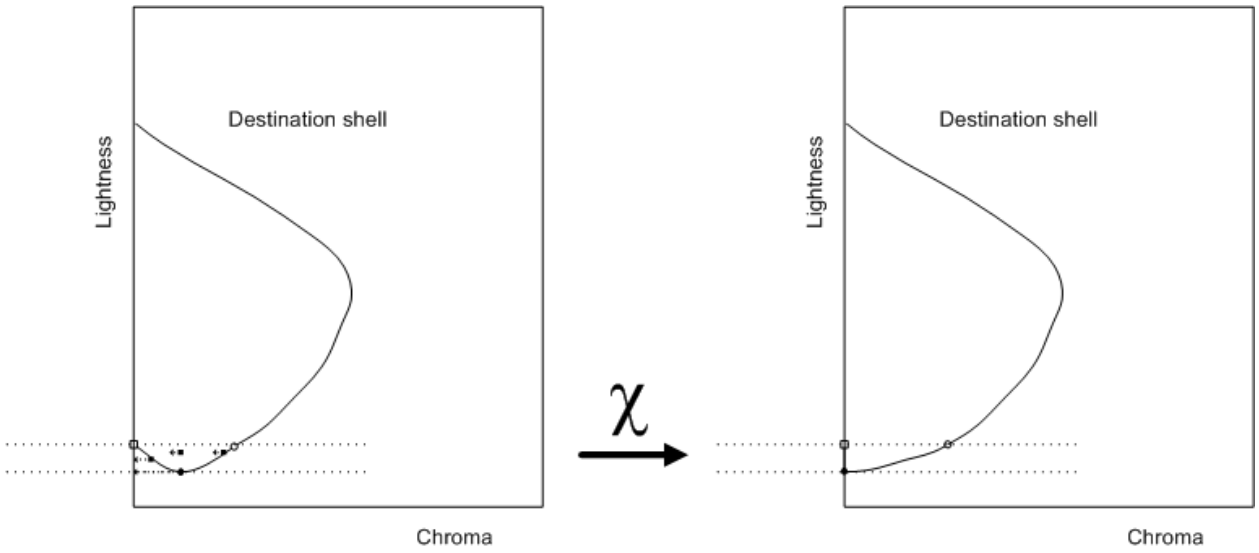


Figure 8 : Géométrie à l'aide du scaler de luminosité modifié avec point noir de l'appareil de destination

Après avoir appliqué l'algorithme de compression chromatique habituel, le point doit être « déplacé en arrière », c'est-à-dire que la transformation inverse doit être appliquée pour obtenir la couleur mappée finale.

$$J_S = \begin{cases} J_{r,\max,ref} + \dfrac{J_{r,\max,pla} - J_{r,\max,ref}}{J_{o,\max,pla} - J_{o,\max,ref}}(J_O - J_{o,\max,ref}) & \text{if } J_O \geq J_{o,\max,ref} \\ S(J_O; J_{o,\min,ref}, J_{o,\max,ref}; J_{r,\min,ref}, J_{r,\max,ref}) & \text{if } J_{o,\min,ref} < J_O < J_{o,\max,ref} \\ J_{r,\min,ref} + \dfrac{J_{r,\min,pla} - J_{r,\min,ref}}{J_{o,\min,pla} - J_{o,\min,ref}}(J_O - J_{o,\min,ref}) & \text{if } J_O \leq J_{o,\min,ref} \end{cases}$$

Le cas des interpréteurs de commandes à double gamut

L'objectif est de généraliser SIG_KNEE pour l'interpréteur de commandes à gamut unique au cas où le GBD de l'appareil source ou le GBD de l'appareil de destination a une structure à deux interpréteurs de commandes. L'interpréteur de commandes interne sera appelé l'interpréteur de commandes de référence, tandis que l'interpréteur de commandes externe sera appelé l'interpréteur de commandes plausible. Vous souhaitez prendre en compte les cas suivants.

(a) Le GBD source et le GBD de destination ont une structure à deux interpréteurs de commandes.

(b) Le GBD source a une structure à deux interpréteurs de commandes ; le GBD de destination n'a qu'un seul interpréteur de commandes.

(c) Le GBD source n'a qu'un seul interpréteur de commandes ; le GBD de destination a une structure à deux interpréteurs de commandes.

(d) Le GBD source et le GBD de destination n'ont qu'un seul interpréteur de commandes.

Le cas (d) est le cas de l'interpréteur de commandes à gamut unique qui a été abordé précédemment. Pour les cas (a), (b) et (c), vous pouvez généraliser la mise à l'échelle de légèreté pour utiliser les informations supplémentaires de la structure du double interpréteur de commandes. Dans les cas (b) et (c) où la source ou la destination n'a qu'un seul interpréteur de commandes, vous introduisez un « interpréteur de commandes de référence induit » qui sera abordé dans une section suivante, « Interpréteur de commandes de référence induit ». L'algorithme général pour deux interpréteurs de commandes sera décrit pour le cas (a). Une fois la construction de l'interpréteur de commandes de référence induit expliquée, l'algorithme peut également être appliqué aux cas (b) et (c). Comme pour la compression chromatique, le taux de compression est déterminé par les plus grandes coques disponibles. En d'autres termes, si l'interpréteur de commandes plausible et l'interpréteur de commandes de référence sont disponibles, l'interpréteur de commandes plausible sera utilisé ; sinon, l'interpréteur de commandes de référence sera utilisé.

Mise à l'échelle de la légèreté généralisée

L'existence de deux interpréteurs de commandes pour le GBD source et le GBD de destination signifie que vous devez mapper un ensemble de quatre points du GBD source à un ensemble correspondant dans le GBD de destination.

$$(J_{o,\min,pla}, J_{o,\min,ref}, J_{o,\max,ref}, J_{o,\max,pla}) \rightarrow (J_{r,\min,pla}, J_{r,\min,ref}, J_{r,\max,ref}, J_{r,\max,pla})$$

Les indices ont les significations suivantes.

o ou r : « original » (source) ou « reproduction » (destination)

min ou max : luminosité neutre minimale ou luminosité neutre maximale

pla ou ref : Interpréteur de commandes plausible ou interpréteur de commandes de référence

L'ordre de chaque quadruplé est également l'ampleur relative attendue de ces points.

La carte Lightness Rescaling utilise les mêmes deux premières équations que l'interpréteur de commandes unique, mais *JS* est défini de manière fragmenté comme suit.

$$J_S = \begin{cases} J_{r,\max,ref} + \dfrac{J_{r,\max,pla} - J_{r,\max,ref}}{J_{o,\max,pla} - J_{o,\max,ref}} (J_O - J_{o,\max,ref}) & \text{if } J_O \geq J_{o,\max,ref} \\ S(J_O; J_{o,\min,ref}, J_{o,\max,ref}; J_{r,\min,ref}, J_{r,\max,ref}) & \text{if } J_{o,\min,ref} < J_O < J_{o,\max,ref} \\ J_{r,\min,ref} + \dfrac{J_{r,\min,pla} - J_{r,\min,ref}}{J_{o,\min,pla} - J_{o,\min,ref}} (J_O - J_{o,\min,ref}) & \text{if } J_O \leq J_{o,\min,ref} \end{cases}$$

(7)

En d'autres termes, il est sigmoïdal dans l'interpréteur de commandes de référence et linéaire à l'extérieur. Voir figure 9.

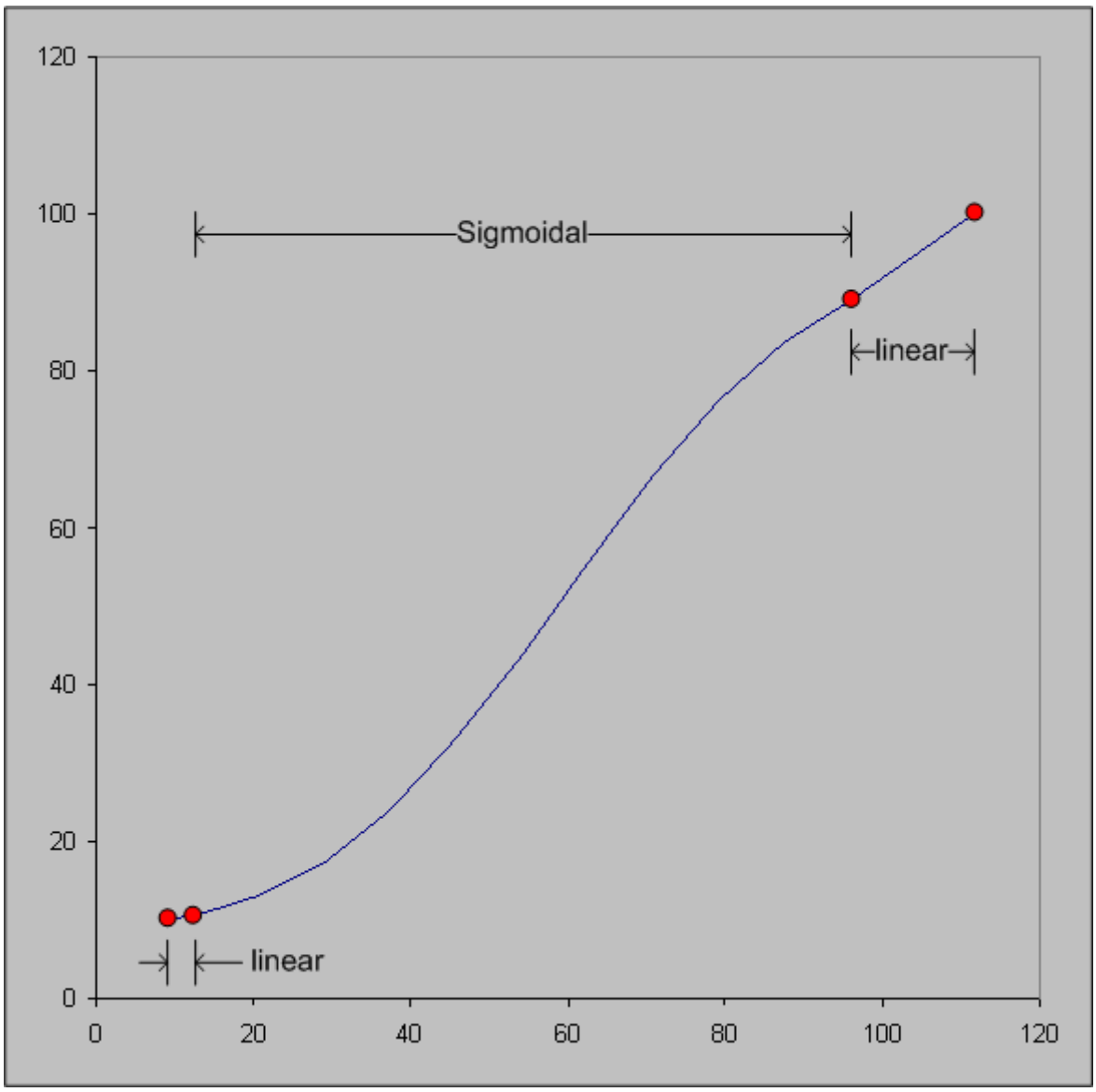


Figure 9 : Fonction de mise à l'échelle de la légèreté pour les GBD à deux shells

INTERPRÉTEUR DE COMMANDES DE RÉFÉRENCE INDUIT

Si un GBD a un interpréteur de commandes et l'autre GBD a deux interpréteurs de commandes, vous devez créer un « interpréteur de commandes de référence » pour le GBD avec un seul interpréteur de commandes. L'interpréteur de commandes existant,

qui serait appelé Interpréteur de commandes de référence, sera remplacé par « Interpréteur de commandes plausible ». En fait, vous n'avez pas besoin de créer un interpréteur de commandes dans l'espace Jab complet. Étant donné que la mise à l'échelle légère utilise uniquement *J max* et *J min*, vous n'avez qu'à créer ces valeurs pour l'interpréteur de commandes de référence induit. Il existe deux cas, selon le GBD qui a deux interpréteurs de commandes.

Cas 1 : Le GBD source a deux interpréteurs de commandes ; le GBD de destination a un interpréteur de commandes.

Déterminer l'interpréteur de commandes de référence induit par la destination sur l'axe neutre ; c'est-à-dire les références $J_{r,\ min,\ ref}$ et $J_{r,\ max,\ de}$ l'interpréteur de commandes. Pour ce faire, utilisez l'algorithme suivant.

$$J_{r,min,ref} = \frac{J_{r,min,pla} + J_{r,max,pla}}{2} + \lambda_{low}\left(J_{r,min,pla} - \frac{J_{r,min,pla} + J_{r,max,pla}}{2}\right)$$

$$J_{r,max,ref} = \frac{J_{r,min,pla} + J_{r,max,pla}}{2} + \lambda_{high}\left(J_{r,max,pla} - \frac{J_{r,min,pla} + J_{r,max,pla}}{2}\right)$$

Les facteurs ? $_{bas}$ et ? $_{les\ hauts}$ contrôlent la séparation entre l'interpréteur de commandes plausible et l'interpréteur de commandes de référence. La valeur 1 signifie que les valeurs J_{min} ou $J\ m_{ax}$ valeurs coïncident. Leurs valeurs sont « induites » à partir de l'interpréteur de commandes de référence source et de l'interpréteur de commandes plausible source.

$$J_{mid} = \frac{J_{o,min,ref} + J_{o,max,ref}}{2}$$

$$\lambda_{low} = F_{low} + (1 - F_{low})\frac{J_{mid} - J_{o,min,ref}}{J_{mid} - J_{o,min,pla}}$$

$$\lambda_{high} = F_{high} + (1 - F_{high})\frac{J_{o,max,ref} - J_{mid}}{J_{o,max,pla} - J_{mid}}$$

Les « facteurs fudge » F_{low} et F_{high} sont *des paramètres paramétrables* qui doivent se situer entre 0 et 1. Si la valeur est 0, les J_{min} ou $J\ m_{ax}$ sont directement induits à partir des interpréteurs de commandes source. Dans ce cas, choisissez F_{faible} = 0,95 et $F_{élevé}$ = 0,1.

Cas 2 : Le GBD source a un interpréteur de commandes ; le GBD de destination a deux interpréteurs de commandes.

Déterminer l'interpréteur de commandes de référence induit par la source sur l'axe neutre ; $_{c'est\text{-}à\text{-}dire\ les\ valeurs\ J\ o,\ min,\ ref}$ et $J_{o,\ max,\ de}$ l'interpréteur de commandes. Pour ce faire, utilisez l'algorithme suivant.

$$J_{o,min,ref} = \frac{J_{o,min,pla} + J_{o,max,pla}}{2} + \lambda_{low}\left(J_{o,min,pla} - \frac{J_{o,min,pla} + J_{o,max,pla}}{2}\right)$$

$$J_{o,max,ref} = \frac{J_{o,min,pla} + J_{o,max,pla}}{2} + \lambda_{high}\left(J_{o,max,pla} - \frac{J_{o,min,pla} + J_{o,max,pla}}{2}\right)$$

Là encore, les facteurs ? $_{bas}$ et ? $_{high}$ contrôle la séparation entre l'interpréteur de commandes plausible et l'interpréteur de commandes de référence. La valeur 1 signifie que les valeurs J $_{min}$ ou J m_{ax} coïncident. Leurs valeurs sont « induites » à partir de l'interpréteur de commandes de référence source et de l'interpréteur de commandes plausible source :

$$J_{mid} = \frac{J_{r,min,ref} + J_{r,max,ref}}{2}$$

$$\lambda_{low} = \frac{J_{mid} - J_{r,min,ref}}{J_{mid} - J_{r,min,pla}}$$

$$\lambda_{high} = \frac{J_{r,max,ref} - J_{mid}}{J_{r,max,pla} - J_{mid}}$$

Raisons des modifications apportées aux recommandations du CIE TC8-03

BasicPhoto diffère des recommandations CIE TC8-03 des manières suivantes.

1. La chroma n'est pas compressée vers le cusp, mais sur des lignes de légèreté constante.
2. La plage de luminosité utilise la légèreté de la couleur la plus foncée de la gamme plutôt que le point auquel la limite de gamut traverse l'axe neutre.
3. BasicPhoto prend en charge à la fois un interpréteur de commandes de gamut de référence et un interpréteur de commandes de gamut plausible, si l'une ou l'autre limite de gamut dans la transformation a deux interpréteurs de commandes.
4. BasicPhoto utilise CIECAM02 ; au lieu d'utiliser CIECAM97s pour convertir en D65 à 400 cd/m2, puis d'utiliser l'espace de couleur IPT RIT.

La première modification a été apportée pour éviter les problèmes d'inversion de tonalité qui peuvent se produire lors de l'utilisation de la compression vers un cusp. Comme le montre la figure 10, la compression du cusp peut provoquer des inversions de tonalité. Cela peut se produire lorsque les couleurs de haute couleur sont plus claires que les couleurs de couleur inférieure. Étant donné que SGCK compresse chaque pixel indépendamment en légèreté et en couleur, il n'est pas garanti de conserver la relation de légèreté entre les valeurs de pixels après la compression. L'inconvénient bien connu de cette décision de compresser sur des lignes de légèreté constante est que vous

pouvez subir des pertes de chroma, en particulier dans les zones où la limite de gamut de destination est très plate, comme c'est le cas avec les jaunes brillants.

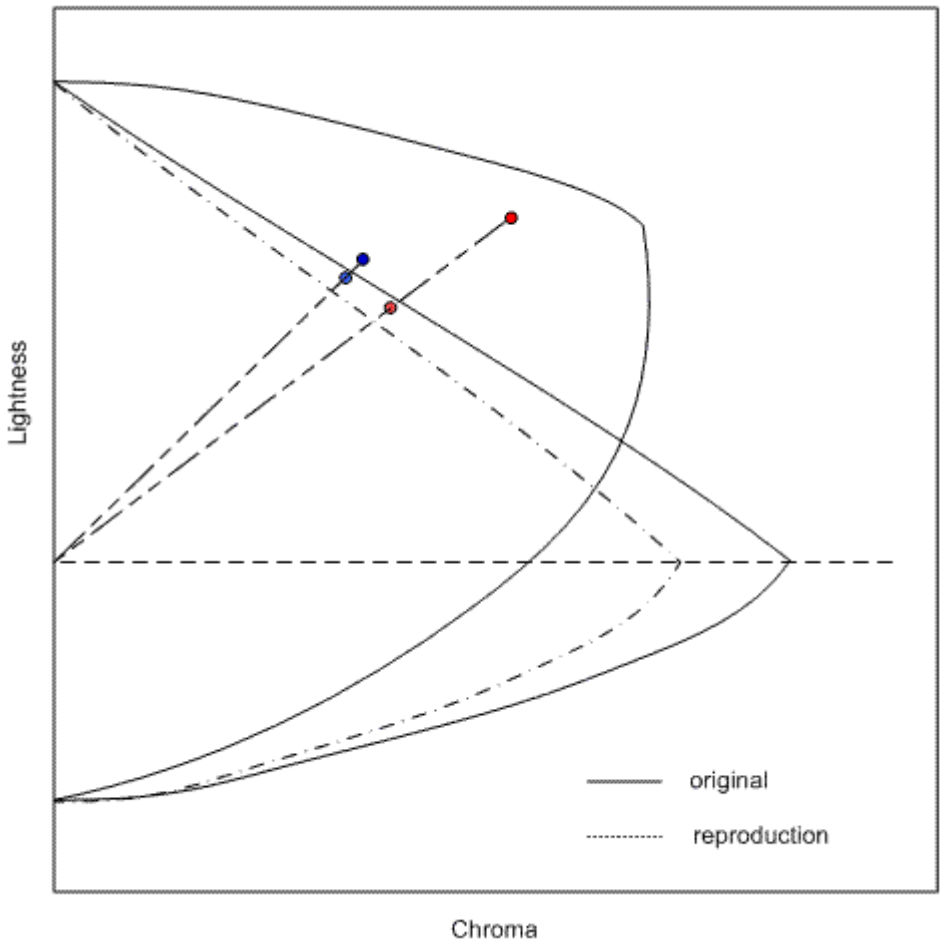


Figure 10 : Inversion de tonalité provoquée par SGCK

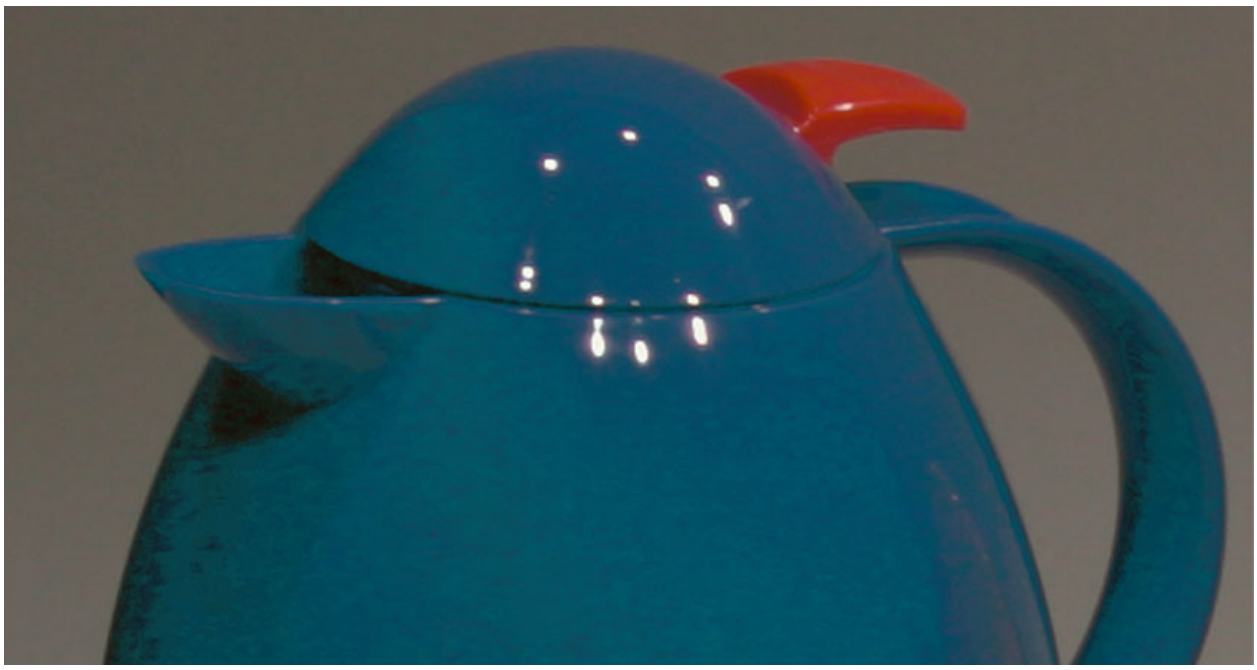


Figure 11 : image d'origine, résultat SGCK et résultat BasicPhoto

La figure 11 illustre cette inversion de tonalité. À gauche se trouve une image originale capturée par un appareil photo numérique ; au centre, l'image telle que reproduite par SGCK ; et à droite, l'image telle que reproduite par BasicPhoto. L'image à gauche se trouve dans l'espace de couleurs de l'appareil photo numérique, les images centre et droite se trouvent dans l'espace de couleurs d'un écran vidéo LCD. Dans l'image d'origine, la partie supérieure de la théière est plus foncée que le bas, car le bas reflète la nappe sur laquelle elle est assise. Dans l'image SGCK, la partie supérieure est en fait plus légère que la partie inférieure, en raison de l'inversion de tonalité. En outre, il est difficile de voir les articles reflétés dans la partie inférieure de la théière. À droite,

BasicPhoto a fixé l'inversion de ton et les articles réfléchis sont plus clairement reconnaissables.

La deuxième modification a été apportée pour améliorer la reproduction des couleurs presque noires sur les imprimantes où le noir le plus noir ne tombe pas directement sur l'axe neutre CIECAM02. La figure 12 suivante montre une image convertie en sRGB ; reproduit pour une imprimante jet d'encre RVB à l'aide de SGCK ; et reproduit pour la même imprimante à l'aide de BasicPhoto. L'image au centre n'utilise pas le noir complet de l'appareil, et elle n'a donc pas le contraste observé dans l'original. Le contraste est restauré avec BasicPhoto.

Figure 12 : Noir amélioré

La troisième modification a été apportée pour améliorer la reproduction des couleurs pour les appareils photo numériques. En particulier dans les cas où l'appareil photo numérique a été profilé à l'aide d'une cible de référence, une description de limite de gamut créée à partir de couleurs mesurées peut ne pas inclure toutes les couleurs qui peuvent être capturées dans une scène du monde réel. Au lieu de couper toutes les couleurs sur la gamut de la cible de couleur mesurée, vous autorisez l'extrapolation à produire une limite de gamut plausible. L'algorithme BasicPhoto est conçu pour prendre en charge une telle limite de gamuts extrapolée.

La quatrième modification a été apportée, car CIECAM02 fonctionne bien pour le mappage de gamuts. Le processus recommandé par TC8-03 pour convertir les couleurs des appareils en D65 à 400 cd/m2, puis utiliser l'espace de couleurs RIT IPT est à la fois gourmand en calculs et chronophage.

Mappage Hue

HueMap est l'équivalent de l'intention de saturation ICC.

Si la limite de gamut source ou la limite de gamut de destination ne contient pas de valeurs primaires, ce modèle revient au modèle MinCD (relatif) décrit dans une section précédente ; par exemple, les appareils pour lesquels les primaires ne peuvent pas être déterminées (profils ICC avec plus de quatre canaux) ou les profils ICC monochromes.

Cet algorithme ajuste d'abord la teinte de la valeur de couleur d'entrée. Ensuite, il ajuste simultanément la légèreté et la chromaité, à l'aide d'un mappage de cisaillement. Enfin, il clipse la valeur de couleur pour s'assurer qu'elle se trouve dans gamut.

La première étape consiste à déterminer les « roues Hue ». Recherchez les valeurs JCh pour les couleurs primaires et secondaires pour l'appareil source et l'appareil de destination. Vous n'envisagez que les composants hue. Il en résulte une roue de teinte primaire ou secondaire avec six points de couleur pour chaque appareil. (Voir la figure 13.)

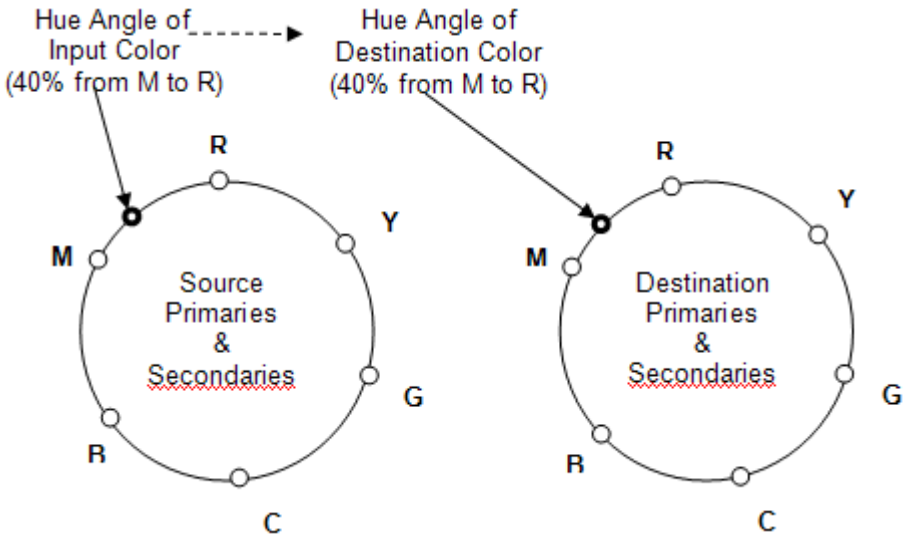


Figure 13 : Roues Hue

De meilleurs résultats peuvent être obtenus si le principal bleu source n'est pas pivoté vers le principal bleu de destination. Au lieu de cela, l'angle de teinte primaire bleu source est utilisé comme angle de teinte primaire bleu de destination.

Ensuite, effectuez les rotations de teinte pour chaque couleur d'entrée à partir de l'image source,

a)À l'aide de l'angle de teinte de la couleur d'entrée, déterminez l'emplacement de la couleur sur la roue de teinte source par rapport aux deux couleurs primaires ou secondaires adjacentes. L'emplacement peut être considéré comme un pourcentage de la distance entre les primaires. Par exemple, la teinte de couleur d'entrée est de 40 % du chemin entre la valeur de teinte de Magenta et la valeur de teinte rouge.

b)Sur la roue de teinte de destination, recherchez l'angle de teinte associé, par exemple, de 40 % de Magenta à Rouge. Cette valeur sera l'angle de teinte de destination.

En général, les primaires sources et les secondaires ne seront pas aux mêmes angles de teinte que les primaires et les secondaires de destination; autrement dit, l'angle de teinte de destination sera généralement différent de l'angle de teinte source.

Par exemple, supposons que les roues hue produisent les valeurs suivantes :

Source M = 295 degrés, Source R = 355 degrés.

Destination M = 290 degrés, Destination R = 346 degrés.

Si l'angle de teinte de la couleur d'entrée est de 319 degrés, il correspond à 40 % de l'angle (24 degrés) de la source M à la source R. L'angle de M à R est de 60 degrés et l'angle de M à la teinte d'entrée est de 24 degrés. Calculez l'angle de la destination qui est de 40 % de la destination M à la destination R (22 degrés), de sorte que l'angle de teinte de la couleur de destination est de 312 degrés.

Ensuite, calculez les points de référence hue pour la teinte source et la teinte de destination. Pour calculer le point de référence de teinte pour une valeur h (hue) particulière, vous souhaitez rechercher la valeur J (légèreté) et la valeur C (chroma).

- Recherchez la valeur J du point de référence de teinte en interpolant entre les valeurs J des points primaires ou secondaires adjacents, à l'aide de la position relative de la teinte ; par exemple, 40 % dans cet exemple.
- Recherchez la valeur C maximale à cette valeur J et h. Vous avez maintenant le JCh du point de référence hue pour cette teinte.

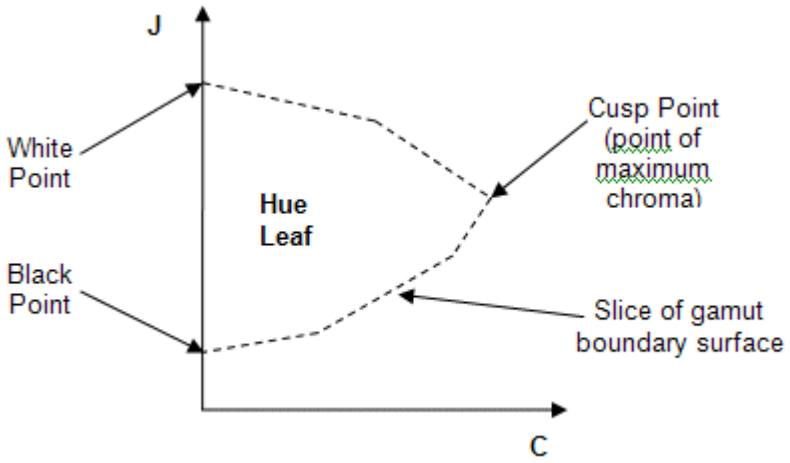


Figure 14 : Feuille de teinte (visualisation d'une tranche de limite de gamut à une teinte spécifique)

L'étape suivante consiste à calculer le mappage de cisaillement pour chaque pixel. Tout d'abord, visualisez une feuille de teinte à partir du gamut source pour l'angle de teinte de la couleur source et une feuille de teinte du gamut de destination pour l'angle de teinte de destination calculé pendant la rotation de teinte. Les feuilles de teinte sont créées en prenant une « tranche » à partir de la surface limite du gamut JCh à un angle de teinte spécifique (voir figure 14).

REMARQUE : Pour des raisons d'optimisation des performances, les feuilles hue ne sont pas réellement créées ; ils sont décrits et affichés ici à des fins de visualisation uniquement. Les opérations sont effectuées directement sur la surface de limite de gamut à la teinte spécifiée. Vous calculez ensuite les points de référence hue pour déterminer le mappage de cisaillement.

- Effectuez une mise à l'échelle légère pour mapper les points noir et blanc de la feuille source à la feuille de destination (voir figure 15). Les points noir et blanc de la feuille de teinte source sont mappés linéairement aux points noir et blanc de la feuille de teinte de destination, en mettant à l'échelle toutes les coordonnées J de la limite source. La valeur de couleur d'entrée mappée est mise à l'échelle de la même manière.

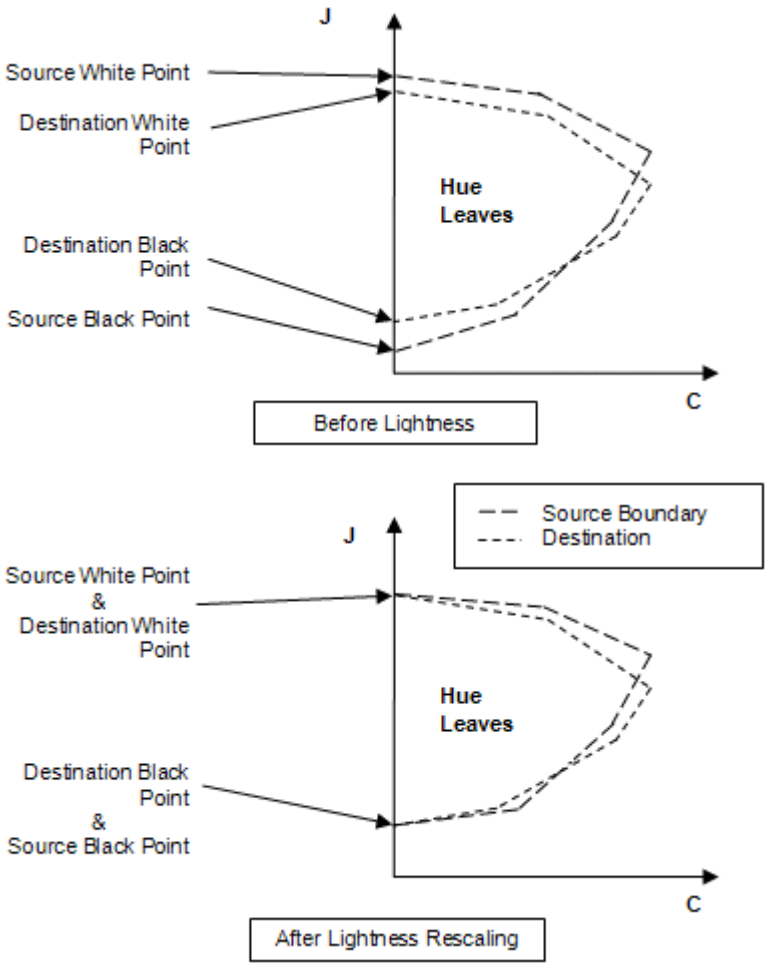


Figure 15 : Mappage de luminosité

- Déterminez les points de référence de teinte pour chaque feuille de teinte. Appliquez un mappage de cisaillement à la feuille source afin que les points de référence source et de destination coïncident (voir la figure 16). Le point de référence d'un gamut à une teinte spécifique est le point de référence de teinte interpolé entre les primaires adjacentes. Le point de référence de la feuille de teinte source est mappé linéairement au point de référence de la feuille de teinte de destination, à l'aide d'une opération de cisaillement qui verrouille l'axe J, en

gardant les points noirs et les points blancs à l'arrêt. Les points noirs, les points blancs et les points de référence des feuilles de teinte source et de destination doivent coïncider.
- Appliquez le mappage de cisaillement à la valeur de couleur d'entrée ajustée en légèreté. Les coordonnées J et C de la valeur de couleur source sont mises à l'échelle proportionnellement, par rapport à sa distance par rapport à l'axe J.
- Ensuite, une compression subtile de la légèreté chromao-dépendante vers la valeur J du point de référence de teinte est effectuée sur le point de couleur mappé de cisaillement. La compression vers la référence de teinte J est effectuée de manière gamma, où le blanc, le noir, les gris et les points sur la référence de teinte J ne sont pas affectés. Tous les autres points tendent vers la référence de teinte J d'une manière lisse, se regroupe légèrement près de la référence de teinte J, la chroma restant constante. La dépendance chromatique garantit que les couleurs neutres ne sont pas affectées et que l'effet est accru sur les couleurs ayant une couleur plus élevée.

Voici une description mathématique de la compression de légèreté vers la référence de teinte J, ou de l'ajustement de la valeur J du point de destination. Il est appelé point de destination, car il a été mappé au gamut de destination.

Tout d'abord, calculez « factorC » (facteur de dépendance chromatique) pour le point de destination, ce qui détermine l'effet de la compression de légèreté. Les points proches ou sur l'axe J auront peu ou pas de compression ; les points plus éloignés de l'axe J (haute chromaie) auront plus de compression appliquée. Multipliez par 0,5 pour vous assurer que factorC est inférieur à 1, car il est possible que sourceC soit légèrement supérieur à référenceC, mais pas deux fois plus grand.

factorC = (destinationC / referenceC) ? 0.5

où :

destinationC est la valeur C du point de destination.

referenceC est la valeur C du point de référence Hue.

Ensuite, déterminez si le point de destination J est au-dessus ou au-dessous de la référence de teinte J. S'il est ci-dessus, procédez comme suit :

1. Calculez « factorJ » pour le point de destination par rapport à la référenceJ. Cette valeur factorJ sera comprise entre 0 et 1 (0 si sur la référenceJ; 1 si à maxJ).

2. factorJ = (destinationJ - référenceJ) / (maxJ - referenceJ)

   où :

destinationJ est la valeur J du point de destination.

referenceJ est la valeur J du point de référence hue.

maxJ est la valeur J maximale du gamut.

3. Appliquez une fonction d'alimentation de type gamma à factorJ, ce qui réduit factorJ d'une certaine quantité. Cet exemple utilise la puissance de 2 (le carré). Soustrayez le facteur J réduit du facteur J d'origine et multipliez le résultat par la plage J totale au-dessus de la référence primaireJ pour rechercher le « deltaJ », qui représente la modification de J après la compression de légèreté, sans inclure la dépendance chromatique.

4. deltaJ = (factorJ - (factorJ ? factorJ)) ? (maxJ - referenceJ)

5. Appliquez factorC au deltaJ (plus la chroma est élevée, plus l'effet est important) et calculez la nouvelle valeur J pour le point de destination.

6. destinationJ = destinationJ - (deltaJ ? factorC)

Si la valeur J pour le point de destination se trouve sous référenceJ, un calcul similaire aux étapes précédentes A-C est effectué, en utilisant minJ au lieu de maxJ pour trouver la plage dans J pour calculer le facteurJ, et en tenant compte de la polarité des opérations « en dessous » de la référenceJ.

factorJ = (referenceJ - destinationJ) / (referenceJ - minJ)

deltaJ = (factorJ - (factorJ ? factorJ)) ? (referenceJ - minJ)

destinationJ = destinationJ + (deltaJ ? factorC)

où :

minJ est la valeur J minimale du gamut.

La chroma pour les points de couleur d'entrée est développée linéairement (si possible) le long de la luminosité constante proportionnelle à la valeur de chroma maximale des gamuts source et de destination à cette teinte et à cette légèreté. Combiné à la compression de légèreté dépendante des chromas précédente, cela permet de préserver la saturation, car le mappage de cisaillement à l'aide des points de référence entraîne parfois une compression excessive du point source dans la chroma (voir la figure 16).

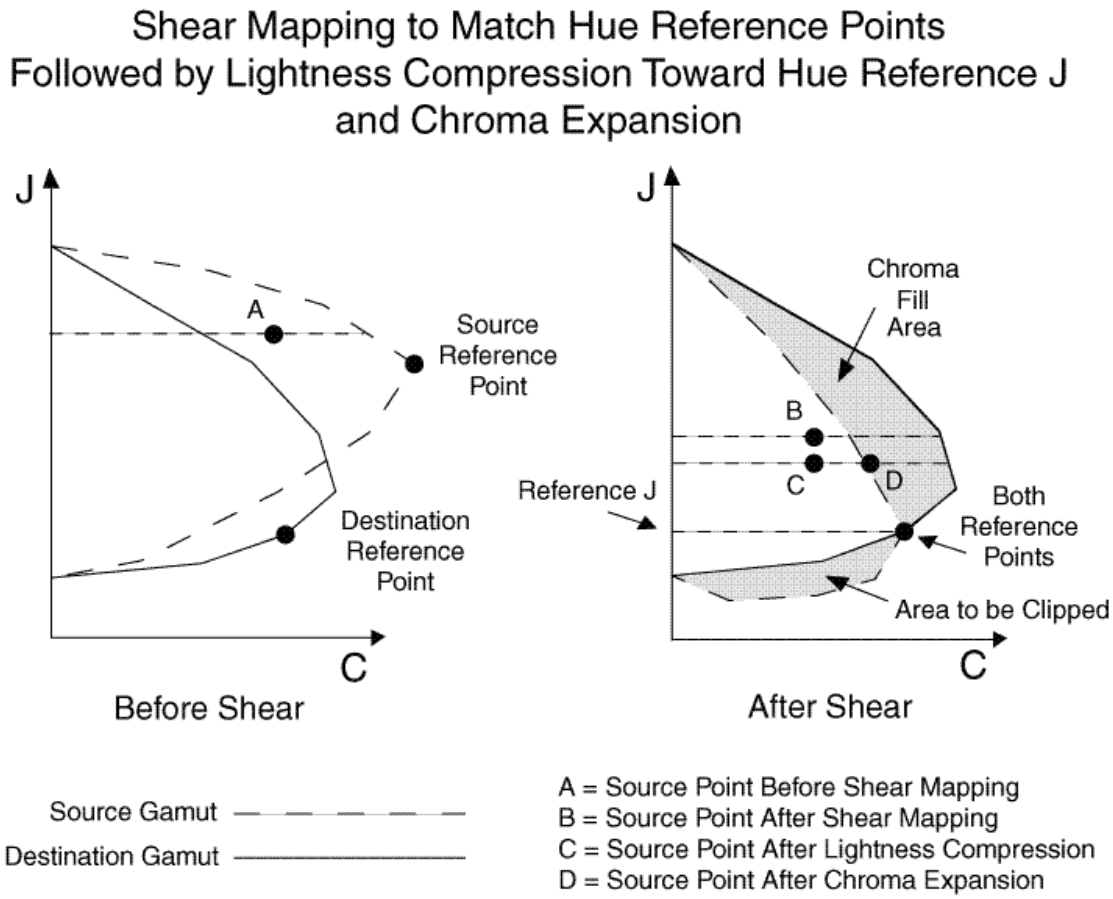


Figure 16 : Mappage de cisaillement, compression de la légèreté vers la référence de teinte J et expansion de la chroma

Voici une description mathématique du processus d'expansion chromatique ou l'ajustement de la valeur C du point de destination. Il est appelé point de destination, car il a été mappé et la légèreté compressée dans le gamut de destination.

1. Avant le mappage de cisaillement, déterminez sourceExtentC (l'étendue chromatique à la luminosité et à la teinte du point source).

2. Après le mappage de cisaillement et la compression de légèreté qui transforme le point source en point de destination, déterminez le destExtentC (l'étendue chromatique à la luminosité et à la teinte du point de destination).

3. Si la valeur sourceExtentC est supérieure à la valeur destExtentC, aucun ajustement chromatique du point de destination n'est nécessaire et vous pouvez ignorer l'étape suivante.

4. Ajustez destinationC (la chroma du point de destination) pour qu'il s'adapte à l'étendue chromatique de destination à cette légèreté et à cette teinte.

5. destinationC = destinationC ? (destExtentC / sourceExtentC)

   où :

destinationC est la valeur C du point de destination.

sourceExtentC est la valeur C maximale du gamut source à la luminosité et à la teinte du point source.

destExtentC est la valeur C maximale du gamut de destination à la luminosité et à la teinte du point de destination.

Enfin, effectuez le découpage de distance mimimum. Si la couleur d'entrée pivotée, ajustée à la légèreté et mappée au cisaillement est toujours légèrement en dehors de la plage de destination, attachez-la (la déplacer) au point le plus proche de la limite de gamut de destination (voir figure 17).

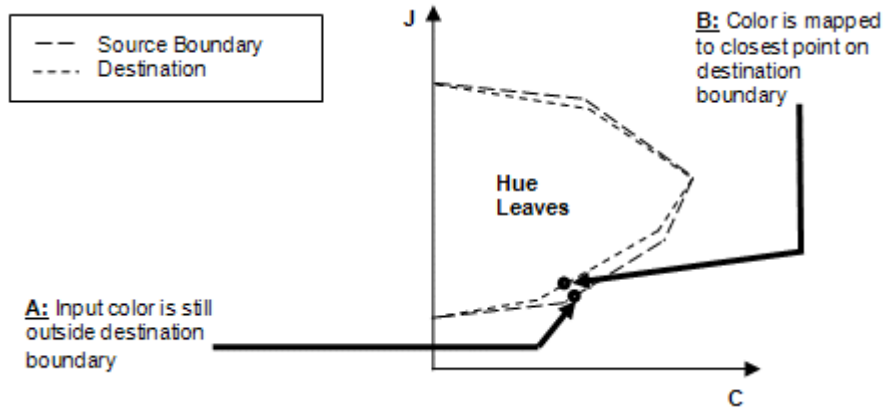


Figure 17 : Découpage de distance minimale

Description de la limite de gamut et algorithmes d'interpréteur de commandes Gamut

La fonction de limite de gamut d'appareil utilise le moteur de modèle d'appareil et les paramètres analytiques pour dériver une limite de gamut d'appareil de couleur, décrite comme une liste de vertex indexées de la coque de la gamut de l'appareil. La coque est calculée différemment selon que vous travaillez avec des appareils additifs, tels que des moniteurs et des projecteurs, ou des appareils soustractifs. La liste de vertex indexée est stockée dans CIEJab. La structure de la liste de vertex indexées est optimisée pour l'accélération matérielle par DirectX.

Cette approche propose de nombreuses solutions connues. Si vous recherchez « directX de coque convexe » sur le Web, vous obtenez plus de 100 coups. Par exemple, il existe une référence de 1983 sur ce sujet spécifique (Computer Graphics Theory and Application, « Shiphulls, b-spline surfaces, and cadcam », p. 34-49) avec des références datant de 1970 à 1982 sur le sujet.

La collection de points peut être déterminée à partir d'informations disponibles en externe, comme suit :

- Les points de l'interpréteur de commandes de référence pour les moniteurs sont générés à l'aide d'un échantillonnage du cube de couleur dans l'espace colorant de l'appareil.
- Des points pour l'interpréteur de commandes de référence pour les imprimantes et les périphériques de capture sont obtenus à partir des exemples de données utilisées pour initialiser le modèle.
- Les points de l'interpréteur de commandes de référence pour scRGB et sRGB sont générés à l'aide d'un échantillonnage du cube de couleur pour sRGB.
- Des points pour l'interpréteur de commandes plausible pour les appareils de capture sont générés à l'aide d'un échantillonnage du cube de couleur dans l'espace colorant de l'appareil.
- Des points pour l'interpréteur de commandes de référence pour les projecteurs sont générés à l'aide d'un échantillonnage d'un polyèdre dans le cube de couleur dans l'espace colorant de l'appareil.
- Des points pour l'interpréteur de commandes possible pour les espaces de couleurs à large plage dynamique sont générés à l'aide d'un échantillonnage du cube de couleur dans l'espace lui-même.

Vous pouvez créer une liste de vertex qui décrit efficacement la gamme de couleurs des appareils, en fonction d'un profil d'appareil et des services de support système.

Pour les appareils de sortie, la limite de gamut décrit la plage de couleurs qui peut être affichée par l'appareil. Une limite de gamut est générée à partir des mêmes données que celles utilisées pour modéliser le comportement de l'appareil. Conceptuellement, vous générez un échantillonnage de la plage de couleurs que l'appareil peut produire, mesurez les couleurs, convertissez les mesures en espace d'apparence, puis utilisez les résultats pour créer la limite de gamut.

Les périphériques d'entrée sont plus délicats. Chaque pixel d'une image d'entrée doit avoir une valeur. Chaque pixel doit être en mesure de représenter n'importe quelle couleur trouvée dans le monde réel d'une manière ou d'une autre. En ce sens, aucune couleur n'est « hors gamut » pour un appareil d'entrée, car elles peuvent toujours être représentées.

Tous les formats d'image numérique ont une plage dynamique fixe. En raison de cette limitation, il existe toujours des stimuli distincts qui correspondent à la même valeur numérique. Vous ne pouvez donc pas établir un mappage un-à-un entre les couleurs du monde réel et les valeurs d'appareil photo numérique. Au lieu de cela, la limite de gamut est formée en estimant une plage de couleurs réelles qui peuvent produire les

réponses numériques de l'appareil photo. Vous utilisez cette plage estimée comme gamut pour le périphérique d'entrée.

Les primaires sont incluses pour fournir un mappage de gamut de type intention graphique métier.

Dans un style orienté objet vrai, vous abstraitz la représentation sous-jacente de la limite de gamut. Cela vous permet de modifier la représentation à l'avenir. Pour comprendre le descripteur de limite de gamut (GBD) utilisé dans le nouveau CTE, vous devez d'abord comprendre le fonctionnement des algorithmes de mappage de gamut (GMA). Traditionnellement, les GMA ont été conçus pour répondre aux besoins de la communauté des arts graphiques; autrement dit, pour reproduire des images qui ont déjà été correctement rendues pour l'appareil sur lequel l'image d'entrée a été créée. L'objectif des GMAs d'arts graphiques est d'obtenir la meilleure reproduction possible de l'image d'entrée sur l'appareil de sortie. Le nouveau GBD CTE est conçu pour résoudre quatre problèmes clés.

Étant donné que l'image d'entrée est rendue pour l'appareil d'entrée, toutes les couleurs s'inscrivent dans la plage entre le point blanc et le point noir du support. Supposons que l'image soit une photo d'une scène dans laquelle il y a un blanc diffus, comme une personne dans un tee-shirt blanc, et un surbrillance spéculaire, comme la lumière qui se reflète sur une fenêtre ou un pare-chocs chromé. La scène est rendue sur le support d'entrée afin que la mise en évidence spéculaire soit mappée au point blanc du milieu et que le blanc diffus soit mappé à une couleur neutre plus foncée que le point blanc du milieu. Le choix de la façon de mapper les couleurs de la scène au support d'entrée est à la fois une décision dépendante de la scène et une décision esthétique. Si le surbrillance spéculaire était absent de la scène d'origine, le blanc diffus serait probablement mappé au point blanc du support. Dans une scène avec beaucoup de détails en surbrillance, il reste plus de plage entre le blanc spéculaire et le blanc diffus. Dans une scène où la mise en évidence n'est pas significative, très peu de plage peut être laissée.

Pour les images pré-rendues, le mappage de gamuts est relativement simple. Fondamentalement, le point blanc du support d'origine est mappé au point blanc du support de reproduction, le point noir source est mappé au point noir de destination et la plupart du mappage est terminé. Les différents GMA existants fournissent des variations pour le mappage d'autres points sur l'échelle de tonalité du support source et différentes façons de gérer les valeurs de chroma hors gamut. Mais le mappage du blanc au blanc et du noir au noir est cohérent tout au long de ces variations. Cette implémentation nécessite que le blanc soit au-dessus d'un J* de 50 et le noir au-dessous d'un J* de 50.

Tous les encodages de couleurs ne limitent pas les plages de couleurs pour les images d'entrée. L'encodage de couleur standard IEC scRGB (IEC 61966-2-2) fournit 16 bits pour chacun des trois canaux de couleur rouge, vert et bleu (RVB). Dans cet encodage, le noir de référence n'est pas encodé en tant que triple RVB (0, 0, 0), mais en tant que (4096, 4096, 4096). Le blanc de référence est encodé comme (12288, 12288, 12288). L'encodage scRGB peut être utilisé pour représenter des surbrillances spéculaires et des détails d'ombre. Il comprend des triples RVB qui ne sont pas physiquement possibles, car ils nécessitent des quantités négatives de lumière et des encodages qui se trouvent en dehors du locus spectral CIE. De toute évidence, aucun appareil ne peut produire toutes les couleurs dans la gamut scRGB. En fait, aucun appareil ne peut produire toutes les couleurs qu'un être humain peut voir. Par conséquent, les appareils ne peuvent pas remplir la gamut scRGB, et il serait utile de pouvoir représenter la partie de la gamut qu'ils remplissent. Chaque appareil a une plage de valeurs dans l'espace scRGB qu'il peut produire. Il s'agit des couleurs « attendues » pour l'appareil ; il serait surprenant que l'appareil produise des couleurs en dehors de cette gamme. Il existe une transformation définie de l'espace scRGB en espace d'apparence, de sorte que chaque appareil a également une plage de valeurs d'apparence qu'il est censé reproduire.

Dans scRGB et l'entrée à partir d'appareils de capture caractérisés avec une cible fixe, il est possible d'obtenir une valeur en dehors de la plage des valeurs attendues. Si quelqu'un étalonne une caméra sur une cible de test ; et capture ensuite une scène avec des surbrillances spéculaires, il peut y avoir des pixels qui sont plus lumineux que le point blanc de la cible. La même chose peut se produire si un rouge naturel est plus chromatique que le rouge cible. Si quelqu'un prend une image scRGB à partir d'un appareil, puis modifie manuellement les couleurs de l'image, il est possible de créer des pixels qui se situent en dehors de la plage attendue de la gamut de l'appareil, même s'ils se trouvent dans la gamme scRGB complète.

Un deuxième problème ne semble pas, à première vue, être lié à cela. Cela se produit lorsque vous utilisez une cible de couleur pour caractériser un appareil d'entrée, tel qu'un appareil photo ou un scanneur. Les cibles réfléchissantes sont généralement produites sur papier et contiennent un certain nombre de patchs colorés. Les manufaturers fournissent des fichiers de données avec des mesures de couleur prises sous une condition d'affichage fixe pour chaque correctif de couleur. Les outils de profilage des couleurs créent un mappage entre ces valeurs mesurées et les valeurs retournées par les capteurs de couleur dans les appareils. Le problème est que souvent ces cibles de couleur ne couvrent pas la plage complète des valeurs de l'appareil. Par exemple, le scanneur ou l'appareil photo peut retourner une valeur de (253, 253, 253) pour le point blanc de référence, et un patch rouge de référence peut avoir une valeur RVB de (254, 12, 4). Elles représentent la plage de valeurs attendues pour l'appareil d'entrée, en fonction des valeurs cibles. Si vous caractérisez l'appareil d'entrée en

fonction des réponses à la cible, vous n'attendez que des couleurs dans cette plage étroite. Cette plage n'est pas seulement plus petite que la plage de couleurs que les humains peuvent voir, elle est plus petite que la plage de couleurs que l'appareil peut produire.

Dans les deux cas, il est difficile d'estimer la gamut de l'appareil ou de l'image d'entrée, malgré l'existence d'une ou de mesures de référence. Dans le premier problème, la gamut plausible de l'appareil d'entrée est inférieure à la gamme complète de scRGB. Dans le deuxième problème, le gamut de référence de la cible est inférieur à la gamme complète possible de l'appareil d'entrée.

Le troisième problème concerne le mappage des tonalités. De nombreux modèles de limites de gamut qui peuvent représenter adéquatement des images pré-rendues utilisées dans les arts graphiques ont été proposés, par exemple, le GBD braun et Fairchild Mountain Range (Braun[97]) et le descripteur de limites segment maxima de Morovic (Morovic[98]). Mais ces modèles fournissent uniquement des informations sur les extrêmes de la gamme de l'appareil ; il manque des informations sur d'autres points dans le mappage tonal. Sans ces informations, les GMA ne peuvent que faire des estimations approximatives du mappage de tonalité optimal. Pire encore, ces modèles ne fournissent aucune aide pour la plage dynamique étendue dans scRGB et dans les images d'appareil photo numérique.

Comment ce problème est-il résolu dans les industries photographique et vidéographique ? L'appareil photo capture une image. Les experts peuvent débattre de la quantité de rendu qui se produit dans l'appareil de capture ; mais ils conviennent que ce n'est pas un montant significatif. Les deux technologies ne mappent pas un blanc diffus dans une scène capturée au point blanc du milieu. De même, ils ne mappent pas le point noir de la scène au point noir du milieu. Le comportement du film photographique est décrit dans l'espace de densité à l'aide d'une courbe caractéristique, souvent appelée Hurter et Driffield, ou courbe HD&. La courbe montre la densité de la scène d'origine et la densité résultante sur le film. La figure 18 montre une courbe HD&classique. L'axe x représente une exposition croissante du journal. L'axe y représente la densité sur la diapositive. Cinq points de référence sont marqués sur la courbe : noir sans détail, qui représente la densité minimale sur le négatif ; noir avec des détails; référencez les carte gris moyen ; blanc avec détail et blanc sans détail. Notez qu'il existe un espace entre le noir sans détail (qui représente le noir de l'appareil) et le noir avec des détails (noir ombre). De même, il existe un espace entre le blanc avec des détails (blanc diffus) et le blanc sans détail (qui représente le blanc de l'appareil).

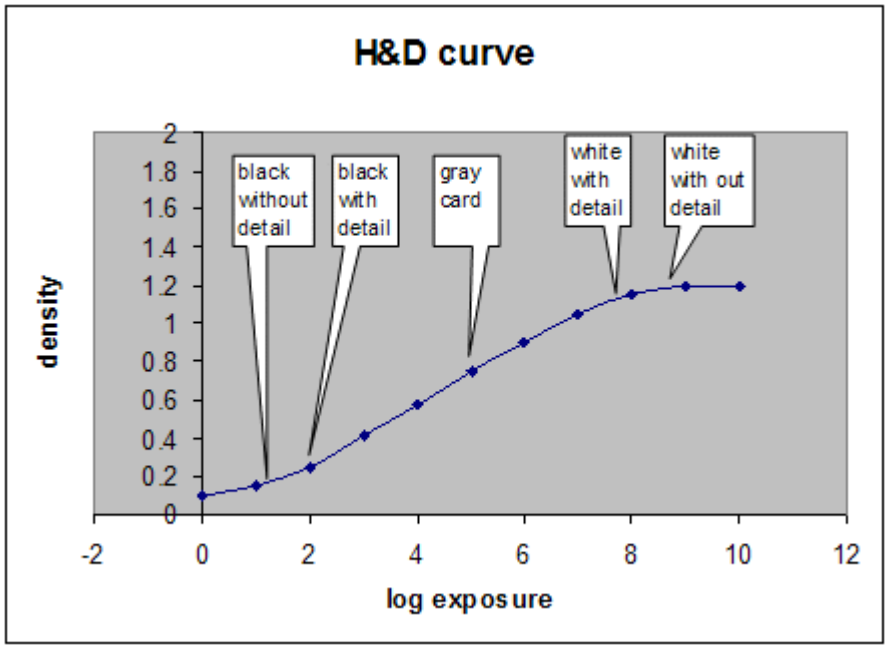


Figure 18 : Courbe HD&pour le film de diapositive

L'industrie de la vidéo fournit « headroom » et « footroom » dans les images. Dans la spécification ITU 709, la luminance (appelée Y) est encodée en 8 bits, avec une plage comprise entre 0 et 255. Toutefois, le noir de référence est encodé à 15 et le blanc de référence est encodé en 235. Cela laisse la plage d'encodage comprise entre 236 et 255 pour représenter les surbrillances spéculaires.

L'industrie vidéo présente un système de boucle essentiellement fermée. Bien qu'il existe de nombreux fournisseurs d'équipement différents, les systèmes vidéo sont basés sur des appareils de référence. Il existe un encodage standard pour les images vidéo. Il n'est pas nécessaire de communiquer une limite de gamut avec des images vidéo, car toutes les images sont encodées pour la reproduction sur le même appareil de référence. Film est également une boucle fermée, car il n'est pas nécessaire de transmettre des données intermédiaires entre différents composants. Vous souhaitez une solution qui permet de reproduire les images d'appareils avec des gamuts différents et représentant des scènes pré-rendues et non rendues sur sortie avec des gamuts variables.

Un quatrième problème que le nouveau CTE doit résoudre est que les couleurs visuellement grises produites par un appareil, par exemple, quand rouge=vert=bleu sur un moniteur, ne tombent souvent pas sur l'axe neutre du CAM (lorsque la chromaie = 0,0). Cela entraîne de grandes difficultés pour les GMA. Pour que les GMA fonctionnent correctement, vous devez ajuster la description de la gamme de l'appareil et des points d'entrée afin que l'axe neutre de l'appareil tombe sur l'axe neutre de l'espace d'apparence. Vous devez ajuster les points hors de l'axe neutre d'un montant similaire. Sinon, vous ne pouvez pas effectuer de gradations fluides via l'image. En sortant de la GMA, vous annulez ce mappage par rapport à l'axe neutre de l'appareil de sortie. Il

s’agit d’un redressement « chiropratique » de l’axe. Comme un chiropraticien, non seulement vous redressez le squelette (axe neutre), mais vous ajustez le reste du corps pour se déplacer avec le squelette. Comme un chiropraticien, vous n’ajustez pas le squelette de la même quantité à travers tout l’espace. Au lieu de cela, vous ajustez les différentes sections différemment.

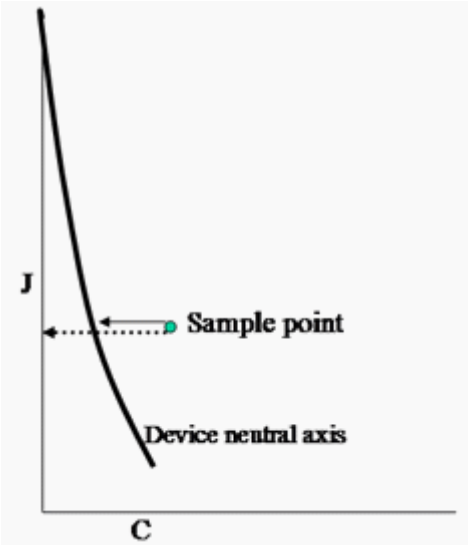


Figure 19 : Courbure de l’axe neutre de l’appareil par rapport à l’axe neutre CIECAM

Le nouveau CTE nécessite un modèle de limite de gamut qui peut être utilisé pour représenter à la fois des images sources rendues et non rendues, fournir des informations sur l’apparence des neutres d’appareil et fournir des informations pour le mappage de tonalités avec une large plage de luminance.

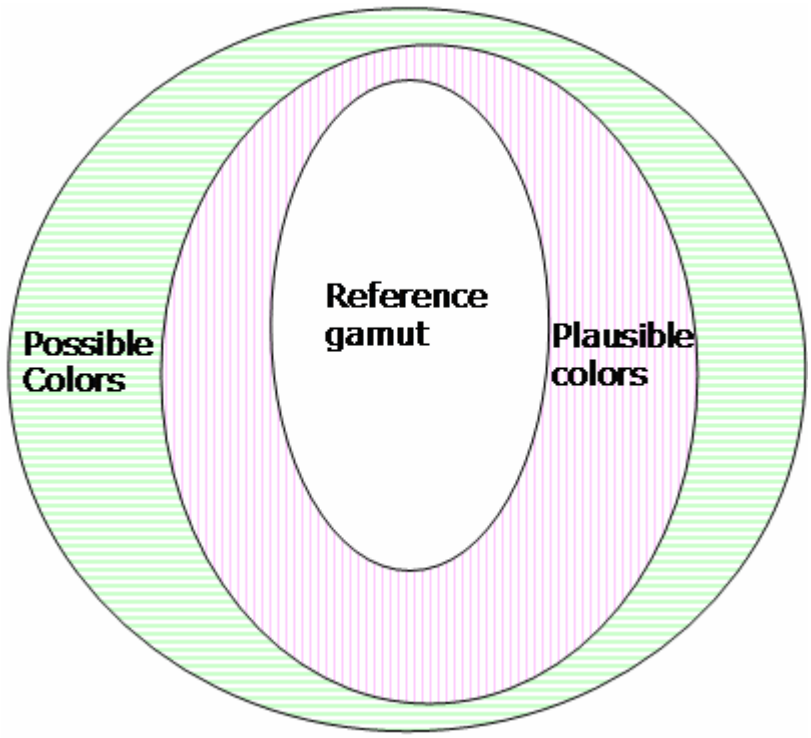


Figure 20 : Trois interpréteurs de commandes gamut

La limite de gamut est composée de trois interpréteurs de commandes qui définissent trois régions.

Dans le nouveau CTE, l'enveloppe externe du gamut est formée avec une coque convexe faite à partir de points d'échantillon dans la gamut de l'appareil. Une coque est formée en prenant un ensemble de points d'échantillonnage et en les entourant par une surface. Une coque convexe a la propriété supplémentaire d'être convexe partout. Par conséquent, il ne s'agit pas de la plus petite coque possible qui peut être adaptée aux données. Mais l'expérimentation a montré que l'ajustement trop serré des points d'échantillon entraîne des artefacts peu appétissants dans les images, comme un manque de trame lisse. La coque convexe semble résoudre ces problèmes.

Dans l'algorithme, les valeurs d'apparence de couleur sont obtenues pour un ensemble de points échantillonné à partir de l'appareil. Pour les moniteurs et les imprimantes, les valeurs d'apparence des couleurs sont obtenues en sortant des échantillons, puis en les mesurant. Vous pouvez également créer un modèle d'appareil, puis exécuter des données synthétiques via le modèle d'appareil pour prédire les valeurs mesurées. Les valeurs mesurées sont ensuite converties de l'espace colorimétrique (XYZ) en espace d'apparence (Jab), et la coque est encapsulée autour des points.

Le point clé de cet algorithme est que le point blanc adopté utilisé dans la conversion de l'espace colorimétrique en espace d'apparence n'a pas besoin d'être le point blanc du support. Au lieu de cela, vous pouvez sélectionner un point plus loin à l'intérieur de la gamme et sur (ou près) de l'axe neutre. Ce point aura alors une valeur J de 100. Les échantillons dont la valeur Y mesurée est supérieure au point blanc adopté se retrouvent avec une valeur J supérieure à 100.

Si vous placez le point blanc diffus de la scène comme point blanc adopté pour la conversion de l'espace de couleurs, les surbrillances spéculaires dans la scène seront facilement détectées comme ayant une valeur J supérieure à 100.

Étant donné que le modèle de couleur CIECAM02 est basé sur le système visuel humain, une fois qu'un blanc adopté est sélectionné, le niveau de luminance du point noir (J = 0) est automatiquement déterminé par le modèle. Si l'image d'entrée a une large plage dynamique, il est possible qu'il y ait des valeurs qui mappent à des valeurs J inférieures à zéro.

La figure 21 suivante montre les appareils neutres qui s'exécutent au centre des gamuts plausibles et de référence.

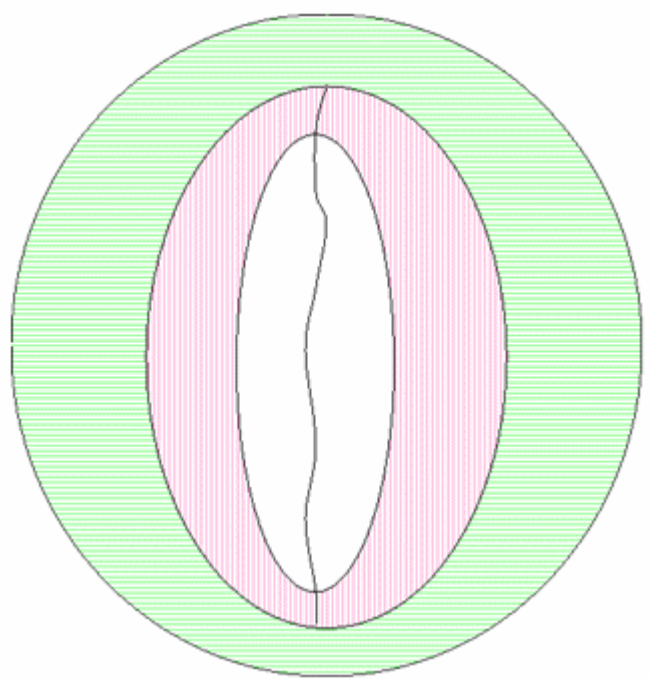


Figure 21 : Axe neutre de l'appareil ajouté à la limite de gamut

Tous les mappages de gamut impliquent soit de couper une plage d'entrée sur un gamut de sortie, soit de compresser le gamut d'entrée pour qu'il s'adapte au gamut de sortie. Des algorithmes plus complexes sont formés en compressant et en coupant dans différentes directions, ou en divisant la gamut en différentes régions, puis en effectuant un découpage ou une compression dans les différentes régions.

Le nouveau CTE étend ce concept pour prendre en charge les régions d'un gamut possible, d'un gamut plausible et d'un gamut de référence, et permet aux GMA de les mapper de différentes manières. En outre, les GMA disposent d'informations sur l'axe neutre de l'appareil. La discussion suivante explique comment gérer les situations où les gamuts plausibles et les gamuts de référence se sont réduits les uns sur les autres.

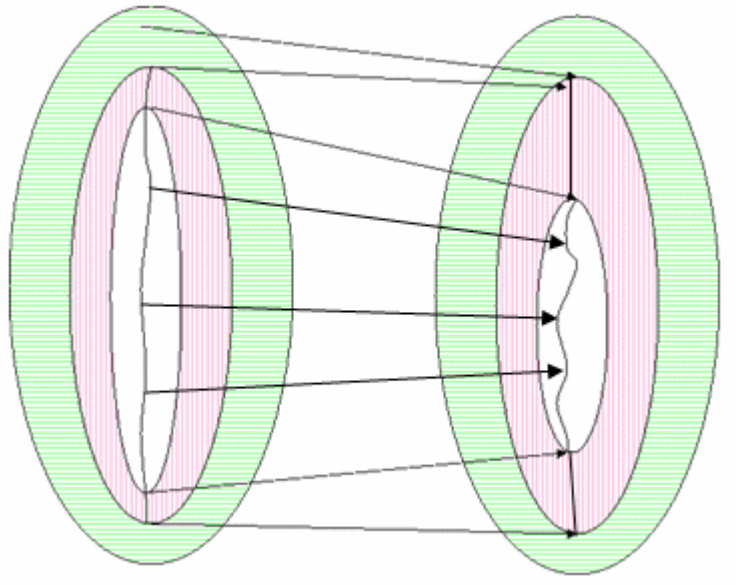


Figure 22 : GMA avec deux descripteurs gamut non réduits

Vous pouvez voir cet exemple si vous mappez à partir d'un périphérique d'entrée, tel qu'un appareil photo ou un scanneur caractérisé par une cible réfléchissante, à l'espace scRGB. Ici, les couleurs plausibles qui sont plus claires que le blanc de référence sont des surbrillances spéculaires. Dans la pratique, la caractérisation d'une caméra avec une cible peut ne pas générer la plage complète de valeurs possible dans la caméra ; cependant, les surbrillances spéculaires et les couleurs très chromatiques trouvées dans la nature le feraient. (Les cibles transmissives ont généralement un patch qui correspond à la densité minimale possible sur le milieu. Avec une telle cible, les surbrillances spéculaires se situent dans la plage de la cible.) Le noir de référence pour une cible réfléchissante serait le début de la zone noire d'ombre. Autrement dit, il est probable que les couleurs des ombres soient plus foncées que le noir sur la cible. Si l'image contient beaucoup de contenu intéressant dans cette région, il peut être utile de conserver cette variation tonale.

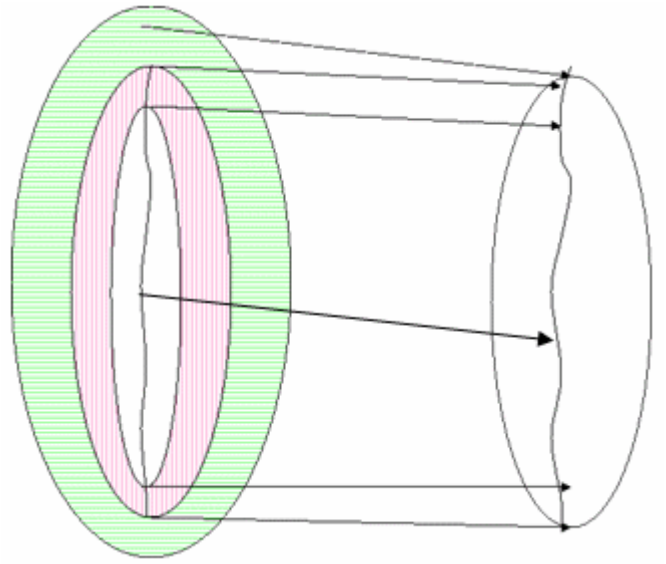


Figure 23 : GMA avec gamut de destination réduit

La figure 23 montre un algorithme de mappage de gamut possible lorsque le gamut de destination fournit uniquement la plage du blanc de l'appareil au noir, et qu'il n'existe aucune couleur possible en dehors de cette gamut. Cela est susceptible de se produire pour les périphériques de sortie classiques, tels que les imprimantes. Les couleurs possibles sont mappées au bord du gamut de destination. Mais il manque une courbe de tonalité pour l'appareil de sortie. La GMA doit sélectionner un point neutre de luminance inférieure à utiliser comme destination de mappage pour le blanc de référence. Un algorithme sophistiqué peut le faire en histogramme les légèretés dans l'image source et en voyant combien tombent dans la plage attendue mais plus claire que le blanc de référence. Plus il y a de légèreté, plus l'appareil de destination nécessite d'espace entre les points mappés pour les surbrillances spéculaires et le blanc de référence. Un algorithme plus simple peut choisir une distance arbitraire vers le bas de

l'échelle de luminosité à partir du blanc de l'appareil, par exemple 5 %. Une approche similaire s'applique au mappage du noir maximal et du noir d'ombre.

Après avoir généré la courbe de tonalité de destination, vous pouvez mapper dans une méthode similaire à celle utilisée dans la figure 23 précédente. Tous les points de la courbe de tonalité de destination se situent dans le gamut de l'appareil, et tous les points du mappage doivent se trouver dans le gamut de l'appareil.

Si vous inversez les figures de gauche et de droite, ainsi que les directions des flèches de la figure 23, vous pouvez décrire le cas où l'image source n'a qu'un gamut de référence, et où les trois gamuts de l'appareil de sortie ne se sont pas réduits les uns sur les autres. Un exemple de ceci peut être le mappage d'un moniteur à scRGB. Là encore, la GMA doit synthétiser les points de contrôle des cinq points de la courbe de tonalité de l'image source. Certains mappages peuvent placer tous les points de la courbe de tonalité dans le gamut d'appareil scRGB, tandis que d'autres mappages peuvent utiliser davantage de la gamut scRGB en mappant le blanc diffus au blanc de référence et en autorisant le blanc spéculaire à mapper à une valeur plus claire.

Enfin, vous avez le cas où les deux appareils ont uniquement le gamut de référence, ce qui est la façon dont la plupart des algorithmes de mappage de gamut fonctionnent. Vous pouvez donc résoudre ce problème en revenir simplement aux algorithmes actuels. Si vous disposez d'un moyen raisonnable de déterminer les cinq points de référence pour les appareils source et de destination, vous pouvez également vous organiser pour mapper les points de référence.

Les gamuts d'appareils contiennent plus que les cinq points de référence sur l'axe neutre. Celles-ci représentent simplement les limites entre les régions potentielles dans l'image. Entre chacun des points de référence, vous pouvez utiliser l'une des techniques de mappage de gamut existantes. Ainsi, vous pouvez couper la plage de couleurs inattendues et compresser toutes les couleurs entre le blanc et le noir attendu, ou vous pouvez couper toutes les couleurs en dehors de la plage de référence et compresser dans cette plage. Il existe de nombreuses possibilités, qui peuvent être mises en œuvre dans différents GMA. De plus, les MGM peuvent compresser et couper de différentes manières. Toutes ces combinaisons sont couvertes par cette invention.

Jusqu'à présent, dans cette discussion, la gamut a été traitée comme s'il s'agissait uniquement d'une fonction de l'appareil sur lequel l'image a été créée, capturée ou affichée. Toutefois, il n'y a aucune raison pour que toutes les images d'un appareil aient la même gamut. Les GMA dépendent des données dans le GBD. Si le descripteur est modifié d'une image à l'autre, il n'y a aucun moyen pour les GMAs de le savoir. En particulier, si les images n'ont pas de surbrillances spéculaires, les MGM fonctionnent

mieux si le descripteur gamut ne montre pas qu'il existe des couleurs plus claires que le blanc diffus.

Dans la nouvelle architecture CTE, il est possible d'utiliser plusieurs GMA. L'utilisation de plusieurs GMA est intrinsèquement mal définie. Par exemple, si un appareil de capture associe une GMA à son « apparence », il a tendance à le faire avec un gamut de destination « ciblé ». Il en va de même pour les appareils de sortie et les gamuts sources « ciblés ». Le gamut sRGB est un gamut implicite fréquemment ciblé. Par conséquent, il est fortement recommandé d'utiliser une seule GMA, si la prévisibilité est une priorité. Un flux de travail GMA unique doit être la valeur par défaut pour tous les flux de travail, en particulier les workflows de consommateurs et de prosommateurs. Bien que le mappage de gamut pour la reproduction préférée doit être effectué une seule fois, il existe des cas où plusieurs processus de mappage sont inclus. Tout d'abord, pour la vérification linguistique, vous effectuez un mappage préféré au gamut de l'appareil cible final, puis un rendu colorimétrique sur le gamut de l'appareil de vérification linguistique. Deuxièmement, certains types de mappage sont utilisés pour modifier les caractéristiques de l'image, mais ne sont pas inclus pour mapper à un gamut d'appareil, par exemple, en ajustant la courbe de tonalité ou la chromaticité. Si plusieurs GMA sont utilisés, l'interface de transformation prend un tableau de cartes de gamut liées, c'est-à-dire des cartes de gamut qui ont été initialisées avec une paire de descriptions de limites de gamut. Lorsqu'il existe plusieurs mappages de gamut, la limite de gamut d'entrée d'une carte de gamuts suivante doit être identique à la limite de gamut de sortie de son prédécesseur.

La fonction de limite de gamut d'appareil prend le moteur du modèle d'appareil et les paramètres analytiques et dérive une limite de gamut d'appareil de couleur décrite comme une liste de vertex ordonnée de la coque convexe de la gamut de l'appareil. La liste de vertex ordonnée est stockée dans CIEJab. La structure de la liste de vertex ordonnée est optimisée pour l'accélération matérielle par DirectX. Cette approche a de nombreuses solutions bien connues (recherchez « directX de coque convexe » sur le web et vous obtenez plus de 100 coups). Il existe également une référence de 1983 sur ce sujet (Computer Graphics Theory and Application, « Shiphulls, b-spline surfaces and cadcam » pp. 34-49), avec des références datant de 1970 à 1982 sur le sujet.

Deux techniques différentes peuvent être utilisées pour calculer les triangles dans l'interpréteur de commandes gamut. Pour d'autres appareils que les appareils RVB additifs, vous calculez une coque convexe. Vous pouvez envisager d'examiner la prise en charge de la coque non convexe pour d'autres appareils si vous avez un accès direct à ces appareils pour valider la robustesse, les performances et la fidélité des algorithmes. Il s'agit d'un processus bien connu qui ne nécessite pas de description supplémentaire. La technique utilisée pour les appareils RVB additifs est décrite comme suit.

Différents GBD présentent des avantages et des inconvénients. La représentation de coque convexe garantit de belles propriétés géométriques, telles que des tranches de teinte convexe qui fournissent un point d'intersection unique avec un rayon émanant d'un point sur l'axe neutre. L'inconvénient de la représentation de la coque convexe est également la convexité. Il est connu que de nombreux appareils, en particulier les appareils d'affichage, ont des gamuts qui sont loin d'être convexes. Si le gamut réel s'écarte considérablement de l'hypothèse de convexité, la représentation de la coque convexe serait inexacte, peut-être dans la mesure où elle ne représente pas la réalité.

Après avoir adopté un GBD qui donne une représentation assez précise de la gamut réelle, d'autres problèmes se posent, certains en raison du concept même de tranche de teinte. Il existe au moins deux situations pathologiques. Dans la figure 24 suivante, une gamme CRT donne lieu à des tranches de teinte avec des « îlots ». Dans la figure 25, un gamut d'imprimante donne lieu à une tranche de teinte avec une partie de l'axe neutre manquante. Les tranches de teinte pathologique ne sont pas causées par des limites de gamut particulièrement pathologiques dans ces cas. Elles sont provoquées par le concept même de tranche de teinte, car (a) elle est prise le long de la teinte constante, et (b) elle ne prend qu'une moitié du plan qui correspond à l'angle de teinte.

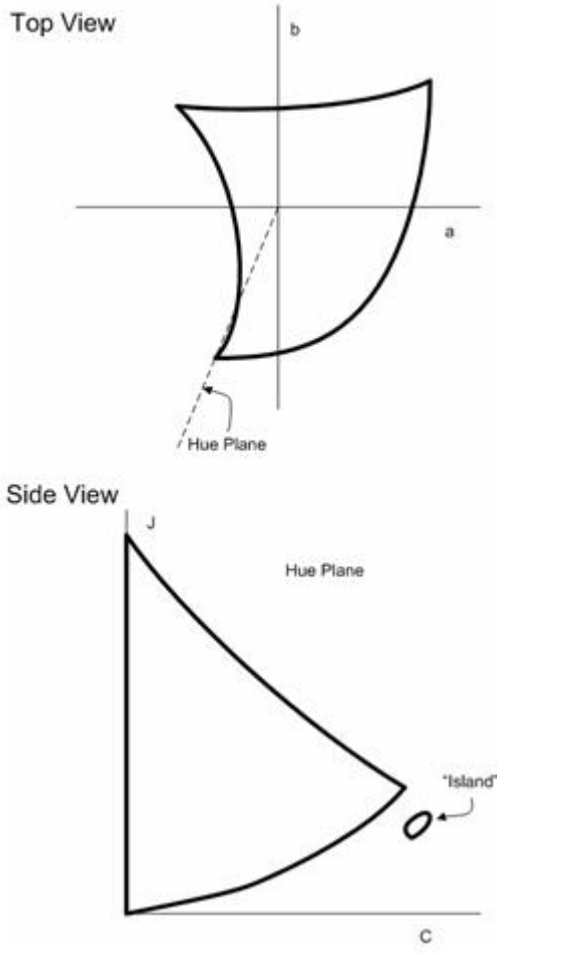


Figure 24 : Un moniteur CRT classique a un gamut qui montre une « courbure » particulière dans les teintes bleues. Si des tranches de teinte sont prises dans cette plage de teintes, des îlots isolés peuvent apparaître dans les tranches de teinte.

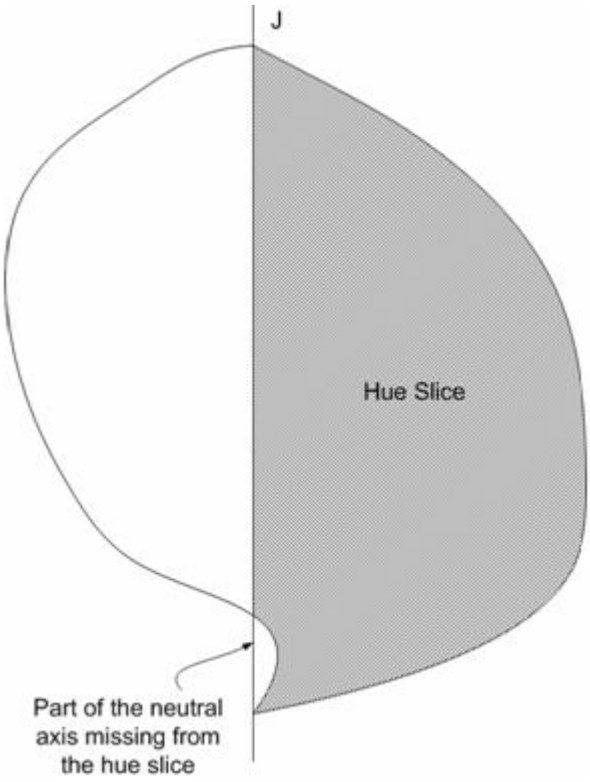


Figure 25 : Une imprimante peut avoir un gamut dont l'axe neutre est « gap ». Lorsqu'une tranche de teinte est prise (qui n'est qu'une moitié du plan), il y a une « dent » sur la partie de la limite qui est l'axe neutre. Cela peut être difficile à résoudre de manière algorithmique.

Pour résoudre ces pathologies, un nouveau framework est proposé qui abandonne le concept de tranche de teinte utilisé comme point de départ. Au lieu de cela, l'infrastructure utilise l'ensemble des « éléments de ligne de limite », ou des lignes qui se trouvent sur la limite de gamut. Ils ne fournissent pas nécessairement une visualisation géométrique cohérente comme les tranches de teinte, mais ils prennent en charge toutes les opérations de gamut courantes. En plus de résoudre les problèmes mentionnés précédemment, cette approche suggère également que la construction de tranches de teintes, même lorsqu'elle est possible, est un gaspillage de calcul.

Triangulation de la limite gamut

Le point de départ est un GBD constitué d'une triangulation de la limite de gamut. Les méthodes connues de construction de GBD fournissent généralement cette triangulation. Pour plus de détails, une méthode de construction de GBD pour les appareils additifs son espace d'appareil est décrite ici. Ces appareils incluent des moniteurs (crt et LCD) et des projecteurs. La géométrie simple du cube vous permet d'introduire un réseau régulier sur le cube. Les faces limites du cube peuvent être triangulées de plusieurs manières, comme celle illustrée dans la figure 26. L'architecture fournit un modèle d'appareil pour l'appareil afin que les valeurs colorimétriques des points de réseau puissent être obtenues de manière algorithmique, ou des mesures ont

été effectuées directement pour ces points. L'architecture fournit également CIECAM02, de sorte que vous pouvez supposer que les données de démarrage ont déjà été mappées dans l'espace Jab CIECAM02. Ensuite, chaque point de réseau sur les faces limites du cube RVB a un point correspondant dans l'espace Jab. Les connexions de points qui forment l'ensemble des triangles dans l'espace RVB induisent également un ensemble de triangles dans l'espace Jab. Cet ensemble de triangles forme une triangulation raisonnable de la limite de gamut si (a) le réseau du cube RVB est suffisamment fin et (b) la transformation de l'espace de l'appareil en espace de couleurs uniforme est topologiquement bien conduite ; autrement dit, il mappe la limite à la limite, et il ne tourne pas la gamut à l'intérieur vers l'extérieur de sorte que les points intérieurs deviennent des points limites.

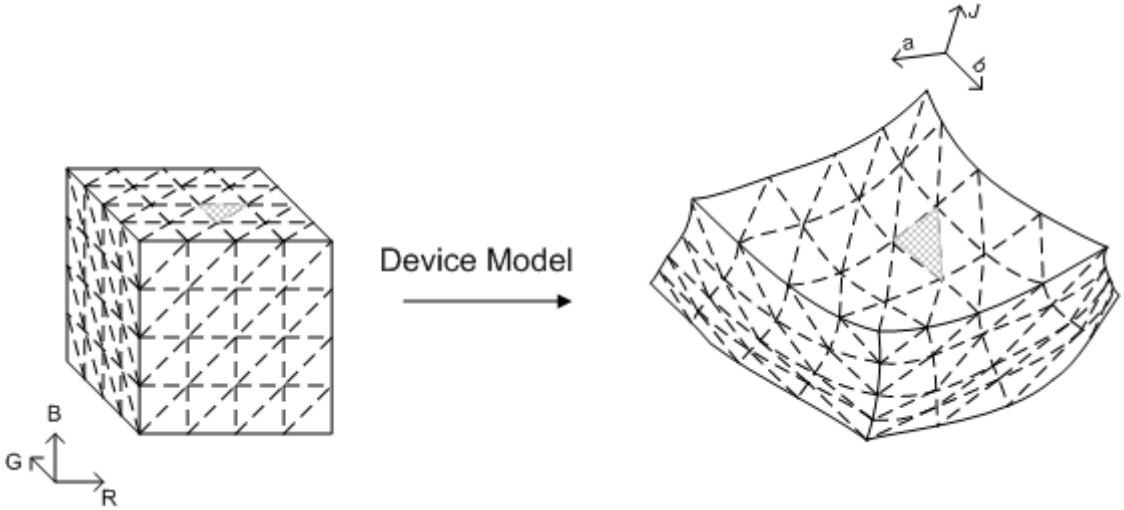


Figure 26 : Méthode simple pour trianguler la limite de gamut d'un appareil avec RVB comme espace d'appareil

Éléments de ligne de limite

Le concept d'éléments de ligne de limite est central dans ce framework ; ensemble de segments de ligne qui (a) se trouvent sur la limite de gamut et (b) se trouvent sur un plan. Dans ce cas, le plan est un plan de teinte. Chaque segment de ligne est le résultat de l'intersection du plan avec un triangle de limite de gamut. Bien que de nombreux chercheurs aient utilisé la construction de l'intersection d'un plan avec des triangles limites, ils analysent généralement la relation entre ces segments de trait et tentent de construire un objet géométrique cohérent à partir des segments de ligne. Différents algorithmes ont été conçus pour suivre ces segments de ligne les uns après les autres jusqu'à ce qu'une tranche de teinte entière soit obtenue, et de nombreuses tentatives ont été effectuées pour accélérer le processus de recherche.

Cette approche est différente. Vous croisez le plan avec les triangles pour obtenir les segments de ligne. Vous considérez ensuite ces segments de ligne comme les objets

conceptuels *de base* . Il est nécessaire d'analyser la relation entre les segments de ligne ; vous n'avez pas besoin de savoir comment ils sont interconnectés les uns avec les autres. Ce point de vue résout le problème de la tranche de teinte avec des îlots. Les approches connues qui tentent de construire une tranche de teinte supposent que, si l'on commence par un segment de ligne et le suit jusqu'au segment de ligne suivant, et ainsi de suite ; il finit par revenir au point de départ, à partir duquel une tranche de teinte entière serait construite. Malheureusement, cette approche manquerait l'île (et dans le pire des scénarios, le continent). En n'insistant pas sur l'obtention d'une image géométrique cohérente; c'est-à-dire, tranche de teinte, vous pouvez gérer le problème d'île sans effort. Une autre différence importante dans cette approche est que, pour accélérer la construction des segments de ligne, elle utilise un « filtre triangle ». Le filtre triangle supprime certains triangles qui ne produiront certainement pas de segments de ligne qui seraient utiles dans l'opération de gamut actuelle. Étant donné que l'intersection d'un triangle avec le plan est coûteuse en calcul, cela améliore la vitesse. Un effet secondaire est que vous ne pouvez pas construire de tranche de teinte, car certains segments de ligne seraient manquants en raison du filtrage de triangle.

Opération Gamut : CheckGamut

L'exemple suivant explique comment fonctionne l'infrastructure et comment CheckGamut est effectué, c'est-à-dire l'opération de vérification si une couleur est in-gamut.

L'infrastructure générale est illustrée dans la figure 27 suivante. Il existe différents composants. Les composants étiquetés en italique sont des composants qui peuvent être différents dans l'implémentation en fonction de l'opération gamut en question. Les autres composants sont invariants pour toutes les opérations de gamut. Pour commencer, ***l'entrée*** est un ensemble d'attributs de couleur. Dans le cas de CheckGamut, il s'agit de la couleur de la requête. Dans la figure 27 et la discussion suivante, il est supposé que l'angle de teinte fait partie des attributs de couleur d'entrée ou peut être obtenu à partir de ceux-ci. C'est clairement le cas si l'entrée est le point de couleur entier, soit dans Jab ou JCh, à partir duquel vous pouvez calculer l'angle de teinte. Notez que l'angle de teinte n'est nécessaire que parce que des plans de teinte sont utilisés. Selon l'opération de gamut en question, il peut ne pas être nécessaire d'utiliser le plan de teinte. Par exemple, dans la construction de la routine CheckGamut, vous pouvez utiliser des plans de J constant. Il s'agit d'une généralité qui ne sera pas utilisée ni abordée plus en détail; mais il peut être utile de rappeler cette flexibilité de la méthodologie pour prendre en charge d'autres opérations de gamut lorsque le plan de teinte n'est peut-être pas le meilleur choix.

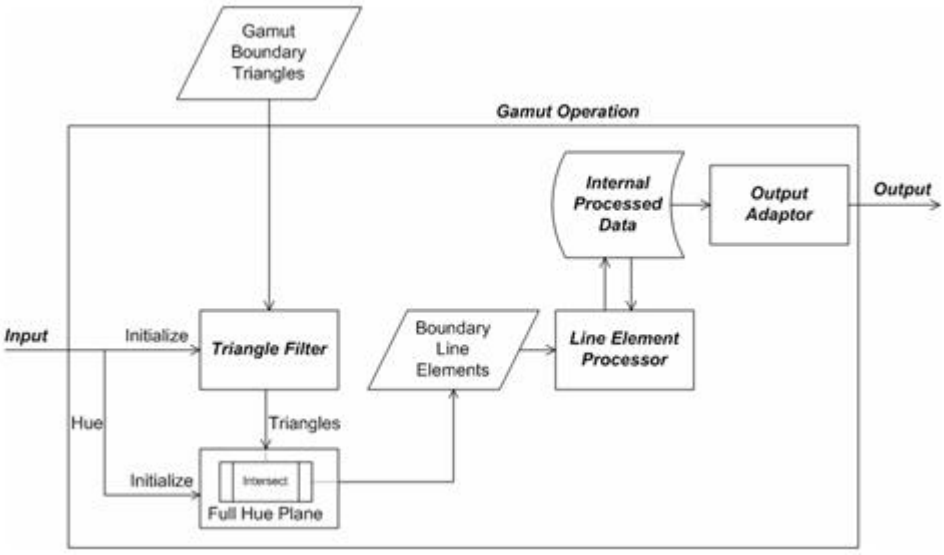


Figure 27 : Infrastructure pour prendre en charge les opérations gamut

L'angle de teinte, qui est obtenu directement à partir des entrées ou calculé à partir des entrées, est utilisé pour initialiser le plan de teinte étiqueté **Plan hue complet** dans la figure. « Full » est souligné parce que c'est le plan complet, pas seulement le demi-plan contenant la teinte. Le plan complet contient à la fois l'angle de teinte d'entrée et l'angle de 180 degrés qui lui est opposé. La fonctionnalité clé du plan hue est la fonction Intersect, qui est expliquée dans la sous-section suivante, Full Hue Plane: Intersect. Supposons que le GBD a déjà été construit et que l'ensemble de **triangles de limites gamut est** disponible. Croisez les triangles qui ont survécu au ***filtre triangle*** avec le plan de teinte à l'aide de l'intersection. Le composant *Filtre triangle* est étiqueté en italique, ce qui signifie que le composant varie en implémentation pour différentes opérations de gamut. Le *filtre triangle* pour CheckGamut est expliqué dans la section Opération Gamut : CheckGamut (suite). Le résultat de l'intersection d'un triangle avec le plan de teinte est soit vide, soit un **élément de ligne limite** , c'est-à-dire une paire de points distincts. Si le résultat n'est pas vide, il est passé dans le ***processeur d'éléments de ligne*** , qui effectue à nouveau des opérations différentes en fonction de l'opération gamut. Le *processeur d'éléments de ligne* met à jour la structure de données interne, ***Données traitées internes*** , dont le contenu ou la disposition dépend également de l'opération gamut. En règle générale, les *données traitées internes* contiennent la « réponse » au problème, qui est continuellement mise à jour avec chaque nouvel élément de ligne de limite trouvé. Lorsque tous les éléments de ligne de limite ont été traités, la réponse a été trouvée. Il reste à y accéder via ***l'adaptateur de sortie***. Étant donné que les *données traitées internes* sont spécifiques à l'opération gamut, l'adaptateur de *sortie* est également spécifique à l'opération de gamut.

Plan Hue complet : intersecter

La fonction Intersect calcule l'intersection du plan de teinte et d'un triangle. Aussi simple que cela puisse paraître, cette fonction est importante pour deux raisons.

Tout d'abord, l'intersection de chaque bord du triangle avec le plan peut produire trois points d'intersection, une situation géométriquement impossible. La raison pour laquelle cela peut se produire dans le calcul est que, lorsque les calculs sont effectués en virgule flottante, par exemple au format IEEE, il existe des incertitudes, ou « bruit numérique », dans chaque étape qui affecte la conclusion de si un bord croise le plan. Lorsque le plan croise les bords dans une situation de quasi-absence, les points d'intersection sont proches les uns des autres et la détermination si un point d'intersection se trouve à l'intérieur du bord est aléatoire. Bien que le bruit dans les valeurs numériques des points soit faible, la conclusion qualitative selon laquelle il y a plus de deux points d'intersection est géométriquement impossible et difficile à gérer correctement dans l'algorithme.

Deuxièmement, cette fonction se trouve dans la boucle critique pour chaque bord de chaque triangle filtré. Il est donc important d'optimiser son efficacité autant que possible.

Pour résoudre le premier problème de bruit numérique, effectuez les calculs en entiers. Pour résoudre le deuxième problème d'optimisation de son efficacité, mettez en cache l'attribut le plus utilisé de chaque sommet, ou le « produit point » associé à chaque sommet. Le passage en entiers est un moyen classique de garantir la cohérence géométrique. L'idée de base est que si vous devez quantifier, faites-le au début. Ensuite, les calculs suivants peuvent être effectués en entiers, et si les entiers sont suffisamment larges pour qu'il n'y ait aucun risque de dépassement de capacité, les calculs peuvent être effectués avec une précision infinie. La fonction de quantification suivante est utile à cet effet.

ScaleAndTruncate(x) = Partie entière de x*10000

Le facteur de mise à l'échelle 10000 signifie que le nombre à virgule flottante d'entrée a quatre décimales, ce qui est suffisamment précis pour cette application. En fonction de la plage de valeurs de l'espace d'apparence de couleur, vous souhaitez choisir un type entier avec des bits suffisamment larges pour contenir les calculs intermédiaires. Dans la plupart des espaces d'apparence de couleur, la plage de chaque coordonnée est comprise entre -1 000 et 1 000. La coordonnée quantifiée a une valeur absolue maximale possible de 1 000 *10 000 = 10 000 000. Comme vous le verrez, la quantité intermédiaire est un produit à points, qui est une somme de deux produits de coordonnées, de sorte qu'elle a une valeur absolue maximale possible de 2* (10 000 000) $_{2}$ = 2?10 $_{14}$. Le nombre de bits requis est log $_{2}$ (2?10 $_{14}$) = 47,51. Un choix pratique pour le type entier est donc des entiers 64 bits.

Pour garantir que l'intersection d'un plan avec un triangle donne toujours un ensemble vide ou un ensemble de deux points, vous devez considérer le triangle comme un ensemble, et non comme des bords individuels du triangle séparément. Pour comprendre la situation géométrique, considérez les « distances signées » des sommets du triangle du plan de teinte. Ne calculez pas ces distances signées directement ; au lieu de cela, calculez les produits de points des vecteurs de position des sommets avec le vecteur normal quantifié vers le plan. Plus précisément, lors de l'initialisation du plan de teinte, le vecteur normal quantifié est calculé comme suit.

NormalVector = (ScaleAndTruncate(-sin(hue)), ScaleAndTruncate(cos(hue)))

Notez que ce vecteur est un vecteur à deux dimensions. Vous pouvez utiliser un vecteur à deux dimensions, car le plan de teinte est vertical, de sorte que le troisième composant du vecteur normal est toujours égal à zéro. En outre, une table de recherche de produits à points est initialisée pour avoir une entrée pour chaque sommet à partir des triangles de limites gamut et le produit à points correspondant défini sur une valeur non valide.

Lors d'une opération d'intersection du plan de teinte avec un triangle, le produit de points de chaque sommet du triangle est recherché. Si la valeur de la table de recherche est la valeur non valide, le produit point est calculé à l'aide de l'expression suivante.

NormalVector.a*ScaleAndTruncate(vertex.a) +
NormalVector.b*ScaleAndTruncate(vertex.b)

Là encore, le composant J du vertex n'est jamais utilisé, car le vecteur normal est horizontal. Ce produit à points est ensuite enregistré dans la table de recherche afin qu'il n'ait pas besoin d'être calculé à nouveau si le produit point du vertex est interrogé ultérieurement.

La mise en cache permet de déterminer rapidement si un bord croise le plan, une fois que les produits point ont été tabulés dans la table de recherche, qui est générée progressivement au fur et à mesure que les sommets sont traités.

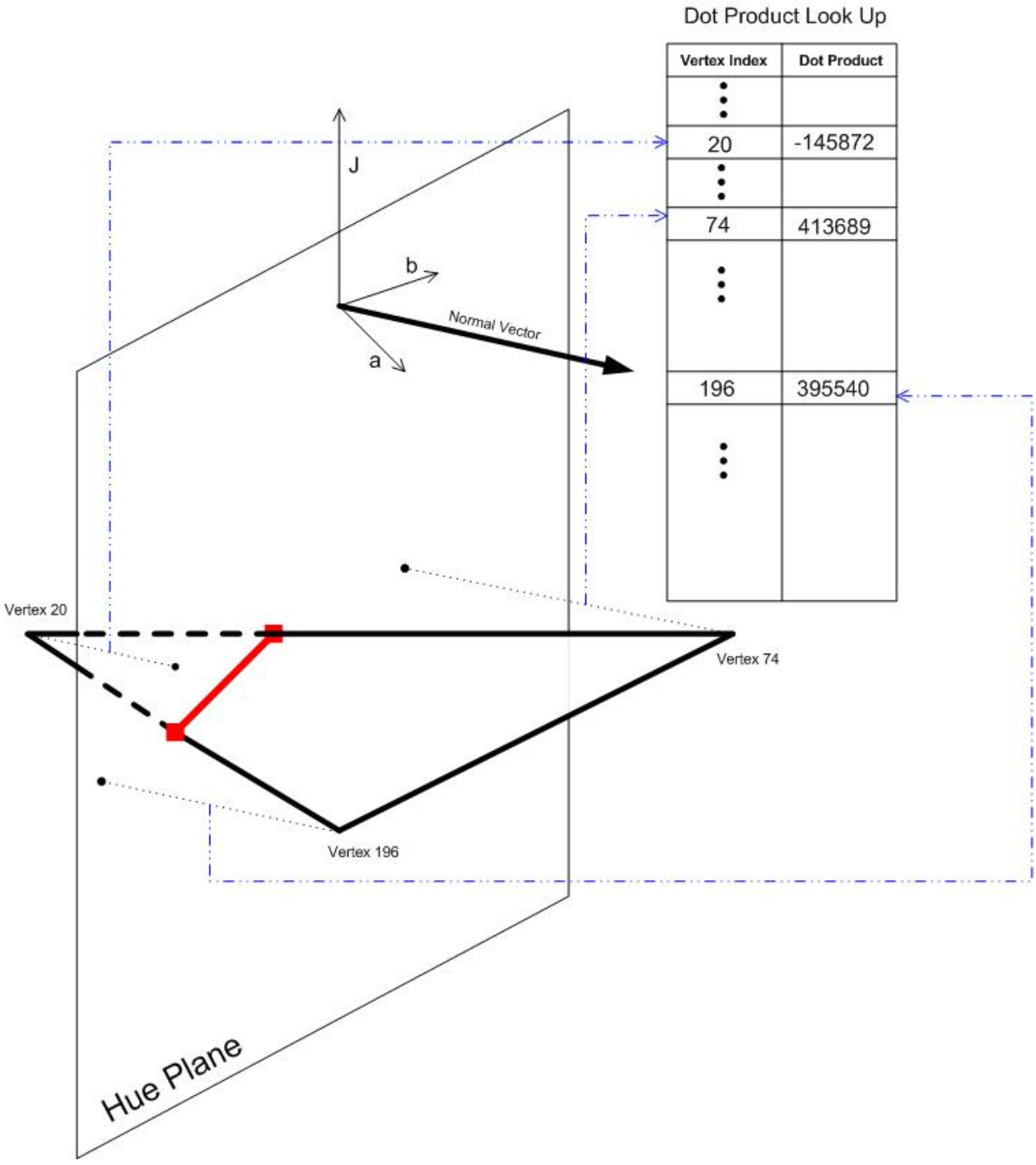


Figure 28 : Intersection du plan de teinte avec un triangle

Pour que le triangle de la figure 28 croise le plan de teinte dans un segment de ligne non dégénéré, les produits de points des sommets doivent être dans l'un des modèles suivants, lorsqu'ils sont triés dans l'ordre croissant.

0,0,+; -,0,0; -,0,+; -,-,+; -,+,+

Un point de terminaison du segment de ligne apparaît lorsque le plan est croisé par un bord avec des sommets qui ont des signes différents dans le produit point. Si le signe est égal à zéro, le sommet se trouve à droite sur le plan, et l'intersection du bord avec le plan est le sommet lui-même. Notez également que les cas 0,0,0; -,-,0; 0,+,+ ne sont pas signalés. Le premier cas (0,0,0) signifie que l'ensemble du triangle se trouve sur le plan.

Cela n'est pas signalé, car chaque bord du triangle doit appartenir à un triangle voisin qui ne se trouve pas également entièrement sur le plan. L'arête est signalée lorsque ce triangle est pris en compte. Les deux autres cas (-,-,0 et 0,++) correspondent à la configuration géométrique selon laquelle le triangle touche le plan dans un sommet. Ces cas ne sont pas signalés, car ils ne donnent pas lieu à un segment de ligne non dégénéré.

L'algorithme précédent détermine quand calculer une intersection entre un bord du triangle et le plan de teinte. Une fois qu'un bord est déterminé, l'intersection est calculée à l'aide d'équations paramétriques. Si l'un des produits à points est égal à zéro, l'intersection est le sommet lui-même, donc aucun calcul n'est nécessaire. En supposant que les deux produits à points des sommets du bord sont non nuls , vertex1 est le vertex avec point *négatif* produit dotProduct1; et vertex2 est le sommet avec point *positif* produit dotProduct2. Cet ordre est important pour s'assurer que le point d'intersection calculé ne dépend pas de la façon dont l'ordre des sommets apparaît dans la représentation de l'arête. Le concept géométrique du bord est symétrique par rapport à ses sommets. L'aspect informatique de l'utilisation d'équations paramétriques du bord introduit une asymétrie (choix du sommet de départ), qui peut donner un point d'intersection légèrement différent en raison du bruit numérique et du conditionnement des équations linéaires à résoudre. Ceci étant dit, le point d'intersection, intersection, est donné par ce qui suit.

t = dotProduct1/(dotProduct1 - dotProduct2)

Intersection. J = vertex1. J + t * (vertex2. J - vertex1. J)

intersection.a = vertex1.a + t * (vertex2.a - vertex1.a)

intersection.b = vertex1.b + t * (vertex2.b - vertex1.b)

Opération Gamut : CheckGamut (suite)

L'algorithme géométrique de base utilisé pour la vérification de la gamut consiste à compter le nombre de croisements de rayons. Pour un point de requête donné, envisagez un rayon commençant par le point de requête et pointant vers le haut (direction J). Comptez le nombre de fois où ce rayon franchit la limite de gamut. Si ce nombre est pair, le point de requête est hors gamut. Si ce nombre est impair, le point se trouve à l'intérieur. En principe, cet algorithme peut être implémenté en 3D, il est généralement en proie à des difficultés causées par des situations dégénérées, telles que le rayon se trouve (en partie) sur un triangle de limite, ou une dégénération de dimension inférieure, comme le rayon qui se trouve (en partie) sur un bord d'un triangle

limite. Même en 2D, vous devez faire face à ces situations dégénérées; mais le problème est plus simple et a été traité de manière satisfaisante. Voir [O'Rourke].

Pour un point d'entrée donné Jab, déterminez son angle de teinte h comme suit.

h = atan(b/a),

Initialisez le plan de teinte, puis déterminez les éléments de ligne limite correspondant à ce plan de teinte. Étant donné que les éléments de ligne limite ne sont pertinents que s'ils croisent le rayon vers le haut, configurez un filtre triangle pour supprimer les triangles qui donnent des éléments de ligne qui ne croiseront certainement pas le rayon vers le haut. Dans ce cas, considérez le cadre englobant du triangle. Le rayon vers le haut n'entrecroisera pas le triangle si le point de requête se trouve en dehors de l'« ombre » projetée par le cadre englobant si une source de lumière se trouvait directement au-dessus. Gonflez légèrement cette valeur avec une tolérance prédéfinie pour permettre le bruit numérique afin de ne pas jeter par inadvertance des triangles qui pourraient donner des éléments de trait utiles. Le résultat est le cylindre rectangulaire semi-infini illustré dans la figure 29. Vérifier si le point de requête se trouve à l'intérieur ou à l'extérieur de ce cylindre peut être implémenté efficacement à l'aide de simples inégalités.

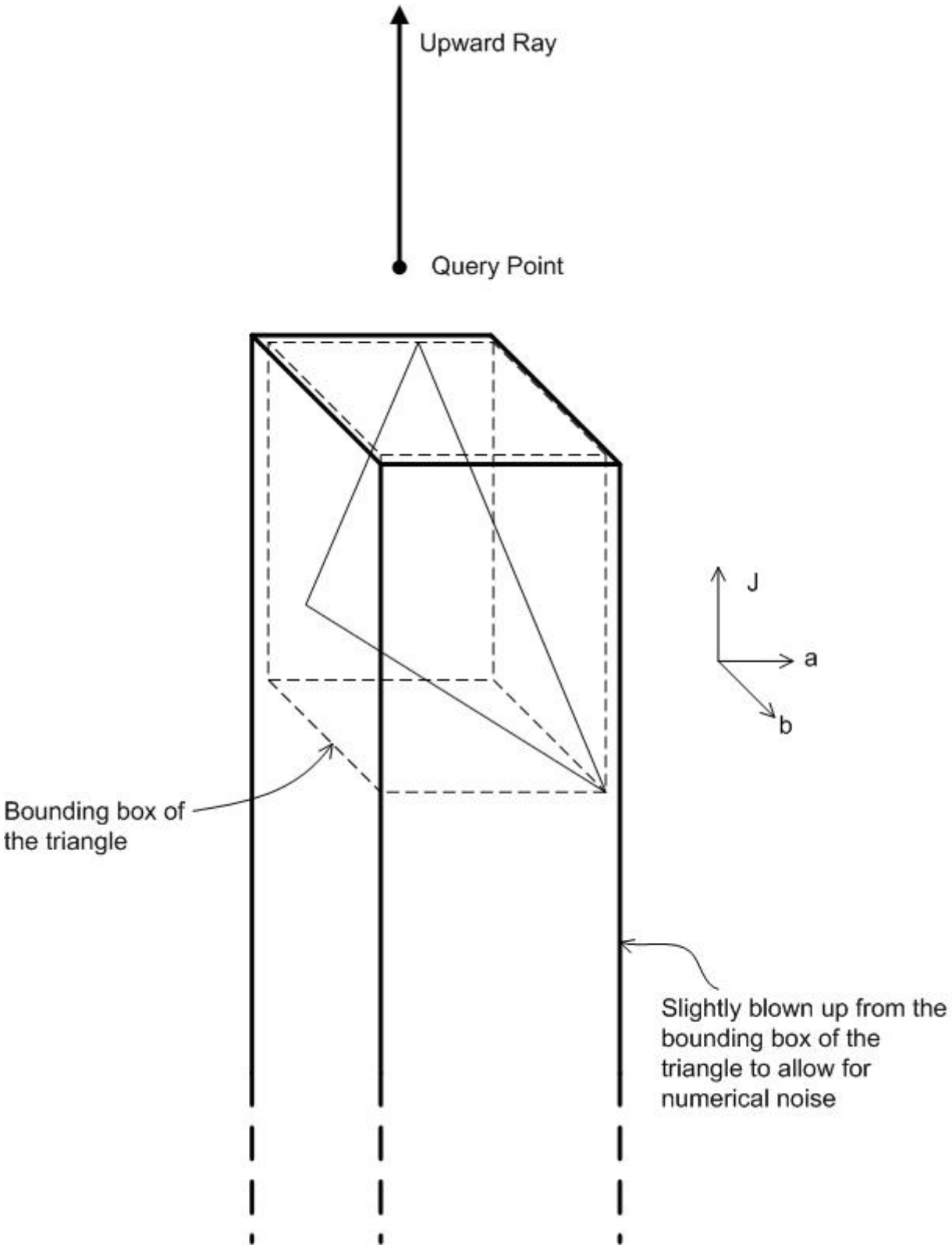


Figure 29 : Filtre triangle pour CheckGamut

CheckGamut comporte trois composants spécifiques aux opérations de gamut : *Données traitées internes*, *Processeur d'éléments de ligne* et *Adaptateur de sortie*. Les *données traitées internes* sont une liste d'éléments de ligne qui ont été traités par *le processeur d'éléments de ligne*. Dans ce cas, le *processeur d'éléments de* ligne ajoute simplement un élément de ligne à la liste. La structure de données interne pour *les données traitées internes* peut être une liste liée ou un tableau qui peut augmenter en taille.

L'adaptateur de sortie est un module qui accède à la liste des éléments de ligne, détermine si un élément de ligne traverse le rayon vers le haut (nombre 1) ou non (nombre 0). La somme de tous ces nombres donne un nombre total. *L'adaptateur de sortie* génère finalement une réponse « oui » (in-gamut) ou « non » (hors gamut), selon que le nombre total est impair ou pair. L'étape où vous déterminez si un élément de ligne traverse le rayon vers le haut mérite une certaine attention, car c'est là que se pose le problème de la dégénération et aussi le problème du surcomptent. Après [O'Rourke], pour qu'un élément de ligne traverse le rayon, le point d'extrémité droit (point d'extrémité avec une chromaie plus grande) doit être strictement sur le côté droit du rayon. Cela garantit que, si un point de terminaison se trouve exactement sur le rayon, il n'est compté qu'une seule fois. La même règle résout également la situation dégénérée où l'élément de ligne se trouve exactement sur le rayon. Vous n'incrémentez pas le nombre pour cet élément de ligne.

La figure 30 montre les éléments de ligne résultants d'un exemple de gamut avec le point de requête dans différentes positions.

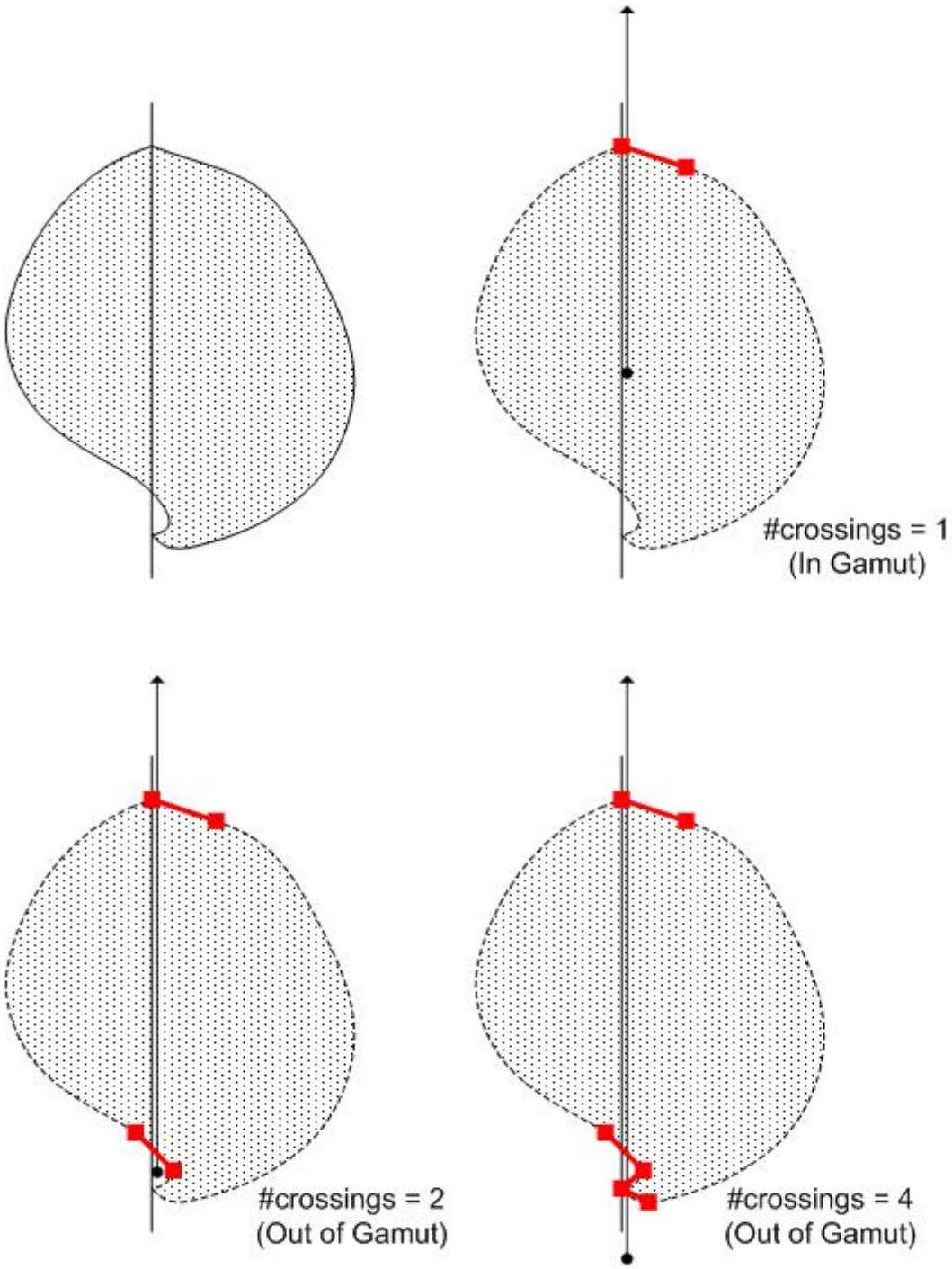


Figure 30 : Fonctionnement de CheckGamut

Mappage de la gamut de différence de couleur minimale

Le mappage minimum de la différence de couleur gamut, MinDEMap, a une spécification simple : si une couleur est in-gamut, ne rien faire. Si une couleur est hors gamut, projetez-la sur le point « le plus proche » de la limite de gamut. La mot clé « la plus proche » n'est pas bien définie tant que vous n'avez pas spécifié l'équation de

différence de couleur à utiliser. Dans la pratique, pour faciliter et accélérer le calcul, la distance euclidienne de l'espace d'apparence de couleur choisi, ou une variante de celui-ci, est utilisée comme métrique de différence de couleur. L'avantage de la métrique euclidienne est qu'elle est compatible avec le produit point de l'espace, ce qui permet d'utiliser l'algèbre linéaire. En détail, si un « produit point » est défini dans l'espace, une distance peut être définie comme racine carrée du produit point du vecteur de différence avec lui-même. Un produit à points peut généralement être défini par une matrice A 3x3 définie positive.

$u?v = u_T Av$

où le côté droit est la multiplication de matrice habituelle. Si A est la matrice d'identité, le produit à points standard est récupéré. Dans la pratique, si Jab est l'espace de couleurs, vous ne souhaitez pas mélanger les composants. Une matrice diagonale autre que la matrice d'identité peut donc être utilisée. En outre, vous pouvez conserver l'échelle sur un et b inchangé afin que la mesure de teinte soit conservée. Par conséquent, une variante utile du produit à points euclidien standard est la suivante.

w_J (composant J de vous)(composant J de v) + (un composant de vous)(un composant de v) + (b composant de vous)(b composant de v)

où w_J est un nombre positif. Une autre variante consiste à laisser w_J varier avec le point de requête d'entrée :

$w_{J\backslash =} w_J$ (queryPoint)

Le résultat final est une mesure de la distance asymétrique par rapport aux deux points, et avec des pondérations relatives différentes sur la légèreté et la chroma ou la teinte à mesure que le point de requête d'entrée varie. Cela est conforme à certaines observations sur la perception des couleurs humaines selon laquelle les différences de couleur ne sont pas pondérées de façon égale dans toutes les dimensions. Il a été constaté que les gens sont moins sensibles aux différences de légèreté qu'aux différences de teinte et de chroma.

La fonction de pondération suivante est utile.

$w_J = k_2 - k_1 (C - C m_{ax}) n$

où k = 1, k_1 = 0,75/(C m_{ax}) n, C m_{ax} = 100, n = 2 et C est le plus petit de la chroma du point de requête et C m_{ax}.

de sorte qu'un poids de 0,25 est placé sur le terme J lorsque la chroma est égale à zéro, et un poids de 1 lorsque la couleur est 100. La tendance à mettre moins de poids sur J

lorsque la chroma est petite, et plus de poids sur J lorsque la couleur est grande suit l'utilisation recommandée pour CMC et CIEDE2000.

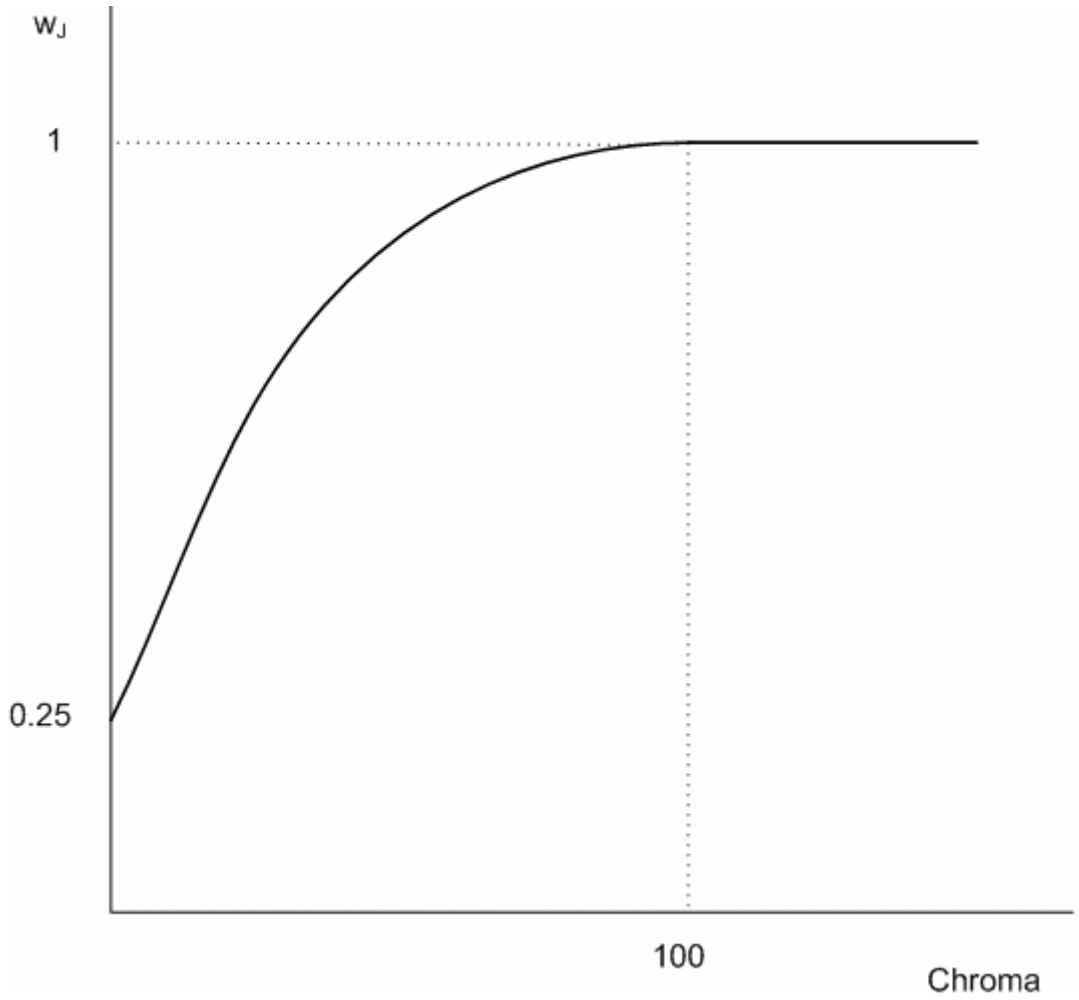


Figure 31 : Fonction de pondération sur le composant J de la métrique

Utilisez l'espace Jab pour l'exemple suivant. Il est difficile d'effectuer une recherche dans tous les triangles de limite pour déterminer le point le plus proche de la métrique euclidienne. Il s'agit d'une approche simple pour rendre ce processus aussi efficace que possible, sans introduire d'hypothèses supplémentaires qui peuvent accélérer le processus, mais aussi aboutir à une réponse approximative. Tout d'abord, il est nécessaire de comprendre la procédure géométrique de projection d'un point sur le triangle donné. Une description est donnée ici.

Une projection orthogonale sur le plan infini contenant le triangle est d'abord effectuée. La distance la plus courte du point de requête par rapport au plan peut être déterminée en deux étapes.

(a) Calculez l'unité vecteur normal du triangle.

(b) Calculer le produit de point du vecteur normal unitaire et un vecteur formé à partir du point de requête et d'un point sur le triangle; c'est-à-dire un de ses sommets. Étant

donné que le vecteur normal a une longueur d'unité, la valeur absolue de ce produit point est la distance du point de requête par rapport au plan.

Le point projeté peut ne pas être la réponse, car il peut se trouver en dehors du triangle. Vous devez donc d'abord effectuer une case activée. Le calcul équivaut à calculer les coordonnées barycentriques du point projeté par rapport au triangle. Si le point projeté est déterminé comme étant à l'intérieur du triangle, il s'agit de la réponse. Si ce n'est pas le cas, le point le plus proche est acquis sur l'un des bords du triangle. Effectuez une recherche sur chacun des trois bords. La détermination de la projection du point de requête sur un bord est un processus similaire à la projection sur le triangle, mais une dimension en moins. Une projection orthogonale est d'abord calculée. Si le point projeté se trouve sur le bord, il s'agit de la réponse. Si ce n'est pas le cas, le point le plus proche est acquis sur l'un des deux points de terminaison. Effectuer une recherche sur les deux points de terminaison ; autrement dit, calculez la distance du point de requête par rapport à chacun d'eux et comparez celui qui est plus petit.

Un examen attentif révèle qu'il y a beaucoup de recherches répétées lorsque vous passez par tous les triangles, car un bord est toujours partagé par deux triangles et un sommet partagé par au moins trois bords. En outre, vous n'êtes pas très intéressé à trouver le point le plus proche d'un triangle particulier; au lieu de cela, vous souhaitez trouver le point le plus proche de la limite de la gamme entière. Toutefois, un triangle particulier serait celui dans lequel cela est réalisé. Vous pouvez utiliser deux stratégies pour accélérer la recherche.

Stratégie I. Chaque sommet sera traité, au maximum, une fois. Chaque arête sera traitée, au maximum, une fois.

Stratégie II. À tout moment dans la recherche, vous avez un meilleur candidat avec la meilleure distance correspondante. Si vous pouvez déterminer, par un case activée rapide, qu'un triangle n'est pas capable de donner une meilleure distance, il n'est pas nécessaire de poursuivre le calcul. Vous n'avez pas besoin du point et de la distance les plus proches pour ce triangle.

MinDEMap

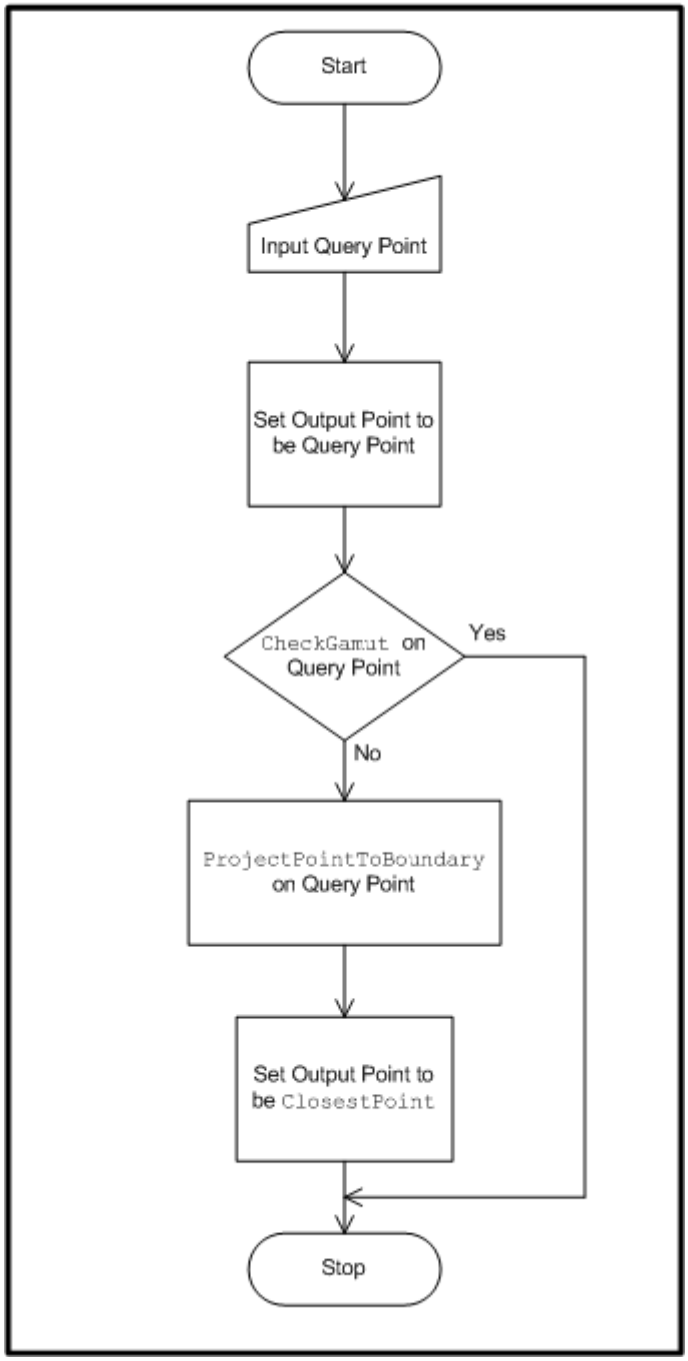


Figure 32 : Schémas de mappage DE minimum

La figure 32 montre le flux général de logique pour la carte de gamut MinDEMap. Pour un point de requête, la fonction CheckGamut est d'abord appelée. Si le point est in-gamut, la carte est un no-op. Si le point est hors gamme, appelez ProjectPointToBoundary. Passez maintenant à la figure 33. À ce stade, il est supposé que les valeurs suivantes ont été calculées.

(a) Unité du vecteur normal à chaque triangle de limite de gamut par rapport au produit à points standard.

(b) Liste de vertex et liste d'arêtes, en plus de la liste de triangles.

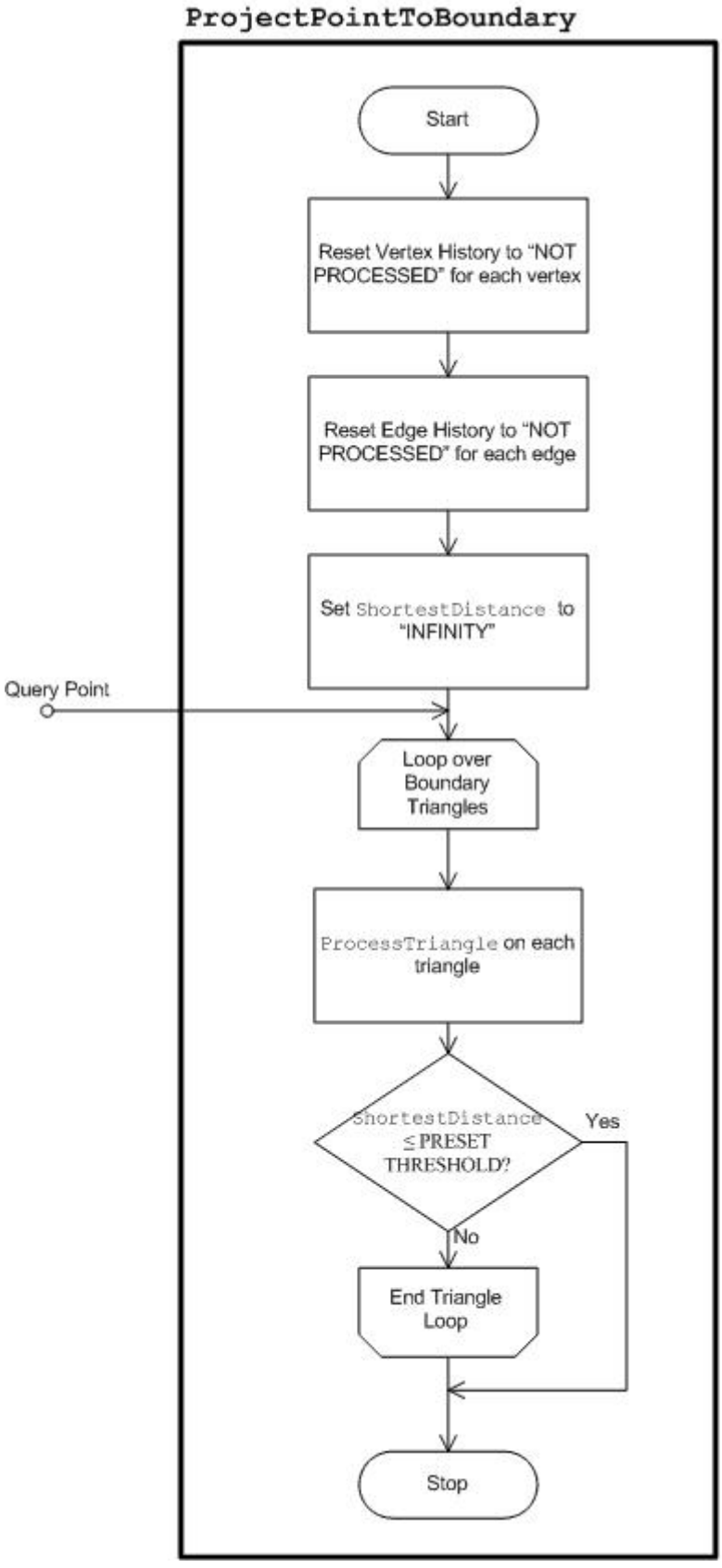


Figure 33 : Routine ProjectPointToBoundary

Toutes ces opérations représentent une surcharge constante et auraient un coût décroissant si suffisamment de requêtes à cette limite de gamme sont effectuées. En règle générale, c'est le cas lorsque vous générez un LUT de transformation d'un appareil à un autre, où il n'y a que deux gammes fixes, et où le LUT de transformation s'exécute à travers des points sur la grille échantillonné uniformément. Vous calculez au préalable

les vecteurs normaux par rapport au produit dot standard, même si la notion de perpendicularité sera basée sur le produit de points pondéré, qui dépend du point de requête comme expliqué précédemment. La raison en est qu'un vecteur normal par rapport au produit à points pondéré peut être obtenu facilement à partir du vecteur normal par rapport au produit à points standard. Si n_0 est un vecteur normal par rapport au produit de point standard, alors

n = (J-component of n_0 /w_J, a-component of n_0, b-component of n_0)

est normal au triangle par rapport au produit à points pondéré. En raison de cette relation, il est toujours avantageux de précalcaliser n_0 même s'il doit être ajusté en fonction du point de requête.

La routine ProjectPointToBoundary commence par réinitialiser l'« historique traité » des sommets et des arêtes. Il s'agit de tables d'indicateurs BOOLEAN qui indiquent si un sommet ou un bord a déjà été visité. Il réinitialise également la variable ShortestDistance à « INFINITY », qui est la valeur maximale encodée dans le système de nombre à virgule flottante utilisé. Ensuite, il s'exécute à travers une boucle, recherchant le point le plus proche de chaque triangle à l'aide de l'appel ProcessTriangle. ProcessTriangle est la routine permettant de mettre à jour la variable ShortestDistance et se trouve clairement dans la boucle critique. Une optimisation consiste à s'arrêter lorsque le résultat est suffisamment bon. Après chaque appel à ProcessTriangle, la variable ShortestDistance est examinée. S'il satisfait à un seuil prédéfini, vous pouvez arrêter. Le seuil prédéfini dépend de l'espace de couleurs utilisé et de la précision requise du système d'imagerie des couleurs. Pour une application classique, vous ne souhaitez pas effectuer de travail inutile si la différence de couleur est inférieure à ce qui peut être discerné par la vision humaine. Pour CIECAM02, cette différence de couleur est 1. Utilisez toutefois une valeur de seuil de 0,005 dans l'implémentation pour conserver la précision des calculs, car il peut ne s'agir que d'une étape intermédiaire dans une chaîne de transformations.

ProcessTriangle implémente la stratégie PRÉCÉDENTE. En obtenant un vecteur normal du vecteur normal d'unité précalpliquée vers le triangle par rapport au produit de point standard, il calcule la distance du point de requête au plan infini contenant le triangle en formant le produit de point du vecteur normal d'unité et le queryVector, le vecteur de l'un des sommets du triangle, vertex1, au point de requête, queryPoint.

queryVector = queryPoint - vertex1

distance = | normalVector * queryVector |/|| normalVector||

Il s'agit d'un calcul relativement peu coûteux, et la distance est nécessaire pour effectuer d'autres calculs. Si cette distance n'est pas inférieure à la meilleure distance actuelle,

ShortestDistance, ce triangle ne produira pas une meilleure distance, car il ne donnera pas une meilleure distance que le plan qui le contient. Dans ce cas, vous retournez le contrôle à la boucle triangle. Si la distance est inférieure à ShortestDistance, vous avez potentiellement un point plus proche, si ce point se trouve à l'intérieur du triangle. Vous devez effectuer des calculs « durs » (mais rien au-delà de l'algèbre linéaire) pour le déterminer. Si les deux autres sommets du triangle sont vertex2 et vertex3, formez les vecteurs de base d'abordBasisVector et secondBasisVector.

firstBasisVector = vertex2 - vertex1

secondBasisVector = vertex3 - vertex1

Utilisez le système linéaire d'équations suivant pour résoudre les inconnues vous et v.

firstBasisVector * queryVector = (firstBasisVector * firstBasisVector)u + (firstBasisVector * secondBasisVector)v

secondBasisVector * queryVector = (secondBasisVector * firstBasisVector)u + (secondBasisVector * secondBasisVector)v

et les conditions pour que le point projeté se trouve à l'intérieur du triangle sont les suivantes :

0 ≤ u ≤ 1, 0 ≤ v ≤ 1 et vous + v ≤ 1

Après ce calcul, s'il est déterminé que le point projeté se trouve dans le triangle, vous avez trouvé un nouveau point le plus proche; la distance que vous avez calculée au début est la nouvelle distance la plus courte. Dans ce cas, mettez à jour les variables ShortestDistance et ClosestPoint. Si le point projeté se trouve à l'extérieur du triangle, vous pouvez trouver un point plus proche sur l'un de ses bords. Vous pouvez donc appeler la routine ProcessEdge sur chacun des trois bords.

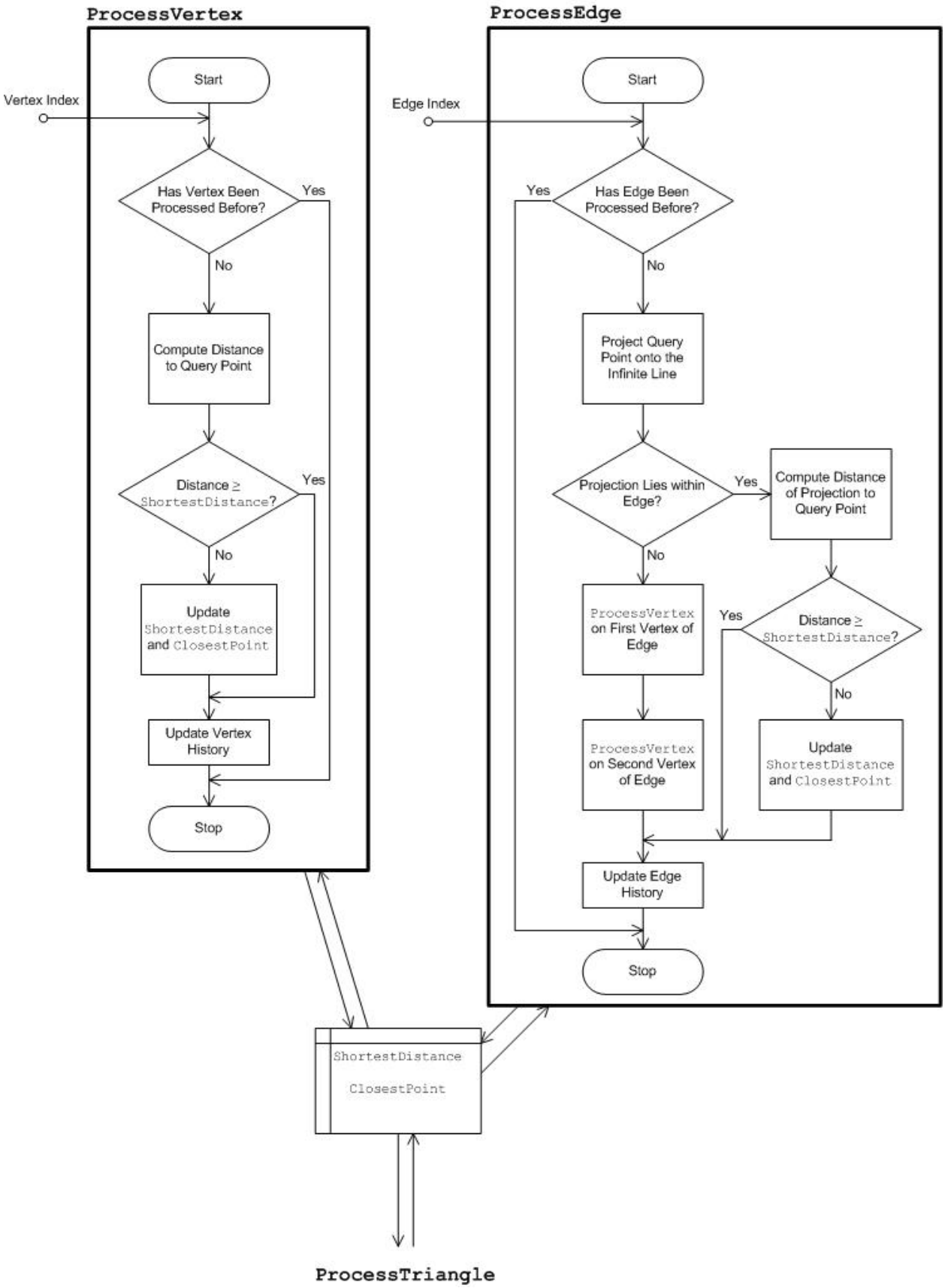


Figure 34 : Routines ProcessEdge et ProcessVertex

La routine ProcessEdge implémente la stratégie I, illustrée à la figure 34. ProcessEdge commence par vérifier si l'arête a déjà été traitée. Si c'est le cas, aucune autre mesure n'est prise. Si ce n'est pas le cas, il calcule la projection orthogonale du point de requête sur la ligne infinie contenant le bord. L'algèbre linéaire impliquée dans le calcul est

similaire aux équations de triangle précédentes. Toutefois, le calcul est plus simple, il n'est pas décrit ici. Si le point projeté se trouve dans le bord, vous trouvez la distance entre le point projeté et le point de requête. Si cette distance est inférieure à ShortestDistance, vous avez trouvé un nouveau point le plus proche. Mettez à jour ShortestDistance et ClosestPoint. Si le point projeté se trouve en dehors du bord, appelez ProcessVertex sur les deux points de terminaison. Avant de retourner le contrôle, mettez à jour l'historique des arêtes afin que ce bord soit marqué comme « TRAITÉ ».

Enfin, vous fournissez une description de ProcessVertex. La routine ProjectVertex implémente également la stratégie I et gère une table d'historique des vertex. Comme illustré dans la figure 34, il vérifie d'abord si le vertex a été traité auparavant. Si c'est le cas, aucune autre mesure n'est prise. Si ce n'est pas le cas, il calcule la distance du sommet par rapport au point de requête. Si la distance est inférieure à ShortestDistance, mettez à jour ShortestDistance et ClosestPoint. À la fin, il met à jour l'historique des vertex afin que ce sommet soit marqué comme « TRAITÉ ».

Lorsque la boucle de contrôle externe a épuisé tous les triangles ou s'est terminée avant que le seuil de différence de couleur n'ait été atteint, la variable ClosestPoint est accessible. Il s'agit du résultat de MinDEMap. L'appelant peut également récupérer ShortestDistance s'il est intéressé par la distance entre la couleur mappée et la couleur de requête.

Lissage des teintes

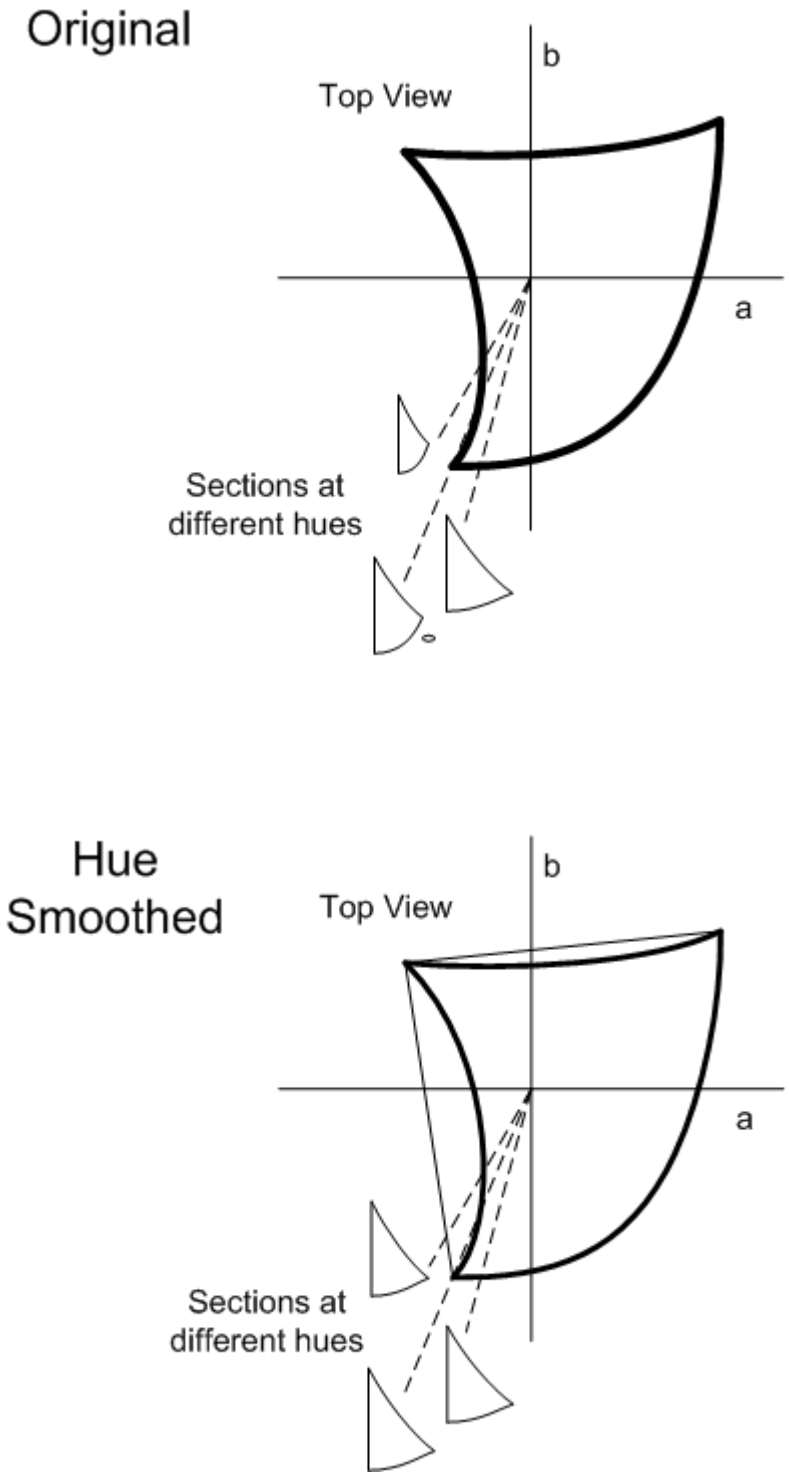


Figure 35 : Lissage des teintes

Un problème se produit avec les opérations qui sont limitées par la teinte ; autrement dit, l'opération prend uniquement en compte les variables au sein d'un plan de teinte. La figure 35 montre un exemple de gamme présentant des tranches de teinte « discontinues » dans les teintes bleues. Dans cette plage de teintes, pour certains angles de teinte, la limite de la gamme est tangentielle au plan de teinte. En effet, cela entraîne une modification de la structure topologique des tranches de teinte. Dans l'exemple illustré, à mesure que le plan de teinte balaye cette plage de teintes, une « île » émerge et submerge. Cette modification de la topologie entraîne la discontinuité des opérations

spécifiques à la teinte. Par exemple, le cusp à teinte fixe change brusquement à mesure que l'angle de teinte change dans cette plage.

Il existe une raison de la science des couleurs pour laquelle il est souhaitable de conserver la teinte dans certaines opérations. Pour résoudre le problème précédent, les triangles de limites de gamut d'origine doivent être « lissés de teinte ». En règle générale, un lissage de teinte d'un ensemble de triangles de limites de gamut est un ensemble de triangles de telle sorte que (a) il forme la limite d'un nouveau « gamut », qui peut ne pas correspondre au gamut réel de l'appareil et qui contient la gamme définie par l'ensemble de triangles d'origine ; et (b) les triangles du nouvel ensemble ne sont pas parallèles aux plans de teinte.

Une façon pratique d'obtenir une teinte lissée ensemble de triangles est de prendre la coque convexe des sommets d'origine. Comme l'illustre la figure 35, les tranches de teinte de la coque convexe varient en douceur dans la plage de teintes problématique sans changer soudainement la topologie.

Définition des valeurs primaires et secondaires dans la description de la limite de gamut

Certaines méthodes de mappage de gamut, telles que HueMap, dépendent de l'emplacement des primaires et des secondaires des appareils. Pour les appareils additifs, les primaires sont rouges, vertes et bleues (R, G et B); et les secondaires sont cyan, magenta et jaune (C, M et Y). Pour les appareils soustractifs, les primaires sont C, M et Y; et les secondaires sont R, G et B. Le GBD effectue le suivi de ces six valeurs, ainsi que le blanc et le noir (W et K), dans un tableau de valeurs de couleur Jab. Ces valeurs sont définies dans la description de la limite de gamut lors de sa création. Pour les appareils de sortie, les valeurs primaires peuvent être déterminées en exécutant des combinaisons de valeurs de contrôle d'appareil via le modèle d'appareil. Pour les appareils de capture, cette approche n'est pas bien adaptée à la création du GBD de référence, car il est presque impossible de capturer une image qui produit une valeur d'appareil pur entièrement saturée, telle que (0.0, 0.0, 1.0). Les profils d'appareil WCS contiennent les index des primaires dans la cible de capture. Étant donné que ces valeurs ne sont pas contenues dans un profil ICC, utilisez des valeurs mesurées à partir d'une cible de scanneur classique après la conversion en Jab, par rapport aux conditions d'affichage ICC.

Définition de l'axe neutre dans la description de la limite de gamut

Les méthodes de mappage de la gamme HueMap et Relative MinCD utilisent l'axe neutre de l'appareil pour le redressement. Pour les périphériques de sortie de base, l'axe neutre peut être déterminé en exécutant des valeurs neutres d'appareil (R=G=B ou C=M=Y) via la méthode DeviceToColorimetric, puis via la méthode ColorimetricToAppearance de l'objet CIECAM02. Toutefois, les appareils de capture ne retournent pas toujours une valeur neutre d'appareil lorsqu'ils sont présentés avec un échantillon neutre. Cela est particulièrement vrai lorsque l'éclairage ambiant n'est pas parfaitement neutre. Les profils d'appareil WCS contiennent les index des échantillons neutres dans la cible. Utilisez ces exemples pour définir l'axe neutre. Étant donné que ces informations ne sont pas disponibles pour les profils ICC, vous devez utiliser la même méthode que celle utilisée pour les périphériques de sortie ; exécutez des exemples de périphérique neutres via la méthode DeviceToColorimetric, puis couplez les valeurs d'entrée et les résultats colorimétriques.

Rubriques connexes

Concepts de base de la gestion des couleurs

Schémas et algorithmes du système de couleurs Windows

Commentaires

Algorithmes de création de transformation WCS

Article • 13/06/2023

Création de transformations

Pour expliquer correctement le fonctionnement des transformations de couleur, il est utile d'expliquer le chemin de traitement complet à la fois par ICM 2.0 et les éléments internes de l'objet CTE. La fonction **CreateColorTransformW** ICM 2.0 crée une transformation de couleur que les applications peuvent utiliser pour effectuer la gestion des couleurs. Cette fonction crée un contexte de couleur à partir des entrées **LOGCOLORSPACE** et intention. Les intentions sont mappées à des corrélations d'algorithme de mappage de gamut ICC de base. La fonction appelle ensuite la fonction ICM 2.0 **CreateMultiProfileTransform** pour un traitement cohérent des couleurs. La fonction **CreateColorTransform** copie généralement les données dans la structure de transformation optimisée interne.

La fonction CreateMultiProfileTransform ICM 2.0 accepte un tableau de profils et un tableau d'intentions ou un profil de liaison d'appareil unique, et crée une transformation de couleur que les applications peuvent utiliser pour effectuer un mappage de couleurs. Il traite ces profils et intentions d'entrée pour créer des modèles d'appareil, des modèles

d'apparence de couleur, des descriptions de limites de gamut et des modèles de mappage de gamut. Voici comment procéder :

- Les modèles d'appareil sont initialisés directement à partir des profils DM. Un modèle d'appareil est créé pour chaque profil dans l'appel à **CreateMultiProfileTransform**.
- Les modèles d'apparence de couleur sont initialisés directement à partir des profils CAM. Il existe un profil CAM pour chaque profil dans l'appel à **CreateMultiProfileTransform**. Toutefois, le même profil CAM peut être spécifié pour plusieurs profils.
- Les descriptions des limites de gamut sont initialisées à partir d'un objet de modèle d'appareil et d'un objet CAM. Il existe une description de limite de gamut pour chaque profil dans l'appel à **CreateMultiProfileTransform**.
- Les modèles de mappage de gamut sont initialisés à partir de deux limites de gamut et d'une intention. Vous devez créer un modèle de mappage de gamut entre chaque paire de modèles d'appareil créée à partir de l'appel à **CreateMultiProfileTransform**. Notez que cela signifie que vous utilisez un modèle de carte de gamut de moins que le modèle d'appareil. Étant donné que le nombre d'intentions correspond au nombre de modèles d'appareil, il y a également une intention de plus que nécessaire. La première intention de la liste est ignorée. Vous parcourez la liste des modèles d'appareils et des intentions, en créant des modèles de mappage de gamut. Récupérez les premier et deuxième modèles d'appareil et la deuxième intention, puis initialisez le premier modèle de mappage de gamut. Récupérez les deuxième et troisième modèles d'appareil et la troisième intention, puis initialisez le deuxième modèle de mappage de gamut. Continuez de cette manière jusqu'à ce que vous ayez créé tous les modèles de mappage de gamut.

Lorsque les profils ont été correctement traités et que tous les objets intermédiaires ont été créés et initialisés, vous pouvez créer la transformation CITE avec l'appel suivant. Les *valeurs pDestCAM* et *pDestDM* sont celles associées au dernier profil dans l'appel à **CreateMultiProfileTransform**.

C++

```
HRESULT CreateCITEColorTransform(
 __inout     IDeviceModel          *pSourceDM,
 __inout     IColorAppearanceModel *pSourceCAM,
 __in        GamutMapArray         *pGamutMapArray,
 __inout     IColorAppearanceModel *pDestCAM,
 __inout     IDeviceModel          *pDestDM,
             EColorTransformMode    eTransformMode,
 __deref_out IColorTransform      **ppCTS
 );
```

Prise en charge des plug-ins

Un problème lié à la configuration de la liste de transformation consiste à vérifier si un plug-in requis est disponible. Le commutateur de modèle suivant fournit cette stratégie pour contrôler ce comportement. La gestion de cette liste de transformations est une méthode dans la structure de transformation optimisée interne, mais chaque méthode de modèle fournit le pointeur vers elle-même et son propre ensemble de valeurs de paramètres.

Le mode doit être l'un des suivants.

- TfmRobust : si un profil de mesure spécifie un plug-in préféré et que le plug-in n'est pas disponible, le nouveau système DTE utilise le plug-in de base. Si aucun plug-in n'est disponible, la transformation signale une erreur.
- TfmStrict : si ColorContext spécifie un plug-in préféré, le plug-in doit être disponible. Si aucun plug-in préféré n'est trouvé, le plug-in de base est utilisé. Si aucun plug-in n'est disponible, la transformation signale une erreur.
- TfmBaseline : seuls les plug-ins de base peuvent être utilisés dans AddMeasurementStep. Si colorContext spécifie un plug-in préféré, le plug-in est ignoré. Si le plug-in de base n'est pas disponible, la transformation signale une erreur.

Exécution de la transformation

La fonction **TranslateColors** de l'API ICM 2.0 traduit un tableau de couleurs de l'espace de couleur source vers l'espace de couleur de destination tel que défini par une transformation de couleur. Cette fonction vérifie en interne un tableau de couleurs mises en cache pour permettre une correspondance immédiate des couleurs couramment transformées. Cette transformation prend en charge les tableaux d'octets 8 bits par canal et les tableaux float 32 bits par canal. Tous les autres formats seront convertis avant de passer à la nouvelle table DTE.

La fonction **TranslateBitmapBitBitBits** de l'API ICM 2.0 traduit les couleurs d'une bitmap ayant un format défini pour produire une autre bitmap dans un format demandé. Cette fonction vérifie en interne un tableau de couleurs mises en cache pour permettre une correspondance immédiate des couleurs couramment transformées. Pour éviter un trop grand nombre de chemins de code, la prise en charge et la complexité des tests, seuls un nombre limité de formats bitmap sont réellement pris en charge dans le moteur de transformation et d'interpolation. Cette fonction doit traduire les formats bitmap entrants et sortants non natifs en formats pris en charge en mode natif pour le traitement. Cette transformation prend uniquement en charge les bitmaps d'octets 8

bits par canal et les bitmaps flottantes 32 bits par canal. Tous les autres formats seront convertis avant de passer à la nouvelle table DTE.

Exécution séquentielle de transformation

Si le paramètre *dwFlags* a le SEQUENTIAL_TRANSFORM bit défini lorsque les fonctions ICM CreateColorTransformW ou **CreateMultiProfileTransform** sont appelées, les étapes de transformation sont exécutées séquentiellement. Cela signifie que le code parcoure séparément chaque modèle d'appareil, modèle d'apparence de couleur et modèle de mappage de gamut, comme spécifié par l'appel **CreateColorTransform** ou **CreateMultiProfileTransform** . Cela peut être utile pour le débogage des modules de plug-in, mais il est beaucoup plus lent que l'exécution via une transformation optimisée. L'exécution en mode séquentiel n'est donc pas recommandée pour les logiciels de production. En outre, il peut y avoir de légères différences dans les résultats obtenus en mode séquentiel et en mode optimisé. Cela est dû aux variations introduites lorsque les fonctions sont concaténées ensemble.

Création de transformations optimisées

Une transformation optimisée est une table de choix multidimensionnelle. La table peut être traitée par un moteur d'interpolation multidimensionnel, tel que l'interpolation tétraédrale, qui applique les couleurs d'entrée à la transformation. La section suivante décrit comment les tables de choix optimisées sont créées. La section suivante décrit comment interpoler dans les tables de choix optimisées.

Tables de choix éparses

Les imprimantes conventionnelles ont des encres CMJN. Pour étendre la gamme, une approche consiste à ajouter de nouvelles entrées manuscrites au système. Les encres généralement ajoutées sont des couleurs que les encres CMJN ont des difficultés à reproduire. Les choix courants sont l'orange, le vert, le rouge, le bleu, etc. Pour augmenter la « résolution apparente », vous pouvez utiliser des encres avec des teintes différentes, par exemple du cyan clair, du magenta clair, etc. En effet, le périphérique d'imprimante a plus de quatre canaux.

Bien que les imprimantes soient des périphériques de sortie, elles effectuent également la conversion de couleur de l'espace de l'appareil vers un autre espace de couleurs. Dans le cas d'une imprimante CMJN, il s'agit d'une transformation de CMJN en XYZ, ou le « modèle avant » de l'imprimante. En combinant le modèle de transfert avec d'autres

transformations, il est possible d'émuler une impression CMJN sur un autre appareil. Par exemple, une imprimante CMJN sur un moniteur RVB rend possible un mécanisme de vérification qui émule une impression de cette imprimante CMJN sur un moniteur. De même, il en va de même pour les imprimantes hi-fi. Une conversion CMYKOG en RVB permet la vérification de l'imprimante CMYKOG sur un moniteur.

L'approche conventionnelle pour implémenter une telle conversion de couleur consiste à utiliser un LUT uniforme. Par exemple, dans un profil ICC pour une imprimante CMYKOG, la spécification ICC impose une étiquette A2B1 qui stocke un LUT uniforme représentant un échantillonnage uniforme dans l'espace d'appareil CMYKOG du modèle avant, qui va de CMYKOG à l'espace de connexion du profil ICC (CIELAB ou CIEXYZ). Le profil de liaison d'appareil ICC permet une transformation directe de l'espace d'appareil CMYKOG vers n'importe quel espace de couleur, y compris un espace d'appareil, également sous la forme d'un LUT échantillonné uniformément dans l'espace CMYKOG. L'échantillonnage n'est jamais effectué avec 256 niveaux (profondeur de bits 8) en raison de l'énorme LUT résultant, sauf dans le cas des appareils monochromes (1 canal). Au lieu de cela, l'échantillonnage avec une profondeur de bits inférieure est utilisé ; certains choix typiques sont 9 (profondeur de bits 3), 17 (profondeur de bits 4), 33 (profondeur de bits 5). Avec le nombre de niveaux inférieur à 256 dans chaque canal, le LUT est utilisé conjointement avec un algorithme d'interpolation pour produire le résultat si un niveau se trouve entre deux niveaux échantillonné.

Bien qu'un LUT uniforme soit conceptuellement simple à implémenter et que l'interpolation sur un LUT uniforme soit généralement efficace, la taille du LUT augmente de façon exponentielle avec la dimension d'entrée. En fait, si d est le nombre d'étapes utilisé dans le LUT uniforme et n le nombre de canaux dans l'espace de couleur source, le nombre de nœuds dans le LUT d^n. De toute évidence, le nombre de nœuds demande rapidement tellement de stockage en mémoire que même les systèmes informatiques de pointe ont des difficultés à gérer la demande. Pour les appareils avec six ou huit canaux, une implémentation ICC du profil d'appareil nécessite l'utilisation de quelques étapes dans le LUT, parfois même jusqu'à cinq étapes dans la table A2B1 pour conserver le profil en mégaoctets au lieu de gigaoctets. De toute évidence, l'utilisation d'un plus petit nombre d'étapes augmente l'erreur d'interpolation, car il y a maintenant moins d'échantillons. Étant donné que le LUT doit être uniforme, la précision sur l'ensemble de l'espace colorimétrique est dégradée même dans les régions de l'espace où une différence de couleur significative peut être causée par une petite modification de la valeur de l'appareil.

Dans les appareils avec plus de quatre colorants, certains sous-espaces de l'espace de l'appareil sont plus importants que d'autres. Par exemple, dans l'espace CMYKOG, les encres cyan et verte sont rarement utilisées ensemble, car leurs teintes se chevauchent en grande partie. De même, les encres jaunes et oranges se chevauchent en grande

partie. Une réduction uniforme du nombre d'étapes peut être vue comme une dégradation globale de la qualité dans l'ensemble de l'espace de couleur, ce qui est quelque chose que vous pouvez vous permettre pour les combinaisons d'encre improbables, mais pas pour les combinaisons probables ou importantes.

Bien qu'un LUT échantillonné uniformément soit simple et efficace pour l'interpolation, il impose des exigences de mémoire énormes à mesure que la dimension augmente. En réalité, alors qu'un appareil peut avoir six ou huit canaux, ils sont rarement utilisés simultanément. Dans la plupart des cas, la couleur d'entrée en transformation de couleur n'a que quelques colorants « actifs » et réside donc dans un espace de couleurs de dimension inférieure. Cela signifie également que l'interpolation peut être effectuée plus efficacement dans cet espace de dimension inférieure, car l'interpolation est plus rapide lorsque la dimension est inférieure.

Par conséquent, l'approche consiste à stratifier l'ensemble de l'espace de l'appareil en sous-espaces de différentes dimensions. Et parce que les dimensions inférieures (celles combinant trois ou quatre colorants) sont plus importantes, en stratifiant l'espace, vous pouvez également appliquer des taux d'échantillonnage différents; c'est-à-dire un nombre différent d'étapes, pour les pièces; augmenter les taux d'échantillonnage pour les dimensions inférieures, les réduire pour les dimensions plus élevées.

Pour corriger les notations, *n* est le nombre de canaux dans l'espace de couleur source de la transformation de couleur que vous souhaitez échantillonner. Vous pouvez également faire référence à *n* comme dimension d'entrée, et $n \geq 5$, sauf indication contraire.

Les blocs de construction de base sont des unités logiques de *dimensions* et de tailles d'entrée différentes, au lieu d'un seul LUT uniforme avec la dimension *d'entrée n*. Pour être plus précis, un*LUT* est un réseau rectangulaire imposé sur un cube d'unité ; autrement dit, toutes les coordonnées de l'appareil sont normalisées à la plage [0, 1]). si est la dimension d'entrée du lut (notez qu'il n'est pas nécessaire d'être égal à *n*, bien que $\nu \leq n$ soit requis), il se compose de ν grilles d'échantillonnage unidimensionnelles :

Samp *i* : $x_1, x_2, \ldots, x_{d(i)}$

où tous les x_j*s* doivent se trouver dans la plage [0, 1], $d(i)$ représente le nombre d'étapes pour l'échantillonnage du *i* ième canal qui doit être d'au moins 1, et $x_{d(i)}$ doit être égal à 1. En revanche, x_1 Il n'est pas nécessaire d'avoir la valeur 0.

Seuls les deux cas spéciaux suivants de LUT seront définis.

LUT fermé : il s'agit d'un LUT avec l'exigence supplémentaire que pour chaque Samp*ᵢ*, $x_1 = 0$, et $d(i) \geq 2$. Un LUT fermé uniforme est un LUT fermé qui a le même $d(i)$ pour

chaque canal, et les nœuds sont espacés uniformément entre 0 et 1.

Open LUT : il s'agit d'un LUT avec l'exigence supplémentaire qui, pour chaque Samp *i*, $x_1 > 0$. Il est correct d'avoir $d(i) = 1$.

L'objectif est de stratifier le cube d'unités [0, 1] *n* en une collection de LTU fermées et de LUT ouvertes, de sorte que la collection entière couvre le cube d'unités. Il est conceptuellement plus simple d'organiser ces « strates LUT » en fonction de leurs dimensions, de sorte qu'au niveau supérieur :

$$[0,1]^n = \bigcup_{k=3}^{n} \Sigma_k$$

où Σ_k est la collection « *k* -dimensional strata ». Notez que la dimension de strates commence par 3 au lieu de 0 ; c'est-à-dire des points, car l'interpolation de combinaisons de trois colorants peut être gérée sans trop de besoin de mémoire.

Description des strates LUT

Dans cette implémentation :

1. Σ_3 se compose de LUT fermés avec trois entrées, une parmi chaque combinaison possible de trois colorants choisis parmi les *n* colorants.

2. Σ_4 se compose d'un LUT fermé pour la combinaison CMJN (ou les quatre premiers colorants), ainsi que $\binom{n}{4} - 1$ luts ouverts pour toutes les autres combinaisons de quatre colorants. En sélectionnant la combinaison CMJN, vous reconnaissez qu'il s'agit d'une combinaison importante.

3. Pour $k = 5, \ldots, n$, Σ_k se compose de $\binom{n}{k}$ open LUTs, un pour chaque combinaison possible de choix *k* colorants parmi le total de *n* colorants.

Il reste à spécifier les tailles des LUT. La principale différence entre les luts ouverts et fermés est que les luT ouverts ne se chevauchent pas, et que les luT fermés peuvent se chevaucher aux faces de la limite. Le fait que l'échantillonnage unidimensionnel dans un LUT ouvert ne contienne pas 0 signifie essentiellement qu'un LUT ouvert manque la moitié des faces limites, d'où le nom « open ». Si deux luT ne se chevauchent pas, vous pouvez utiliser un nombre différent d'étapes ou d'emplacements de nœud dans chaque canal. La même chose n'est pas vraie si deux luT se chevauchent. Dans ce cas, si le nombre d'étapes ou les emplacements des nœuds sont différents, un point situé à

l'intersection des deux luT reçoit une valeur d'interpolation différente selon le LUT utilisé dans l'interpolation. Une solution simple à ce problème consiste à utiliser un échantillonnage uniforme avec le même nombre d'étapes chaque fois que deux luT se chevauchent. En d'autres termes :

Tous les LUT fermés (tous les LUT à trois colorants et le LUT CMJN dans cette implémentation) doivent être uniformes et avoir le même nombre d'étapes, qui sont indiquées *d*.

Les deux algorithmes suivants peuvent être utilisés pour déterminer le nombre d'étapes *d* pour les unités logiques fermées et le nombre d'étapes pour les unités logiques ouvertes.

Algorithme n° 1

Cet algorithme ne nécessite pas d'entrée externe.

Toutes les unités logiques fermées sont uniformes avec *un nombre d'étapes* .

Toutes les unités logiques ouvertes de la dimension *k* auront le même nombre d'étapes $d(k)$ dans chaque canal d'entrée, et les nœuds sont également espacés ; autrement dit, pour chaque $i = 1,2,\ldots,k$.

Samp *i* : $1/d(k), 2/d(k), \ldots, (d(k)-1)/d(k), 1$

Enfin, spécifiez *d* et *d* (*k*) dans le tableau 1 suivant. Les trois modes , « preuve », « normal » et « meilleur » sont les paramètres de qualité ICM 2.0. Dans cette implémentation, le mode de preuve a l'encombrement mémoire le plus faible, et le meilleur mode a l'encombrement mémoire le plus élevé.

Pour implémenter cet algorithme, vous devez appeler l'algorithme suivant n°2. Les utilisateurs peuvent spécifier leurs propres emplacements d'échantillonnage, en utilisant les tables comme guide.

Algorithme n°2

Cet algorithme nécessite une entrée externe sous la forme d'une liste d'emplacements d'échantillonnage « importants », mais il est plus adaptatif et peut potentiellement économiser de l'espace mémoire.

L'entrée requise est un tableau de valeurs d'appareil fourni par l'utilisateur. Ces valeurs d'appareil indiquent quelle région de l'espace de couleur de l'appareil est importante ; autrement dit, quelle région doit être échantillonné le plus.

Toutes les unités logiques fermées seront uniformes avec *un nombre d'étapes* , comme décrit dans Algorithme n°1. Les valeurs de *d* sont fournies dans le tableau 1.

(a) Uniform Closed LUT

	Mode preuve	**Mode Normal**	**Meilleur mode**
D	9	17	33

(b) Ouvrir LUT

Dimension d'entrée	**Mode preuve**	**Mode Normal**	**Meilleur mode**
4	5	7	9
5	2	3	3
6	2	3	3
7	2	2	2
8 ou plus	2	2	2

Tableau 1 : Tailles LUT utilisées dans l'algorithme

Chaque LUT ouvert peut avoir un nombre différent d'étapes dans chaque canal d'entrée, et les emplacements d'échantillonnage n'ont pas besoin d'être également espacés. Pour une strate LUT ouverte donnée, il existe une combinaison de colorant associée, par exemple, $c_1,\cdots,c_k$, où les c_i s sont des entiers distincts compris entre 1 et *n*. Il s'agit des indices de canal correspondant aux colorants « actifs » de cette strate.

ÉTAPE 1 : Filtrez le tableau entré de valeurs d'appareil qui ne sont pas contenues dans cette couche. Une valeur d'appareil $(x_1, x_2, \cdots, x_n)$ est contenu dans les strates, si et uniquement si $x_{c_1} > 0, x_{c_2} > 0, \ldots, x_{c_k} > 0$ tous les autres canaux sont 0. Si l'ensemble filtré a *N* entrées, laissez

$$d_{tentative}(k) = \min(dMax(k), \max(1, \text{int}(N^{1/k})))$$

Pour chaque $i = 1,2,\ldots,k$, itérer les étapes suivantes 2 à 5 :

ÉTAPE 2 : Si $d_{tentative}(k) = 1$, Samp *i* n'a que 1 point, qui doit être 1.0. Passez au *i* suivant. Sinon, passez à l'ÉTAPE 3.

ÉTAPE 3 : Triez les exemples filtrés dans l'ordre croissant dans le canal c_i

ÉTAPE 4 : Définir la grille d'échantillonnage « provisoire » à l'aide des nœuds

$x_j = (j-1) \cdot 100/(d_{tentative}(k)-1)$

où $j = 1, 2, \ldots, d_{tentative}(k)$

ÉTAPE 5 : Régulariser la grille provisoire pour vous assurer qu'elle est conforme à la monotonicité stricte et qu'elle se termine par 1.0. Étant donné que le tableau est déjà trié, les nœuds de la grille provisoire sont déjà monotoniques nondécasés. Toutefois, les nœuds adjacents peuvent être identiques. Vous pouvez résoudre ce problème en supprimant des nœuds identiques, si nécessaire. Enfin, après cette procédure, si le point de terminaison est inférieur à 1.0, remplacez-le par 1.0.

Notez que l'ÉTAPE 5 est la raison pour laquelle les couches LUT peuvent avoir un nombre d'étapes différent dans chaque canal. Après la régularisation, le nombre d'étapes dans un canal peut être inférieur à $d_{tentative}(k)$

Interpolation

Vous pouvez construire la stratification du cube d'unités en utilisant des strates LUT ouvertes et des strates LUT fermées. Pour effectuer l'interpolation à l'aide de cette « structure LUT éparse », procédez comme suit. Supposons qu'une valeur de périphérique d'entrée donnée $(x_1, x_2, \ldots, x_n)$.

ÉTAPE 1 : Déterminez le nombre de canaux « actifs ». Il s'agit du nombre de canaux autres que zéro. Cela détermine la dimension *k* des strates à rechercher pour la strate contenante. Plus précisément, la dimension des strates est 3 si le nombre de canaux actifs est ≤ 3 Sinon, la dimension des strates est identique au nombre de canaux actifs.

ÉTAPE 2 : Dans Σ_k , recherchez la strate contenante. Une valeur d'appareil est contenue dans une couche ouverte si tous les canaux correspondant à la strate ont une valeur différente de zéro, et si tous les autres canaux sont zéro. Une valeur d'appareil est contenue dans une couche fermée si chaque canal non représenté par la strate est égal à zéro. Si aucune strate contenante n'est trouvée, il existe une condition d'erreur. Annuler et signaler l'échec. Si une couche contenante est trouvée, passez à l'étape suivante.

ÉTAPE 3 : Si la strate contenante est fermée, l'interpolation au sein de la strate peut être effectuée par n'importe quel algorithme d'interpolation connu. Dans cette implémentation, le choix de l'algorithme est l'interpolation tétraédrale. Si la strate

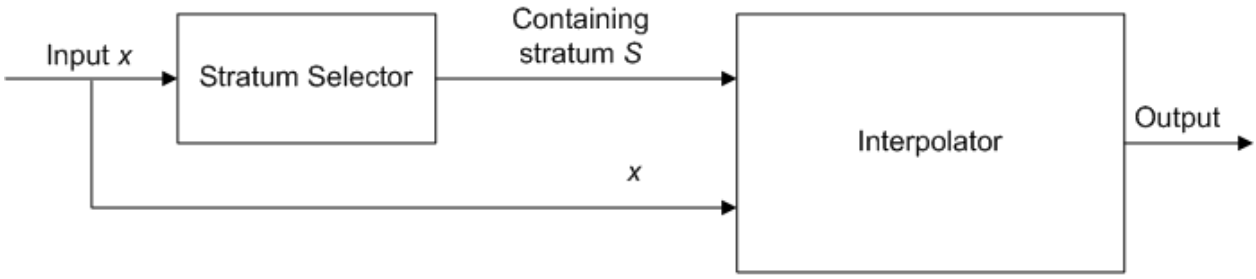
contenante est ouverte et que la valeur de l'appareil se trouve strictement à l'intérieur de la strate, c'est-à-dire,

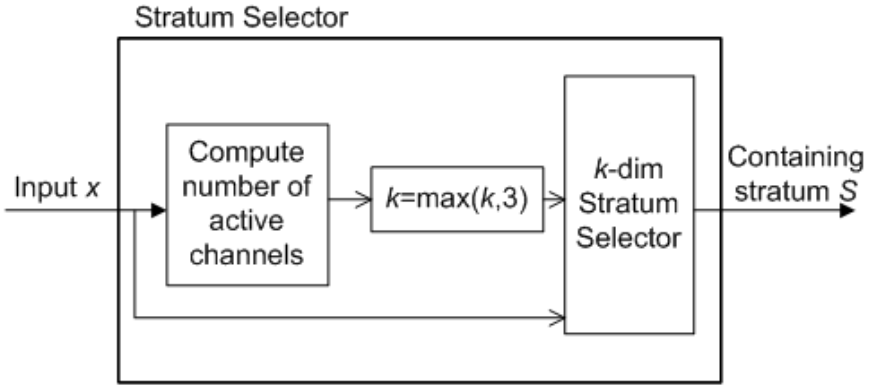
$x_i \geq$ premier nœud dans i ième canal

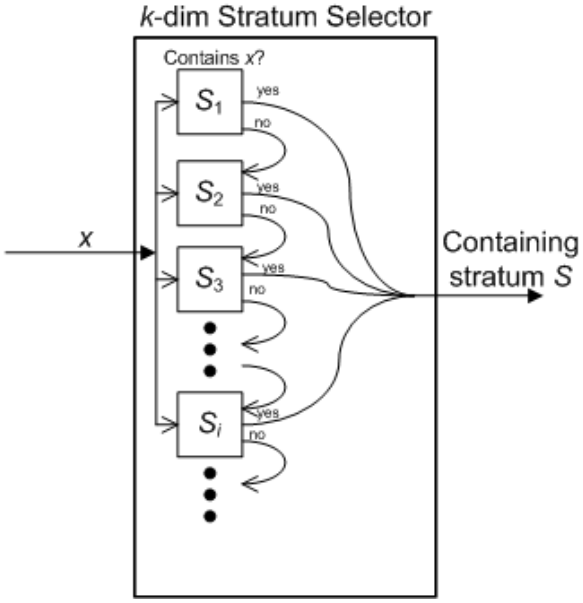
où i est un index de canal pour la strate, puis l'algorithme d'interpolation standard, tel que l'interpolation tétraédrale, fonctionne.

Si $x_i <$ premier nœud dans i th canal pour certains i, la valeur de l'appareil tombe dans l'« écart » entre la strate et les sous-espaces de dimension inférieure. Ce MOI n'est pas concerné par un algorithme d'interpolation en soi. Par conséquent, n'importe quel algorithme d'interpolation peut être utilisé pour interpoler dans cette « lacune », bien que l'algorithme préféré soit l'interpolation transfinite suivante.

L'architecture du module d'interpolation est illustrée dans les deux parties de la figure 1.


Input x
Stratum Selector
Containing stratum S
x
Interpolator
Output


Stratum Selector
Input x
Compute number of active channels
k=max(k,3)
k-dim Stratum Selector
Containing stratum S


k-dim Stratum Selector
Contains x?
S1
S2
S3
Si
yes
no
x
Containing stratum S

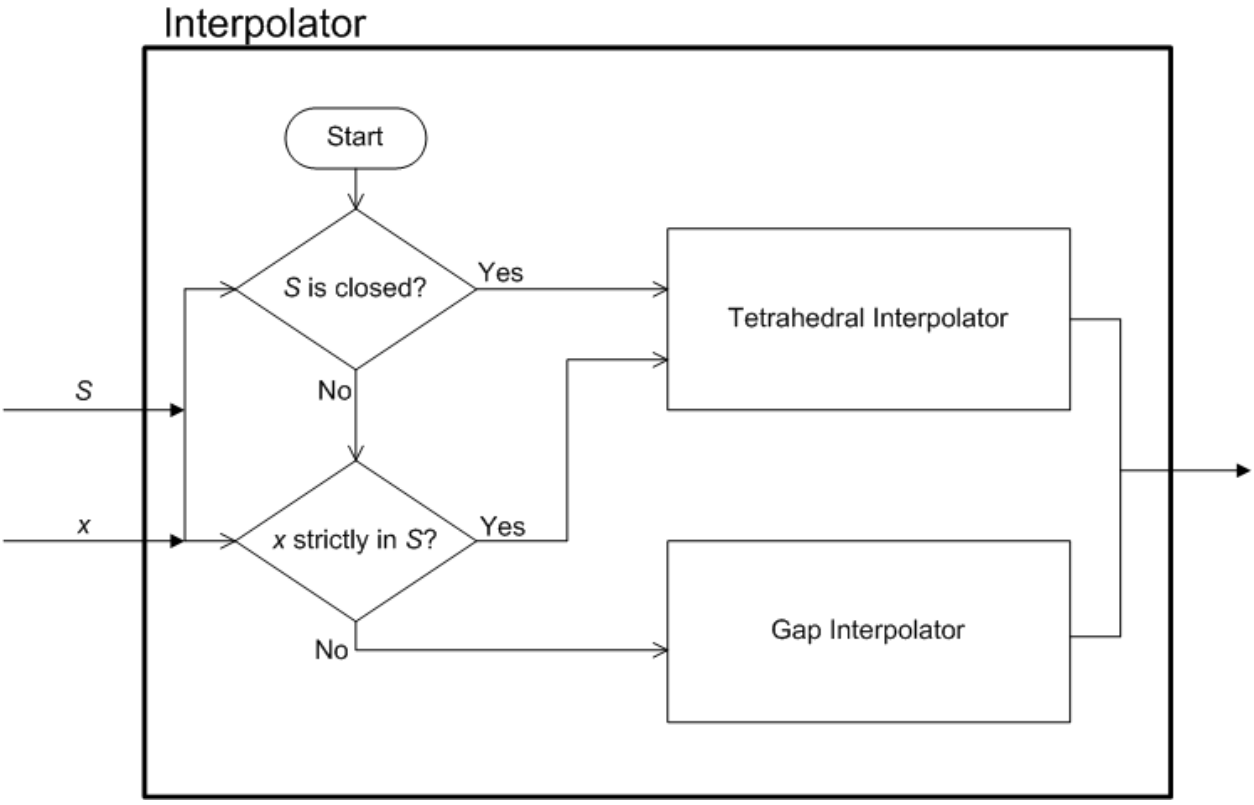


Figure 1 : Architecture du module d'intepolation

Comme expliqué précédemment, cet algorithme est capable d'effectuer un échantillonnage relativement dense dans les régions de l'espace de l'appareil qui contiennent une combinaison importante de colorants, tout en réduisant la taille totale des LUT nécessaires. Le tableau suivant présente une comparaison du nombre de nœuds nécessaires pour l'implémentation LUT éparse (à l'aide de l'algorithme n°1 et du mode normal) et de l'implémentation uniforme LUT correspondante.

Nombre de canaux d'entrée	LUT partiellement alloué	LUT uniforme
5	142498	1419857
6	217582	24137567
7	347444	410338673
8	559618	6975757441

Interpolation dans un cube d'unité

Une étape de base dans le cas d'une grille rectangulaire est l'interpolation dans une cellule englobante. Pour un point d'entrée, vous pouvez facilement déterminer la cellule englobante. Dans une grille rectangulaire, la valeur de sortie à chacun des sommets

(points d'angle) de la cellule englobante est spécifiée. Il s'agit également des seules conditions limites (BC) auxquelles un interpolant doit satisfaire : l'interpolant doit passer par tous ces points. Notez que ces conditions limites se trouvent sur des points « discrets », dans ce cas les 2n points d'angle de la cellule, où n est la dimension de l'espace de couleurs.

Il est utile de formaliser le concept de conditions limites avant de passer à autre chose. Pour tout sous-ensemble S de la limite de la cellule englobante (cube d'unité dans n dimensions), une condition de limite sur S est une spécification d'une fonction BC : S → Rm, où m est la dimension de sortie. En d'autres termes, un interpolant, qui peut être nommé Interp : [0,1]n→ Rm, est requis pour satisfaire : Interp(x) = BC(x) pour tous les x dans S.

Dans le scénario standard d'interpolation sur le cube d'unité, S est l'ensemble de points discrets qui sont les 2n sommets du cube.

Vous pouvez maintenant généraliser les conditions limites pour résoudre les problèmes décrits précédemment et fournir un nouvel algorithme d'interpolation dans le cube d'unité. Au lieu d'autoriser uniquement des points de limite discrets, des conditions limites peuvent être imposées à une face limite entière du cube. Les hypothèses précises sont les suivantes :

(a) Le point vn =(1,1,...,1) est spécial et seule une condition de limite discrète est autorisée. En d'autres termes, aucune condition de limite continue ne peut être imposée aux n faces limites xi=1 (i=1,...,n).

(b) Pour chacune des n faces de limite restantes xi=0 (i=1,...,n), une condition limite peut être imposée à l'ensemble de la face, avec la condition de compatibilité selon laquelle si deux faces se croisent, les conditions limites des visages doivent convenir de l'intersection.

(c) Tous les sommets non contenus dans les visages avec condition limite auront une condition limite individuelle (discrète).

Vous pouvez faire référence à une condition de limite discrète en tant que données finies et à une condition de limite continue en tant que données transfinites pour discuter de l'interpolation sur les données finies et transfinites.

Tout d'abord, passez en revue l'interpolation tétraédrique standard (comme celle utilisée dans le brevet de Sakamoto) qui permet de définir les notations pour cette formulation particulière du problème. On sait que le cube d'unité [0,1]n peut être subdivisé en n! tétraèdre, paramétré par l'ensemble de permutations sur n symboles. Plus spécifiquement, chacun de ces tétraèdre est défini par les inégalités

$$x_{\sigma(1)} \geq x_{\sigma(2)} \geq \ldots \geq x_{\sigma(n)}$$

où σ:{1,2,..,n}→{1,2,...,n} est une permutation de « symboles » 1, 2, ..., n, c'est-à-dire un mappage bijectif de l'ensemble de n symboles. Par exemple, si n = 3 et σ = (3, 2, 1), ce qui signifie σ(1)=3, σ(2)=2, σ(3)=1, le tétraèdre correspondant est défini par z≥y≥x, où la notation commune x, y, z est utilisée pour x1, x2, x3. Notez que ces tétraèdres ne sont pas disjoints les uns des autres. Aux fins de l'interpolation, les points situés sur une face commune de deux tétraèdres distincts auront la même valeur d'interpolation, quel que soit le tétraèdre utilisé dans l'interpolation. Néanmoins, dans le scénario standard d'interpolation sur des points finis, pour un point d'entrée donné (x1, ..., xn), déterminez d'abord le tétraèdre dans lequel il se trouve, ou de façon équivalente, la permutation correspondante σ, puis l'interpolant tétraédral est défini comme étant

$$Interp(\mathbf{x}) = BC(\mathbf{v}_0) + \sum_{i=1}^{n} x_i \left[BC(\mathbf{v}_i) - BC(\mathbf{v}_{i-1}) \right]$$

où $\mathbf{v}_0 = \mathbf{0}, \mathbf{v}_i = \sum_{j=1}^{i} \mathbf{e}_{\sigma(j)}$ pour i=1, ..., n et e1, ..., en sont les vecteurs de base standard. Avant de passer à la généralisation, notez que v0, v1, ..., vn sont les sommets du tétraèdre et $1 - x_{\sigma(1)}, x_{\sigma(1)} - x_{\sigma(2)}, x_{\sigma(2)} - x_{\sigma(3)}, \ldots, x_{\sigma(n)}$

Pour le cas général des contrôleurs de bord sur les faces limites, vous pouvez utiliser le concept de projection barycentrique. Comme précédemment, pour un point d'entrée donné (x1, ..., xn), déterminez d'abord dans quel tétraèdre il se trouve, ou de manière équivalente, la permutation correspondante σ. Effectuez ensuite une série de projections barycentriques, comme suit. La première projection $\mathrm{BProj}_1(\mathbf{x})$ envoie le point au plan $x_{\sigma(1)} = 0$ Sauf si $\mathbf{x} = \mathbf{v}_n$ Dans ce cas, il n'est pas modifié. La définition précise de la carte BProj est définie comme suit :

$$\mathrm{BProj}_k(\mathbf{x}) = \begin{cases} \mathbf{p}_k + (\mathbf{x} - \mathbf{p}_k)/(1 - x_{\sigma(k)}) & \text{if } \mathbf{x} \neq \mathbf{p}_k \\ \mathbf{x} & \text{if } \mathbf{x} = \mathbf{p}_k \end{cases}$$

avec $\mathbf{p}_k = \mathbf{v}_n - \sum_{i=1}^{k-1} \mathbf{e}_{\sigma(n+1-i)}$ et k = 1, 2, ..., n.

Dans le cas $\mathbf{x} = \mathbf{v}_n$, vous pouvez arrêter, car BC est défini à vn par l'assomption (a). Dans le cas $\mathbf{x} \neq \mathbf{v}_n$, il est clair que $\mathrm{BProj}_1(\mathbf{x})$ a le composant σ(1)th annihilé. En d'autres termes, il se trouve sur l'une des faces limites. Soit il se trouve sur une face sur laquelle BC est défini, auquel cas vous pouvez arrêter, soit vous effectuez une autre projection barycentrique $\mathrm{BProj}_2(\mathbf{x}')\ \mathbf{x}' = \mathrm{BProj}_1(\mathbf{x})$ Et si $\mathbf{x}'' = \mathrm{BProj}_1(\mathbf{x}')$ est sur un visage sur lequel BC est défini, vous pouvez arrêter ; sinon, effectuez une autre projection $\mathrm{BProj}_3(\mathbf{x}'')$

Étant donné que chaque projection annihile un composant, la dimension effective diminue. Vous savez donc que le processus doit s'arrêter. Dans le pire des cas, vous effectuez n projections jusqu'à la dimension 0, c'est-à-dire des sommets sur le cube, qui, selon l'hypothèse (c), vous savez que BC sera défini sur.

En supposant que K projections ont été effectuées, avec

$$\mathbf{x}^{(k)} = \mathrm{BProj}_k(\mathbf{x}^{(k-1)}), k = 1,\ldots, K,$$

x(0)= x, le point d'entrée et BC est défini à x(k). Ensuite, déroulez les projections en définissant une série de vecteurs de sortie :

$$\mathbf{y}^{(k-1)} = x^{(k-1)}_{\sigma(k)} BC(\mathbf{p}_k) + (1 - x^{(k-1)}_{\sigma(k)})\mathbf{y}^{(k)}, k = K, K-1,\ldots,1.$$

où $\mathbf{y}^{(K)} = BC(\mathbf{x}^{(K)})$, et vous obtenez enfin la réponse

$$Interp(\mathbf{x}) = \mathbf{y}^{(0)}$$

Exemple pratique

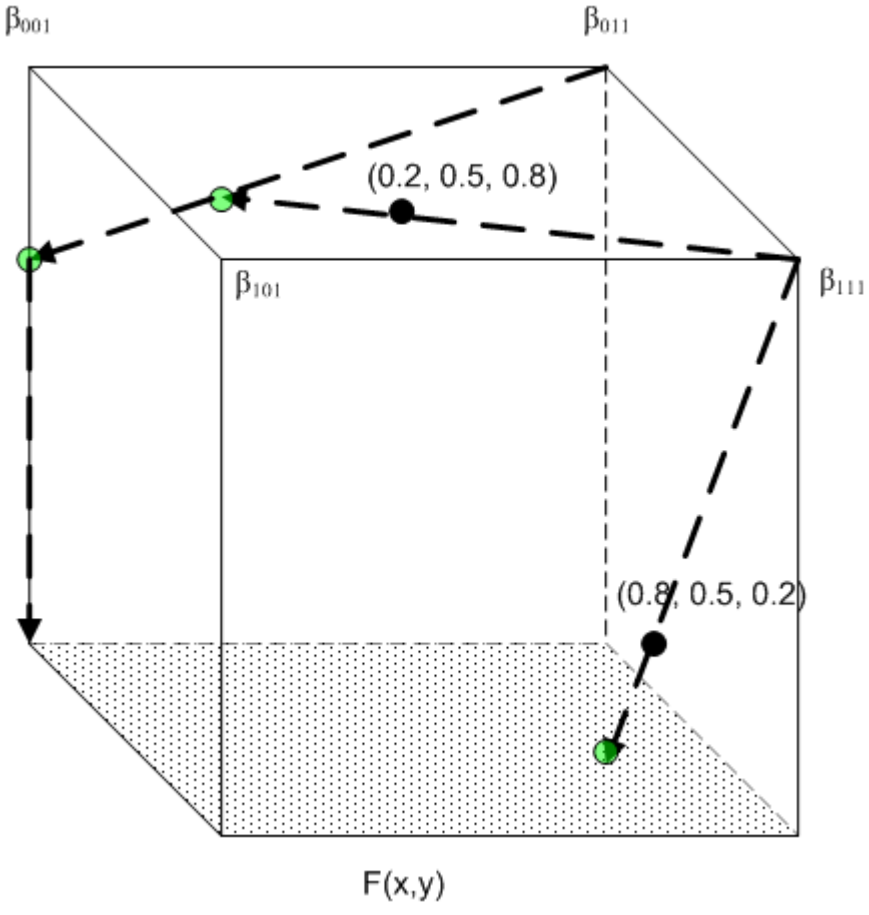


Figure 2 : Exemple de travail

Considérez la situation illustrée à la figure 2, où n = 3, m = 1, et vous avez les contrôleurs de base suivants :

a) Quatre BCs discrets sur les sommets

(0, 0, 1): β001

(0, 1, 1): β011

(1, 0, 1): β101

(1, 1, 1): β111

(b) Bc continue sur le visage x3=0 : F(x1, x2)

Calcul n°1 : point d'entrée x = (0,8, 0,5, 0,2). Le tetraèdre englobant est associé à la permutation <1, 2, 3>.

1ère projection : $\mathbf{p}_1 = (1,1,1), \mathbf{x}^{(1)} = \mathbf{p}_1 + (\mathbf{x} - \mathbf{p}_1)/(1-0.2) = (0.75, 0.375, 0)$

C'est déjà sur le visage x3=0, vous pouvez donc vous arrêter. La substitution vers l'arrière donne alors

$\mathbf{y}^{(0)} = 0.2\beta_{111} + 0.8F(0.75, 0.375)$ qui est la réponse.

Calcul n°2 : point d'entrée x = (0,2, 0,5, 0,8). Le tétraèdre englobant est associé à la permutation <3, 2, 1>.

1ère projection : $\mathbf{p}_1 = (1,1,1), \mathbf{x}^{(1)} = \mathbf{p}_1 + (\mathbf{x} - \mathbf{p}_1)/(1-0.2) = (0, 0.375, 0.75)$

2e projection : $\mathbf{p}_2 = (0,1,1), \mathbf{x}^{(2)} = \mathbf{p}_2 + (\mathbf{x}^{(1)} - \mathbf{p}_2)/(1-0.375) = (0,0,0.6)$

3e projection : $\mathbf{p}_3 = (0,0,1), \mathbf{x}^{(3)} = \mathbf{p}_3 + (\mathbf{x}^{(2)} - \mathbf{p}_3)/(1-0.6) = (0,0,0)$, qui est sur la face x3=0. La substitution vers l'arrière donne alors

$\mathbf{y}^{(2)} = 0.6\beta_{001} + 0.4F(0,0)$

$\mathbf{y}^{(1)} = 0.375\beta_{011} + 0.625\mathbf{y}^{(2)} = 0.375\beta_{011} + 0.375\beta_{001} + 0.25F(0,0)$

$\mathbf{y}^{(0)} = 0.2\beta_{111} + 0.8\mathbf{y}^{(1)} = 0.2\beta_{111} + 0.3\beta_{011} + 0.3\beta_{001} + 0.2F(0,0)$, qui est la réponse finale.

Applications

(a) Interpolation séquentielle de type tetraédral

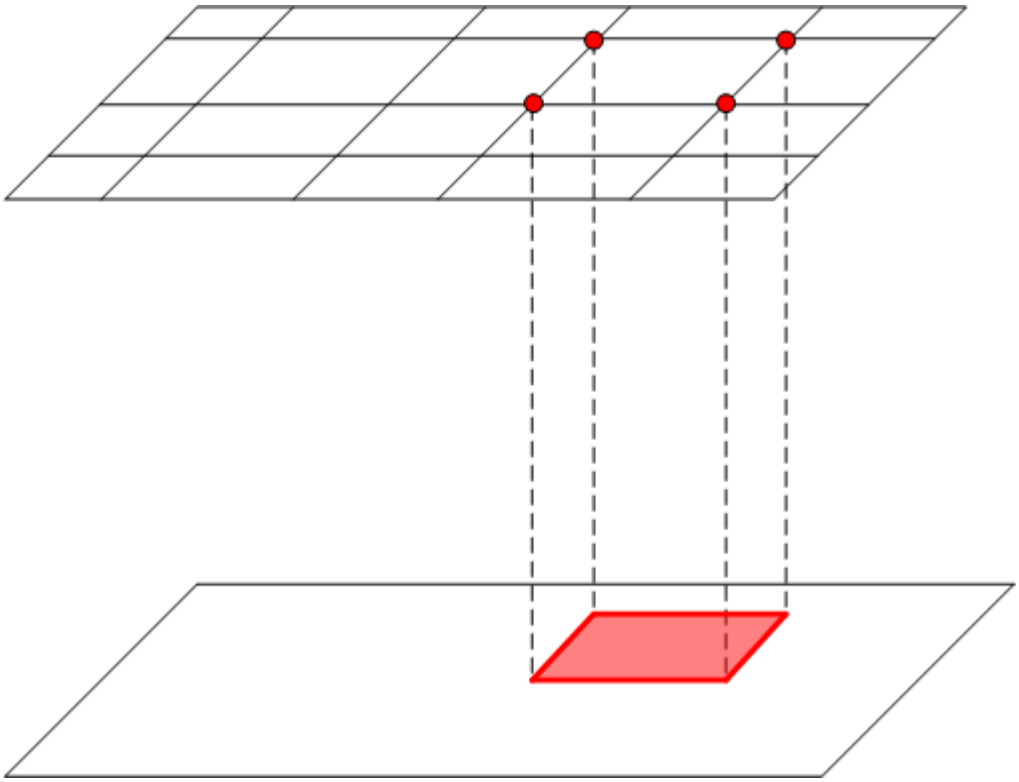


Figure 3 : Interpolation quadriédrale séquentielle

Reportez-vous à la figure 3. Pour interpoler entre deux plans sur lesquels des grilles incompatibles ont été imposées, considérez une cellule englobant un point P donné illustré dans la figure. Les sommets « supérieurs » de la cellule proviennent directement de la grille dans le plan supérieur. Les sommets dans la face inférieure ne sont pas compatibles avec la grille dans le plan inférieur, de sorte que la face entière est traitée comme ayant une bc avec des valeurs obtenues par interpolation sur la grille dans le plan inférieur. Il est alors clair que cette configuration répond aux hypothèses (a), (b) et (c) ci-dessus, et vous pouvez appliquer l'algorithme d'interpolation.

Il est également clair que l'algorithme a réduit la dimension du problème d'interpolation de 1, car le résultat est une combinaison linéaire de valeurs aux sommets dans la grille supérieure et l'interpolation dans le plan inférieur, qui a une dimension moins 1. Si une configuration de plan de sandwich similaire existe dans le plan inférieur, vous pouvez appliquer la procédure dans ce plan, en réduisant la dimension de 1. Cette procédure peut continuer jusqu'à ce que vous atteigniez la dimension 0. Cette cascade de projections et d'interpolations peut être appelée « Interpolation séquentielle de type tetraédral ».

(b) Interpolation d'écart

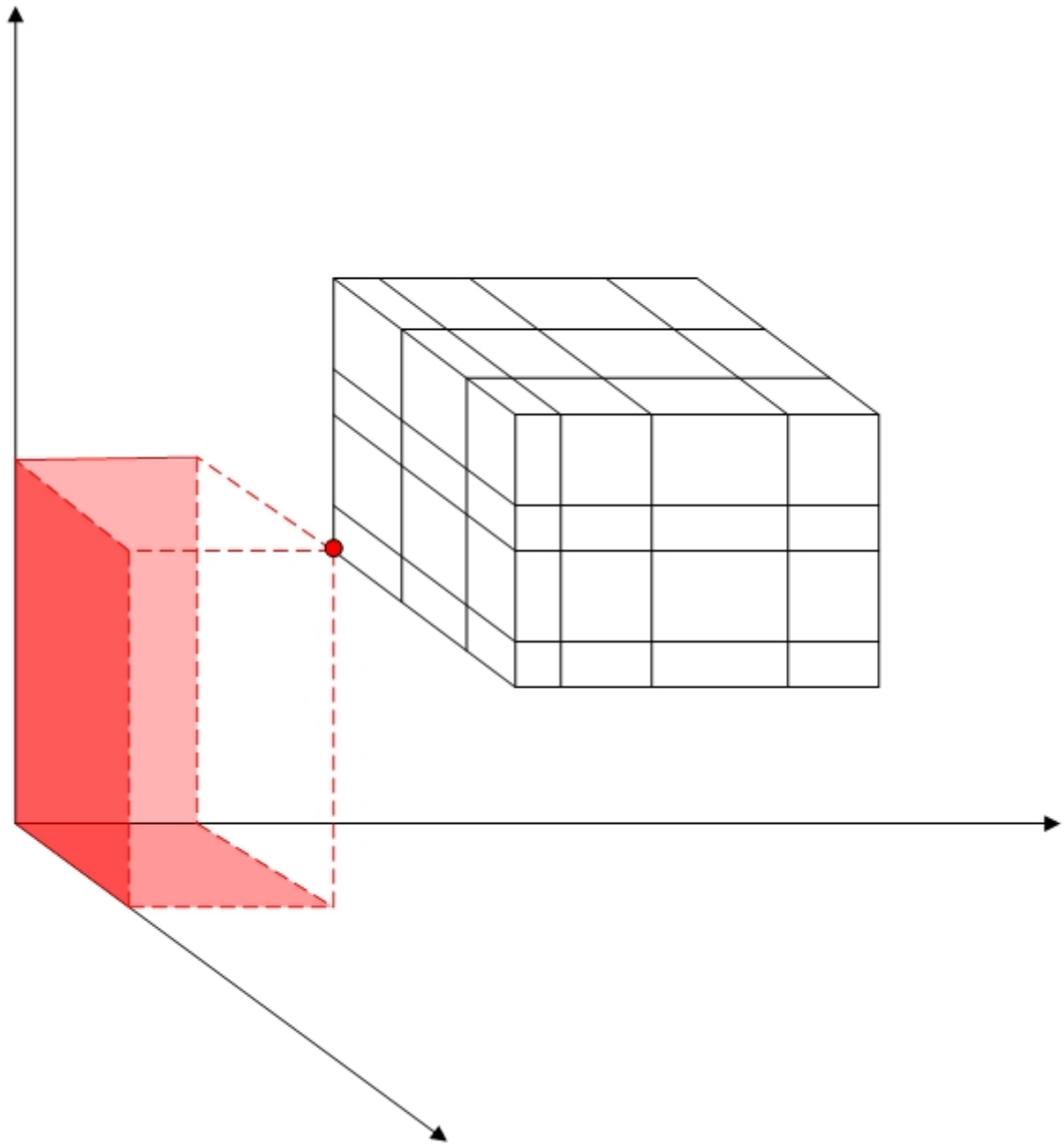


Figure 4 : Interpolation d'écart

Il s'agit d'une grille imposée sur un cube se trouvant strictement à l'intérieur du quadrant positif. Le cube lui-même a une grille, et chaque plan de coordonnées a des grilles qui ne sont pas nécessairement compatibles. L'« écart » entre le cube et les plans de coordonnées a une section croisée « en forme de L » et n'est pas accessible aux techniques standard. Toutefois, avec la technique introduite ici, vous pouvez facilement introduire des cellules qui couvrent cette lacune. La figure 4 illustre l'un de ces éléments. Les grilles sur les plans de coordonnées prennent en charge l'interpolation qui fournit les BC nécessaires pour toutes les faces inférieures de la cellule, avec un sommet restant dont bc est fourni par le coin inférieur du cube.

Note finale sur l'implémentation

Dans l'application réelle, le « cube d'unité » qui est le paramètre de base de l'algorithme est extrait de lattics plus grands, et les valeurs aux sommets peuvent nécessiter un calcul

coûteux. D'un autre côté, il est également clair que l'interpolation de la forme de l'unité ne nécessite que les valeurs au niveau des sommets du quadrièdre, qui est un sous-ensemble de tous les sommets du cube d'unité. Par conséquent, il est plus efficace de mettre en œuvre ce qu'on peut appeler une « évaluation différée ». Dans une implémentation logicielle de l'algorithme précédent, il est courant d'avoir une sous-routine qui prend le cube d'unité et les valeurs à ses sommets comme entrée. L'évaluation différée signifie qu'au lieu de transmettre les valeurs au niveau des sommets, les informations nécessaires à l'évaluation des valeurs des sommets sont passées, sans réellement effectuer l'évaluation. À l'intérieur de la sous-routine, l'évaluation réelle de ces valeurs ne sera effectuée que pour les sommets qui appartiennent au tétraèdre englobant, une fois le tétraèdre englobant déterminé.

Table de recherche à utiliser avec des appareils sources RVB virtuels à plage dynamique élevée

Dans le cas où une transformation est construite avec un appareil source modélisé comme un appareil RVB virtuel, il est possible que les valeurs de colorant source soient négatives ou supérieures à unity (1.0). Lorsque cela se produit, l'appareil source est appelé avoir une plage dynamique élevée (HDR). Une attention particulière est apportée à ce cas.

Dans le cas de transformations HDR, les valeurs minimales et maximales de chaque canal colorant peuvent être déterminées à partir de la limite de la gamme de l'appareil. En utilisant ces valeurs, une mise à l'échelle simple pour chaque canal colorant est appliquée afin que les valeurs de colorant égales au colorant minimal soient converties en 0,0, et que les valeurs de colorant égales au colorant maximal soient converties en 1,0, avec une mise à l'échelle linéaire des valeurs entre pour mapper linéairement entre 0,0 et 1,0.

ICCProfileFromWCSProfile

Étant donné que l'objectif main de cette fonctionnalité est de prendre en charge les versions antérieures à Vista de Windows, vous devez générer des profils ICC version 2.2 comme défini dans la spécification ICC.1:1998-09. Dans certains cas (voir le tableau suivant « Mappage de classe de profil de base de l'appareil à la classe de profil ICC »), vous pouvez créer une matrice ou un profil ICC basé sur TRC à partir d'un profil WCS. Dans d'autres cas, le profil de la CPI se compose de luts. Le processus suivant décrit comment créer les unités logiques AToB et BToA. Bien sûr, les profils icc ont d'autres domaines aussi. Certaines données peuvent être dérivées du profil WCS. Pour les autres données, vous devez développer des valeurs par défaut intelligentes. Le droit d'auteur

sera attribué à Microsoft ; puisque c'est la technologie Microsoft qui est utilisée pour créer les LUT.

Cette conception doit fonctionner pour tous les types de modèles d'appareils, y compris les plug-ins. Tant que le plug-in a un modèle d'appareil de base associé, le type d'appareil sous-jacent peut être déterminé.

La partie difficile de la création d'un profil ICC consiste à créer les tables de recherche AToB et BToA. Ces tableaux sont mappés entre l'espace de l'appareil, par exemple RVB ou CMJN, et l'espace de connexion de profil (PCS), qui est une variante de CIELAB. Cela est fondamentalement identique au processus de gestion des couleurs utilisé dans la transformation CITE pour mapper de l'espace de l'appareil à l'espace de l'appareil. Toutefois, vous devez disposer des informations suivantes pour effectuer la transformation.

1. Référencer les conditions d'affichage du PCS.

2. Référencer la gamme PCS.

3. Modèle d'appareil qui convertit les valeurs PCS et la colorimétrie.

Le profil WCS et sa CAM associée sont fournis en tant que paramètres. Il existe deux modèles d'appareil de base qui effectuent une conversion entre la colorimétrie et l'encodage PCS. La raison pour laquelle vous en avez besoin est expliquée ci-dessous.

1. Vous pouvez obtenir les conditions d'affichage de référence pour le PCS à partir de la spécification du format de profil ICC. Les informations fournies dans la spécification du format de profil ICC sont suffisantes pour calculer toutes les données requises pour initialiser la CAM utilisée par le CMS. Par souci de cohérence et de flexibilité, ces informations sont stockées dans un profil de couleur WCS.

2. Vous pouvez également utiliser un profil WCS pour stocker des exemples qui définissent la gamme de référence du PCS. Le système de gestion des couleurs CITE (CMS) offre deux façons de créer des limites de gamme. L'une consiste à échantillonner l'espace complet de l'appareil et à utiliser le modèle d'appareil pour créer des valeurs de mesure. La deuxième méthode consiste à utiliser des échantillons mesurés à partir du profil pour créer une limite de gamut de référence. Étant donné que la gamme du PCS ICC est trop grande pour en faire une gamme de référence utile, la première méthode est inappropriée. Mais la deuxième méthode est une approche flexible basée sur des profils. Pour redéfinir la gamme PCS de référence, vous pouvez modifier les données de mesure dans le profil d'appareil PCS.

3. Le PCS ICC est une modélisation d'un appareil idéal. En créant un modèle du PCS en tant qu'appareil réel, vous pouvez tirer parti du processus de gestion des couleurs utilisé dans la CMM intelligente. La création d'un modèle d'appareil à partir de la colorimétrie vers l'encodage PCS est simple. Il vous suffit de mapper entre les vraies valeurs colorimétriques et les valeurs encodées PCS. Étant donné que l'interface CMS pour les modèles d'appareil prend uniquement en charge les valeurs XYZ, vous devrez peut-être également mapper xyz et LAB. Il s'agit d'une transformation bien connue. Ce modèle est décrit dans le document 2.2.02 « Modèles d'appareil de base » dans les sections 7.9 et 7.10.

Vous devrez peut-être effectuer un mappage de gamut si la gamme de l'appareil est supérieure à la gamme du PCS. Les MGM de base peuvent être utilisés à cette fin. Notez qu'un profil ICC correctement créé comporte des tables de recherche pour les intentions colorimétriques, perceptuelles et de saturation relatives, bien qu'elles puissent toutes pointer vers le même LUT en interne.

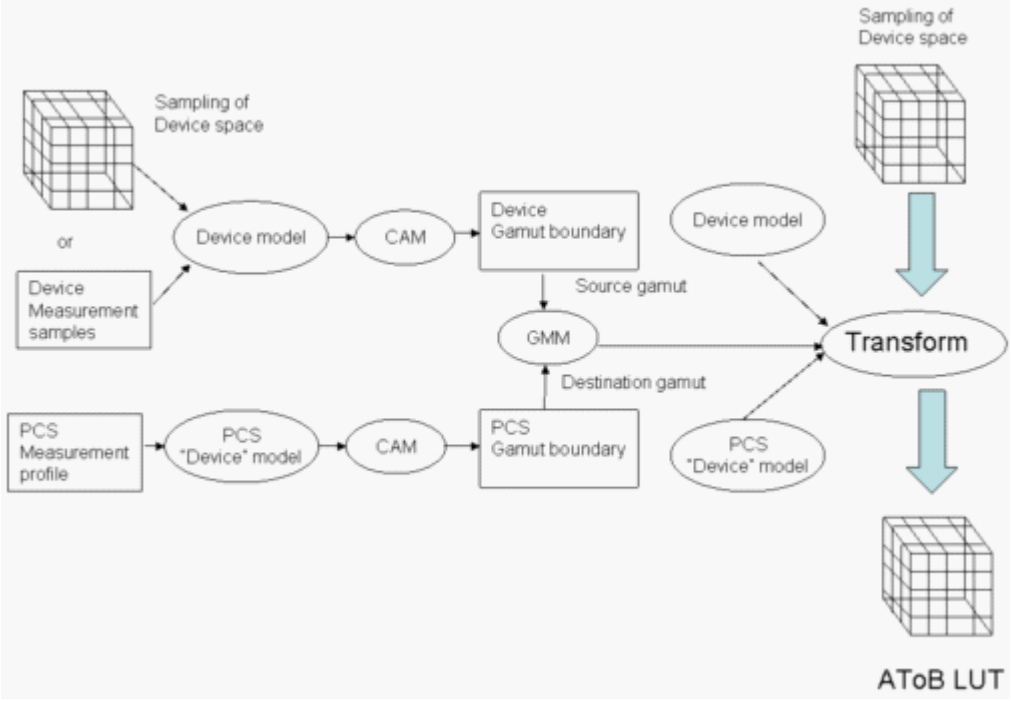


Figure 5 : Création d'un LUT AToB

Ce processus est illustré dans la figure 5. Tout d'abord, le modèle d'appareil est initialisé à partir des données du profil DM. Ensuite, construisez une limite de gamme d'appareils comme suit. Un échantillonnage des données du modèle d'appareil est exécuté via le modèle d'appareil pour obtenir des données colorimétriques. Les données colorimétriques sont exécutées via la CAM pour créer des données d'apparence. Les données d'apparence sont utilisées pour créer la limite de la gamme d'appareils.

Ensuite, utilisez les données du profil de mesure PCS de référence pour créer une limite de gamut pour le PCS.

Utilisez les deux limites de gamut que vous venez de créer pour initialiser un GMM. Ensuite, utilisez le modèle d'appareil, le GMM et le modèle d'appareil PCS pour créer une transformation. Exécutez un échantillonnage de l'espace d'appareil via la transformation pour créer un LUT AToB.

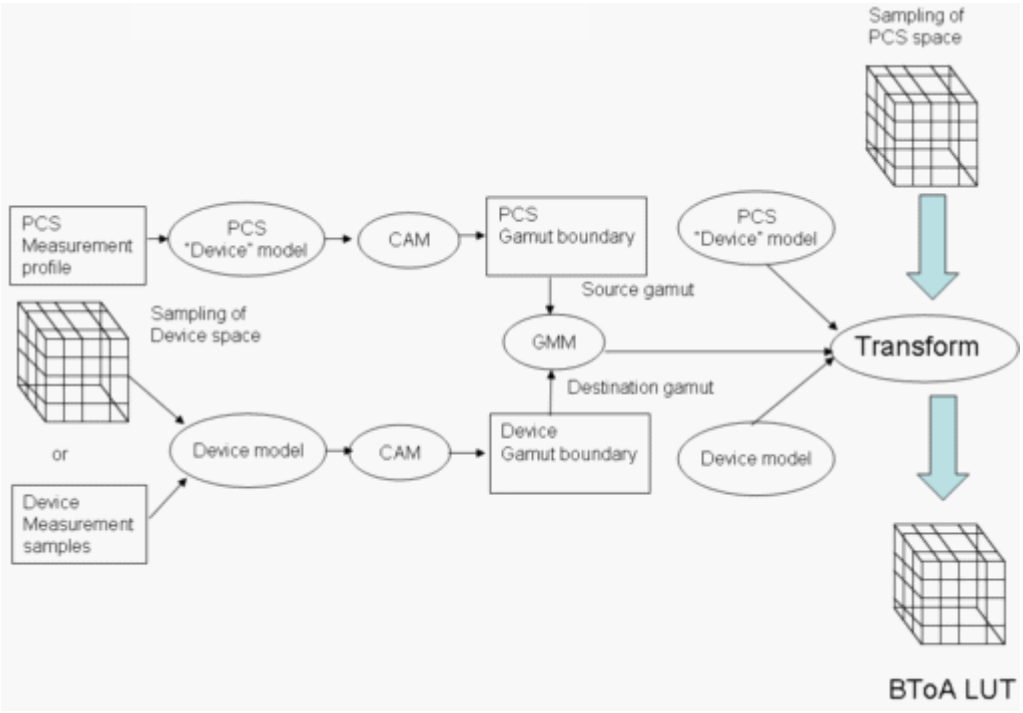


Figure 6 : Création d'un LUT BToA

La figure 6 illustre la création du LUT BToA. Cela est presque identique à la création d'un LUT AToB, avec les rôles de source et de destination échangés. En outre, vous devez échantillonner la gamme PCS complète pour créer le LUT.

Notez que étant donné que le CAM (CIECAM02 dans WCS) est impliqué dans le processus, l'adaptation chromatique entre le point blanc du média et le point blanc PCS (mandaté par icc comme étant celui de D50) est effectuée de manière transparente par le CAM.

Appareils RVB virtuels HDR

Une attention particulière doit être prise en compte lors de la génération de profils pour les appareils RVB virtuels HDR ; c'est-à-dire les appareils pour lesquels les valeurs de colorant peuvent être inférieures à 0,0 ou supérieures à 1,0. Dans la génération du LUT ATOB, un plus grand ensemble de LUT d'entrée 1D est généré. Les valeurs de colorant sont mises à l'échelle et décalées à la plage 0 .. 1 en utilisant les valeurs minimales et maximales de colorant dans le profil WCS.

Étant donné que l'espace colorant des appareils HDR n'est probablement pas complètement rempli, une prise en charge spéciale est fournie dans le LUT 3D pour

l'étiquette. Afin de gérer les couleurs dans la région peu peuplée, les colorants sont recodés afin d'obtenir une extrapolation au-delà de 0,0 et 1,0. La plage utilisée est -1 .. +4.

En raison de la mise à l'échelle appliquée pour le LUT 3D, un ensemble de luts de sortie 1D est créé pour mapper le résultat à la plage 0 .. 1.

Plusieurs PCS

La CPI a conclu qu'un PCS n'était pas suffisamment souple pour répondre à toutes les utilisations prévues d'un CMS. Dans la version 4 de la spécification de profil, l'ICC a précisé qu'il existe en fait deux encodages PCS. Un est utilisé pour les intentions colorimétriques; une autre est utilisée pour l'intention perceptuelle. (Aucun PCS n'est spécifié pour l'intention Saturation. La CPI a laissé cette partie ambiguë.) Le PCS colorimétrique a une légèreté minimale et maximale spécifiée, mais les valeurs de chroma et de teinte sont comprises à environ ± 127. Ce PCS ressemble à un prisme rectangulaire. Comme mentionné précédemment, le volume de PCS perceptuel ressemble à la gamme d'une imprimante à jet d'encre.

Les deux PCS ICC ont également deux encodages numériques différents. Dans le PCS perceptuel, une valeur de zéro représente une légèreté de zéro. Dans le PCS colorimétrique, la valeur zéro représente la légèreté minimale du PCS, qui est supérieure à zéro. Vous pouvez résoudre ce problème en ayant un modèle d'appareil différent pour chacun des encodages PCS.

Mappage de gamuts

Pour créer les unités logiques AToB dans un profil ICC, vous mappez de la gamme d'appareils à l'espace PCS approprié. Pour créer les unités logiques BToA, vous mapper de l'espace PCS à la gamme d'appareils. Le mappage des unités logiques AToB est assez similaire à celui utilisé dans un CMS basé sur les mesures. Pour le PCS perceptuel, mappez la gamme plausible de l'appareil à la limite de la gamme PERCEPTual PCS, en utilisant le découpage ou la compression pour toutes les couleurs hors gamut. Pour les intentions colorimétriques, vous devrez peut-être découper la légèreté, mais les valeurs de chroma et de teinte vont toutes s'adapter à la gamme colorimétrique PCS.

Le mappage des unités logiques BToA est un peu différent. Les intentions colorimétriques sont toujours faciles; il vous suffit de découper les valeurs PCS sur la gamme de l'appareil. Toutefois, l'ICC exige que toutes les valeurs PCS possibles soient mappées à une valeur d'appareil, et pas seulement à celles figurant dans la gamme de référence du PCS perceptuel. Par conséquent, vous devez vous assurer que les MGM

peuvent gérer les couleurs sources qui se trouvent en dehors de la gamme de référence. Pour ce faire, vous pouvez découper ces couleurs sur la limite de la gamme de l'appareil.

Mappage d'appareil de base à classe de profil ICC

Type d'appareil de référence	Classe de profil ICC	Remarque
Périphérique de capture RVB	Périphérique d'entrée (« scnr »)	PCS est CIELAB. AToB0Tag est Appareil vers PCS avec intention colorimétrique relative.
CRT, moniteur LCD	Périphérique d'affichage (« mntr »)	PCS est CIEXYZ. Consultez les rubriques suivantes pour la conversion de modèle.
Projecteur RVB	Espace de couleur (« espace »)	PCS est CIELAB.
Imprimante RVB et CMJN	Périphérique de sortie (« prtr »)	PCS est CIELAB.
Appareil virtuel RVB (cas non HDR)	Périphérique d'affichage (« mntr »)	PCS est CIEXYZ.
Appareil virtuel RVB (cas HDR)	Espace de couleur (« espace »)	PCS est CIELAB.

La conversion des profils d'analyse n'implique pas la création de luts, mais consiste plutôt à créer une matrice ou un modèle TRC. Le modèle utilisé dans ICC est légèrement différent de celui utilisé dans la modélisation WCS CRT ou LCD en ce que le terme « correction noire » est manquant. Plus précisément :

Modèle WCS : $$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = M_{WCS} \begin{pmatrix} \rho_{WCS}(R) \\ \gamma_{WCS}(G) \\ \beta_{WCS}(B) \end{pmatrix} + \begin{pmatrix} X_{black} \\ Y_{black} \\ Z_{black} \end{pmatrix}$$

Modèle ICC : $$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = M_{ICC} \begin{pmatrix} \rho_{ICC}(R) \\ \gamma_{ICC}(G) \\ \beta_{ICC}(B) \end{pmatrix}$$

La conversion du modèle WCS en modèle ICC s'effectue comme suit.

Définissez de nouvelles courbes :

$$\begin{pmatrix} \overline{\rho}_{WCS}(R) \\ \overline{\gamma}_{WCS}(G) \\ \overline{\beta}_{WCS}(B) \end{pmatrix} = \begin{pmatrix} \rho_{WCS}(R) \\ \gamma_{WCS}(G) \\ \beta_{WCS}(B) \end{pmatrix} + M_{WCS}^{-1} \begin{pmatrix} X_{black} \\ Y_{black} \\ Z_{black} \end{pmatrix}$$

Il ne s'agit pas de courbes de reproduction de tonalités, car elles ne mappent pas 1 à 1. Une normalisation y parvient. Les définitions finales du modèle ICC sont les suivantes :

$$\rho_{ICC}(R) = \overline{\rho}_{WCS}(R) / \overline{\rho}_{WCS}(1)$$

$$\gamma_{ICC}(G) = \overline{\gamma}_{WCS}(G) / \overline{\gamma}_{WCS}(1)$$

$$\beta_{ICC}(R) = \overline{\beta}_{WCS}(B) / \overline{\beta}_{WCS}(1)$$

$$M_{ICC} = M_{WCS} \cdot \begin{pmatrix} \overline{\rho}_{WCS}(1) & 0 & 0 \\ 0 & \overline{\gamma}_{WCS}(1) & 0 \\ 0 & 0 & \overline{\beta}_{WCS}(1) \end{pmatrix}$$

Pour les appareils virtuels RVB non HDR, vous générez également un profil ICC d'affichage pour l'efficacité de l'espace. Dans ce cas, la matrice tristimulus *M ICC* peut être obtenue directement à partir des primaires du profil WCS sans la conversion de modèle ci-dessus. Une dernière remarque, mais importante, est que cette matrice tristimulus doit être adaptée chromatiquement à D50 pour se conformer à la spécification ICC du PCS. En d'autres termes, les entrées sur chaque ligne de la matrice à encoder dans le profil ICC doivent correspondre respectivement à 96,42, 100 et 82,49. Dans l'implémentation actuelle, l'adaptation chromatique est effectuée par CAT02, qui est également la transformation d'adaptation chromatique utilisée dans CAM02.

Black Preservation et Black Generation

L'implémentation de la préservation des noirs est liée à la génération du canal noir dans les appareils qui prennent en charge un canal noir. Pour ce faire, des informations sur chaque couleur de source sont collectées pour permettre aux modèles d'appareils qui prennent en charge un canal noir de déterminer la meilleure façon de définir le canal noir sur la sortie. Bien que la conservation des noirs soit pertinente pour les transformations de couleur qui se convertissent d'un appareil à canal noir à un autre, la génération noire est implémentée pour toutes les transformations impliquant un appareil de destination de canal noir.

Les informations de canal noir sont enregistrées dans une structure de données appelée **BlackInformation**. La structure **BlackInformation** contient une valeur booléenne indiquant si la couleur contient uniquement du colorant noir et une valeur numérique

indiquant le degré de « noirceur » appelé « poids noir ». Pour les appareils sources qui prennent en charge un canal noir, le poids noir est le pourcentage de colorant noir dans la couleur source. Pour les appareils sources qui ne contiennent pas de canal noir, le poids noir est calculé à l'aide des autres colorants et de la valeur d'apparence. Une valeur appelée « pureté de la couleur » est calculée en prenant la différence entre la valeur maximale de colorant et la valeur minimale de colorant divisée par la valeur maximale du colorant. Une valeur appelée « légèreté relative » est calculée en prenant la différence entre la légèreté de la couleur et la légèreté minimale pour l'appareil de destination divisée par la différence entre la luminosité minimale et la luminosité maximale pour l'appareil de destination. Si l'appareil source est un appareil additif (moniteur ou projecteur), le poids noir est déterminé comme étant de 1,0 moins la pureté des couleurs multipliée par la légèreté relative. Par exemple, si l'appareil source est un moniteur RVB, la valeur maximale et la valeur minimale de R, G et B pour chaque couleur sont calculées et le poids noir est déterminé par la formule :

BW = (1,0 – (max(R,G,B) – min(R,G,B)) / max(R, G, B)) * légèreté relative

Si l'appareil source prend en charge la coloration soustractive, par exemple une imprimante CMY, les colorants individuels doivent être « complétés » (soustraits de 1.0) avant d'être utilisés dans la formule précédente. Par conséquent, pour une imprimante CMY, R = 1.0 – C, G = 1.0 – M et B = 1.0 – Y.

Les informations noires pour chaque couleur traitée par la transformation de couleur sont déterminées pendant le processus de traduction de couleur. Les informations en noir uniquement sont déterminées si la conservation des noirs est spécifiée. Le poids noir est toujours déterminé si le modèle d'appareil de destination prend en charge un colorant noir. Les informations noires sont transmises au modèle d'appareil de destination via la méthode **ColorimetricToDeviceColorsWithBlack** , qui utilise le LUT résultant.

Notez que, en raison de l'optimisation de la transformation de couleur, le processus ci-dessus se produit uniquement lors de la création de la LUT de transformation optimisée, et non pendant l'exécution de la méthode TranslateColors.

Optimisation des transformations avec plus de trois canaux sources

La taille de la transformation optimisée est déterminée par plusieurs facteurs : le nombre de canaux de couleur dans l'appareil source, le nombre d'étapes dans le tableau pour chaque canal de couleur source et le nombre de canaux de couleur dans l'appareil de sortie. La formule pour déterminer la taille de la table de transformation est la suivante :

Size = Nombre d'étapes par source de canal $_{\ device(Number\ of\ channels\ in\ source\ device)}$ x nombre de canaux dans l'appareil de sortie

Comme vous pouvez le voir, la taille de la table augmente de façon exponentielle en fonction du nombre de canaux dans l'appareil source. De nombreux appareils sources prennent en charge trois canaux de couleur, par exemple rouge, vert et bleu. Toutefois, si un appareil source prend en charge quatre canaux, tels que CMJN, la taille de la table et le temps nécessaire pour construire la table augmentent d'un facteur du nombre d'étapes. Dans un CMS basé sur les mesures où les transformations sont construites « à la volée », cette fois peut bien être inacceptable.

Pour réduire le temps nécessaire à la construction de la table de conversion de couleurs, il est possible de tirer parti de deux faits. Tout d'abord, alors que l'appareil source peut prendre en charge plus de trois canaux de couleur, l'espace de couleur intermédiaire indépendant du périphérique (CIECAM02 Ja $_{C}$ b $_{C}$) n'a que trois canaux de couleur. Deuxièmement, la partie la plus longue du traitement n'est pas la modélisation de l'appareil (conversion des coordonnées de couleur d'appareil en valeurs tristimulus), mais le mappage de la gamme. À l'aide de ces faits, vous pouvez construire une table de conversion de couleurs préliminaire qui convertit les couleurs dans l'espace de couleurs indépendant de l'appareil par le biais des étapes de mappage de gamut et, enfin, via le modèle de couleur d'appareil de sortie. La construction de cette table est de dimension 3. Ensuite, nous construisons la table de conversion de couleur finale de quatre dimensions en convertissant les combinaisons de couleurs sources en espace intermédiaire indépendant de l'appareil, puis, à l'aide de la table de conversion de couleur préliminaire, terminez la conversion en espace colorimétrique du périphérique de sortie. Ainsi, vous réduisez du calcul (nombre d'étapes dans la table de choix) $_{le\ nombre\ de\ canaux}$ de mappage de gamuts au nombre d'étapes dans la table intermédiaire $_{3}$ calculs de mappage de gamuts. Même si vous devez effectuer un nombre d'étapes dans le nombre de canaux (table de choix) $_{nombre\ de\ calculs}$ de la modélisation d'appareil et des recherches de tables tridimensionnelles, cela reste beaucoup plus rapide que le calcul d'origine.

Le processus précédent fonctionne correctement, à condition qu'il n'y ait pas besoin d'informations à transmettre entre le modèle d'appareil source et tout autre composant de la transformation de couleur. Toutefois, si le dispositif de sortie et le dispositif source prennent tous deux en charge un colorant noir et que le colorant noir source est utilisé pour déterminer le colorant noir de sortie, le processus ne parvient pas à communiquer correctement les informations sur le noir source. Un autre processus consiste à construire une table de conversion de couleurs préliminaire qui convertit les couleurs dans l'espace de couleurs indépendant de l'appareil par le biais des étapes de mappage de gamut uniquement. Ensuite, construisez la table de conversion de couleur finale de la dimension quatre en procédant comme suit : a) convertissez les combinaisons de

couleurs sources en espace intermédiaire indépendant du périphérique, b) effectuez les étapes de mappage de gamut en interpolant dans la table de couleurs préliminaire au lieu d'appliquer les processus de mappage de gamut réels, et c) utilisez les valeurs résultantes des étapes de mappage de gamut et des informations de canal noir source pour calculer les colorants de périphérique de sortie à l'aide du modèle d'appareil de sortie. Ce processus peut également être utilisé quand des informations sont transférées entre les modèles d'appareil source et de sortie, même s'il n'y a pas de canal noir ; par exemple, si les deux modules sont implémentés avec une architecture de plug-in qui permet l'échange de données entre les modules.

Les deux processus précédents peuvent être utilisés pour améliorer efficacement le temps nécessaire à la construction de la table de transformation de couleurs à quatre dimensions.

CheckGamut

L'ICM appelle CreateTransform et **CreateMultiProfileTransform** prennent un mot de valeurs d'indicateur, dont l'une est ENABLE_GAMUT_CHECKING. Lorsque cet indicateur est défini, CITE doit créer la transformation différemment. Les étapes initiales sont les mêmes : les cartes d'accès client source et de destination doivent être initialisées, puis les descripteurs de limites de gamut source et de destination doivent être initialisés. Quelle que soit l'intention spécifiée, le GMM CheckGamut doit être utilisé. Le GMM CheckGamut doit être initialisé à l'aide des modèles d'appareil source et de destination et des descripteurs de limites de gamut. Toutefois, la transformation doit ensuite créer une transformation tronquée comprenant le modèle d'appareil source, la cam source, toutes les machines gmms intermédiaires et checkGamut GMM. Cela garantit que les valeurs delta J, delta C et delta h sorties par checkGamut CMM deviennent les valeurs résultantes finales.

La signification de CheckGamut est claire lorsqu'il n'y a que deux profils d'appareil dans la transformation. Lorsqu'il existe plus de deux profils d'appareil et plus de deux MMM, CheckGamut indique si les couleurs qui ont été transformées via le premier modèle d'appareil et toutes les GMM sauf le dernier sont comprises dans la gamme de l'appareil de destination.

Rubriques connexes

Concepts de base de la gestion des couleurs

Schémas et algorithmes du système de couleurs Windows

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

À propos de Windows Color System version 1.0

Article • 13/06/2023

La technologie Microsoft Windows Color System (WCS) garantit qu'un objet image couleur, graphique ou texte est rendu aussi près que possible de son intention d'origine sur n'importe quel appareil, malgré les différences dans les technologies d'imagerie et les fonctionnalités de couleur entre les appareils. Que vous scanniez une image ou un autre graphique sur un scanneur de couleurs, que vous la téléchargez sur Internet, que vous la visionnez ou la modifiez à l'écran, ou que vous la mettez sur papier, film ou autre média, WCS 1.0 vous aide à garder ses couleurs cohérentes et précises.

WCS 1.0 fait partie intégrante de Microsoft Windows Étant donné que WCS 1.0 est intégré à la famille Windows de systèmes d'exploitation en tant qu'ensemble de fonctions d'API Win32, il est facilement disponible pour n'importe quelle application, pilote de périphérique, outil d'étalonnage de périphérique ou module de gestion des couleurs (CMM).

La version 1.0 de la gestion des couleurs des images (ICM) a été fournie dans Microsoft Windows 95 et fournit des fonctionnalités de gestion des couleurs de base dans les contextes d'appareils Windows.

ICM version 2.0 est inclus dans Windows 98, Windows Millennium Edition, Windows 2000 et Windows XP, et inclut des fonctions qui implémentent la gestion des couleurs en dehors des contextes d'appareil. Ces fonctions étaient adaptées aux exigences de gestion des couleurs plus exigeantes et donnent aux applications un plus grand contrôle sur le processus de gestion des couleurs.

LA version 1.0 de WCS est incluse dans Windows Vista et inclut une variété de nouvelles fonctions qui offrent des améliorations significatives en matière de flexibilité, de transparence, de prévisibilité et d'exttensiblité pour les fournisseurs.

Quels types d'applications bénéficieront de l'utilisation de WCS 1.0 ? Presque toutes les applications s'exécutant dans un environnement Windows Vista aujourd'hui. La fonctionnalité de gestion des couleurs de base devient une exigence pour les applications de tous types.

L'affichage et l'impression en couleur se sont améliorés ces dernières années au point où les images couleur photo-réalistes et les graphiques en couleur complexes sont désormais couramment utilisés non seulement dans l'édition de bureau, mais également dans un large éventail d'applications de données et de présentation. Les technologies

d’objet telles que COM et ActiveX ont des objets de couleur incorporés dans pratiquement tous les types d’espace de données.

Les catalogues de vêtements, l’art en ligne, les albums photo de famille, les logos d’entreprise, les graphiques, les graphiques, les présentations, les aperçus imprimés et les simulations de couleurs ne sont que quelques exemples d’applications quotidiennes où la couleur est importante.

Par conséquent, de plus en plus d’utilisateurs exigent que leurs applications soient capables d’afficher des couleurs précises. Un mauvais rendu des couleurs ruine les images et les graphiques qui sont de plus en plus importants pour les utilisateurs.

Sur un plan fondamental, presque toutes les applications doivent être en mesure d’ajuster automatiquement la couleur afin que sa sortie soit identique sur différents moniteurs et imprimantes. WCS 1.0 fournit un ensemble de fonctions pour fournir ce type de gestion des couleurs qui est transparent pour un utilisateur et qui nécessite peu de surcharge dans l’application.

À un niveau supérieur, WCS 1.0 fournit des fonctions supplémentaires qui fournissent une gestion des couleurs plus complexe et contrôlable. Les applications graphiques et de publication de bureau ont besoin de ces fonctionnalités supplémentaires pour permettre à leurs utilisateurs de travailler précisément avec des couleurs cohérentes sur de nombreux appareils tout au long d’un processus de production.

En général, les éditeurs de logiciels dont les produits traitent l’entrée ou la sortie des couleurs et les fournisseurs de matériel de périphériques de couleur trouveront WCS 1.0 une technologie clé pour simplifier la livraison de produits réussis. Les sections suivantes incluent :

- Nouveautés de la version 1.0 de WCS
- Disponibilité de WCS 1.0

Commentaires

Nouveautés de Windows Vista

Article • 03/06/2023

La version 1.0 de la gestion des couleurs d'image (ICM) a été fournie dans Microsoft Windows 95 et fournit des fonctionnalités de gestion des couleurs de base dans les contextes des appareils Windows.

ICM version 2.0 a été fournie dans Windows 98, Windows Millennium Edition, Windows 2000 et WindowsXP et inclut une variété de nouvelles fonctions qui ont implémenté la gestion des couleurs en dehors des contextes d'appareil. Ces nouvelles fonctions conviennent aux exigences de gestion des couleurs plus exigeantes et donnent aux applications un meilleur contrôle sur le processus de gestion des couleurs.

Avec la version de Windows Vista, ICM 2.0 est désormais inclus dans Windows Color System (WCS) 1.0, ce qui ajoute plus de fonctionnalités. Le tableau suivant répertorie les nouvelles interfaces de programmation d'applications (API) qui sont fournies dans Windows Vista.

Nouvelle expédition d'API dans Windows Vista

Énumérations

Nom de l'API

En-tête

Bibliothèque

COLORDATATYPE

icm.h

mscms.lib

COLORPROFILESUBTYPE

icm.h

mscms.lib

COLORPROFILETYPE

icm.h

mscms.lib

WCS_PROFILE_MANAGEMENT_SCOPE

icm.h

mscms.lib

Fonctions

Nom de l'API

En-tête

Bibliothèque

WcsAssociateColorProfileWithDevice

icm.h

mscms.lib

WcsCheckColors

icm.h

mscms.lib

WcsCreateIccProfile

icm.h

mscms.lib

WcsDisassociateColorProfileFromDevice

icm.h

mscms.lib

WcsEnumColorProfiles

icm.h

mscms.lib

WcsEnumColorProfilesSize

icm.h

mscms.lib

WcsGetDefaultColorProfile

icm.h

mscms.lib

WcsGetDefaultColorProfileSize

icm.h

mscms.lib

WcsGetDefaultRenderingIntent

icm.h

mscms.lib

WcsGetUsePerUserProfiles

icm.h

mscms.lib

WcsOpenColorProfileW

icm.h

mscms.lib

WcsSetDefaultColorProfile

icm.h

mscms.lib

WcsSetDefaultRenderingIntent

icm.h

mscms.lib

WcsSetUsePerUserProfiles

icm.h

mscms.lib

WcsTranslateColors

icm.h

mscms.lib

Interfaces et leurs fonctions

Nom de l’API

En-tête

Bibliothèque

IDeviceModelPlugin

WcsPlugIn.h

N/A

IDeviceModelPlugin::ColorimetricToDeviceColors

WcsPlugIn.h

N/A

IDeviceModelPlugin::ColorimetricToDeviceColorsWithBlack

WcsPlugIn.h

N/A

IDeviceModelPlugin::D eviceToColorimetricColors

WcsPlugIn.h

N/A

IDeviceModelPlugin::GetGamutBoundaryMesh

WcsPlugIn.h

N/A

IDeviceModelPlugin::GetGamutBoundaryMeshSize

WcsPlugIn.h

N/A

IDeviceModelPlugin::GetNeutralAxis

WcsPlugIn.h

N/A

IDeviceModelPlugin::GetNeutralAxisSize

WcsPlugIn.h

N/A

IDeviceModelPlugin::GetNumChannels

WcsPlugIn.h

N/A

IDeviceModelPlugin::GetPrimarySamples

WcsPlugIn.h

N/A

IDeviceModelPlugin::Initialize

WcsPlugIn.h

N/A

IDeviceModelPlugin::SetTransformDeviceModelInfo

WcsPlugIn.h

N/A

IGamutMapModelPlugin

WcsPlugIn.h

N/A

IGamutMapModelPlugin::Initialize

WcsPlugIn.h

N/A

IGamutMapModelPlugin::SourceToDestinationAppearanceColors

WcsPlugIn.h

N/A

Structures

Nom de l'API

En-tête

Bibliothèque

BlackInformation

WcsPlugIn.h

N/A

GamutBoundaryDescription

WcsPlugIn.h

N/A

XYZColorF

WcsPlugIn.h

N/A

JChColorF

WcsPlugIn.h

N/A

JabColorF

WcsPlugIn.h

N/A

GamutShell

WcsPlugIn.h

N/A

GamutShellTriangle

WcsPlugIn.h

N/A

PrimaryJabColors

WcsPlugIn.h

N/A

PrimaryXYZColors

WcsPlugIn.h

N/A

Commentaires

Nouveautés de la version 1.0 de WCS

Article • 13/06/2023

La version 1.0 de la gestion des couleurs des images (ICM) a été fournie dans Microsoft Windows 95 et fournit des fonctionnalités de gestion des couleurs de base dans les contextes d'appareils Windows.

ICM version 2.0 est incluse dans Windows 98, Windows Millennium Edition, Windows 2000 et Windows XP et inclut une variété de nouvelles fonctions qui implémentent la gestion des couleurs en dehors des contextes d'appareil. Ces nouvelles fonctions conviennent aux exigences de gestion des couleurs plus exigeantes et donnent aux applications un meilleur contrôle sur le processus de gestion des couleurs.

Des fonctionnalités supplémentaires sont incluses dans la version Windows Vista de Microsoft Windows.

- choix d'un flux de travail CMM/Profil statique/monolithique et dynamique/modulaire
  - statique fournie par HeidelbergerDruckmaschinen dans Win98 ICM2, dynamique fournie par Canon Inc. à l'aide de la technologie Kyuanos de Canon
  - appareils de base, apparence des couleurs et profils de modèle de mappage de gamut,
  - prise en charge de la génération noire,
  - prise en charge de la préservation des noirs,
  - Prise en charge de la plage dynamique élevée,
  - virgule flottante supérieure à 16 bits par canal,
  - prise en charge par défaut du profil d'espace de travail large gamut,
  - flexibilité pour contrôler les caractéristiques intra-appareil objectives distinctes des caractéristiques inter-appareils,
  - possibilité de contrôler l'algorithme de mappage de gamut unique entre deux appareils,
  - prise en charge du mappage des primaires d'appareils sources aux primaires de destination dans un workflow géré par les couleurs,
- choix des formats de profil de couleur binaire et texte
  - binaire conforme au format de profil basé sur le TIFF ICC et au texte à l'aide de schémas XML
- Prise en charge de la version 4 d'ICC,
- panneau de configuration central de gestion des couleurs,
- compatibilité héritée maintenue pour les applications et les appareils,
- extensibilité via des modèles d'appareil de plug-in tiers et des modèles de mappage de gamut,

- prise en charge du mappage d’appareils/gamuts avancé en activant la communication directe entre les plug-ins DM et GMM,
- La CMM statique (ICM2) est désormais définie par défaut sur l’intention colorimétrique relative au lieu de l’intention colorimétrique absolue.

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l’aide sur Microsoft Q&A

Disponibilité de WCS 1.0

Article • 13/06/2023

ICM1 est disponible pour Microsoft Windows 95. Il n'est pas disponible pour MS DOS, Windows 3. *x*, ou Windows NT 4.0 ou version antérieure.

ICM2 est disponible pour Microsoft Windows 98, Windows Millennium Edition (Me), Windows 2000 et Windows XP. Il n'est pas disponible pour MS DOS, Windows 3. *x*, Windows 95 ou Windows NT 4.0 ou version antérieure.

WCS 1.0 est disponible pour Microsoft Windows Vista. Il n'est pas disponible pour Windows XP ou version antérieure.

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

Utilisation de WCS 1.0

Article • 13/06/2023

Cette section présente une vue d'ensemble du fonctionnement de WCS et de la façon dont il peut être utilisé pour créer des applications qui utilisent la gestion des couleurs. Elle contient les rubriques suivantes :

- Utilisation de profils d'appareil avec WCS
- Utilisation des modules de gestion des couleurs (CMM)
- Rendu des intentions
- Utilisation du processus de mappage des couleurs avec WCS
- Utilisation de structures dans WCS
- Utilisation de la gestion des couleurs sur Internet
- Utilisation de fonctions GDI avec WCS

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

Utilisation de profils d'appareil avec WCS

Article • 13/06/2023

Les profils d'appareil sont un outil de base pour la gestion des couleurs. Un profil d'appareil est un fichier qui contient des informations sur la conversion des couleurs dans l'espace de couleurs et la gamme de couleurs d'un appareil spécifique en un espace de couleurs indépendant de l'appareil. L'espace de couleurs indépendant de l'appareil utilisé par ICM2 est appelé espace de connexion de profil (PCS). Un profil d'appareil contient également des informations sur la conversion des couleurs du PCS dans l'espace de couleurs et la gamme de couleurs d'un appareil spécifique.

Ces deux ensembles d'informations de conversion sont utilisés pour la conversion des couleurs et la gestion des couleurs. Par instance, une image peut être créée dans l'espace de couleurs et la gamme de couleurs d'un affichage vidéo. Les informations contenues dans les fichiers de profil d'appareil peuvent être utilisées pour afficher une représentation d'une image imprimée. Les utilisateurs veulent souvent voir à l'écran comment les couleurs changent lorsqu'une image est imprimée. C'est ce qu'on appelle la vérification. Une image peut être prouvée en convertissant les couleurs de la gamme de couleurs source (l'écran) en couleurs PCS , puis en les convertissant à partir du PCS en la gamme de couleurs de destination (l'imprimante). L'image résultante peut être affichée à l'écran, ce qui permet à l'utilisateur de voir à quoi ressemblera l'image finale lors de son impression.

Les profils d'appareil permettent également des utilisations plus complexes. Par exemple, ils peuvent être utilisés pour avoir une idée de ce à quoi ressemblerait une image créée sur un écran vidéo lors de l'impression sur une imprimante laser haute résolution. L'exemple devient plus complexe s'il n'existe qu'une imprimante jet d'encre standard sur laquelle le prouver. ICM2 convertit l'image de la gamme de l'affichage en gamut de l'imprimante à jet d'encre. À partir de là, il est converti en gamut de l'imprimante laser. L'image résultante peut être imprimée sur l'imprimante à jet d'encre. Bien sûr, l'image serait à une résolution plus élevée lors de l'impression sur l'imprimante laser couleur. Toutefois, les couleurs de l'image d'épreuve imprimée sur l'imprimante jet d'encre correspondent étroitement aux couleurs que l'imprimante laser imprimerait.

Les conversions à partir d'un profil d'appareil peuvent être concaténées ensemble dans un seul fichier, appelé *profil de lien d'appareil*. Si une série de conversions est utilisée à plusieurs reprises, la création d'un profil de lien d'appareil pour celles-ci réduit le temps de conversion.

Les profils d’appareil eux-mêmes peuvent être gérés. La gestion des profils est le processus d’association de profils de couleur à des instances d’appareil. N’importe quel appareil de sortie de couleur peut avoir un ensemble d’un ou plusieurs profils de couleur qui lui sont associés. Les fabricants de matériel et les tiers qui fournissent des profils de couleur doivent effectuer la gestion des profils.

Les jeux de profils peuvent être partagés entre des appareils ou peuvent être dédiés à des appareils spécifiques. Par exemple, supposons que vous disposez de deux imprimantes couleur du même modèle. Chaque imprimante nécessite qu’un ensemble de profils de couleur lui soit associé. L’ensemble de profils de couleur d’une imprimante correspond aux différentes configurations possibles dans ce modèle. Étant du même modèle, les deux imprimantes peuvent partager un ensemble de profils. L’alternative consiste à donner à chaque imprimante son propre jeu de profils distinct. Dans ce dernier cas, les profils de couleur de l’ensemble peuvent être étalonnés précisément en fonction de la sortie individuelle de l’imprimante, et pas seulement des caractéristiques de sortie générales du modèle.

Il existe trois classes différentes de profils ICC : les profils d’entrée, les profils d’affichage et les profils de sortie. Les profils d’entrée sont généralement associés à un appareil, tel qu’un scanneur. Les profils d’affichage sont généralement associés à un moniteur d’ordinateur. Les profils de sortie sont couramment associés aux imprimantes.

La spécification autorise également les profils d’appareil, les profils abstraits, les profils de liaison d’appareil, les profils de couleur nommés et les profils d’espace de couleurs.

Un profil d’appareil décrit l’espace de couleurs d’un appareil particulier.

Un profil abstrait fournit une méthode générique permettant aux utilisateurs d’apporter des modifications subjectives de couleur aux images ou aux objets graphiques en transformant les données de couleur dans le PCS.

Comme mentionné précédemment, un profil de liaison d’appareil concatène une série de profils en une seule transformation.

Les profils de couleur nommés peuvent être considérés comme des profils frères aux profils d’appareil. Pour un appareil donné, il existe un ou plusieurs profils d’appareil pour gérer les conversions de couleurs de processus et un ou plusieurs profils de couleur nommés pour gérer les couleurs nommées. Il peut y avoir plusieurs profils de couleur nommés pour prendre en compte différents consommables ou plusieurs fournisseurs de couleurs nommées.

Un profil d’espace de couleurs décrit un espace de couleurs indépendant de l’appareil.

Le tableau suivant récapitule les différents types de profils.

Type de profil	Description
Profil abstrait	Profil qui a été ajusté en fonction des préférences particulières d'un utilisateur.
Profil d'espace de couleurs	Profil qui décrit un espace de couleurs indépendant de l'appareil.
Profil Device Link	Série de profils concaténés ensemble en un seul profil.
Profil d'appareil	Profil qui décrit l'espace de couleurs d'un appareil particulier.
Profil d'affichage	Classe ou catégorie de profils qui inclut tout type de profil associé à un affichage.
Profil d'entrée	Classe ou catégorie de profils qui inclut tout type de profil associé à un périphérique d'entrée, tel qu'un scanneur.
Profil de couleur nommé	Profil d'un espace de couleurs qui se compose de couleurs nommées.
Profils de sortie	Classe ou catégorie de profils qui inclut n'importe quel type de profil associé à un périphérique de sortie de copie papier, tel qu'une imprimante.

Les transformations de couleur peuvent être créées avec un ou plusieurs profils. Si une transformation est créée avec un seul profil, le profil doit être un profil de lien d'appareil. Une transformation peut également avoir au moins deux profils dans une chaîne. Si c'est le cas, il ne peut contenir aucun profil de liaison d'appareil. Les profils abstraits ne peuvent se trouver qu'au milieu de la chaîne. Le premier et le dernier profil doivent être des profils d'appareil ou des profils d'espace de couleurs.

Utilisation des modules de gestion des couleurs (CMM)

Article • 13/06/2023

Les modules de gestion des couleurs (CMM) sont des modules de code WCS qui utilisent les informations contenues dans les profils d'appareil pour effectuer la conversion des couleurs et le mappage des couleurs. Les développeurs d'applications ne doivent pas avoir à implémenter des MCM. Microsoft fournit la gestion CMM par défaut. Toutefois, si vous écrivez des logiciels qui nécessitent l'utilisation d'algorithmes de conversion de couleurs et de mappage de couleurs spécialisés, vous pouvez en créer un.

> **Notes**
>
> Les points d'entrée CMM *ne sont pas* des fonctions API et ne doivent pas être appelés par les applications.

Lorsque les machines virtuelles sont installées, le programme d'installation les inscrit dans le Registre Windows. Les applications peuvent énumérer les machines virtuelles inscrites et en sélectionner une à l'aide de la fonction **SelectCMM** . L'exemple d'application suivant montre comment énumérer toutes les MCM inscrites.

C++

```
#include <stdio.h>
#include <stdlib.h>
#include <stddef.h>
#include <string.h>
#include <mbstring.h>
#include <windows.h>
#include <icm.h>

#ifdef WINDOWS_98
TCHAR  gszICMatcher[] = __TEXT(
  "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\ICM\\ICMatchers");
#else
TCHAR  gszICMatcher[] = __TEXT(
  "SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\ICM\\ICMatchers");
#endif

_CRTAPI1 main (int argc, char *argv[])
{
    DWORD dwNumCMM = 0;
```

```
    HANDLE hkCMM;
    DWORD dwErr = RegCreateKey(HKEY_LOCAL_MACHINE,
                gszICMatcher, &hkCMM);
    DWORD dwMaxName, dwMaxValue;
    DWORD dwInfoErr = RegQueryInfoKey(&hkCMM, NULL, NULL,
                                NULL, NULL, NULL, NULL, NULL,
                                &dwMaxName, &dwMaxValue,
                                NULL, NULL);
    TCHAR chCMM[dwMaxName];
    ULONG cjCMM = sizeof(chCMM)/sizeof(chCMM[0]);
    DWORD dwType;
    TCHAR chCMMFile[dwMaxValue];
    ULONG cjCMMFile = sizeof(chCMMFile)/sizeof(chCMMFile[0]);

    if (dwErr != ERROR_SUCCESS)
    {
        printf("Could not open ICMatcher registry key: %d\n", dwErr);
    }

    if (dwErr == ERROR_SUCCESS)
    {
        while (RegEnumValue(
                    hkCMM,dwNumCMM,chCMM,
                    &cjCMM,NULL,&dwType,
                    chCMMFile,&cjCMMFile) == ERROR_SUCCESS)
        {
            if (dwType == REG_SZ)
            {
                printf("%d: %-80s - %-80s\n",dwNumCMM,chCMM,chCMMFile);
            }
            else
            {
                printf("%d: error\n");
            }

            dwNumCMM++;
            cjCMM = sizeof(chCMM);
            cjCMMFile = sizeof(chCMMFile);
        }
    }

    RegCloseKey(hkCMM);
}
```

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l’aide sur Microsoft Q&A

Rendu des intentions

Article • 13/06/2023

L'International Color Consortium (ICC) a défini quatre valeurs différentes *appelées intentions de rendu*. Celles-ci représentent quatre approches différentes pour créer un rendu des couleurs. Ces quatre intentions et les constantes utilisées pour y faire référence dans le code sont les suivantes.

Intentionnel	**Nom ICC**	**Description**
Image	Perception	INTENT_PERCEPTUAL
Graphic	Saturation	INTENT_SATURATION
Proof	Colorimétrique relative	INTENT_RELATIVE_COLORIMETRIC
Faire correspondre	Colorimétrique absolue	INTENT_ABSOLUTE_COLORIMETRIC

La spécification du format de profil ICC version 3.4, qui décrit ces intentions, peut être téléchargée à partir de color.org.

Intention de l'image

Appelée intention perceptuelle dans la clause de spécification ICC 4.9, une intention Image entraîne la compression ou l'extension de la gamme complète de l'image pour remplir la gamme de l'appareil de destination, de sorte que l'équilibre des gris soit conservé, mais que la précision colorimétrique ne soit pas conservée.

En d'autres termes, si certaines couleurs d'une image sortent de la plage de couleurs que le périphérique de sortie peut afficher, l'intention de l'image entraîne l'ajustement de toutes les couleurs de l'image afin que chaque couleur de l'image se situe dans la plage qui peut être rendue et que la relation entre les couleurs soit conservée autant que possible.

Cette intention est la plus appropriée pour l'affichage de photographies et d'images, et est généralement l'intention par défaut.

Intention graphique

La clause 4.12 de la spécification ICC appelle l'intention Graphic une intention de saturation . Il préserve la chromaie des couleurs dans l'image au détriment possible de la teinte et de la légèreté.

La mise en œuvre de cette intention reste quelque peu problématique, et la CPI travaille toujours sur des méthodes pour obtenir les effets souhaités.

Cette intention convient le mieux aux graphiques d'entreprise tels que les graphiques, où il est plus important que les couleurs soient vives et contrastées les unes avec les autres plutôt qu'une couleur spécifique.

Intention de preuve

L'intention Proof, appelée intention colorimétrique dans la spécification ICC, est définie de telle sorte que toutes les couleurs qui se trouvent en dehors de la plage que le périphérique de sortie peut afficher soient ajustées à la couleur la plus proche qui peut être restituée, tandis que toutes les autres couleurs restent inchangées.

L'intention de preuve ne conserve pas le point blanc.

Par exemple, le blanc le plus blanc d'un papier est plus jaune que le blanc le plus blanc d'un moniteur d'ordinateur. Une image convertie dans la gamme de l'imprimante à l'aide d'une intention colorimétrique relative entraînerait une augmentation du jaune dans toutes les couleurs. Le point blanc de l'image est déplacé pour correspondre au point blanc de l'imprimante. Toutes les autres couleurs de l'image conservent leur position par rapport au point blanc. Cela produit une image qui reflète plus précisément à quoi ressemblera l'image imprimée. Toutefois, l'utilisateur peut trouver cela visuellement déconcertant.

Faire correspondre l'intention

Dans une intention de correspondance, toutes les couleurs qui se trouvent en dehors de la plage que le périphérique de sortie peut afficher sont ajustées à la couleur la plus proche qui peut être rendue, tandis que toutes les autres couleurs restent inchangées. La spécification ICC appelle l'intention colorimétrique absolue de correspondance.

L'intention de correspondance conserve le point blanc.

Par exemple, le blanc le plus blanc d'un papier est plus jaune que le blanc le plus blanc d'un moniteur d'ordinateur. Une image convertie dans le gamut de l'imprimante à l'aide de l'intention de correspondance entraîne la conversion et la correspondance de toutes les couleurs dans la gamme de l'imprimante. Le point blanc de l'image n'est pas déplacé

pour correspondre au point blanc de l'imprimante. Par conséquent, la distance entre les couleurs et le point blanc peut changer. Cela produit une image moins déconcertante visuellement pour l'utilisateur, mais qui est également un rendu moins précis de la sortie de l'imprimante.

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

Utilisation du processus de mappage des couleurs avec WCS

Article • 13/06/2023

Le mappage des couleurs WCS est basé sur les profils d'appareil. Ceux-ci sont fournis par les fournisseurs d'appareils matériels de couleur et installés lorsqu'un appareil est installé. Lorsque le mappage de couleurs est utilisé par un programme d'application, WCS accède au profil d'appareil de l'image pour obtenir les informations nécessaires pour convertir l'image en PCS. La conversion est effectuée par la CMM.

Un profil d'appareil peut être incorporé dans l'image elle-même. Ainsi, le profil de l'appareil voyage avec l'image, même sur Internet. Un utilisateur n'a pas besoin de l'appareil source pour obtenir un mappage de couleurs précis. Si une image n'a pas de profil d'appareil, l'espace sRGB est utilisé par défaut. Pour plus d'informations, consultez Utilisation de la gestion des couleurs sur Internet.

Notez que les applications qui utilisent WCS ne doivent jamais incorporer le profil sRGB dans une image. L'espace colorimétrique sRGB fournit un espace de couleurs standardisé qui est portable sur tous les appareils. Il est automatiquement disponible pour les utilisateurs de Windows 98 et versions ultérieures, ainsi que de Windows 2000 et versions ultérieures. Par conséquent, il n'a pas besoin de voyager avec l'image.

Une fois les couleurs de l'image dans le PCS, WCS accède au profil d'appareil de l'appareil de destination. Il obtient la CMM pour convertir les couleurs d'image de PCS vers la gamme de l'appareil de destination.

Un mappage de couleurs plus complexe peut également être effectué avec WCS. Par exemple, il peut être utilisé pour avoir une idée de ce à quoi ressemblerait une image créée sur un écran vidéo lors de l'impression sur une imprimante laser haute résolution. L'exemple devient plus complexe s'il n'existe qu'une imprimante jet d'encre standard sur laquelle afficher un aperçu. L'image peut être convertie à partir de la gamme de l'écran vers la gamme de l'imprimante à jet d'encre. À partir de là, il peut être converti dans la gamme de l'imprimante laser. L'image obtenue peut être imprimée sur l'imprimante jet d'encre. Bien sûr, l'image serait à une résolution plus élevée lors de l'impression sur l'imprimante laser couleur. Toutefois, les couleurs de l'image de vérification imprimée sur l'imprimante jet d'encre correspondraient étroitement aux couleurs que l'imprimante laser imprimerait.

La façon dont les conversions dans l'exemple sont effectuées consiste à convertir les couleurs de l'image de la gamme de l'affichage dans le PCS. Une fois les couleurs de l'image converties en PCS, le profil d'appareil de l'imprimante jet d'encre est utilisé pour

les transformer en gamme de l'imprimante à jet d'encre. Ensuite, la transformation de la gamme en PCS est utilisée pour déplacer les couleurs vers le PCS. À partir de là, le profil d'appareil de l'imprimante laser est utilisé pour convertir les couleurs du PCS dans la gamme de l'imprimante laser.

La possibilité de transformer facilement les couleurs d'une gamme d'appareil vers le PCS et de revenir en arrière permet de vérifier les couleurs d'image destinées à un appareil de sortie de couleur sur presque tous les autres.

Dans l'exemple précédent, la description diffère quelque peu de la procédure réelle pour plus de clarté. En réalité, toutes les transformations mentionnées dans le paragraphe précédent seraient concaténées en une seule transformation.

Commentaires

Utilisation de structures dans WCS 1.0

Article • 13/06/2023

La plupart des structures utilisées par WCS 1.0 sont très simples et nécessitent peu d'explications. Ils sont documentés dans la section de référence wcs 1.0 intitulée Structures.

Les exceptions sont la structure **COLORMATCHSETUPW** utilisée par la fonction **SetupColorMatchingW** et les structures Windows suivantes définies dans Wingdi.h :

- BITMAPV5HEADER
- **LOGCOLORSPACE**
- **CIEXYZ**
- **CIEXYZTRIPLE**

Les sujets suivants sont abordés plus longuement :

- Structures d'en-têtes Bitmap Windows
- Différences entre les en-têtes V4 et V5
- Bitmap.exe : utilitaire Command-Line pour la conversion d'en-têtes bitmap

Structures d'en-têtes Bitmap Windows

WCS 1.0 permet aux profils de couleur ICC d'être liés ou incorporés dans des bitmaps indépendantes des appareils (DIB). Cela permet de caractériser les couleurs DIB plus précisément que ce qui était possible avec WCS dans Windows 95. BITMAPV5HEADER , la nouvelle structure d'en-tête bitmap, est définie dans Wingdi.h dans la version de Windows 98. À des fins de développement, il est également inclus dans le fichier Icm.h avec cette référence du programmeur. La structure **BITMAPV5HEADER** est la suivante :

C++

```
typedef struct {
    DWORD        bV5Size;
    LONG         bV5Width;
    LONG         bV5Height;
    WORD         bV5Planes;
    WORD         bV5BitCount;
    DWORD        bV5Compression;
    DWORD        bV5SizeImage;
    LONG         bV5XPelsPerMeter;
    LONG         bV5YPelsPerMeter;
    DWORD        bV5ClrUsed;
    DWORD        bV5ClrImportant;
```

```
    DWORD        bV5RedMask;
    DWORD        bV5GreenMask;
    DWORD        bV5BlueMask;
    DWORD        bV5AlphaMask;
    DWORD        bV5CSType;
    CIEXYZTRIPLE bV5Endpoints;
    DWORD        bV5GammaRed;
    DWORD        bV5GammaGreen;
    DWORD        bV5GammaBlue;
    DWORD        bV5Intent;         // Rendering intent for bitmap
    DWORD        bV5ProfileData;    // Offset to profile data
    DWORD        bV5ProfileSize;    // Size of embedded profile data
    DWORD        bV5Reserved;       // Should be zero
} BITMAPV5HEADER, FAR *LPBITMAPV5HEADER, *PBITMAPV5HEADER;
```

Le membre **bV5CSType** peut avoir les valeurs PROFILE_EMBEDDED ou PROFILE_LINKED pour spécifier si un profil est incorporé ou lié à la DIB. Le membre **bV5ProfileData** est le décalage en octets entre le début de la structure **BITMAPV5HEADER** et le début des données de profil. Si le profil est incorporé, les données de profil sont le profil réel, et si elles sont liées, les données de profil sont le nom de fichier terminé par null du profil. Il ne peut pas s'agir d'une chaîne Unicode. Il doit être composé exclusivement de caractères du jeu de caractères Windows (page de codes 1252).

Lorsqu'une DIB est chargée en mémoire, les données de profil (le cas échéant) doivent suivre la table de couleurs, et **bV5ProfileData** doit donner le décalage des données de profil à partir du début de la structure **BITMAPV5HEADER** . La valeur de ce membre sera différente maintenant, car les bits bitmap ne suivent pas le tableau de couleurs en mémoire. Les applications doivent modifier le membre **bV5ProfileData** après avoir chargé la DIB en mémoire.

Pour les DIB compressés, les données de profil doivent suivre les bits bitmap similaires au format de fichier. Le membre **bV5ProfileData** doit toujours donner le décalage des données de profil à partir du début de la structure **BITMAPV5HEADER** .

Les applications doivent accéder aux données de profil uniquement lorsque **bV5Size** == **sizeof** (**BITMAPV5HEADER**) **ANDbV5CSType** est PROFILE_EMBEDDED ou PROFILE_LINKED.

Si un profil est lié, le chemin d'accès du profil peut être n'importe quel nom complet (y compris un chemin réseau) qui peut être ouvert à l'aide de la fonction **Win32 CreateFile** .

Différences entre les en-têtes V4 et V5

Lors de l'utilisation de la nouvelle structure bitmap, il est utile de reconnaître les différences dans la façon dont les structures **BITMAPV4HEADER** et **BITMAPV5HEADER** sont configurées :

En-tête V4	Signification
bV4CSType	LCS_CALIBRATED_RGB. Cette valeur implique que les points de terminaison et les gammas sont donnés dans les champs appropriés. Les valeurs fausses provoquent des problèmes.
bV4CSType	LCS_sRGB. Cette valeur implique que la bitmap se trouve dans l'espace de couleur sRGB (gamma et points de terminaison ignorés).
bV4CSType	LCS_WINDOWS_COLOR_SPACE. Cette valeur implique que la bitmap se trouve dans l'espace de couleurs par défaut de Windows.

En-tête V5	Signification
bV5CSType	LCS_CALIBRATED_RGB. Cette valeur implique que les points de terminaison et les gammas sont donnés dans les champs appropriés. Les valeurs fausses provoquent des problèmes.
bV5CSType	LCS_sRGB. Cette valeur implique que la bitmap se trouve dans l'espace de couleur sRGB (gamma et points de terminaison ignorés).
bV5CSType	PROFILE_EMBEDDED. Cette valeur implique que **bV5ProfileData** pointe vers une mémoire tampon qui contient le profil à utiliser (les gamma et les points de terminaison sont ignorés).
bV5CSType	PROFILE_LINKED. Cette valeur implique que **bV5ProfileData** pointe vers le nom de fichier du profil à utiliser (les gamma et les points de terminaison sont ignorés).
bV5CSType	LCS_WINDOWS_COLOR_SPACE. Cette valeur implique que la bitmap se trouve dans l'espace de couleurs par défaut de Windows.

Pour convertir des bitmaps plus anciennes vers et à partir de la nouvelle structure **BITMAPV5HEADER** , un fichier utilitaire de conversion de ligne de commande nommé Bitmap.exe est inclus dans la référence du programmeur WCS 1.0.

BitMap.exe : utilitaire Command-Line pour la conversion d'en-têtes bitmap

Bitmap.exe est un utilitaire de ligne de commande situé dans le dossier \Bin sous le dossier d'installation que vous avez spécifié. Il modifie les en-têtes des bitmaps Windows, ce qui vous permet de convertir les bitmaps existantes des structures d'en-tête **BITMAPINFOHEADER** et **BITMAPV4HEADER** vers la structure **BITMAPV5HEADER** plus récente, puis de revenir à nouveau. La syntaxe de ligne de commande est la suivante :

```
C++

BITMAP [/d] [/1|4|5] [/s] [/f]
filename
```

Les commutateurs de ligne de commande ont les effets suivants.

Commutateur	Signification
/d	Les valeurs par défaut sont automatiquement entrées dans les en-têtes convertis.
/1	Convertir les bitmaps spécifiées en **BITMAPINFOHEADER**
/4	Convertir les bitmaps spécifiées en **BITMAPV4HEADER**
/5	Convertir les bitmaps spécifiées en **BITMAPV5HEADER**
/f	Force la conversion, même si la bitmap a déjà l'en-tête approprié
/s	Convertit les bitmaps dans le dossier spécifié et tous les sous-répertoires qu'il contient

Commentaires

Utilisation de la gestion des couleurs sur Internet

Article • 13/06/2023

Les images en couleur et les graphiques sont un moyen de plus en plus important de communiquer des informations pour les utilisateurs d'Internet. Les informations de conversion de couleur peuvent être incorporées dans certains formats de fichiers. Les techniques d'incorporation actuellement utilisées ne sont pas compactes. Les considérations relatives à la bande passante Internet les rendent souvent peu pratiques. Les considérations relatives à Internet incluent les rubriques suivantes :

- sRGB : espace de couleurs standard
- Valeurs par défaut de WCS 1.0 pour l'espace de couleur d'entrée et le profil de sortie
- sRGB et profils incorporés

Commentaires

sRGB : espace de couleurs standard

Article • 16/03/2023

En raison des considérations relatives à la bande passante Internet, Hewlett-Packard et Microsoft ont proposé l'adoption d'un espace de couleurs prédéfini standard appelé *sRGB* (IEC 61966-2-1), afin de permettre un mappage de couleurs précis avec très peu de surcharge de données.

Une version de fichier d'aide d'un livre blanc traitant des détails techniques de sRGB, sRGB.hlp, est disponible dans le dossier \Help de la référence du programmeur WCS 1.0.

Différents formats de fichier peuvent utiliser ou ajouter un indicateur pour spécifier que l'image se trouve dans l'espace de couleur sRGB. Au format DIB (Device-Independent Bitmap) Windows, la définition du membre **bV5CSType** de la structure **BITMAPV5HEADER** sur **LCS_sRGB** spécifie que les couleurs DIB se trouvent dans l'espace de couleurs sRGB.

WCS 1.0 fournit une prise en charge native de sRGB. Il existe deux façons d'utiliser WCS 1.0 pour le rendu d'une image définie dans l'espace de couleurs sRGB :

Pour afficher une image dans le contexte de l'appareil

1. Créez un contexte d'appareil (DC) sur l'appareil d'affichage.
2. Définissez la gestion des couleurs à l'aide de la fonction **SetICMMode** .
3. Utilisez la fonction **SetDIBitsToDevice** pour transférer la DIB dans le contrôleur de domaine. Tant que le membre **bV5CSMType** de la structure **BITMAPV5HEADER** est défini sur **LCS_sRGB**, le système effectue la gestion des couleurs appropriée.

Pour afficher une image en dehors du contexte de l'appareil

1. Créez une transformation à l'aide de **CreateColorTransformW**. Le membre **lcsCSType** de la structure **LOGCOLORSPACE** pointée vers le paramètre *pLogColorSpace* doit être défini sur **LCS_sRGB**. Le paramètre *hDestProfile* indique l'espace de couleurs de l'appareil d'affichage.
2. Utilisez la transformation de couleur créée pour colorier l'image avant de l'afficher sur l'appareil.

Valeurs par défaut de WCS 1.0 pour l'espace de couleur d'entrée et le profil de sortie

Quand aucun espace de couleur d'entrée n'est spécifié, PAR défaut, WCS 1.0 utilise l'espace de couleur sRGB comme espace de couleur d'entrée pour le mappage des couleurs.

Lorsqu'aucun profil de sortie n'est spécifié, mais qu'un appareil par défaut est spécifié, WCS 1.0 sélectionne un profil de sortie par défaut. Si l'appareil par défaut n'a pas de profil associé, WCS 1.0 utilise l'espace de couleur sRGB comme profil de sortie.

Le tableau suivant montre les transformations de couleurs résultantes lorsqu'un appareil par défaut n'est pas disponible.

	Profil de sortie spécifié	**Profil de sortie non spécifié**
Espace de couleur d'entrée spécifié	La transformation utilise les profils spécifiés.	La transformation convertit l'espace de couleur d'entrée connu en sRGB.
Espace de couleur d'entrée non spécifié	La transformation convertit de sRGB en profil de sortie connu.	La transformation de sRGB en sRGB est supposée ; rien n'est fait.

Profils sRGB et incorporés

À compter d'ICM version 2.0, les applications qui utilisent WCS peuvent incorporer des profils dans des images. Les profils incorporés aident les applications des utilisateurs à maintenir une apparence de couleur cohérente, même si les images sont transmises sur Internet.

Les images qui utilisent l'espace colorimétrique sRGB n'ont pas besoin d'un profil de couleur incorporé. Étant donné qu'elles n'ont pas de profil incorporé, les images sRGB sont plus petites et plus facilement transférables sur les canaux de données avec une bande passante limitée.

Les applications doivent définir l'indicateur **LCS_sRGB** dans l'en-tête bitmap de l'image pour indiquer que l'image utilise l'espace de couleur sRGB. Pour plus d'informations, consultez Structures d'en-têtes bitmap Windows et LOGCOLORSPACE.

Utilisation des fonctions GDI avec WCS

Article • 13/06/2023

Il existe différentes fonctions dans l'interface de périphérique graphique (GDI) qui utilisent ou opèrent sur des données de couleur. Certains sont activés pour une utilisation avec WCS et d'autres ne le sont pas. Les fonctions GDI suivantes sont pertinentes pour ICM :

- Fonctions de contexte d'appareil avec WCS
- Fonctions stylet et pinceau avec WCS
- Fonctions de sortie de texte avec WCS
- Fonctions bitmap avec WCS

Fonctions de contexte d'appareil avec WCS

Fonction	Description
CreateCompatibleDC	Si le contexte d'appareil (DC) qui est passé à cette fonction via son paramètre hdc est activé pour ICM, le contrôleur de domaine créé par la fonction est également activé pour ICM. Les espaces de couleurs source et de destination sont spécifiés dans le contrôleur de domaine.
CreateDC	ICM peut être activé en définissant le membre dmICMMethod de la structure DEVMODE pointée par le paramètre pInitData sur la valeur appropriée. Pour plus d'informations, consultez la documentation du Kit de développement logiciel (SDK) de plateforme sur la structure DEVMODE.
ResetDC	Le profil de couleur du contexte d'appareil spécifié par le paramètre hdc est réinitialisé en fonction des informations contenues dans la structure DEVMODE spécifiée par le paramètre lpInitData.

Fonctions stylet et pinceau avec WCS

Fonction	Description
Fonctions Brush	Aucune gestion des couleurs n'est effectuée lors de la création du pinceau. Toutefois, la gestion des couleurs est effectuée lorsque le pinceau est sélectionné dans un contrôleur de domaine avec ICM.

Fonction	Description
CreatePen	Aucune gestion des couleurs n'est effectuée lors de la création du stylet. Toutefois, la gestion des couleurs est effectuée lorsque le pinceau est sélectionné dans un contrôleur de domaine avec ICM.
ExtCreatePen	Aucune gestion des couleurs n'est effectuée lors de la création du stylet. Toutefois, la gestion des couleurs est effectuée lorsque le pinceau est sélectionné dans un contrôleur de domaine avec ICM.
SelectObject	Si l'objet sélectionné est un pinceau ou un stylet, la gestion des couleurs est effectuée.
SetDCBrushColor	La gestion des couleurs est effectuée si WCS est activé.
SetDCPenColor	La gestion des couleurs est effectuée si WCS est activé.

Fonctions de sortie de texte avec WCS

Fonction	Description
SetBkColor	La gestion des couleurs est effectuée si WCS est activé.
SetTextColor	La gestion des couleurs est effectuée si WCS est activé.

Fonctions bitmap avec WCS

Fonction	Description
BitBlt	Aucune gestion des couleurs n'est effectuée lorsque des fentes se produisent.
CreateDIBitmap	Le paramètre fuUsage spécifie que le membre bmiColors de la structure BITMAPINFO pointée par le paramètre lpbmi ne contient pas d'informations de couleur. Si ce n'est pas le cas, aucune gestion des couleurs n'est effectuée pour cette bitmap. La bitmap doit utiliser la version 4 ou la version 5 de la structure BITMAPINFO pour que la gestion des couleurs soit activée. Le contenu de la bitmap résultante ne correspond pas aux couleurs après la création de la bitmap.
CreateDIBSection	Si la structure BITMAPINFO transmise via le paramètre pbmi n'est pas la version 4 ou la version 5, aucune gestion des couleurs n'est effectuée. S'il s'agit de la version 4 ou 5, la gestion des couleurs est activée et l'espace de couleurs spécifié est associé à la bitmap.

Fonction	Description
MaskBlt	Aucune gestion des couleurs n'est effectuée lorsque des fentes se produisent.
SelectObject	Si l'objet est une bitmap créée avec CreateDIBSection, la gestion des couleurs est effectuée. L'espace de couleurs de la bitmap est utilisé comme espace de couleur de destination.
SetDIBits	La gestion des couleurs est effectuée. Si la structure BITMAPINFO spécifiée n'est pas la version 4 ou la version 5, le profil de couleur du contrôleur de domaine actuel est utilisé comme profil d'espace de couleurs source. S'il n'en a pas, l'espace sRGB est utilisé. Si la structure BITMAPINFO spécifiée est la version 4 ou la version 5, le profil d'espace de couleurs spécifié dans l'en-tête bitmap est utilisé comme profil d'espace de couleurs source.
SetDIBitsToDevice	La gestion des couleurs est effectuée. Si la structure BITMAPINFO spécifiée n'est pas la version 4 ou la version 5, le profil de couleur du contexte d'appareil actuel est utilisé comme profil d'espace de couleur source. S'il n'en a pas, l'espace de couleur sRGB est utilisé. Si la structure BITMAPINFO spécifiée est la version 4 ou la version 5, le profil d'espace de couleurs associé à la bitmap est utilisé comme espace de couleur source.
SetDIBColorTable	Aucune gestion des couleurs n'est effectuée.
StretchBlt	Aucune gestion des couleurs n'est effectuée lorsque des fentes se produisent.
StretchDIBits	La gestion des couleurs est effectuée. Si la structure BITMAPINFO spécifiée n'est pas la version 4 ou la version 5, le profil de couleur du contrôleur de domaine actuel est utilisé comme profil d'espace de couleurs source. S'il n'en a pas, l'espace sRGB est utilisé. Si la structure BITMAPINFO spécifiée est la version 4 ou la version 5, le profil d'espace de couleurs spécifié dans l'en-tête bitmap est utilisé comme profil d'espace de couleurs source.

Comportement du profil ICC avec Advanced Color

Article • 13/06/2023

Advanced Color est un terme générique des technologies de système d'exploitation pour les écrans avec une fidélité des couleurs considérablement supérieure à celle des écrans standard. Pour plus d'informations, consultez Utiliser DirectX avec une couleur avancée sur des affichages à plage dynamique élevée/standard. La gestion avancée des couleurs et la gestion automatique des couleurs garantissent une couleur d'affichage cohérente et colorimétriquement précise pour toutes les applications: héritées et modernes. Toutefois, votre application peut déjà effectuer sa propre gestion explicite des couleurs à l'aide de profils de couleur ICC (International Color Consortium).

Quand Advanced Color est actif sur les affichages SDR ou HDR, le comportement des profils ICC d'affichage change de manière non compatible avec les versions descendantes. Si votre application fonctionne avec des profils ICC d'affichage, Windows propose des assistances de compatibilité pour garantir que votre application continue d'obtenir un comportement correct. Les applications avancées prenant en charge les couleurs doivent passer de l'interaction directe avec les profils ICC d'affichage, car Windows fournit des API orientées application de remplacement qui éliminent le profil ; Des conseils complets sont disponibles dans Utiliser DirectX avec une couleur avancée sur des affichages à plage dynamique haute/standard.

Cette rubrique décrit les modifications apportées au comportement du profil ICC. En outre, si votre application gérée par les couleurs doit continuer à utiliser des profils ICC d'affichage, cette rubrique montre comment adapter votre application pour tirer parti de manière incrémentielle des avantages de couleur avancée.

Comportement de gestion des couleurs Windows hérité

Lorsque la couleur avancée est inactive, Windows n'effectue aucune gestion des couleurs sur la sortie du contenu visuel de votre application (par exemple, GDI hDC, chaîne d'échange DirectX ou visuel de composition) ; Dans la pratique, il suppose que le contenu de votre application se trouve dans l'espace de couleurs sRGB standard. Si vous souhaitez une reproduction précise des couleurs sur l'affichage actif, votre application doit effectuer sa propre gestion des couleurs, le plus souvent à l'aide de profils de couleurs ICC (International Color Consortium). Les étapes conceptuelles main sont les suivantes :

1. Obtenez les caractéristiques de couleur de l'affichage.
2. Effectuer la conversion de l'espace de couleur en espace colorimétrique de l'affichage.
3. Effectuez un mappage de gamut afin de limiter le gamut de l'affichage.

Voici plus d'informations sur l'ech des trois étapes.

Obtenir les caractéristiques de couleur de l'affichage

Une application Win32 utilise les fonctions de gestion des profils Windows Color System pour obtenir le profil ICC par défaut, qui indique les caractéristiques de couleur de l'affichage, y compris sa gamme de couleurs disponible.

Une application plateforme Windows universelle utilise la méthode DisplayInformation.GetColorProfileAsync à la place.

Effectuer la conversion de l'espace de couleur en espace colorimétrique de l'affichage

Si l'espace de couleur de l'affichage ne correspond pas à l'espace de couleurs de votre contenu, vous devez effectuer une conversion d'espace de couleur. Par exemple, le contenu numérique est souvent encodé en sRGB, mais votre affichage peut être large DCI-P3. Vous utilisez généralement une bibliothèque de gestion des couleurs ICC qui lit le profil ICC et transforme les valeurs de couleurs de votre contenu pour qu'elles correspondent. Windows fournit plusieurs moteurs de gestion des couleurs ICC ; par exemple, l'effet de gestion des couleurs Direct2D.

Il est important de noter que la gestion des couleurs basée sur un profil ICC est *référencée par affichage* ou *par sortie*. Cela signifie que les valeurs de couleur ne sont pas stockées en tant que couleurs absolues (*référencées par la scène*), mais qu'elles sont encodées par rapport à l'espace de couleurs de l'affichage (le périphérique de sortie). Par exemple, si votre application affiche sRGB en rouge, celui-ci est représenté comme `RGB(1, 0, 0)` dans votre sortie rendue. Mais si vous affichez ce contenu sur un écran Adobe RVB, `RGB(1, 0, 0)` il est simplement interprété par l'affichage comme son rouge le plus saturé (rouge Adobe RVB), ce qui est incorrect. Lorsque vous appliquez une transformation de couleur ICC, elle réencode la couleur en tant `RGB(0.858659, 0, 0)` que , et quand elle est rendue par l'affichage Adobe RVB, elle est correctement reproduite en rouge sRGB.

Effectuer un mappage de gamut pour limiter à la gamut de l'affichage

En plus de réinterpréter les valeurs de couleur pour qu'elles correspondent à l'espace de couleur de l'affichage, vous devez gérer le cas où l'affichage ne peut pas reproduire physiquement toutes les couleurs de votre contenu ; si la gamme de couleurs de votre contenu est supérieure à celle de l'affichage. Ce processus est appelé mappage de gamut.

Le mappage de gamuts est perdant, car vous devez faire un compromis sur la façon d'approcher la plus grande gamme du contenu. La méthode la plus simple est la colorimétrie, où les couleurs qui se trouvent dans la gamme de l'affichage sont conservées et les couleurs qui sont hors de la gamme sont clippées à la valeur in-gamut la plus proche.

Dans un workflow basé sur un profil ICC, le mappage de gamut est généralement géré automatiquement dans la bibliothèque de gestion des couleurs. Vous avez un certain contrôle sur le comportement de mappage en sélectionnant l'intention de rendu (voir Modes d'intention de rendu).

> ⓘ **Notes**
>
> Lorsque vous êtes dans un workflow de couleurs avancé, nous vous déconseillons généralement d'utiliser l'intention de rendu perceptuel, ni pour la source ou la destination, car il a été conçu pour les sources et destinations SDR qui ont des gammes de couleurs plus petites que celles utilisées pour HDR et certains affichages WCG ; de sorte que leur utilisation peut entraîner un comportement inattendu.

Gestion automatique des couleurs système Windows

Lorsque La couleur avancée est active, Windows effectue une gestion automatique des couleurs système, ce qui garantit que le contenu des couleurs de votre application est reproduit avec précision sur l'écran. Cela simplifie considérablement les actions requises sur votre application, bien que les applications avancées puissent continuer à effectuer un traitement supplémentaire pour une couleur maximale et une précision perceptive. Pour plus d'informations, consultez Utiliser DirectX avec une couleur avancée sur des affichages à plage dynamique élevée/standard.

Obtenir les caractéristiques de couleur de l'affichage

Les applications avancées prenant en charge les couleurs ne doivent pas interagir directement avec le profil ICC d'affichage. Au lieu de cela, vous pouvez obtenir les propriétés de couleur de l'affichage à l'aide de DisplayInformation::GetAdvancedColorInfo ou d'IDXGIOutput6.

Effectuer la conversion de l'espace de couleur en espace colorimétrique de l'affichage

Windows effectue la conversion de l'espace de couleur en espace de couleurs de l'affichage déterminé par le profil de couleur par défaut actuel. En l'absence de profil, les données de colorimétrie EDID sont utilisées. Votre application obtient automatiquement le comportement des couleurs *référencées* par la scène, par exemple, si vous affichez le rouge sRGB encodé en tant que `RGB(1, 0, 0)` et qu'il s'affiche sur un moniteur Adobe RVB, Windows le reproduit correctement en rouge sRGB. Les applications avancées prenant en charge les couleurs doivent baliser leur contenu avec l'espace de couleur approprié pour informer Windows à l'aide d'IDXGISwapChain3::SetColorSpace1. Pour toutes les applications prenant en charge les couleurs non avancées qui s'affichent au format de pixels entiers standard (par exemple, RVBA 8 bits), Windows traite explicitement l'application comme sRVB. Si vous souhaitez afficher adobeRGB rouge dans un scénario Couleur avancée, vous devez effectuer le rendu `RGB(1.158157,0,0)` dans une surface marquée par scRGB (elle est limitée par la gamme de l'affichage).

Effectuer un mappage de gamut pour limiter à la gamut de l'affichage

Le pipeline d'affichage du GPU effectue un découpage numérique sur les couleurs hors gamme. Si votre application souhaite utiliser un mappage plus sophistiqué, vous devez le faire vous-même.

Comportement par défaut du profil ICC avec Advanced Color

La gestion automatique des couleurs système a nécessairement un impact sur le comportement des applications basées sur un profil ICC existantes, car elles effectuent elles-mêmes de nombreuses actions qui sont désormais gérées par le système d'exploitation Windows applique le comportement par défaut (décrit ci-dessous) aux applications basées sur un profil ICC. Cela garantit que ces applications n'ont pas de

comportement incorrect. Toutefois, sans autre travail, ils n'auront accès à aucune des fonctionnalités de couleur étendues.

En particulier, par défaut, votre application basée sur un profil ICC est limitée à la gamme sRGB, même si le moniteur est en fait plus large. Windows fournit également une assistance de compatibilité ICC qui peut permettre à votre application ICC d'accéder à l'ensemble de la gamme de l'affichage. Pour plus d'informations, consultez la section Afficher l'assistance sur la compatibilité du profil ICC dans cette rubrique.

Obtenir les caractéristiques de couleur de l'affichage

Lorsque Advanced Color est actif, tous les appels aux API de gestion des profils de couleur pour obtenir le profil par défaut d'un affichage retournent « aucun profil », quels que soient les profils réellement installés. Par convention, « aucun profil » doit être interprété comme sRGB.

Les profils ICC d'affichage sont toujours valides et utilisés avec Advanced Color, mais ils sont utilisés uniquement au niveau du système, et la plupart des applications ne doivent pas interagir directement avec eux. Les informations ci-dessous sont généralement nécessaires uniquement si votre application est un utilitaire qui énumère tous les profils d'affichage ou si elle crée/installe des profils.

Pour appliquer cela, Windows ajoute le concept de sous-types de `STANDARD` profil de couleur et `EXTENDED` . Cela s'applique à toutes les API de gestion des profils de couleurs qui utilisent COLORPROFILESUBTYPE :

```
C++

CPST_STANDARD_DISPLAY_COLOR_MODE
CPST_EXTENDED_DISPLAY_COLOR_MODE
```

> **Notes**
>
> `STANDARD` et `EXTENDED` les sous-types ne sont pas une propriété stockée dans le profil lui-même ; ils s'appliquent plutôt à l'association du profil à un affichage (autrement dit, lorsque le profil est ajouté à la liste d'association de profil de l'affichage). Un profil unique peut être associé à la fois `STANDARD` aux sous-types et `EXTENDED` pour un affichage, ce qui signifie qu'il serait disponible à la fois pour les scénarios Standard et Advanced Color.

Les associations de profil d'affichage destinées à être utilisées dans le SDR (SDR standard ou Advanced Color SDR) ont un sous-type `STANDARD` (thiat est la valeur par défaut si aucune valeur n'est spécifiée). Les associations de profil d'affichage à utiliser en mode HDR sont de sous-type `EXTENDED`. Si votre application ne spécifie pas de sous-type, cela est interprété comme `STANDARD`.

Toute API *getter* utilisant COLORPROFILESUBTYPE retourne uniquement les profils avec la correspondance `STANDARD` ou `EXTENDED` le sous-type. Par exemple, si HDR est actif, les seuls profils d'affichage avec le `EXTENDED` sous-type sont valides pour une utilisation, et `STANDARD` les profils de sous-type ne sont pas utilisés. *Setter* Les API peuvent spécifier le sous-type (`STANDARD` est la valeur par défaut).

Effectuer la conversion de l'espace de couleur en espace colorimétrique de l'affichage

Étant donné que les API de gestion des profils ICC retournent sRGB quand Advanced Color est actif, votre application basée sur un profil ICC gère les couleurs sur sRGB, et Windows le reproduit correctement en tant que sRGB à l'écran.

Effectuer un mappage de gamut pour limiter à la gamut de l'affichage

Tout comportement de mappage de gamut existant est conservé.

Afficher l'assistance de compatibilité du profil ICC

Lorsque La couleur avancée est active, Windows fournit une assistance de compatibilité pour les profils ICC d'affichage qui permet d'accéder à l'ensemble de la gamme de l'affichage. De cette façon, votre application continue d'obtenir des couleurs précises et larges jusqu'à la fonctionnalité signalée de l'affichage, la même fonctionnalité que celle disponible sur les moniteurs de gamut large étalonnés en mode couleur non avancé hérité aujourd'hui. Sans cette assistance, votre application sera limitée au comportement par défaut, à savoir sRGB (consultez Comportement par défaut du profil ICC avec Advanced Color).

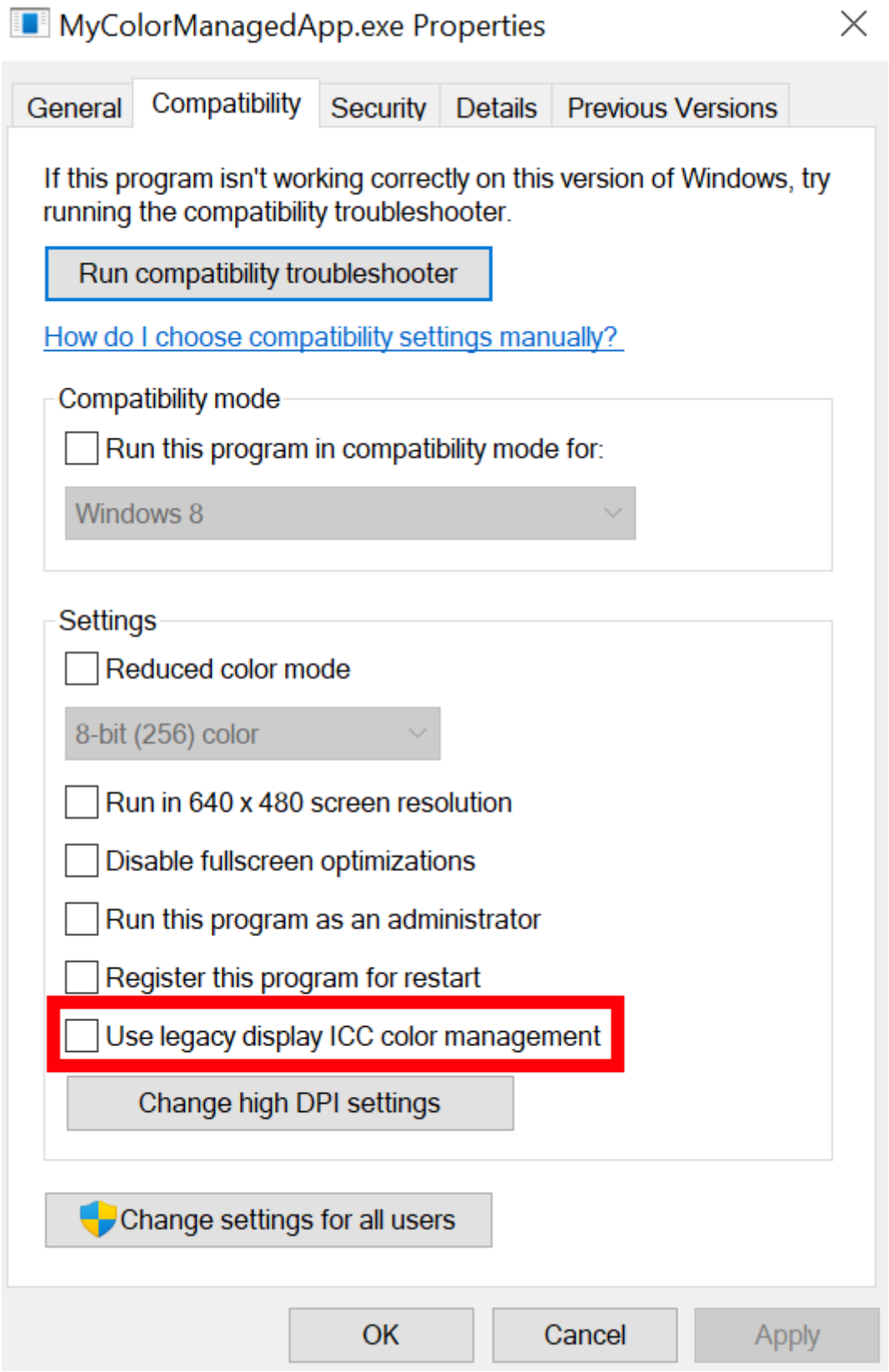
Cette assistance est disponible à partir de Windows 11. Il ne fournit pas d'autres avantages de la couleur avancée, notamment l'accès à une précision/profondeur de bits

ou à une plage dynamique élevée. Vous devez modifier votre application pour qu'elle prenne en charge les couleurs avancées.

Activation de l'assistance de compatibilité ICC d'affichage

L'assistance de compatibilité ICC d'affichage est activée par application. Il n'est pas activé par défaut.

Les utilisateurs peuvent l'activer pour une application en accédant à l'onglet Compatibilité des propriétés de l'exécutable, puis en sélectionnant **Utiliser la gestion des couleurs ICC d'affichage hérité**. L'assistance de compatibilité est appliquée à l'ensemble du processus et est active uniquement lorsque la couleur avancée est activée pour l'affichage. Elle n'a aucun effet sur un affichage SDR standard.



Windows active automatiquement l'assistance pour certaines applications populaires connues pour utiliser la gestion des profils de couleurs ICC.

Il n'existe aucun moyen par programme d'activer cette assistance de compatibilité pour votre application.

Obtenir les caractéristiques de couleur de l'affichage

Si l'assistant de compatibilité est actif, lorsque votre application interroge le profil de couleur par défaut `STANDARD` à l'aide des fonctions de gestion des profils du système de couleurs Windows, Windows construit un profil ICC synthétique à l'aide des mêmes données que celles qui remplissent les API des fonctionnalités d'affichage des couleurs avancées. Les données du profil synthétique peuvent provenir d'une combinaison du profil de couleur actuel, de l'EDID ou displayID de l'affichage ou d'autres sources.

Si votre application interroge le profil de couleur par défaut `EXTENDED` , cela indique que votre application prend en charge les couleurs avancées et qu'elle recevra le profil réel `EXTENDED` .

Effectuer la conversion de l'espace de couleur en espace de couleurs de l'affichage

Si l'assistance de compatibilité est active, votre application doit utiliser la gestion des couleurs ICC pour cibler le profil d'affichage synthétique. Windows suppose que votre application cible cet espace de couleurs et effectue la conversion correcte de l'espace de couleur pour s'assurer qu'il est correctement affiché sur l'écran.

La conversion de l'espace de couleurs s'applique à l'ensemble du processus d'application, de sorte que tout le contenu visuel de votre application est traité comme ciblant l'espace de couleurs de l'affichage, même si certaines d'entre elles ne sont pas gérées par les couleurs et ciblent nominalement sRGB (par exemple, l'interface utilisateur). La conversion d'espace de couleurs est également appliquée indépendamment de l'API graphique (GDI, DirectX, XAML, etc.), du format de pixels ou d'autres caractéristiques de votre contenu rendu.

Effectuer le mappage de gamuts pour limiter le gamut de l'affichage

Tout comportement de mappage de gamuts existant est conservé.

Changements de comportement visibles par l'utilisateur

Les utilisateurs peuvent vérifier si l'assistance de compatibilité ICC d'affichage est active pour un exécutable en vérifiant son onglet Propriétés de compatibilité. Si votre application affiche des informations sur le profil ICC d'affichage par défaut, les utilisateurs verront qu'il s'agit d'un profil synthétique. Le contenu descriptif du profil (nom compris) est un détail d'implémentation.

Le comportement réel des couleurs doit être identique au moment où la couleur avancée est désactivée. Dans les deux cas, votre application affiche des couleurs précises qui peuvent accéder à la gamme complète de l'affichage, comme décrit par le profil ICC.

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

Pipeline d'étalonnage des couleurs d'affichage matériel Windows

Article • 14/07/2023

Cette rubrique traite de l'étalonnage des couleurs d'affichage à l'aide d'un nouveau pipeline de transformation des couleurs d'affichage GPU pris en charge par Windows 10, version 2004 (20H1) et versions ultérieures. Le pipeline offre une précision des couleurs considérablement améliorée par rapport aux chemins existants, tels que le pipeline *de rampe gamma* GDI, et ajoute la prise en charge des affichages HDR.

Cette rubrique s'adresse aux fabricants d'affichage et de PC et aux fournisseurs d'étalonnage d'affichage qui souhaitent mieux étalonner les écrans de leurs clients. La plupart des applications Windows n'ont pas besoin de faire quoi que ce soit pour tirer parti du pipeline ; Mais si vous développez des applications gérées par les couleurs, vous voudrez peut-être être conscient du fonctionnement de cette technologie.

Le nouveau pipeline de couleurs est disponible pour n'importe quel affichage si le GPU répond à la configuration système requise. Si l'affichage est HDR ou utilise la gestion automatique des couleurs, il existe d'autres considérations et exigences. qui peut être trouvé dans Utiliser DirectX avec couleur avancée sur les affichages de plage dynamique haute/standard.

Introduction

L'étalonnage des couleurs d'affichage est le processus qui permet de s'assurer qu'un affichage correspond exactement à son espace de couleurs signalé ; par exemple, sRGB ou DCI-P3 D65. En raison de variations dans le processus de fabrication et d'autres sources, un panneau d'affichage individuel peut s'écarter de sa spécification. Une fois qu'un affichage a été étalonné, vos applications et votre contenu peuvent cibler en toute confiance l'espace de couleur de l'affichage sans se soucier de cette variabilité ou de cette inexactitude.

À un niveau élevé, l'étalonnage des couleurs d'affichage implique les étapes suivantes :

1. Effectuez des mesures optiques de la sortie de couleur réelle d'un affichage lors du rendu d'un ensemble de valeurs de couleur connues.
2. En fonction des données de mesure, générez une transformation de couleur qui corrige les inexactitudes dans l'affichage et générez des métadonnées qui décrivent le volume de couleurs résultant de l'affichage.

3. Stockez les données de transformation de couleur et affichez les métadonnées pour une utilisation ultérieure.
4. Au moment de l'exécution, chargez et appliquez la transformation de couleur au framebuffer d'affichage (valeurs de couleur envoyées à l'affichage) et signalez les métadonnées d'affichage aux applications.

Windows 10, la version 2004 fournit des fonctionnalités améliorées pour les étapes 3 et 4, tandis que les fabricants d'affichage et les fournisseurs d'étalonnage sont responsables des étapes 1 et 2.

Configuration système requise

Le nouveau pipeline de transformation de couleur nécessite un GPU et un pilote d'affichage compatibles. Les architectures GPU prises en charge sont les suivantes :

- AMD:
  - AMD RX 500 série 400 ou ultérieure
  - Processeurs AMD Ryzen avec Graphics
- Intel:
  - Intégré : Gpu Intel 10e génération (Ice Lake) ou version ultérieure
  - Discret : Intel DG1 ou version ultérieure
- NVIDIA GTX 10xx ou version ultérieure (Pascal+)
- Qualcomm 8CX Gen 3 ou version ultérieure ; 7C Gen 3 ou version ultérieure

> ⓘ **Notes**
>
> Les chipsets De nom de code Intel Comet Lake (code de modèle à 5 chiffres) ne sont pas pris en charge.

Un pilote Windows Display Driver Model (WDDM) 2.6 ou version ultérieure est nécessaire (publié avec Windows 10, version 1903). Certains fournisseurs de GPU ont besoin d'un pilote plus récent, potentiellement aussi nouveau que WDDM 3.0 (publié avec Windows 11, version 21H2).

Pour plus d'informations sur la façon dont une application peut déterminer si le nouveau pipeline de transformation de couleur est disponible sur un système, consultez Nouvelles API de gestion de profil ICC d'affichage .

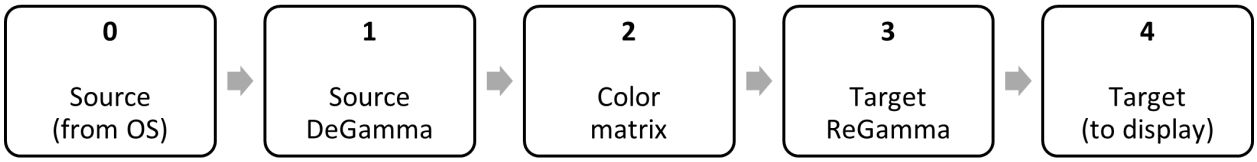
Nouveau pipeline de transformation de couleur GPU

Windows 10, la version 2004 expose un pipeline de transformation des couleurs d'affichage accéléré par GPU composé d'une matrice de couleurs gamma linéaire et de 1DLUT. Par rapport au pipeline *de rampe gamma* existant, il offre une précision, une précision et une prise en charge supérieures pour les affichages de larges gammes de couleurs. En outre, il ajoute la prise en charge de nouvelles technologies telles que les affichages HDR qui utilisent la signalisation BT.2100.

Le pipeline n'est pas directement programmable par les applications, mais il est exposé uniquement via des profils MHC ; pour plus d'informations, voir ci-dessous. D'autres fonctionnalités de système d'exploitation, telles que l'éclairage nocturne, peuvent également utiliser ce pipeline, et le système d'exploitation gère la façon de partager (composer) et/ou de rationaliser l'accès au pipeline entre plusieurs scénarios.

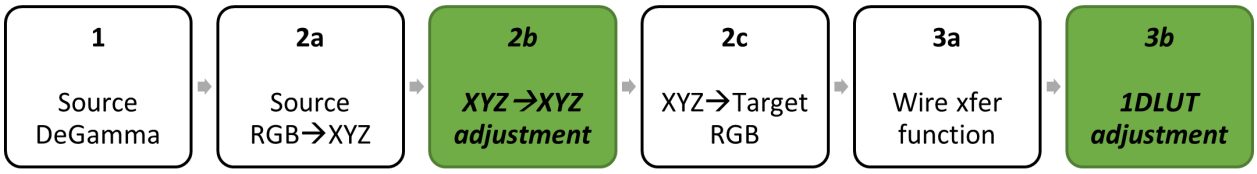
Description du pipeline de transformation de couleur

Le pipeline de transformation de couleur est basé sur le modèle conceptuel standard pour les conversions d'espace de couleurs :



Le modèle peut convertir entre deux espaces de couleurs RVB (ou autres 3 canaux), tels que sRGB en P3 D65. Il peut également corriger les types les plus courants de variation de couleur de panneau.

Le pipeline de transformation des couleurs Windows prend le modèle conceptuel, développe les phases 2 (matrice de couleurs) et 3 (regamma cible) en sous-étapes, et expose un sous-ensemble des étapes (**2b et 3b, en vert**) pour les applications à programmer, tout en laissant le reste (blanc) contrôlé par le pilote :



Ces modifications permettent au pipeline de couleurs d'être indépendant de l'espace de couleur du contenu source, qui peut changer d'image à image. En outre, il améliore la compatibilité avec les espaces de couleurs d'affichage tels que BT.2100 ST.2084, qui nécessitent des optimisations opaques afin de préserver la précision.

Étape 0 : Source (entrée graphique)

L'entrée est le framebuffer rendu à partir du système d'exploitation. Il peut se trouver dans l'un de plusieurs espaces de couleur selon le scénario, y compris sRGB, sYCC, HDR10 ou scRGB, et peut changer d'image à image.

Étape 1 : Source DeGamma

Le pilote d'affichage convertit automatiquement le contenu source en gamma linéaire, et cette étape n'est pas programmable par les applications.

Étape 2 : matrice de conversion d'espace de couleur

Dans le modèle de conversion d'espace de couleurs standard, la phase de matrice peut être divisée en trois matrices, qui sont composées (multipliées) ensemble :

- **2a :** Convertir à partir de l'espace de couleur RVB de contenu source (gamma linéaire) en espace de couleurs absolu ; dans le pipeline Windows, l'espace de couleurs absolu est CIEXYZ.
- **2b :** Effectuez des ajustements dans l'espace CIEXYZ, comme l'étalonnage.
- **2c :** Convertissez à partir de CIEXYZ en espace de couleurs RVB cible (gamma linéaire). L'espace de couleurs RVB cible est défini comme l'encodage utilisé lors de la transmission des couleurs sur le fil d'affichage, généralement bt.709 ou BT.2020 primaires. Il ne s'agit pas des primaires réelles mesurées du panel physique.

La matrice 2a est déterminée par le contenu source, et la matrice 2c est déterminée par le mode de signalisation de l'affichage ; seule la matrice 2b est accessible aux applications. Le pilote multiplie les trois ensembles pour générer la matrice réelle à exécuter dans le matériel :

```
FinalMatrix = SourceRGBtoXYZ * XYZtoXYZAdjust * XYZtoTargetRGB
```

> ⓘ **Notes**
>
> Étant donné que le pilote d'affichage est responsable de la source RVB vers XYZ et des conversions XYZ vers RVB cibles, la matrice que vous programmez (étape 2b) ne doit pas inclure non plus.

Exemple 1 : Si vous n'effectuez aucun ajustement des couleurs (pass-through), votre matrice doit être une identité, quel que soit le type d'affichage vers lequel vous

effectuez des opérations.

Exemple 2 : Si vous utilisez un affichage SDR P3 D65 et que vous implémentez un profil de « vérification sRGB » qui émule sRGB sur le panneau, votre matrice doit se composer d'une rotation des primaires de sRGB vers P3 D65.

Étape 3 : ReGamma cible

Cette étape peut être divisée en deux RVB 1DLUT, qui sont composées ensemble :

- **3a :** Encodez les données RVB linéaires de l'étape 2c dans la fonction de transfert/gamma du signal sur le câble d'affichage.
- **3b :** Effectuez les ajustements dans l'espace gamma cible, comme l'étalonnage.

1DLUT 3a est déterminé par l'espace de couleur du format de fil d'affichage ; le plus souvent, il s'agit de sRGB pour les affichages SDR et ST.2084 pour les affichages HDR. 3b est programmable par les applications et se produit après l'application de la fonction de transfert de format filaire. Le pilote compose les deux 1DLUT pour générer le 1DLUT réel à exécuter dans le matériel :

```
Final1DLUT = Adjustment1DLUT(TargetReGamma(input))
```

> ⓘ **Notes**
>
> Étant donné que le pilote est responsable de la programmation de la fonction de transfert de signal d'affichage, le 1DLUT que vous programmez (3b) ne doit pas inclure cet encodage. Par exemple, si vous n'effectuez aucun ajustement des couleurs (pass-through), votre 1DLUT doit être une identité, quel que soit l'espace de couleur de format de fil d'affichage.

Étape 4 : Cible (sortie à analyser)

Il s'agit du framebuffer à analyser sur le fil par le GPU ; dans l'espace de couleurs natif de l'affichage, et après tous les ajustements que vous avez programmés. Des opérations supplémentaires telles que l'encodage YCbCr peuvent se produire par la suite.

Précision et précision supérieures

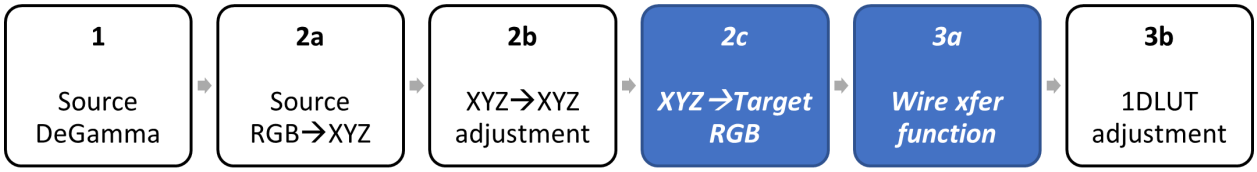
La phase de matrice gamma linéaire (ajustement XYZ à XYZ) a été introduite dans Windows 10, version 1709. La fonctionnalité vous permet d'effectuer des ajustements pour les primaires de couleur et le point blanc, ainsi que des conversions d'espace de couleur RVB arbitraires.

L'étape d'ajustement 1DLUT est conceptuellement similaire à la *rampe gamma* existante 1DLUT, mais offre une précision améliorée, avec jusqu'à 4 096 entrées LUT avec une précision de point fixe de 16 bits maximum.

> ⓘ **Notes**
>
> Tous les matériels ne prennent pas en charge le nombre total d'entrées ou la précision exposées par le pipeline de couleurs.

Prise en charge des affichages HDR (BT.2100)

Une limitation du pipeline *de rampe gamma* existant est qu'il a un comportement non défini lorsque l'affichage utilise la signalisation HDR (BT.2100 ST.2084). Le nouveau pipeline de transformation de couleur prend explicitement en charge la signalisation SDR (BT.1886 ou sRGB) et HDR, et les mises à l'échelle pour prendre en charge les futurs espaces de couleurs de format de fil. Pour ce faire, il effectue les étapes « XYZ to Target RVB » et « Wire transfer function » (**bleu**) dans le diagramme de blocs :



Ces deux étapes, qui sont contrôlées automatiquement par le pilote, sont responsables de l'encodage des couleurs dans l'espace de couleurs de format de fil : par exemple, sRGB ou BT.2020 ST.2084.

Par conséquent, lorsque vous programmez le pipeline de transformation de couleur, vous obtenez un comportement bien défini en fonction de l'espace de couleur de format de fil actif de l'affichage.

Nouvelle balise « MHC2 » pour les profils ICC

Windows ne fournit pas d'API pour contrôler directement le nouveau pipeline de transformation de couleur au moment de l'exécution. Au lieu de cela, votre application accède au pipeline en écrivant un profil de couleur ICC (International Color Consortium)

correctement mis en forme avec des données supplémentaires stockées dans une nouvelle balise privée « Microsoft Hardware Calibration » (« MHC2 »). Il s'agit d'un modèle similaire au pipeline *de rampe gamma* existant, qui utilise des balises ICC privées « VCGT ». Les profils ICC avec des données de balise SCHL2 valides sont appelés « profils ICC SCHL » ou « profils MHC ».

> ⓘ **Notes**
>
> MHC2 fait référence à la deuxième version de la balise privée, qui est disponible pour tous les appareils Windows 10, version 2004 ; SCHL1 fourni sur une version antérieure de Windows avec des PC OEM spécifiques.

Métadonnées statiques HDR supplémentaires ST.2086

En plus de programmer le nouveau pipeline de transformation de couleur, les profils ICC MHC contiennent également des métadonnées statiques HDR ST.2086. Il s'agit de valeurs qui décrivent la plage dynamique (luminance) et la gamme de couleurs d'un affichage. Ils sont largement implémentés avec les affichages HDR, mais sont utiles pour n'importe quel affichage. Les valeurs sont :

- Luminance maximale (nits)
- Luminosité pleine image maximale (nits)
- Luminance minimale (nits)
- Couleurs primaires RVB (coordonnées xy)
- Point blanc (coordonnées xy)

Le point blanc, la luminosité maximale de l'image complète et les primaires de couleur RVB sont décrits à l'aide de balises ICC standard. La luminosité maximale et la luminosité minimale sont décrites dans la balise SCHL2. Un profil doit contenir toutes ces informations pour que le système d'exploitation accepte le profil et l'utilise pour les scénarios De couleur avancée.

Windows rationalise les métadonnées ST.2086 à partir de plusieurs sources, notamment le profil ICC MHC, le pilote graphique et le microprogramme EDID ou DisplayID. Les profils ICC MHC sont traités comme la source la plus fiable et remplacent d'autres sources. Windows expose ces informations via les API de capacité HDR, comme décrit dans Utiliser DirectX avec une couleur avancée sur des affichages à plage dynamique élevée/standard. De cette façon, les applications HDR reçoivent les meilleures informations d'affichage HDR disponibles.

Définition de la luminance ST.2086 pour les écrans de rétro-éclairage réglables

Un écran peut avoir un rétro-éclairage réglable, par exemple contrôlé par l'utilisateur, ou contrôlé automatiquement par un capteur de lumière ambiante. Cela introduit une ambiguïté quant à la façon dont les valeurs de luminance ST.2086 doivent être interprétées.

Pour les écrans où Windows contrôle le rétro-éclairage (généralement pour les ordinateurs portables et les appareils à panneau intégré), les valeurs de luminosité doivent décrire le moment où ce rétro-éclairage contrôlé par le système d'exploitation est à son maximum ou le plus lumineux.

Pour les écrans où Windows n'a pas de contrôle sur le rétro-éclairage (généralement pour les moniteurs externes), les valeurs de luminosité sont précises uniquement pour l'état d'affichage au moment de la mesure.

Exigences du profil ICC

Un profil ICC MHC doit utiliser la spécification ICC version 2 (ICC.1:2001-04) ou la version 4 (ICC.1:2010-12/ISO 15076-1:2010). Un profil ICC SCHL doit être un profil d'appareil d'affichage.

Un profil ICC MHC peut inclure des données de pipeline de transformation de couleur. Les parties de la structure SCHL2 qui définissent la transformation de couleur peuvent être vides, ce qui indique explicitement une transformation d'identité.

Un profil ICC MHC doit inclure des métadonnées ST.2086. Un profil contenant uniquement des métadonnées ST.2086 et aucune donnée de transformation n'est utilisé pour les scénarios d'étalonnage d'affichage HDR. Dans ce cas, l'étalonnage HDR signifie fournir des informations plus précises sur la luminosité minimale/maximale et la gamme de couleurs pour les applications et les jeux HDR.

Réutilisation des balises publiques existantes

Les profils ICC MHC utilisent des balises publiques existantes pour définir certaines des valeurs de métadonnées ST.2086. Toutes ces étiquettes sont déjà requises pour les profils d'appareil d'affichage. Les définitions de balises et de types de données se trouvent dans les spécifications ICC.

Nom de la balise	Type de données	Valeur ST.2086	Unité signalée par Windows
redColorantTag	XYZNumber	Primaire rouge	Chromaticity (xy)
greenColorantTag	XYZNumber	Primaire verte	Chromaticity (xy)
blueColorantTag	XYZNumber	Primaire bleue	Chromaticity (xy)
mediaWhitePointTag	XYZNumber	Point blanc	Chromaticity (xy)
luminanceTag	XYZNumber	Luminosité pleine image maximale	Luminance (nits)

Définition de balise privée « SCHL2 »

Un profil ICC SCHL doit contenir une structure d'étiquette SCHL2. La matrice et les éléments de transformation de couleur 1DLUT peuvent être définis sur 0 (NULL), ce qui indique explicitement une transformation d'identité pour la phase respective. Les valeurs de métadonnées ST.2086 doivent être remplies avec des données valides.

Position des octets	Longueur du champ (octets)	Contenu	Type de données
0 à 3	4	Signature de type 'SCHL2' (4D484332h)	SCHL2Type
4 à 7	4	Décalage vers le début de l'élément de données de balise	uInt32Number
8 à 13	4	Taille de l'élément de données de balise	uInt32Number

Définition de structure MHC2Type

Position des octets	Longueur du champ (octets)	Contenu	Type de données
0 à 3	4	Signature de type 'SCHL2' (4D484332h)	
4 à 7	4	Réservé, défini sur 0	
8 à 11	4	Nombre d'entrées 1DLUT (4 096 ou moins) **[1]** **FACULTATIF :** 0 = Transformation d'identité	uInt32Number

Position des octets	Longueur du champ (octets)	Contenu	Type de données
12 à 15	4	ST.2086 min luminance in nits	S15Fixed16Number
16 à 19	4	ST.2086 peak luminance in nits	S15Fixed16Number
20 à 23	4	Décalage en octets sur la matrice **[2]** **FACULTATIF :** 0 = Transformation d'identité	uInt32Number
24 à 27	4	Décalage en octets sur 1DLUT rouge **[2]**	uInt32Number
28 à 31	4	Décalage en octets sur 1DLUT vert **[2]**	uInt32Number
32 à 35	4	Décalage en octets sur 1DLUT bleu **[2]**	uInt32Number

[1] Le système d'exploitation interpole les données au nombre d'entrées prises en charge par le matériel.

[2] Les décalages au sein de la structure MHC2Type sont relatifs au début de la structure, et non au fichier.

Définition de matrice

Position des octets	Longueur du champ (octets)	Contenu	Type de données
0 à 23	24	Matrice d'ajustement xyz vers XYZ 3x4 stockée dans l'ordre principal des lignes, la colonne 4 est ignorée **[1]**	s15Fixed16Number

[1] La structure de la matrice est dimensionnée pour s'adapter à 12 éléments pour une matrice 3x4 dans l'ordre principal des lignes. Toutefois, Windows utilise uniquement les données des trois colonnes de gauche, définissant ainsi une matrice 3x3. Par exemple, le stockage de ces 12 valeurs dans l'ordre linéaire :

```
[a, b, c, 0, d, e, f, 0, g, h, i, 0]
```

produit la matrice suivante :

Première colonne	Deuxième colonne	Troisième colonne
a	b	c
d	e	f
g	h	i

> ⓘ Notes
>
> Comme décrit dans **Matrice de conversion d'espace** de couleur, n'incluez pas les transformations RVB sources en XYZ ou XYZ pour cibler les transformations de matrice RVB, car elles sont gérées automatiquement par le pilote. RVB cible est défini comme l'encodage utilisé lors de la transmission des couleurs sur le fil d'affichage ; généralement BT.709 ou BT.2020 primaires.

Définition 1DLUT

Position des octets	Longueur du champ (octets)	Contenu	Type de données
0 à 3	4	'sf32' (73663332h) Signature de type	
4 à 7	4	Réservé, défini sur 0	
8 jusqu'à la fin	Variable (0 à 16384)	Valeurs LUT d'étalonnage normalisées à [0.0, 1.0]	s15Fixed16Number

> ⓘ Notes
>
> Comme décrit dans **Target ReGamma**, ce LUT fonctionne dans l'espace de couleur du format de fil une fois la fonction de transfert encodée.

> ⓘ Notes
>
> Si vos mesures ou courbes d'étalonnage nécessitent moins de 4 096 entrées LUT, stockez uniquement le nombre d'entrées dont vous avez réellement besoin et spécifiez le nombre dans la structure MHC2Type. Par exemple, le LUT d'identité le

plus simple nécessite seulement deux entrées définies sur 0.0 et 1.0. Le système d'exploitation est interpolé sur le nombre d'entrées prises en charge par le matériel.

Nouvelles API de gestion des profils ICC d'affichage

> ⓘ **Notes**
>
> Les instructions de cette section s'appliquent à tout profil ICC d'affichage, qu'il contienne ou non des données SCHL.

Une fois que vous avez généré un profil ICC SCHL, vous le provisionnez sur le système Windows pour l'affichage ciblé. Dans les versions antérieures de Windows, vous utilisiez les fonctions de gestion des profils WCS (Windows Color System) pour ce faire. Bien que vous puissiez continuer à utiliser ces API existantes, Windows 10, la version 2004 ajoute un ensemble de nouvelles API modernisées à WCS qui sont spécialisées pour la gestion des profils de couleurs ICC d'affichage. Ces API sont toutes précédées de « ColorProfile » :

- ColorProfileAddDisplayAssociation
- ColorProfileRemoveDisplayAssociation
- ColorProfileSetDisplayDefaultAssociation
- ColorProfileGetDeviceCapabilities

> ⓘ **Notes**
>
> L'API ci-dessus fournit des fonctionnalités pour lesquelles il n'existe pas d'équivalent d'API WCS.

- ColorProfileGetDisplayList
- ColorProfileGetDisplayDefault
- ColorProfileGetDisplayUserScope

Un flux de travail classique utilisant les API ColorProfile pour provisionner un profil ICC MHC sur le système est le suivant :

1. Utilisez **ColorProfileGetDeviceCapabilities** pour déterminer si le système prend en charge le nouveau pipeline de transformation de couleur. Même si ce n'est pas le

cas, il peut toujours être utile de provisionner le profil pour fournir des métadonnées ST.2086 supplémentaires.
2. Utilisez **InstallColorProfile** (une API WCS existante) pour installer le profil de couleur. Cela ajoute le profil à la liste des profils disponibles pour une utilisation sur le système.
3. Utilisez **ColorProfileGetDisplayUserScope** pour déterminer si l'utilisateur Windows a remplacé les associations de profil par défaut du système et utilise ses propres listes d'associations par utilisateur.
4. Utilisez **ColorProfileAddDisplayAssociation** pour associer le profil de couleur à un affichage (rendre un profil installé sélectionnable pour cet affichage) et éventuellement définir le profil comme profil par défaut (le profil actuellement actif).

Chargeur d'étalonnage d'affichage Windows amélioré

Windows propose un chargeur d'étalonnage des couleurs d'affichage de boîte de réception depuis Windows 7. Ce chargeur d'étalonnage prend en charge la lecture des profils ICC avec des données de pipeline de *rampe gamma* stockées dans des balises de profil ICC privées VCGT ou MS00. Le chargeur *de rampe gamma* doit être explicitement activé en appelant **WcsSetCalibrationManagementState**.

Windows 10, la version 2004 améliore le chargeur d'étalonnage de boîte de réception en ajoutant la prise en charge des profils ICC MHC et du nouveau pipeline de transformation de couleur. L'écriture et l'approvisionnement d'un profil ICC MHC, et le fait que le chargeur Windows applique son état, sont la seule méthode permettant aux applications d'accéder au pipeline de transformation de couleur : il n'existe pas d'API d'accès direct. Contrairement aux profils *de rampe gamma* , la lecture à partir de profils ICC MHC est toujours activée. Par conséquent, une fois qu'un profil ICC MHC est défini par défaut sur un système compatible, son état d'étalonnage est automatiquement chargé.

Scénarios HDR et Couleur avancée avec gestion automatique des couleurs système

Les nouvelles technologies de couleur avancées telles que HDR et la gestion automatique des couleurs ajoutent de nouvelles fonctionnalités à Windows, notamment une précision des couleurs supérieure et l'accès à des gammes de couleurs d'affichage

beaucoup plus grandes ; Pour plus d'informations, consultez Utiliser DirectX avec une couleur avancée sur les affichages de plage dynamique haute/standard.

La gestion avancée des couleurs et automatique des couleurs garantit une couleur d'affichage cohérente et précise du point de vue colorimétrique pour toutes les applications : héritées et modernes. Toutefois, certaines applications peuvent effectuer leur propre gestion explicite des couleurs à l'aide de profils de couleur ICC (International Color Consortium).

Lorsque la couleur avancée est active sur les affichages SDR ou HDR, le comportement des profils ICC d'affichage change de manière non compatible avec les versions descendantes. Si votre application fonctionne avec les profils ICC d'affichage, Windows propose des comportements de compatibilité pour garantir que votre application continue d'obtenir un comportement correct.

Pour plus d'informations sur les modifications apportées au comportement du profil ICC et sur la façon dont vous pouvez adapter votre application pour optimiser la compatibilité avec la couleur avancée, reportez-vous à comportement du profil ICC avec Advanced Color.

Commentaires

Informations de référence sur les éléments d'API pour WCS

Article • 13/06/2023

Les éléments d'API que WCS fournit se répartissent dans les sections suivantes :

- Fonctions
- Fonctions WCS obsolètes
- Énumérations
- Structures
- Macros pour les valeurs et couleurs CMJN
- Constantes WCS
- Schémas WCS
- WCS Interfaces
- Clés de Registre WCS

Commentaires

Fonctions WCS

Article • 13/06/2023

Dans les rubriques suivantes, les fonctions WCS sont organisées par catégorie. Cela vous permet de parcourir les fonctions qui ont des utilisations associées dans une seule rubrique. Les pages de référence de fonction sont organisées par ordre alphabétique.

Les fonctions sont regroupées comme suit :

- Fonctions de base à utiliser dans un contexte d'appareil
- Fonctions avancées pour une utilisation en dehors d'un contexte d'appareil
- Fonctions d'étalonnage et de caractérisation des appareils
- Fonctions de gestion des profils
- Fonctions ICM2 pour les modules de gestion des couleurs (CMM) à implémenter
- Liste alphabétique de toutes les fonctions WCS

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

Fonctions de base à utiliser dans un contexte d'appareil

Article • 13/06/2023

Les fonctions WCS suivantes fournissent des fonctionnalités de mappage des couleurs de base dans les contextes d'appareil. Ils sont utiles pour toutes les applications qui ont besoin d'implémenter la gestion des couleurs avec une faible surcharge et une intervention utilisateur minimale.

Fonction	Description
CheckColorsInGamut	Vérifie si les couleurs données se trouvent dans la gamut d'un appareil.
ColorCorrectPalette	Corrige les entrées d'une palette pour un contexte d'appareil.
ColorMatchToTarget	Effectue le mappage des couleurs à des fins d'aperçu.
CreateColorSpace	Crée un espace de couleurs.
DeleteColorSpace	Supprime un espace de couleurs.
EnumICMProfiles	Énumère les profils de couleur de sortie disponibles pour un contexte d'appareil donné.
EnumICMProfilesProcCallback	Fonction de rappel définie par l'application pour EnumICMProfiles. Le nom de cette fonction est également défini par l'application.
GetColorSpace	Obtient l'espace de couleur d'entrée actuel dans un contexte d'appareil.
GetICMProfile	Obtient le profil de couleur de sortie actuel d'un contexte d'appareil.
GetLogColorSpace	Obtient la structure LOGCOLORSPACE d'un contexte d'appareil.
SetColorSpace	Définit l'espace de couleur d'entrée d'un contexte d'appareil.
SetICMMode	Active ou désactive la gestion des couleurs dans un contexte d'appareil.
SetICMProfile	Définit le profil de couleur de sortie pour un contexte d'appareil donné.
WcsEnumColorProfiles	Énumère tous les profils de couleur qui répondent aux critères d'énumération dans l'étendue de gestion des profils spécifiée.

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

Fonctions avancées pour une utilisation en dehors d'un contexte d'appareil

Article • 13/06/2023

Ces fonctions fournissent des fonctionnalités avancées de gestion des couleurs en dehors des contextes d'appareil.

Fonction	Description
PCMSCALLBACKW	*PCMSCALLBACKW* (ou **ApplyCallbackFunction**) est une fonction de rappel que vous implémentez qui met à jour les données de configuration WCS pendant l'exécution de la boîte de dialogue affichée par la fonction SetupColorMatchingW .
CheckBitmapBits	Vérifie si les pixels d'un bitmap spécifié se trouvent dans la gamut de sortie d'une transformation spécifiée.
CheckColors	Détermine si les couleurs d'un tableau se trouvent dans la gamme de sortie d'une transformation spécifiée.
ConvertColorNameToIndex	Convertit les noms de couleurs d'un espace de couleurs nommé en numéros d'index dans un profil de couleur ICC (International Color Consortium).
ConvertIndexToColorName	Transforme les index d'un espace de couleurs en tableau de noms dans un espace de couleurs nommé.
CreateColorTransformW	Transforme les index d'un espace de couleurs en tableau de noms dans un espace de couleurs nommé.
CreateMultiProfileTransform	Accepte un tableau de profils ou un profil de lien d'appareil unique et crée une transformation de couleur que les applications peuvent utiliser pour effectuer le mappage des couleurs.
DeleteColorTransform	Supprime une transformation de couleur donnée.
GetCMMInfo	Récupère diverses informations sur le module de gestion des couleurs (CMM) qui a créé la transformation de couleur spécifiée.
GetNamedProfileInfo	Récupère des informations sur le profil de couleur nommé ICC (International Color Consortium) spécifié dans le premier paramètre.
ICMProgressProcCallback	Fonction de rappel fournie par l'application pour signaler la progression. Le nom de cette fonction est également défini par l'application.

Fonction	Description
SélectionnerCMM	Vous permet de sélectionner le module de gestion des couleurs (CMM) préféré à utiliser.
SetupColorMatchingW	Fournit un contrôle utilisateur sur la gestion des couleurs au moyen d'une boîte de dialogue.
TranslateBitmapBits	Convertit les couleurs bitmap à l'aide d'une transformation de couleur.
TranslateColors	Traduit un tableau de couleurs de l'espace de couleur source en espace de couleur de destination tel que défini par une transformation de couleur.
WcsCheckColors	Détermine si les couleurs d'un tableau se trouvent dans la gamme de sortie d'une transformation de couleur WCS spécifiée.
WcsTranslateColors	Traduit un tableau de couleurs de l'espace de couleur source en espace de couleur de destination tel que défini par une transformation de couleur.

Commentaires

Fonctions d'étalonnage et de caractérisation des appareils

Article • 13/06/2023

Les fonctions suivantes sont utiles pour écrire les outils d'étalonnage et de caractérisation des appareils nécessaires à l'installation et à l'étalonnage des appareils d'affichage couleur tels que les moniteurs et les imprimantes.

Fonction	Description
CloseColorProfile	Ferme un handle de profil ouvert.
CreateDeviceLinkProfile	Crée un *profil de liaison d'appareil* ICC (International Color Consortium) à partir d'un ensemble de profils de couleur, à l'aide des intentions spécifiées.
GetColorProfileElement	Copie les données d'un élément de profil balisé spécifié d'un profil de couleur spécifié dans une mémoire tampon.
GetColorProfileElementTag	Récupère le nom de balise spécifié par *dwIndex* dans la table de balises d'un profil de couleur ICC (International Color Consortium) donné, où *dwIndex* est un index de base unique dans cette table.
GetColorProfileFromHandle	Récupère le contenu du profil de couleur en fonction d'un handle pour un profil de couleur ouvert.
GetColorProfileHeader	Récupère ou dérive la structure d'en-tête ICC à partir d'un profil de couleur ICC ou d'un profil XML WCS. Les pilotes et les applications doivent supposer que le retour **de TRUE** indique uniquement qu'un en-tête correctement structuré est retourné. Chaque balise doit toujours être validée indépendamment à l'aide des API ICM2 héritées ou des API de schéma XML.
GetCountColorProfileElements	Récupère le nombre d'éléments balisés dans un profil de couleur donné.
GetPS2ColorRenderingDictionary	Récupère le dictionnaire de rendu des couleurs PostScript Level 2 à partir du profil de couleur ICC spécifié.
GetPS2ColorRenderingIntent	Récupère l'intention de rendu des couleurs PostScript niveau 2 à partir d'un profil de couleur ICC.
GetPS2ColorSpaceArray	Récupère le tableau d'espaces colorimétriques PostScript niveau 2 à partir d'un profil de couleur ICC.

Fonction	Description
IsColorProfileTagPresent	Indique si une balise ICC (International Color Consortium) spécifiée est présente dans le profil de couleur spécifié.
IsColorProfileValid	Vous permet de déterminer si le profil spécifié est un profil ICC (International Color Consortium) valide ou un handle de profil Windows Color System (WCS) valide qui peut être utilisé pour la gestion des couleurs.
OpenColorProfileW	Crée un handle pour un profil de couleur spécifié. Le handle peut ensuite être utilisé dans d'autres fonctions de gestion des profils.
SetColorProfileElement	Définit les données d'élément d'un élément de profil balisé dans un profil de couleur ICC.
SetColorProfileElementReference	Crée dans un profil de couleur ICC spécifié une nouvelle balise qui référence les mêmes données qu'une balise existante.
SetColorProfileElementSize	Définit la taille d'un élément balisé dans un profil de couleur ICC.
SetColorProfileHeader	Définit les données d'en-tête dans un profil de couleur ICC spécifié.
WcsGetCalibrationManagementState	Détermine si la gestion système de l'état d'étalonnage de l'affichage est activée.
WcsSetCalibrationManagementState	Détermine si la gestion système de l'état d'étalonnage de l'affichage est activée.

Commentaires

Fonctions de gestion des profils

Article • 13/06/2023

Fonctions de gestion des profils

Les fonctions d'API suivantes sont utiles dans la gestion des profils.

Fonction	Description
AssociateColorProfileWithDeviceW	Associe un profil de couleurs spécifié à un appareil spécifié.
[CreateProfileFromLogColorSpaceW] ((/windows/win32/api/icm/nf-icm-createprofilefromlogcolorspacew)	Convertit un espace de couleurs logique en profil d'appareil.
DisassociateColorProfileFromDeviceW	Dissocie un profil de couleur spécifié avec un appareil spécifié sur un ordinateur spécifié.
EnumColorProfilesW	Énumère tous les profils répondant aux critères d'énumération donnés.
GetColorDirectoryW	Récupère le chemin d'accès du répertoire Windows COLOR sur un ordinateur spécifié.
GetDeviceGammaRamp	Obtient la rampe gamma à partir de tableaux d'affichage en couleurs directes.
GetStandardColorSpaceProfileW	Récupère le profil de couleur inscrit pour l'espace de couleurs standard spécifié.
InstallColorProfileW	Installe un profil donné à utiliser sur un ordinateur spécifié. Le profil est également copié dans le répertoire COLOR.
RegisterCMMW	Associe une valeur d'identification spécifiée à la bibliothèque de liens dynamiques du module de gestion des couleurs (DLL CMM) spécifiée. Lorsque cet ID apparaît dans un profil de couleur, Windows peut ensuite localiser la CMM correspondante afin de créer une transformation.
SetDeviceGammaRamp	Définit la rampe gamma sur les tableaux d'affichage en couleur directe.
SetStandardColorSpaceProfileW	Inscrit un profil spécifié pour un espace de couleurs standard donné. Le profil peut être interrogé à l'aide de GetStandardColorSpaceProfileW.

Fonction	Description
UninstallColorProfileW	Supprime un profil de couleur spécifié d'un ordinateur spécifié. Les fichiers associés sont éventuellement supprimés du système.
Annuler l'inscriptionCMMW	Dissocie une valeur d'ID spécifiée d'une bibliothèque de liens dynamiques de module de gestion des couleurs (DLL CMM).
WcsAssociateColorProfileWithDevice	Associe un profil de couleur WCS spécifié à un appareil spécifié.
WcsCreateIccProfile	Convertit un profil WCS en profil ICC.
WcsDisassociateColorProfileFromDevice	Dissocie un profil de couleur WCS spécifié avec un appareil spécifié sur un ordinateur spécifié.
WcsEnumColorProfiles	Énumère tous les profils de couleur qui répondent aux critères d'énumération dans l'étendue de gestion des profils spécifiée.
WcsEnumColorProfilesSize	Retourne la taille, en octets, de la mémoire tampon requise par la fonction WcsEnumColorProfiles pour énumérer les profils de couleur.
WcsGetDefaultColorProfile	Récupère le profil de couleur par défaut d'un appareil, ou le profil par défaut indépendant de l'appareil si l'appareil n'est pas spécifié.
WcsGetDefaultColorProfileSize	Retourne la taille, en octets, du nom de profil de couleur par défaut d'un appareil, y compris la marque de fin **NULL** .
WcsGetDefaultRenderingIntent	Récupère l'intention de rendu par défaut dans l'étendue de gestion de profil spécifiée.
WcsGetUsePerUserProfiles	Détermine si l'utilisateur a choisi d'utiliser une liste d'association de profil par utilisateur pour l'appareil spécifié.
WcsOpenColorProfileW	Crée un handle pour un profil de couleur spécifié.
WcsSetDefaultColorProfile	Définit le nom de profil de couleur par défaut du type de profil spécifié dans l'étendue de gestion des profils spécifiée.
WcsSetDefaultRenderingIntent	Définit l'intention de rendu par défaut dans l'étendue de gestion des profils spécifiée.

Fonction	Description
WcsSetUsePerUserProfiles	Permet à l'utilisateur de spécifier s'il faut utiliser une liste d'association de profil par utilisateur pour l'appareil spécifié.

Fonctions de consommation de profil

Les API de consommation de profil sont celles dans ICM2 qui prennent des profils XML ICC ou WCS, des descripteurs de profil ou des intentions de rendu en tant que paramètres, ainsi qu'un ensemble de nouvelles API pour la prise en charge des profils WCS pour le code de gestion des couleurs des applications.

Fonctions de gestion des profils et des profils

Le workflow de gestion des profils est basé sur les API ICM2 existantes qui sont augmentées pour fournir des fonctionnalités supplémentaires pour la révision du code d'application.

Les profils contiennent des informations utilisées par les algorithmes de traitement des couleurs pour traduire la couleur entre différents espaces de couleurs. La gestion des profils permet d'interroger et de spécifier les profils utilisés à différentes étapes par le modèle de traitement des couleurs pour gérer la sortie des couleurs de différents périphériques avec diverses caractéristiques de couleur.

La gestion des profils fournit l'ensemble de fonctionnalités suivant :

1. Installation de profils de couleur à utiliser dans le système.

2. Association d'un ou de plusieurs profils de couleur installés à un appareil particulier.

3. Choix d'un profil de couleur par défaut d'un type particulier parmi les profils disponibles pour une utilisation à une étape particulière du traitement des

couleurs. Il peut s'agir d'un appareil parmi les profils qui lui sont associés, ou parmi les profils installés dans le système et non spécifiques à l'appareil.

4. Énumération des profils de couleur qui répondent à des critères particuliers parmi les profils installés dans le système.

Les extensions de nom de fichier de profil WCS sont « .cdmp » pour les DPM, .camp » pour les camps et .gmmp » pour les MPM.

Gestion des profils par utilisateur et activation de l'exécution dans le contexte LUA

L'objectif de la conception décrite dans le document actif est le suivant :

1. L'implémentation ICM2 héritée ne prend pas en charge la gestion des profils par utilisateur. Différents utilisateurs ne peuvent pas avoir leurs propres paramètres de profil. Dans Vista, l'infrastructure de gestion des profils WCS permet aux utilisateurs de configurer des paramètres de profil individuels pour la plupart des fonctionnalités.

2. Toutes les API de gestion des profils ICM2 héritées modifient les paramètres à l'échelle du système et nécessitent des privilèges d'administration. Dans Windows Vista, tous les utilisateurs s'exécutent dans les paramètres de compte d'utilisateur avec privilèges minimum (LUA) la plupart du temps, et les administrateurs peuvent élever les privilèges de manière sélective pour exécuter des applications qui modifient les paramètres à l'échelle du système. Dans la gestion des profils WCS, tous les paramètres de profil par utilisateur sont configurables dans le contexte LUA. Les applications de gestion des profils peuvent s'exécuter en tant que paramètres LUA, ce qui augmente leur étendue d'utilisation et garantit que la sécurité du système n'est pas compromise.

La gestion des profils dans Vista offre les améliorations suivantes par rapport à l'infrastructure ICM2 héritée :

1. Il permet l'association de profil avec les appareils, les paramètres de profil par défaut et l'énumération des profils dans l'étendue par utilisateur et à l'échelle du

système.

2. L’installation d’un profil reste à l’échelle du système et nécessite des privilèges d’administrateur. Cela est cohérent avec l’installation du profil pendant l’installation de l’appareil, car l’installation de l’appareil est à l’échelle du système et nécessite des privilèges administratifs.

La possibilité d’installer des appareils à partir du contexte LUA est particulière à ce qui est pris en charge pour cette classe d’appareil. Par exemple, dans Vista, il est possible d’effectuer l’installation de l’imprimante à partir du contexte LUA si l’utilisateur a obtenu des droits de copie de fichiers dans le magasin de pilotes par un administrateur de domaine à l’aide de stratégies de magasin de pilotes. L’infrastructure de gestion des profils de couleur n’a pas besoin de faire quelque chose de spécial à cet égard, car l’installation se produit dans le contexte du spouleur.

3. La modification des paramètres de profil dans l’étendue par utilisateur peut être effectuée dans le contexte de LUA ; les modifications à l’échelle du système nécessitaient des privilèges d’administration. Les opérations de gestion des profils qui nécessitent la lecture des informations de configuration peuvent être effectuées dans le contexte de LUA pour les paramètres par utilisateur et à l’échelle du système.

L’étendue de gestion des profils indique l’étendue des opérations effectuées ; par utilisateur ou à l’échelle du système.

Pour chaque opération, il est indiqué si elle peut être effectuée à partir du contexte LUA. Si une opération ne peut pas être effectuée dans le contexte LUA, l’API de gestion de profil correspondante retourne un échec avec accès refusé. Les applications qui utilisent l’API, telles que la gestion des couleurs Panneau de configuration, peuvent permettre à l’utilisateur d’élever vers un contexte administratif (à l’aide d’OTS ou d’une interface utilisateur de consentement), puis d’appeler l’API à partir du contexte avec élévation de privilèges afin que l’opération réussisse.

Opération

Étendue de gestion des profils

Pré-condition

Post-condition

Exécutable dans le contexte LUA

${ROWSPAN2}profil $Install${REMOVE}$

À l'échelle du système

Profil copié, installé dans le système et disponible pour utilisation. Le profil est énumérable à l'échelle du système et dans l'étendue utilisateur actuel pour tous les utilisateurs.

Pendant l'installation du pilote de périphérique, régie par les stratégies d'installation du pilote. « NON » dans le cas contraire.

Utilisateur actuel

Non pris en charge

${ROWSPAN2}profil $Uninstall${REMOVE}$

À l'échelle du système

Le profil est installé dans le système

Profil désinstallé du système et éventuellement supprimé du magasin de profils. Le profil n'est plus disponible pour une utilisation et n'est énumérable dans aucune étendue.

Non

Utilisateur actuel

Non pris en charge

${ROWSPAN2}$Associate profil avec appareil${REMOVE}$

À l'échelle du système

Le profil est installé et est de type ICC ou CDMP

Le profil peut être utilisé avec l'appareil par tous les utilisateurs. Il est énumérable, dans l'étendue du système et également dans l'étendue de l'utilisateur actuel pour tous les utilisateurs, comme associés à l'appareil.

Non

Utilisateur actuel

Le profil est installé. Peu importe que le profil soit déjà associé à l’appareil dans l’étendue du système et qu’il soit de type ICC ou CDMP.

Le profil est disponible pour une utilisation avec l’appareil par l’utilisateur actuel. Il n’est énumérable que dans l’étendue de l’utilisateur actuel (sauf s’il existe également une association à l’échelle du système) comme associé à l’appareil.

Oui

${ROWSPAN2}$Disassociate profil de l’appareil${REMOVE}$

À l’échelle du système

Le profil est associé à l’appareil dans l’étendue du système et est de type ICC ou CDMP

Le profil n’est plus disponible pour l’utilisation (à l’exception des utilisateurs qui ont également cette association dans leur étendue utilisateur actuel). Il n’est pas énumérable dans l’étendue du système. Il peut toutefois être énumérable dans l’étendue de l’utilisateur actuel, pour un utilisateur qui a cette association dans son étendue.

Non

Utilisateur actuel

Le profil est associé à l’appareil dans l’étendue utilisateur actuel (qu’il soit associé ou non dans l’étendue du système) et est de type ICC ou CDMP.

Le profil n’est plus disponible pour l’utilisation, ou énumérable comme associé à l’appareil, par l’utilisateur actuel (sauf s’il est également associé dans l’étendue du système à l’appareil).

Oui

${ROWSPAN2}$Set profil pour un type (DMP ou ICC) par défaut pour un appareil${REMOVE}$

À l’échelle du système

Le profil est de type ICC ou CDMP

Le profil est utilisé par défaut, pour le type particulier avec l’appareil, pour tous les utilisateurs, à l’exception de ceux qui ont remplacé ce paramètre dans leur étendue utilisateur actuel. (Le profil est installé et associé à l’ensemble du système de l’appareil, si ce n’est pas déjà le cas.)

Non

Utilisateur actuel

Le profil est de type ICC ou CDMP

Le profil est utilisé par défaut pour le type particulier avec l'appareil dans le cas de l'utilisateur actuel, quelle que soit la valeur par défaut à l'échelle du système. (Le profil est installé et associé à l'appareil pour l'utilisateur actuel, si ce n'est pas déjà le cas.)

Oui, si le profil est déjà installé

${ROWSPAN2}$Set profil pour une combinaison de type (ICC, DMP, CAMP, GMMP) et de sous-type comme valeur par défaut globale${REMOVE}$

À l'échelle du système

Seuls les profils ICC et CDMP peuvent être associés à des appareils.

Le profil est utilisé par défaut pour le type particulier. Les utilisateurs peuvent remplacer ce paramètre dans l'étendue utilisateur actuel. (Le profil est installé, si ce n'est pas déjà le cas.)

Non

Utilisateur actuel

Seuls les profils ICC et CDMP peuvent être associés à des appareils.

Le profil est utilisé par défaut pour le type particulier de l'utilisateur actuel. (Le profil est installé, si ce n'est pas déjà le cas.)

Oui, si le profil est déjà installé.

${ROWSPAN2}$Erase le remplacement de l'utilisateur actuel pour un paramètre de profil par défaut particulier, afin que la valeur par défaut du système soit toujours utilisée (comme secours) même pour l'étendue utilisateur actuel.${REMOVE}$

À l'échelle du système

Non applicable

Utilisateur actuel

Même pour les requêtes utilisateur actuelles sur les paramètres de profil par défaut, les paramètres à l'échelle du système sont retournés pour utilisation.

Oui

${ROWSPAN2} $Enumerate les profils installés répondant à des critères particuliers (comme la classe d'appareil, la classe de profil, etc.) ${REMOVE}$

À l'échelle du système

Seuls les profils ICC et CDMP peuvent être associés et énumérés pour les appareils.

Les profils installés et qui répondent aux critères spécifiés dans l'étendue du système sont énumérés.

Oui

Utilisateur actuel

Seuls les profils ICC et CDMP peuvent être associés aux appareils et donc énumérés pour les appareils.

Les profils qui sont installés et qui répondent aux critères spécifiés dans l'étendue du système sont énumérés.

Oui

${ROWSPAN2}$Enumerate profils associés à un appareil particulier répondant à des critères particuliers, tels que la classe d'appareil et la classe de profil${REMOVE}$

À l'échelle du système

Seuls les profils ICC et CDMP peuvent être associés et énumérés pour les appareils.

Les profils associés à l'appareil dans l'étendue du système et qui répondent aux critères spécifiés dans l'étendue du système sont énumérés.

Oui

Utilisateur actuel

Seuls les profils ICC et CDMP peuvent être associés et énumérés pour les appareils.

Les profils disponibles comme associés à l'appareil dans l'étendue utilisateur actuel, qui inclut les associations à l'échelle du système et qui répondent aux critères spécifiés dans l'étendue de l'utilisateur actuel sont énumérés.

Oui

Les types de profils de couleur valides sont fournis par l'énumération COLORPROFILETYPE.

Les sous-types de profil de couleur valides sont fournis par l'énumération COLORPROFILESUBTYPE.

Les combinaisons de type de profil/sous-type valides sont indiquées dans le tableau suivant.

COLORPROFILETYPE

COLORPROFILESUBTYPE valide

Notes

Paramètre par défaut de l'appareil

Valeur par défaut globale

Utilisation prévue

Utilisation prévue

CPT_ICC

CPST_NONE

Obtenir/définir le profil ICC par défaut associé à un appareil

CPST_RGBWorkingSpace ou CPST_CustomWorkingSpace

Obtenez/définissez le profil ICC comme profil RVB global ou un profil d'espace de travail personnalisé. Consultez Remarque.

Les CPT_ICC et CPT_DMP COLORPROFILETYPE s'excluent mutuellement. Le profil de couleur par défaut que vous définissez pour un espace de travail donné (RVB ou Personnalisé) peut être un profil ICC ou un profil DMP, mais pas les deux.

CPT_DMP

CPST_NONE

Obtenir/définir le profil DMP par défaut associé à un appareil

CPST_RGBWorkingSpace ou CPST_CustomWorkingSpace

Obtenez/définissez le profil DMP en tant que profil RVB global ou espace de travail personnalisé. Consultez Remarque.

Les CPT_ICC et CPT_DMP COLORPROFILETYPE s'excluent mutuellement. Le profil de couleur par défaut que vous définissez pour un espace de travail donné (RVB ou

Personnalisé) peut être un profil ICC ou un profil DMP, mais pas les deux.

> ⓘ **Notes**
>
> Lorsque WcsSetDefaultColorProfile est appelé pour définir un profil DMP comme profil par défaut pour l'espace de travail RVB ou un espace de travail personnalisé, seul un profil DMP de type RGBVirtualDevice, LCD ou CRT est valide.
>
> Lorsque WcsSetDefaultColorProfile est appelé pour définir un profil ICC comme profil par défaut pour l'espace de travail RVB ou un espace de travail personnalisé, seul un profil ICC dont la classe est « spac » ou « disp » et dont l'espace de couleurs est « RVB » est valide.

L'architecture est conçue en fonction des exigences des opérations, comme indiqué dans les énumérations et les tableaux ci-dessus.

Couche d'API publique de gestion des profils

Étant donné que l'étendue de gestion des profils n'est pas prise en charge par les API ICM2 héritées, un nouvel ensemble d'API de gestion des profils WCS est requis qui définit l'étendue de gestion des profils en tant qu'utilisateur actuel ou à l'échelle du système. ? Les API ICM2 héritées continuent d'être prises en charge à des fins de compatibilité descendante et fonctionnent sur l'étendue de gestion des profils implicite pour l'appel. o LES API ICM2 qui fonctionnent sur l'étendue utilisateur actuel ? Il s'agit des opérations prises en charge à l'échelle du système et de l'étendue utilisateur actuel dans la gestion des profils WCS. Les API ICM2 héritées appellent les nouvelles API WCS avec l'étendue de gestion des profils en tant qu'utilisateur actuel. Cela est logique du point de vue de l'utilisateur, car cela permet d'activer les paramètres par utilisateur à partir d'applications héritées et d'exécuter la plupart des opérations dans le contexte LUA. o LES API ICM2 qui fonctionnent sur l'étendue du système ? Il s'agit des opérations (installer des profils et désinstaller des profils) qui prennent uniquement en charge l'étendue du système. Aucune nouvelle API de gestion de profil WCS n'est créée et les API existantes peuvent être modifiées.

Les implémentations sous-jacentes des opérations de gestion des profils fonctionnent sur les entités de données de configuration suivantes pour créer le contexte pour les

algorithmes de traitement des couleurs afin de fournir des fonctionnalités de gestion des couleurs. Il s'agit de paramètres spécifiques à l'appareil ou globaux (indépendants de l'appareil). o Données de configuration spécifiques à l'appareil : ? Liste des profils associés à un appareil particulier. ? Profil par défaut pour différents types de profils associés à un appareil. ? Mode de correspondance des profils utilisés pour l'énumération. o Données de configuration globale : ? Liste des profils installés dans le système. ? Profil par défaut global pour différents types de profils. ? Les implémentations sous-jacentes du stockage des données de configuration prennent une étendue de stockage pour les données de configuration (indépendantes de l'appareil ou spécifiques à l'appareil), qui peuvent être à l'échelle du système ou utilisateur actuel. Ceci est différent de l'étendue de gestion des profils. Une opération avec l'étendue de gestion du profil utilisateur actuel peut entraîner une lecture à partir d'une étendue de stockage à l'échelle du système si le paramètre utilisateur actuel pour cette opération n'est pas présent. ? La couche API ICM2/WCS appelle dans cette couche de stockage pour obtenir et définir des données avec l'étendue de stockage appropriée. La couche de stockage est transparente pour l'étendue de gestion des profils. Logique permettant de combiner des données provenant d'étendues de stockage utilisateur actuel et à l'échelle du système pour créer ou mettre à jour une configuration en fonction de l'étendue de gestion des profils spécifiée par l'appelant d'API. Cette logique est présente dans la couche API ICM2/WCS.

Couche de stockage spécifique à l'appareil

Le stockage de différentes classes d'appareils comme l'impression, la capture ou l'affichage peut être différent les uns des autres. Par exemple, les données de configuration d'un périphérique d'impression doivent être stockées à l'aide d'API d'impression standard, telles que SetPrinterDataEx et GetPrinterDataEx, pour permettre la copie des profils et le transfert des paramètres vers un ordinateur client pendant la connexion Point-and-Print. ? Cette couche exporte des fonctionnalités pour ouvrir le magasin, obtenir des données, définir des données et fermer le magasin à l'aide d'interfaces prédéfinies courantes afin que la couche de stockage de configuration de gestion des profils puisse les appeler tout en étant transparent sur la façon dont les données sont stockées pour cet appareil.

Le diagramme suivant illustre cette architecture.

Couche d'API publique de gestion des profils

${ROWSPAN2}$Legacy API ICM2 pour les opérations qui prennent uniquement en charge l'étendue de gestion des profils à l'échelle du système dans Vista (installer, désinstaller et obtenir le répertoire de couleurs). Ils appellent la couche de stockage de configuration avec l'étendue de stockage appropriée.${REMOVE}$

API ICM2 héritée pour les opérations qui prennent en charge l'étendue de gestion des profils utilisateur actuels et à l'échelle du système dans Vista (toutes les opérations autres que l'installation, la désinstallation et l'obtention du répertoire de couleurs). Ils fonctionnent implicitement sur l'étendue de l'utilisateur actuel et appellent la nouvelle API WCS avec l'étendue de gestion des profils en tant qu'utilisateur actuel.

Nouvelle API WCS avec prise en charge de l'étendue de gestion des profils utilisateur actuels et à l'échelle du système. Ils appellent la couche de stockage de configuration avec l'étendue de stockage appropriée.

Couche de stockage configuration de la gestion des profils

Routines de configuration globale indépendantes de l'appareil

Routines de configuration spécifiques à l'appareil

${ROWSPAN3}$Profile l'installation et la gestion des paramètres de profil par défaut indépendants de l'appareil, prise en charge dans l'étendue du stockage à l'échelle du système et de l'utilisateur actuel.${REMOVE}$

Gestion des paramètres d'association d'appareils et de profil par défaut spécifiques à l'appareil, prise en charge dans l'étendue du stockage à l'échelle du système et de l'utilisateur actuel.

couche stockage Device-Specific

Imprimer un stockage spécifique

Afficher un stockage spécifique

Capturer un stockage spécifique

Les API ICM2 héritées pour les opérations qui prennent en charge uniquement l'étendue de gestion des profils à l'échelle du système dans Vista n'ont aucun changement de comportement. Les opérations d'installation et de désinstallation appartiennent à cette catégorie.

Les API ICM2 héritées pour les opérations qui prennent en charge l'étendue de gestion des profils utilisateur actuel et à l'échelle du système ont changé de comportement pour interroger et configurer les paramètres de l'utilisateur actuel. Toutes les opérations autres que l'installation et la désinstallation appartiennent à cette catégorie.

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

Fonctions WCS pour les modules de gestion des couleurs (CMM) à implémenter

Article • 13/06/2023

Les fonctions suivantes doivent être implémentées par des modules de gestion des couleurs (CMM) et exportées pour que le système d'exploitation puisse appeler.

Fonction	Description
CMCheckColors	Détermine si les couleurs spécifiées se trouvent dans la gamme de sortie d'une transformation spécifiée.
CMCheckColorsInGamut	Détermine si les triples RVB spécifiés se trouvent dans la gamme de sortie d'une transformation spécifiée.
CMCheckRGBs	Vérifie les couleurs bitmap par rapport à une gamme de sortie.
CMConvertColorNameToIndex	Convertit les noms de couleurs d'un espace de couleurs nommé en nombres d'index dans un profil de couleur
CMConvertIndexToColorName	Transforme les index d'un espace de couleurs en tableau de noms dans un espace de couleurs nommé.
CMCreateDeviceLinkProfile	Crée un profil de liaison d'appareil au format spécifié par l'International Color Consortium dans sa spécification de format de profil ICC.
CMCreateMultiProfileTransform	Accepte un tableau de profils ou un profil de liaison d'appareil unique et crée une transformation de couleur. Cette transformation est un mappage de l'espace de couleurs spécifié par le premier profil à celui du deuxième profil, et ainsi de suite, au dernier.
CMCreateProfile	Crée un profil de couleur d'affichage à partir d'une structure LOGCOLORSPACEA .
CMCreateProfileW	Crée un profil de couleur d'affichage à partir d'une structure LOGCOLORSPACEW .
CMCreateTransform	Action déconseillée. Il n'existe pas d'API de remplacement, car celle-ci n'était plus utilisée. Les développeurs d'autres modules CMM ne sont pas obligés de l'implémenter.

Fonction	Description
CMCreateTransformExt	Crée une transformation de couleur qui mappe d'une entrée LOGCOLORSPACEA à un espace cible facultatif, puis à un appareil de sortie, à l'aide d'un ensemble d'indicateurs qui définissent la façon dont la transformation doit être créée.
CMCreateTransformExtW	Crée une transformation de couleur qui mappe à partir d'une entrée LOGCOLORSPACEW à un espace cible facultatif, puis à un appareil de sortie, à l'aide d'un ensemble d'indicateurs qui définissent la façon dont la transformation doit être créée.
CMCreateTransformW	Action déconseillée. Il n'existe pas d'API de remplacement, car celle-ci n'était plus utilisée. Les développeurs d'autres modules CMM ne sont pas obligés de l'implémenter.
CMDeleteTransform	Supprime une transformation de couleur spécifiée et libère toute la mémoire qui lui est associée.
CMGetInfo	Récupère diverses informations sur le module de gestion des couleurs (CMM).
CMGetNamedProfileInfo	Récupère des informations sur le profil de couleur nommé spécifié.
CMGetPS2ColorRenderingDictionary	Obtient un dictionnaire de rendu des couleurs PostScript.
CMGetPS2ColorRenderingIntent	Récupère l'intention de rendu des couleurs PostScript niveau 2 à partir d'un profil.
CMGetPS2ColorSpaceArray	Obtient un tableau d'espaces colorimétriques PostScript.
CMIsProfileValid	Indique si le profil donné est un profil ICC valide qui peut être utilisé pour la gestion des couleurs.
CMTranslateColors	Convertit un tableau de couleurs d'un espace de couleurs source en espace colorimétrique de destination à l'aide d'une transformation de couleur.
CMTranslateRGB	Convertit un RGBQuad fourni par l'application en espace de couleurs de l'appareil.
CMTranslateRGBs	Convertit une bitmap d'un espace de couleurs à un autre à l'aide d'une transformation de couleur.

Fonction	Description
CMTranslateRGBsExt	Convertit une bitmap d'un format défini dans un autre format défini et appelle régulièrement une fonction de rappel, le cas échéant, pour signaler la progression et permettre à l'application appelante d'arrêter la traduction.

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

Liste alphabétique de toutes les fonctions WCS

Article • 13/06/2023

Voici une liste alphabétique complète des fonctions d'API WCS 1.0 fournies par Windows 98 et versions ultérieures et Windows 2000 et versions ultérieures.

Fonction ou structure	Description
PCMSCALLBACKW	*PCMSCALLBACKW* (ou **ApplyCallbackFunction**) est une fonction de rappel que vous implémentez qui met à jour les données de configuration WCS pendant l'exécution de la boîte de dialogue affichée par la fonction SetupColorMatchingW .
AssociateColorProfileWithDeviceW	Associe un profil de couleur spécifié à un appareil spécifié.
CheckBitmapBits	Vérifie si les pixels d'un bitmap spécifié se trouvent dans la gamut de sortie d'une transformation spécifiée.
CheckColors	Détermine si les couleurs d'un tableau se trouvent dans la gamme de sortie d'une transformation spécifiée.
CheckColorsInGamut	Vérifie si les couleurs données se trouvent dans la gamut d'un appareil.
CloseColorProfile	Ferme un handle de profil ouvert.
CMCheckColors	Détermine si les couleurs données se trouvent dans la gamme de sortie d'une transformation spécifiée.
CMCheckColorsInGamut	Détermine si les triples RVB spécifiés se trouvent dans la gamme de sortie d'une transformation spécifiée.
CMCheckRGBs	Vérifie les couleurs bitmap par rapport à un gamut de sortie.
CMConvertColorNameToIndex	Convertit les noms de couleurs d'un espace de couleurs nommé en numéros d'index dans un profil de couleur
CMConvertIndexToColorName	Transforme les index d'un espace de couleurs en tableau de noms dans un espace de couleurs nommé.

Fonction ou structure	Description
CMCreateDeviceLinkProfile	Crée un profil de liaison d'appareil au format spécifié par l'International Color Consortium dans sa spécification de format de profil ICC.
CMCreateMultiProfileTransform	Accepte un tableau de profils ou un profil de lien d'appareil unique et crée une transformation de couleur. Cette transformation est un mappage de l'espace de couleurs spécifié par le premier profil à celui du deuxième profil et ainsi de suite au dernier.
CMCreateProfile	Crée un profil de couleur d'affichage à partir d'une structure LOGCOLORSPACEA .
CMCreateProfileW	Crée un profil de couleur d'affichage à partir d'une structure LOGCOLORSPACEW .
CMCreateTransform	Action déconseillée. Il n'existe pas d'API de remplacement, car celle-ci n'était plus utilisée. Les développeurs d'autres modules CMM ne sont pas obligés de l'implémenter.
CMCreateTransformExt	Crée une transformation de couleur qui mappe d'une entrée LOGCOLORSPACEA à un espace cible facultatif, puis à un appareil de sortie, à l'aide d'un ensemble d'indicateurs qui définissent la façon dont la transformation doit être créée.
CMCreateTransformExtW	Crée une transformation de couleur qui mappe d'une entrée LOGCOLORSPACEW à un espace cible facultatif, puis à un appareil de sortie, à l'aide d'un ensemble d'indicateurs qui définissent la façon dont la transformation doit être créée.
CMCreateTransformW	Action déconseillée. Il n'existe pas d'API de remplacement, car celle-ci n'était plus utilisée. Les développeurs d'autres modules CMM ne sont pas obligés de l'implémenter.
CMDeleteTransform	Supprime une transformation de couleur spécifiée et libère toute la mémoire qui lui est associée.
CMGetInfo	Récupère diverses informations sur le module de gestion des couleurs (CMM).
CMGetNamedProfileInfo	Récupère des informations sur le profil de couleur nommé spécifié.
CMGetPS2ColorRenderingDictionary	Obtient un dictionnaire de rendu des couleurs PostScript.

Fonction ou structure	Description
CMGetPS2ColorRenderingIntent	Récupère l'intention de rendu de couleur PostScript Niveau 2 à partir d'un profil.
CMGetPS2ColorSpaceArray	Obtient un tableau d'espaces de couleurs PostScript.
CMIsProfileValid	Indique si le profil donné est un profil ICC valide qui peut être utilisé pour la gestion des couleurs.
CMTranslateColors	Traduit un tableau de couleurs d'un espace de couleurs source en espace de couleur de destination à l'aide d'une transformation de couleur.
CMTranslateRGB	Convertit un RGBQuad fourni par l'application en espace de couleurs de l'appareil.
CMTranslateRGBs	Traduit une bitmap d'un espace de couleurs à un autre à l'aide d'une transformation de couleur.
CMTranslateRGBsExt	Convertit une bitmap d'un format défini dans un autre format défini et appelle régulièrement une fonction de rappel, le cas échéant, pour signaler la progression et permettre à l'application appelante d'arrêter la traduction.
ColorCorrectPalette	Corrige les entrées d'une palette pour un contexte d'appareil.
ColorMatchToTarget	Effectue le mappage des couleurs à des fins d'aperçu.
ConvertColorNameToIndex	Convertit les noms de couleurs d'un espace de couleurs nommé en numéros d'index dans un profil de couleur ICC (International Color Consortium).
ConvertIndexToColorName	Transforme les index d'un espace de couleurs en tableau de noms dans un espace de couleurs nommé.
CreateColorSpace	Crée un espace de couleurs.
CreateColorTransformW	Transforme les index d'un espace de couleurs en tableau de noms dans un espace de couleurs nommé.
CreateColorTransformW	Transforme les index d'un espace de couleurs en tableau de noms dans un espace de couleurs nommé.
CreateMultiProfileTransform	Accepte un tableau de profils ou un profil de lien d'appareil unique et crée une transformation de couleur que les applications peuvent utiliser pour effectuer le mappage des couleurs.

Fonction ou structure	Description
[**CreateProfileFromLogColorSpaceW**] ((/windows/win32/api/icm/nf-icm-createprofilefromlogcolorspacew)	Convertit un espace de couleur logique en profil d'appareil.
DeleteColorSpace	Supprime un espace de couleurs.
DeleteColorTransform	Supprime une transformation de couleur donnée.
DisassociateColorProfileFromDeviceW	Dissocie un profil de couleur spécifié avec un appareil spécifié sur un ordinateur spécifié.
EnumColorProfilesW	Énumère tous les profils qui satisfont aux critères d'énumération donnés.
EnumICMProfiles	Énumère les profils de couleur de sortie disponibles pour un contexte d'appareil donné.
EnumICMProfilesProcCallback	Fonction de rappel définie par l'application pour EnumICMProfiles.
GetCMMInfo	Récupère diverses informations sur le module de gestion des couleurs (CMM) qui a créé la transformation de couleur spécifiée.
GetColorDirectoryW	Récupère le chemin d'accès du répertoire Windows COLOR sur un ordinateur spécifié.
GetColorProfileElement	Copie les données d'un élément de profil étiqueté spécifié d'un profil de couleur spécifié dans une mémoire tampon.
GetColorProfileElementTag	Récupère le nom de balise spécifié par *dwIndex* dans la table de balises d'un profil de couleur ICC (International Color Consortium) donné, où *dwIndex* est un index de base unique dans cette table.
GetColorProfileFromHandle	Récupère le contenu du profil de couleur en fonction d'un handle pour un profil de couleur ouvert.
GetColorProfileHeader	Récupère ou dérive la structure d'en-tête ICC à partir d'un profil de couleur ICC ou d'un profil XML WCS. Les pilotes et les applications doivent supposer que le retour **de TRUE** indique uniquement qu'un en-tête correctement structuré est retourné. Chaque balise doit toujours être validée indépendamment à l'aide d'API ICM2 héritées ou d'API de schéma XML.
GetColorSpace	Obtient l'espace de couleur d'entrée actuel dans un contexte d'appareil.

Fonction ou structure	Description
GetCountColorProfileElements	Récupère le nombre d'éléments étiquetés dans un profil de couleur donné.
GetDeviceGammaRamp	Obtient la rampe gamma à partir de tableaux d'affichage couleur directs.
GetICMProfile	Obtient le profil de couleur de sortie actuel d'un contexte d'appareil.
GetLogColorSpace	Obtient la structure LOGCOLORSPACE d'un contexte d'appareil.
GetNamedProfileInfo	Récupère des informations sur le profil de couleur nommé ICC (International Color Consortium) spécifié dans le premier paramètre.
GetPS2ColorRenderingDictionary	Récupère le dictionnaire de rendu de couleur PostScript Niveau 2 à partir du profil de couleur ICC spécifié.
GetPS2ColorRenderingIntent	Récupère l'intention de rendu des couleurs PostScript Niveau 2 à partir d'un profil de couleur ICC.
GetPS2ColorSpaceArray	Récupère le tableau d'espaces de couleurs PostScript Niveau 2 à partir d'un profil de couleur ICC.
GetStandardColorSpaceProfileW	Récupère le profil de couleur inscrit pour l'espace de couleurs standard spécifié.
ICMProgressProcCallback	Rappel fourni par l'application pour signaler la progression.
InstallColorProfileW	Installe un profil donné pour une utilisation sur un ordinateur spécifié. Le profil est également copié dans le répertoire COLOR.
IsColorProfileTagPresent	Indique si une balise ICC (International Color Consortium) spécifiée est présente dans le profil de couleur spécifié.
IsColorProfileValid	Vous permet de déterminer si le profil spécifié est un profil ICC (International Color Consortium) valide ou un handle de profil Windows Color System (WCS) valide qui peut être utilisé pour la gestion des couleurs.
OpenColorProfileW	Crée un handle dans un profil de couleur spécifié. Le handle peut ensuite être utilisé dans d'autres fonctions de gestion de profil.

Fonction ou structure	Description
RegisterCMMW	Associe une valeur d'identification spécifiée à la bibliothèque de liens dynamiques du module de gestion des couleurs (DLL CMM) spécifiée. Lorsque cet ID apparaît dans un profil de couleur, Windows peut ensuite localiser la MMT correspondante afin de créer une transformation.
SélectionnerCMM	Vous permet de sélectionner le module de gestion des couleurs (CMM) préféré à utiliser.
SetColorProfileElement	Définit les données d'élément d'un élément de profil balisé dans un profil de couleur ICC.
SetColorProfileElementReference	Crée dans un profil de couleur ICC spécifié une nouvelle balise qui référence les mêmes données qu'une balise existante.
SetColorProfileElementSize	Définit la taille d'un élément étiqueté dans un profil de couleur ICC.
SetColorProfileHeader	Définit les données d'en-tête dans un profil de couleur ICC spécifié.
SetColorSpace	Définit l'espace de couleur d'entrée d'un contexte d'appareil.
SetDeviceGammaRamp	Définit la rampe gamma sur les panneaux d'affichage en couleur directe.
SetICMMode	Active ou désactive la gestion des couleurs dans un contexte d'appareil.
SetICMProfile	Définit le profil de couleur de sortie pour un contexte d'appareil donné.
SetStandardColorSpaceProfileW	Inscrit un profil spécifié pour un espace de couleurs standard donné. Le profil peut être interrogé à l'aide de GetStandardColorSpaceProfileW.
SetupColorMatchingW	Fournit un contrôle utilisateur sur la gestion des couleurs au moyen d'une boîte de dialogue.
TranslateBitmapBits	Convertit les couleurs bitmap à l'aide d'une transformation de couleur.
TranslateColors	Traduit un tableau de couleurs de l'espace de couleur source en espace de couleur de destination tel que défini par une transformation de couleur.

Fonction ou structure	Description
UninstallColorProfileW	Supprime un profil de couleur spécifié d'un ordinateur spécifié. Les fichiers associés sont éventuellement supprimés du système.
DésinscrireCMMW	Dissocie une valeur d'ID spécifiée d'une bibliothèque de liens dynamiques (DLL CMM) de module de gestion des couleurs donnée.
WcsAssociateColorProfileWithDevice	Associe un profil de couleur WCS spécifié à un appareil spécifié.
WcsCheckColors	Détermine si les couleurs d'un tableau se trouvent dans la gamme de sortie d'une transformation de couleur WCS spécifiée.
WcsCreateIccProfile	Convertit un profil WCS en profil ICC.
WcsDisassociateColorProfileFromDevice	Dissocie un profil de couleur WCS spécifié avec un appareil spécifié sur un ordinateur spécifié.
WcsEnumColorProfiles	Énumère tous les profils de couleur qui répondent aux critères d'énumération dans l'étendue de gestion des profils spécifiée.
WcsEnumColorProfilesSize	Retourne la taille, en octets, de la mémoire tampon requise par la fonction WcsEnumColorProfiles pour énumérer les profils de couleur.
WcsGetCalibrationManagementState	Détermine si la gestion système de l'état d'étalonnage de l'affichage est activée.
WcsGetDefaultColorProfile	Récupère le profil de couleur par défaut d'un appareil, ou le profil par défaut indépendant de l'appareil si l'appareil n'est pas spécifié.
WcsGetDefaultColorProfileSize	Retourne la taille, en octets, du nom de profil de couleur par défaut d'un appareil, y compris la marque de fin **NULL** .
WcsGetDefaultRenderingIntent	Retourne l'intention de rendu à l'échelle de l'utilisateur ou du système.
WcsGetUsePerUserProfiles	Détermine si l'utilisateur a choisi d'utiliser une liste d'association de profil par utilisateur pour l'appareil spécifié.
WcsOpenColorProfileW	Crée un handle pour un profil de couleur spécifié.
WcsSetCalibrationManagementState	Active ou désactive la gestion système de l'état d'étalonnage de l'affichage.

Fonction ou structure	Description
WcsSetDefaultColorProfile	Définit le nom de profil de couleur par défaut du type de profil spécifié dans l'étendue de gestion des profils spécifiée.
WcsSetDefaultRenderingIntent	Définit l'intention de rendu à l'échelle de l'utilisateur ou du système.
WcsSetUsePerUserProfiles	Permet à l'utilisateur de spécifier s'il doit utiliser ou non une liste d'association de profil par utilisateur pour l'appareil spécifié.
WcsTranslateColors	Convertit un tableau de couleurs de l'espace colorimétrique source vers l'espace de couleur de destination tel que défini par une transformation de couleur.

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

Fonction de rappel PBMCALLBACKFN (icm.h)

Article26/02/2024

TBD

Syntaxe

```
C++

PBMCALLBACKFN Pbmcallbackfn;

BOOL Pbmcallbackfn(
  ULONG unnamedParam1,
  ULONG unnamedParam2,
  LPARAM unnamedParam3
)
{...}
```

Paramètres

`unnamedParam1`

TBD

`unnamedParam2`

TBD

`unnamedParam3`

TBD

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 10 Build 20348
Serveur minimal pris en charge	Windows 10 Build 20348

Condition requise	Valeur
En-tête	icm.h

Commentaires

Cette page a-t-elle été utile ? Yes No

Indiquer des commentaires sur le produit | Obtenir de l'aide sur Microsoft Q&A

Fonction de rappel PCMSCALLBACKA (icm.h)

Article24/08/2023

PCMSCALLBACKA (ou **ApplyCallbackFunction**) est une fonction de rappel que vous implémentez qui met à jour les données de configuration WCS pendant l'exécution de la boîte de dialogue affichée par la fonction **SetupColorMatchingW** . Le nom **ApplyCallbackFunction** est un espace réservé. Le nom réel de cette fonction de rappel est fourni par votre application à l'aide d'ICM.

Syntaxe

```
C++

PCMSCALLBACKA Pcmscallbacka;

BOOL Pcmscallbacka(
  _tagCOLORMATCHSETUPA *unnamedParam1,
  LPARAM unnamedParam2
)
{...}
```

Paramètres

`unnamedParam1`

Pointeur vers une structure **COLORMATCHSETUPW** qui contient des données de configuration WCS.

`unnamedParam2`

Contient une valeur fournie par l'application.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. La fonction de rappel peut définir les informations d'erreur étendues en appelant SetLastError.

Notes

La fonction **ApplyCallbackFunction** permet de modifier la configuration WCS d'un appareil pendant que la boîte de dialogue Gestion des couleurs s'affiche. La boîte de dialogue Gestion des couleurs est affichée par la fonction **SetupColorMatchingW** .

Si la fonction de rappel est fournie, un bouton **Appliquer** s'affiche dans le coin inférieur droit de la boîte de dialogue. Lorsque vous sélectionnez le bouton **Appliquer** , la fonction de rappel met immédiatement à jour la configuration de l'appareil en cours de configuration. La boîte de dialogue Gestion des couleurs reste à l'écran.

Une application fournit une fonction de rappel à WCS en stockant l'adresse de la fonction de rappel dans la structure **COLORMATCHSETUPW** qui est passée à la fonction **SetupColorMatchingW** . L'adresse est stockée dans le membre **lPfnApplyCallback** de la structure **COLORMATCHSETUP** . Le membre **dwFlags** doit être défini sur CMS_USEAPPLYCALLBACK, sinon la fonction de rappel est ignorée.

Une valeur fournie par l'application peut être passée à la fonction de rappel. Avant d'appeler la fonction **SetupColorMatchingW** , l'application peut stocker une valeur dans le membre **lParamApplyCallback** de la structure **COLORMATCHSETUPW** . Lorsque la fonction de rappel est appelée, la valeur dans le membre de structure **lParamApplyCallback** est passée à la fonction de rappel dans son paramètre *lParam* .

La fonction de rappel est entièrement facultative. S'il n'est pas fourni, le bouton **Appliquer** n'apparaît pas dans la boîte de dialogue Gestion des couleurs. Microsoft recommande vivement que votre application fournisse une fonction de rappel.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

- SetupColorMatchingW
- COLORMATCHSETUPW

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

Fonction de rappel PCMSCALLBACKW (icm.h)

Article24/08/2023

PCMSCALLBACKW (ou **ApplyCallbackFunction**) est une fonction de rappel que vous implémentez qui met à jour les données de configuration WCS pendant l'exécution de la boîte de dialogue affichée par la fonction **SetupColorMatchingW** . Le nom **ApplyCallbackFunction** est un espace réservé. Le nom réel de cette fonction de rappel est fourni par votre application à l'aide d'ICM.

Syntaxe

```
C++

PCMSCALLBACKW Pcmscallbackw;

BOOL Pcmscallbackw(
  _tagCOLORMATCHSETUPW *unnamedParam1,
  LPARAM unnamedParam2
)
{...}
```

Paramètres

`unnamedParam1`

Pointeur vers une structure **COLORMATCHSETUPW** qui contient des données de configuration WCS.

`unnamedParam2`

Contient une valeur fournie par l'application.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. La fonction de rappel peut définir les informations d'erreur étendues en appelant SetLastError.

Notes

La fonction **ApplyCallbackFunction** permet de modifier la configuration WCS d'un appareil pendant que la boîte de dialogue Gestion des couleurs s'affiche. La boîte de dialogue Gestion des couleurs est affichée par la fonction **SetupColorMatchingW** .

Si la fonction de rappel est fournie, un bouton **Appliquer** s'affiche dans le coin inférieur droit de la boîte de dialogue. Lorsque vous sélectionnez le bouton **Appliquer** , la fonction de rappel met immédiatement à jour la configuration de l'appareil en cours de configuration. La boîte de dialogue Gestion des couleurs reste à l'écran.

Une application fournit une fonction de rappel à WCS en stockant l'adresse de la fonction de rappel dans la structure **COLORMATCHSETUPW** qui est passée à la fonction **SetupColorMatchingW** . L'adresse est stockée dans le membre **lPfnApplyCallback** de la structure **COLORMATCHSETUP** . Le membre **dwFlags** doit être défini sur CMS_USEAPPLYCALLBACK, sinon la fonction de rappel est ignorée.

Une valeur fournie par l'application peut être passée à la fonction de rappel. Avant d'appeler la fonction **SetupColorMatchingW** , l'application peut stocker une valeur dans le membre **lParamApplyCallback** de la structure **COLORMATCHSETUPW** . Lorsque la fonction de rappel est appelée, la valeur dans le membre de structure **lParamApplyCallback** est passée à la fonction de rappel dans son paramètre *lParam* .

La fonction de rappel est entièrement facultative. S'il n'est pas fourni, le bouton **Appliquer** n'apparaît pas dans la boîte de dialogue Gestion des couleurs. Microsoft recommande vivement que votre application fournisse une fonction de rappel.

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

- SetupColorMatchingW
- COLORMATCHSETUPW

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

Fonction AssociateColorProfileWithDeviceA (icm.h)

Article24/08/2023

Associe un profil de couleur spécifié à un appareil spécifié.

> ⓘ **Notes**
>
> Cette API ne prend pas en charge les profils de « couleur avancée » pour les moniteurs HDR. Utilisez **ColorProfileAddDisplayAssociation** pour gérer les profils de couleurs avancés.

Syntaxe

```
C++

BOOL AssociateColorProfileWithDeviceA(
  PCSTR pMachineName,
  PCSTR pProfileName,
  PCSTR pDeviceName
);
```

Paramètres

`pMachineName`

Réservé. Doit avoir la **valeur NULL**. Ce paramètre est destiné à pointer vers le nom de l'ordinateur sur lequel associer le profil et l'appareil spécifiés. Un pointeur **NULL** indique l'ordinateur local.

`pProfileName`

Pointe vers le nom de fichier du profil à associer.

`pDeviceName`

Pointe vers le nom de l'appareil à associer.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, appelez GetLastError.

Notes

La fonction **AssociateColorProfileWithDevice** échoue si le profil n'a pas été installé sur l'ordinateur à l'aide de la fonction **InstallColorProfileW** .

Notez que sous Windows (Windows 95 ou version ultérieure), le pilote de périphérique PostScript pour les imprimantes suppose un modèle couleur CMJN. Par conséquent, toutes les imprimantes PostScript doivent utiliser un profil de couleur CMJN. Windows 2000 n'a pas cette limitation.

Si l'appareil spécifié est un moniteur, cette fonction met à jour le profil par défaut.

Plusieurs profils sont généralement associés aux imprimantes, en fonction des types de papier et d'encre. Il n'y a pas de valeur par défaut. GDI sélectionne le meilleur à partir des profils associés lorsque votre application crée un contexte d'appareil (DC).

Les scanneurs n'ont pas non plus de profil par défaut. Toutefois, il est inhabituel d'associer plusieurs profils à un scanneur.

AssociateColorProfileWithDevice ajoute toujours le profil spécifié à la liste d'association de profils par utilisateur de l'utilisateur actuel pour l'appareil spécifié. Avant d'ajouter le profil à la liste, **AssociateColorProfileWithDevice** détermine si l'utilisateur a déjà exprimé le désir d'utiliser une liste d'associations de profils par utilisateur pour l'appareil. Si c'est le cas, **AssociateColorProfileWithDevice** ajoute simplement le profil spécifié à la liste d'associations de profils par utilisateur existante pour l'appareil. Si ce n'est pas le cas, **AssociateColorProfileWithDevice** crée une liste d'associations de profils par utilisateur pour l'appareil en copiant la liste d'associations à l'échelle du système pour cet appareil. Il ajoute ensuite le profil spécifié à la liste par utilisateur. À partir de ce point, l'utilisateur actuel utilise une liste d'association de profils par utilisateur pour l'appareil spécifié, comme si **WcsSetUsePerUserProfiles** avait été appelé pour *pDevice* avec le paramètre *usePerUserProfiles* défini sur **TRUE**.

Spécifications

Client minimal pris en charge	Windows 10 Build 20348
Serveur minimal pris en charge	Windows 10 Build 20348
En-tête	icm.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- DisassociateColorProfileFromDevice

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

AssociateColorProfileWithDeviceW, fonction (icm.h)

Article24/08/2023

Associe un profil de couleurs spécifié à un appareil spécifié.

> ⓘ **Notes**
>
> Cette API ne prend pas en charge les profils de couleur avancés pour les moniteurs HDR. Utilisez **ColorProfileAddDisplayAssociation** pour gérer les profils de couleurs avancés.

Syntaxe

C++

```
BOOL AssociateColorProfileWithDeviceW(
  PCWSTR pMachineName,
  PCWSTR pProfileName,
  PCWSTR pDeviceName
);
```

Paramètres

`pMachineName`

Réservé. Doit être **NULL**. Ce paramètre est destiné à pointer vers le nom de l'ordinateur sur lequel associer le profil et l'appareil spécifiés. Un pointeur **NULL** indique l'ordinateur local.

`pProfileName`

Pointe vers le nom de fichier du profil à associer.

`pDeviceName`

Pointe vers le nom de l'appareil à associer.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, appelez GetLastError.

Notes

La fonction **AssociateColorProfileWithDevice** échoue si le profil n'a pas été installé sur l'ordinateur à l'aide de la fonction **InstallColorProfileW** .

Notez que sous Windows (Windows 95 ou version ultérieure), le pilote de périphérique PostScript pour les imprimantes suppose un modèle de couleur CMJN. Par conséquent, toutes les imprimantes PostScript doivent utiliser un profil de couleur CMJN. Windows 2000 n'a pas cette limitation.

Si l'appareil spécifié est un moniteur, cette fonction met à jour le profil par défaut.

Plusieurs profils sont généralement associés aux imprimantes, en fonction des types de papier et d'encre. Il n'y a pas de valeur par défaut. Le GDI sélectionne le meilleur parmi les profils associés lorsque votre application crée un contexte d'appareil (DC).

Les scanneurs n'ont pas non plus de profil par défaut. Toutefois, il est atypique d'associer plusieurs profils à un scanneur.

AssociateColorProfileWithDevice ajoute toujours le profil spécifié à la liste d'association de profils par utilisateur de l'utilisateur actuel pour l'appareil spécifié. Avant d'ajouter le profil à la liste, **AssociateColorProfileWithDevice** détermine si l'utilisateur a précédemment exprimé le désir d'utiliser une liste d'association de profil par utilisateur pour l'appareil. Dans ce cas, **AssociateColorProfileWithDevice** ajoute simplement le profil spécifié à la liste d'association de profil par utilisateur existante pour l'appareil. Si ce n'est pas le cas, **AssociateColorProfileWithDevice** crée une liste d'association de profils par utilisateur pour l'appareil en copiant la liste d'association à l'échelle du système pour cet appareil. Il ajoute ensuite le profil spécifié à la liste par utilisateur. À partir de ce point, l'utilisateur actuel utilise une liste d'association de profil par utilisateur pour l'appareil spécifié, comme si **WcsSetUsePerUserProfiles** avait été appelé pour *pDevice* avec le paramètre *usePerUserProfiles* défini sur **TRUE**.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau

	uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- DisassociateColorProfileFromDeviceW

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

CheckBitmapBitBits, fonction (icm.h)

Article24/08/2023

Vérifie si les pixels d'un bitmap spécifié se trouvent dans la gamut de sortie d'une transformation spécifiée.

Syntaxe

```
C++

BOOL CheckBitmapBits(
  HTRANSFORM    hColorTransform,
  PVOID         pSrcBits,
  BMFORMAT      bmInput,
  DWORD         dwWidth,
  DWORD         dwHeight,
  DWORD         dwStride,
  PBYTE         paResult,
  PBMCALLBACKFN pfnCallback,
  LPARAM        lpCallbackData
);
```

Paramètres

`hColorTransform`

Gérez la transformation de couleur à utiliser.

`pSrcBits`

Pointeur vers la bitmap vers case activée par rapport au gamut de sortie.

`bmInput`

Spécifie le format de l'image bitmap. Doit être défini sur l'une des valeurs du type énuméré **BMFORMAT** .

`dwWidth`

Spécifie le nombre de pixels par ligne d'analyse de la bitmap.

`dwHeight`

Spécifie le nombre de lignes d'analyse de l'image bitmap.

`dwStride`

Spécifie le nombre d'octets entre la ligne d'analyse de début et le début de la suivante. Si la valeur est égale à zéro, les lignes d'analyse bitmap sont supposées être rembourrées afin d'être alignées sur **DWORD**.

`paResult`

Pointeur vers un tableau d'octets où les résultats du test doivent être placés. Cette mémoire tampon de résultats doit contenir au moins autant d'octets que de pixels dans la bitmap.

`pfnCallback`

Pointeur vers une fonction de rappel appelée régulièrement par **CheckBitmapBitBits** pour signaler la progression et permettre au processus d'appel d'annuler le test bitmap. (Voir **ICMProgressProcCallback**).

`lpCallbackData`

Données transmises à la fonction de rappel, par exemple, pour identifier le test bitmap sur lequel la progression est signalée.

Valeur retournée

Si cette fonction réussit, la valeur de retour est une valeur différente de zéro.

Si cette fonction échoue, la valeur de retour est zéro. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Notes

Si le format d'entrée n'est pas compatible avec la transformation de couleur, la fonction **CheckBitmapBitBits** échoue.

Cette fonction place les résultats des tests dans la mémoire tampon pointée par *paResult*. Chaque octet de la mémoire tampon correspond à un pixel dans la bitmap et a une valeur non signée comprise entre 0 et 255. La valeur 0 indique que la couleur est en gamut, tandis qu'une valeur différente de zéro indique qu'elle est hors gamut. Pour tout entier n tel que $0 < n < 255$, une valeur de résultat de $n + 1$ indique que la couleur correspondante est au moins aussi éloignée de la gamut que serait indiquée par une valeur de résultat de n.

Lorsque l'un des BMFORMAT à virgule flottante, BM_32b_scARGB ou BM_32b_scRGB est utilisé, les données de couleur vérifiées ne doivent pas contenir de NaN ou d'infini. NaN et infinity ne sont pas considérés comme représentant des valeurs de composants de couleur légitimes, et le résultat de la vérification des pixels contenant naN ou infini n'a aucun sens en termes de couleur. Les valeurs NaN ou infini dans les données de couleur en cours de traitement seront gérées en mode silencieux et aucune erreur ne sera retournée.

Les informations hors gamut dans les étiquettes gamut créées dans WCS utilisent la distance de couleur perceptive dans CIECAM02, qui est la racine carrée moyenne dans CIECAM02 espace Jab. La distance dans les balises de gamut de profil ICC héritées est la racine carrée moyenne dans l'espace CIELAB. Nous vous recommandons d'utiliser l'espace CIECAM02 lorsqu'il est disponible, car il fournit des métriques de distance plus précises.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- ICMProgressProcCallback
- BMFORMAT

Commentaires

Cette page a-t-elle été utile ?

Yes No

Obtenir de l’aide sur Microsoft Q&A

CheckColors, fonction (icm.h)

Article29/02/2024

Détermine si les couleurs d'un tableau se trouvent dans la gamme de sortie d'une transformation spécifiée.

Syntaxe

```
C++

BOOL CheckColors(
  HTRANSFORM hColorTransform,
  PCOLOR     paInputColors,
  DWORD      nColors,
  COLORTYPE  ctInput,
  PBYTE      paResult
);
```

Paramètres

`hColorTransform`

Gérez la transformation de couleur à utiliser.

`paInputColors`

Pointeur vers un tableau de structures *nColors***COLOR** à traduire.

`nColors`

Contient le nombre d'éléments dans les tableaux pointés par *paInputColors* et *paResult*.

`ctInput`

Spécifie le type de couleur d'entrée.

`paResult`

Pointeur vers un tableau *d'octets nColors* qui reçoit les résultats du test.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Remarques

Si le type de couleur d'entrée n'est pas compatible avec la transformation de couleur, **CheckColors** échoue.

La fonction place les résultats des tests dans le tableau pointé vers *paResult*. Chaque octet du tableau correspond à un élément **COLOR** dans le tableau pointé par *paInputColors* et a une valeur non signée comprise entre 0 et 255. La valeur 0 indique que la couleur est en gamut, tandis qu'une valeur différente de zéro indique qu'elle est hors gamut. Pour n'importe quel entier *n* tel que 0 <*n*< 255, une valeur de résultat de *n* +1 indique que la couleur correspondante est au moins aussi éloignée de la gamut que l'indique une valeur de résultat de *n*.

Les informations hors gamut dans les étiquettes gamut créées dans WCS utilisent la distance de couleur perceptive dans CIECAM02, qui est la racine carrée moyenne dans CIECAM02 espace Jab. La distance dans les balises de gamut de profil ICC héritées est la racine carrée moyenne dans l'espace CIELAB. Nous vous recommandons d'utiliser l'espace CIECAM02 lorsqu'il est disponible, car il fournit des métriques de distance plus précises.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- Structure COLOR

Commentaires

Cette page a-t-elle été utile ? Yes No

Indiquer des commentaires sur le produit | Obtenir de l'aide sur Microsoft Q&A

CheckColorsInGamut, fonction (wingdi.h)

Article27/08/2023

La fonction **CheckColorsInGamut** détermine si un ensemble spécifié de triples RVB se trouve dans la gamme de sortie d'un appareil spécifié. Les triples RVB sont interprétés dans l'espace de couleurs logique d'entrée.

Syntaxe

```
C++

BOOL CheckColorsInGamut(
  HDC         hdc,
  LPRGBTRIPLE lpRGBTriple,
  LPVOID      dlpBuffer,
  DWORD       nCount
);
```

Paramètres

`hdc`

Gérez le contexte de l'appareil dont la gamme de sortie doit être vérifiée.

`lpRGBTriple`

Pointeur vers un tableau de triples RVB vers case activée.

`dlpBuffer`

Pointeur vers la mémoire tampon dans laquelle les résultats doivent être placés. Cette mémoire tampon doit être au moins égale à *nCount* octets.

`nCount`

Nombre d'éléments dans le tableau de triples.

Valeur retournée

Si cette fonction réussit, la valeur de retour est une valeur différente de zéro.

Si cette fonction échoue, la valeur de retour est zéro.

Remarques

La fonction place les résultats du test dans la mémoire tampon pointée vers *lpBuffer*. Chaque octet de la mémoire tampon correspond à un *triple RVB* et a une valeur non signée comprise entre CM_IN_GAMUT (= 0) et CM_OUT_OF_GAMUT (= 255). La valeur 0 indique que la couleur est en gamut, tandis qu'une valeur différente de zéro indique qu'elle est hors gamut. Pour n'importe quel entier *n* tel que 0 <*n*< 255, une valeur de résultat *de n* + 1 indique que la couleur correspondante est au moins aussi éloignée de la gamme que l'indique une valeur de résultat *de n*, comme spécifié par la spécification du format de profil ICC. Pour plus d'informations sur la spécification du format de profil ICC, consultez les sources répertoriées dans Informations supplémentaires .

Notez que pour que cette fonction réussisse, WCS doit être activé pour le handle de contexte d'appareil transmis via le paramètre *hDC* . WCS peut être activé pour un handle de contexte d'appareil en appelant la fonction SetICMMode .

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
Plateforme cible	Windows
En-tête	wingdi.h
Bibliothèque	Gdi32.lib
DLL	Gdi32.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- SetICMMode

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l’aide sur Microsoft Q&A

Fonction CloseColorProfile (icm.h)

Article24/08/2023

Ferme un handle de profil ouvert.

Syntaxe

```
C++

BOOL CloseColorProfile(
  HPROFILE hProfile
);
```

Paramètres

`hProfile`

Gérez le profil à fermer. La fonction détermine si le fichier HPROFILE contient des informations de profil ICC ou WCS.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, appelez GetLastError.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib

DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

CMCheckColors, fonction (icm.h)

Article24/08/2023

Détermine si les couleurs spécifiées se trouvent dans la gamme de sortie d'une transformation spécifiée.

Syntaxe

```
C++

BOOL CMCheckColors(
  HCMTRANSFORM hcmTransform,
  LPCOLOR      lpaInputColors,
  DWORD        nColors,
  COLORTYPE    ctInput,
  LPBYTE       lpaResult
);
```

Paramètres

`hcmTransform`

Descripteur en transformation de couleur à utiliser.

`lpaInputColors`

Pointeur vers un tableau de structures **COLOR** pour case activée par rapport à la gamme de sortie.

`nColors`

Spécifie le nombre d'éléments du tableau.

`ctInput`

Spécifie le type de couleur d'entrée.

`lpaResult`

Pointeur vers une mémoire tampon dans laquelle placer un tableau d'octets contenant les résultats des tests. Chaque octet de la mémoire tampon correspond à une structure **COLOR** , et à la sortie a été défini sur une valeur non signée comprise entre 0 et 255. La valeur 0 indique que la couleur est en gamut, tandis qu'une valeur différente de zéro

indique qu'elle est hors gamut. Pour tout entier *n* tel que 0 <*n*< 255, une valeur de résultat de *n* + 1 indique que la couleur correspondante est au moins aussi éloignée de la gamme que l'indique une valeur de résultat de *n*. Ces valeurs sont généralement générées à partir du *gamutTag* dans le profil ICC.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Si la fonction échoue, la CMM doit appeler SetLastError pour définir la dernière erreur sur une valeur d'erreur valide définie dans Winerror.h.

Notes

Chaque CMM est nécessaire pour exporter cette fonction.

Si le type de couleur d'entrée n'est pas compatible avec la transformation de couleur **CMCheckColors** échoue.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

Cette page a-t-elle été utile ?

Yes No

Obtenir de l’aide sur Microsoft Q&A

Fonction CMCheckColorsInGamut (icm.h)

Article26/02/2024

[**CMCheckColorsInGamut** n'est plus disponible pour une utilisation à partir de Windows Vista.]

Détermine si les triples RVB spécifiés se trouvent dans la gamme de sortie d'une transformation spécifiée.

Syntaxe

```
C++

BOOL CMCheckColorsInGamut(
  HCMTRANSFORM hcmTransform,
  RGBTRIPLE    *lpaRGBTriple,
  LPBYTE       lpaResult,
  UINT         nCount
);
```

Paramètres

`hcmTransform`

Spécifie la transformation à utiliser.

`lpaRGBTriple`

Pointe vers un tableau de triples RVB pour case activée.

`lpaResult`

Pointe vers la mémoire tampon dans laquelle placer les résultats.

Les résultats sont représentés par un tableau d'octets. Chaque octet du tableau correspond à un triple RVB et a une valeur non signée comprise entre 0 et 255. La valeur 0 indique que la couleur est en gamut, tandis qu'une valeur différente de zéro indique qu'elle est hors gamut. Pour n'importe quel entier *n* de la plage 0 <*n*< 255, une valeur de résultat *de n* + 1 indique que la couleur correspondante est au moins aussi éloignée de la gamut que l'indique une valeur de résultat de *n*.

`nCount`

Spécifie le nombre d'éléments du tableau.

Valeur retournée

À compter de Windows Vista, la mmao par défaut (Icm32.dll) retourne **FALSE** et GetLastError signale ERROR_NOT_SUPPORTED.

Windows Server 2003, Windows XP et Windows 2000 :

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Appelez GetLastError pour récupérer l'erreur.

Remarques

À compter de Windows Vista, les implémenteurs CMM ne sont plus nécessaires pour implémenter cette méthode.

Windows Server 2003, Windows XP et Windows 2000 :

Les implémenteurs CMM sont nécessaires pour implémenter cette méthode.

Chaque CMM est nécessaire pour exporter cette fonction.

Si la fonction échoue, les machines virtuelles personnalisées doivent appeler SetLastError pour définir la dernière erreur sur une valeur d'erreur valide définie dans Winerror.h.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

Fonction CMCheckRGBs (icm.h)

Article24/08/2023

Vérifie les couleurs bitmap par rapport à un gamut de sortie.

Syntaxe

```
C++

BOOL CMCheckRGBs(
  HCMTRANSFORM  hcmTransform,
  LPVOID        lpSrcBits,
  BMFORMAT      bmInput,
  DWORD         dwWidth,
  DWORD         dwHeight,
  DWORD         dwStride,
  LPBYTE        lpaResult,
  PBMCALLBACKFN pfnCallback,
  LPARAM        ulCallbackData
);
```

Paramètres

`hcmTransform`

`lpSrcBits`

`bmInput`

`dwWidth`

`dwHeight`

`dwStride`

`lpaResult`

`pfnCallback`

`ulCallbackData`

Spécifications

Client minimal pris en charge	Windows 10 Build 20348
Serveur minimal pris en charge	Windows 10 Build 20348
En-tête	icm.h

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

CmConvertColorNameToIndex, fonction (icm.h)

Article29/02/2024

Convertit les noms de couleurs dans un espace de couleurs nommé en nombres d'index dans un profil de couleur.

Syntaxe

```
C++
```

```
BOOL CMConvertColorNameToIndex(
  HPROFILE    hProfile,
  PCOLOR_NAME paColorName,
  PDWORD      paIndex,
  DWORD       dwCount
);
```

Paramètres

`hProfile`

Handle d'un profil de couleur nommé.

`paColorName`

Pointeur vers un tableau de structures de noms de couleur.

`paIndex`

Pointeur vers un tableau de **DWORDS** que cette fonction remplit avec les index.

`dwCount`

Nombre de noms de couleurs à convertir.

Valeur retournée

Si cette fonction réussit avec la conversion, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Dans ce cas, la CMM doit appeler **SetLastError** pour définir la dernière erreur sur une valeur d'erreur valide définie dans Winerror.h.

Remarques

Cette fonction est requise dans la CMM par défaut. Elle est facultative pour toutes les autres machines virtuelles.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

CmConvertIndexToColorName, fonction (icm.h)

Article24/08/2023

Transforme les index d'un espace de couleurs en tableau de noms dans un espace de couleurs nommé.

Syntaxe

```
C++

BOOL CMConvertIndexToColorName(
  HPROFILE    hProfile,
  PDWORD      paIndex,
  PCOLOR_NAME paColorName,
  DWORD       dwCount
);
```

Paramètres

`hProfile`

Handle d'un profil d'espace de couleurs.

`paIndex`

Pointeur vers un tableau de numéros d'index d'espace de couleur.

`paColorName`

Pointeur vers un tableau de structures de noms de couleur.

`dwCount`

Nombre d'index à convertir.

Valeur retournée

Si cette fonction de conversion réussit, la valeur de retour est TRUE.

Si cette fonction échoue, la valeur de retour est FALSE. Dans ce cas, la CMM doit appeler **SetLastError** pour définir la dernière erreur sur une valeur d'erreur valide définie dans Winerror.h.

Notes

Cette fonction est requise dans la CMM par défaut. Elle est facultative pour toutes les autres machines virtuelles.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

CmCreateDeviceLinkProfile, fonction (icm.h)

Article24/08/2023

Crée un profil de liaison d'appareil au format spécifié par l'International Color Consortium dans sa spécification de format de profil ICC.

Syntaxe

```
C++

BOOL CMCreateDeviceLinkProfile(
  PHPROFILE pahProfiles,
  DWORD     nProfiles,
  PDWORD    padwIntents,
  DWORD     nIntents,
  DWORD     dwFlags,
  LPBYTE    *lpProfileData
);
```

Paramètres

`pahProfiles`

Pointeur vers un tableau de handles de profil.

`nProfiles`

Spécifie le nombre de profils dans le tableau.

`padwIntents`

Tableau d'intentions de rendu.

`nIntents`

Nombre d'éléments dans le tableau d'intentions.

`dwFlags`

Spécifie les indicateurs à utiliser pour la création de contrôle de la transformation. Pour plus d'informations, consultez Indicateurs de création de transformation CMM.

`lpProfileData`

Pointeur vers un pointeur vers une mémoire tampon. En cas de réussite, la fonction alloue et remplit cette mémoire tampon. L'application appelante doit libérer cette mémoire tampon quand elle n'est plus nécessaire. Utilisez la fonction **GlobalFree** pour libérer cette mémoire tampon.

Valeur retournée

Si la fonction réussit, la valeur de retour est une valeur différente de zéro.

Si cette fonction échoue, la valeur de retour est zéro. Si la fonction échoue, la CMM doit appeler **SetLastError** pour définir la dernière erreur sur une valeur d'erreur valide définie dans Winerror.h.

Notes

Seule la CMM windows par défaut est requise pour exporter cette fonction ; elle est facultative pour toutes les autres machines virtuelles.

Si une CMM ne prend pas en charge **CMCreateDeviceLinkProfile**, Windows utilise la CMM par défaut pour créer un profil de liaison d'appareil.

Le premier et le dernier profil du tableau doivent être des profils d'appareil. Les autres profils peuvent être des espaces de couleurs ou des profils abstraits. L'espace de couleur de sortie de chaque profil doit être l'espace de couleur d'entrée du profil suivant.

L'application appelante doit libérer la mémoire tampon allouée par cette fonction et pointée par le paramètre *lpProfileData* . Utilisez la fonction GlobalFree pour libérer la mémoire tampon.

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- GlobalFree

Commentaires

CmCreateMultiProfileTransform, fonction (icm.h)

Article24/08/2023

Accepte un tableau de profils ou un profil de liaison d'appareil unique et crée une transformation de couleur. Cette transformation est un mappage de l'espace de couleurs spécifié par le premier profil à celui du deuxième profil, et ainsi de suite, au dernier.

Syntaxe

```
C++

HCMTRANSFORM CMCreateMultiProfileTransform(
  PHPROFILE pahProfiles,
  DWORD     nProfiles,
  PDWORD    padwIntents,
  DWORD     nIntents,
  DWORD     dwFlags
);
```

Paramètres

`pahProfiles`

Pointe vers un tableau de handles de profil.

`nProfiles`

Spécifie le nombre de profils dans le tableau.

`padwIntents`

Pointe vers un tableau d'intentions de rendu. Chaque intention de rendu est représentée par l'une des valeurs suivantes :

INTENT_PERCEPTUAL
INTENT_SATURATION
INTENT_RELATIVE_COLORIMETRIC
INTENT_ABSOLUTE_COLORIMETRIC

Pour plus d'informations, consultez Intentions de rendu.

`nIntents`

Spécifie le nombre d'intentions dans le tableau d'intentions. Peut être 1 ou la même valeur que *nProfiles*.

`dwFlags`

Spécifie les indicateurs à utiliser pour la création de contrôle de la transformation. Pour plus d'informations, consultez Indicateurs de création de transformation CMM.

Valeur retournée

Si cette fonction réussit, la valeur de retour est une transformation de couleur dans la plage de 256 à 65 535. Étant donné que seul le **mot** faible de la transformation est conservé, les transformations valides ne peuvent pas dépasser cette plage.

Si cette fonction échoue, la valeur de retour est un code d'erreur dont la valeur est inférieure à 256. Lorsque la valeur de retour est inférieure à 256, signalant une erreur, la CMM doit utiliser **SetLastError** pour définir la dernière erreur sur une valeur d'erreur valide, comme défini dans Winerror.h.

Notes

Chaque CMM est nécessaire pour exporter cette fonction.

Le tableau d'intentions spécifie la façon dont les profils doivent être combinés. La *nième*intention est utilisée pour combiner le *nième*profil dans le tableau. Si une seule intention est spécifiée, elle est utilisée pour le premier profil et tous les autres profils sont combinés à l'aide de l'intention De correspondance.

Les descripteurs de profil utilisés pour créer la transformation de couleur peuvent être fermés une fois l'appel à **CMCreateMultiProfileTransform** terminé.

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]

En-tête	icm.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

Fonction CMCreateProfile (icm.h)

Article26/02/2024

[**CMCreateProfile** n'est plus disponible pour une utilisation à partir de Windows Vista.]

Crée un profil de couleur d'affichage à partir d'une structure **LOGCOLORSPACEA** .

Syntaxe

```
C++

BOOL CMCreateProfile(
  LPLOGCOLORSPACEA lpColorSpace,
  LPDEVCHARACTER   *lpProfileData
);
```

Paramètres

`lpColorSpace`

Pointeur vers un espace logique de couleur, dont le membre **lcsFilename** sera **NULL**.

`lpProfileData`

Pointeur vers un pointeur vers une mémoire tampon. En cas de réussite, la fonction alloue et remplit cette mémoire tampon. Il incombe à l'application appelante de libérer cette mémoire tampon lorsqu'elle n'est plus nécessaire.

Valeur retournée

À compter de Windows Vista, la mmao par défaut (Icm32.dll) retourne **FALSE** et GetLastError signale ERROR_NOT_SUPPORTED.

Windows Server 2003, Windows XP et Windows 2000 :

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Appelez GetLastError pour récupérer l'erreur.

Remarques

À compter de Windows Vista, les implémenteurs CMM ne sont plus nécessaires pour implémenter cette méthode.

Windows Server 2003, Windows XP et Windows 2000 :

La version Unicode de cette fonction est **CMCreateProfileW** .

Seule la gestion CMM par défaut de Windows est requise pour exporter cette fonction ; elle est facultative pour toutes les autres machines virtuelles.

Si un CMM ne prend pas en charge **CMCreateProfile**, Windows utilise la gestion CMM par défaut pour créer le profil.

Le CMM doit définir tous les champs d'en-tête sur des valeurs par défaut raisonnables. Ce profil doit être utilisable comme profil d'entrée dans une transformation.

L'application appelante doit libérer la mémoire tampon allouée par cette fonction et pointée vers le paramètre *lpProfileData* . Utilisez **GlobalFree** pour libérer la mémoire tampon.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- CMCreateProfileW

Commentaires

Fonction CMCreateProfileW (icm.h)

Article24/08/2023

[**CMCreateProfileW** n'est plus disponible à partir de Windows Vista.]

Crée un profil de couleur d'affichage à partir d'une structure **LOGCOLORSPACEW** .

Syntaxe

```
C++

BOOL CMCreateProfileW(
  LPLOGCOLORSPACEW lpColorSpace,
  LPDEVCHARACTER   *lpProfileData
);
```

Paramètres

`lpColorSpace`

Pointeur vers un espace logique de couleur, dont le membre **lcsFilename** sera **NULL**.

`lpProfileData`

Pointeur vers un pointeur vers une mémoire tampon. En cas de réussite, la fonction alloue et remplit cette mémoire tampon. Il incombe à l'application appelante de libérer cette mémoire tampon lorsqu'elle n'est plus nécessaire.

Valeur retournée

À compter de Windows Vista, la mmao par défaut (Icm32.dll) retourne **FALSE** et GetLastError signale ERROR_NOT_SUPPORTED.

Windows Server 2003, Windows XP et Windows 2000 :

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Appelez GetLastError pour récupérer l'erreur.

Notes

À compter de Windows Vista, les implémenteurs CMM ne sont plus nécessaires pour implémenter cette méthode.

Windows Server 2003, Windows XP et Windows 2000 :

La version Unicode de cette fonction est **CMCreateProfileW** .

Seule la gestion CMM par défaut de Windows est requise pour exporter cette fonction ; elle est facultative pour toutes les autres machines virtuelles.

Si une CMM ne prend pas en charge **CMCreateProfileW**, Windows utilise la gestion CMM par défaut pour créer le profil.

Le CMM doit définir tous les champs d'en-tête sur des valeurs par défaut raisonnables. Ce profil doit être utilisable comme profil d'entrée dans une transformation.

L'application appelante doit libérer la mémoire tampon allouée par cette fonction et pointée vers le paramètre *lpProfileData* . Utilisez **GlobalFree** pour libérer la mémoire tampon.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

CmCreateTransform, fonction (icm.h)

Article27/02/2024

Action déconseillée. Il n'existe pas d'API de remplacement, car celle-ci n'était plus utilisée. Les développeurs d'autres modules CMM ne sont pas obligés de l'implémenter.

Syntaxe

```
C++

HCMTRANSFORM CMCreateTransform(
  LPLOGCOLORSPACEA lpColorSpace,
  LPDEVCHARACTER   lpDevCharacter,
  LPDEVCHARACTER   lpTargetDevCharacter
);
```

Paramètres

`lpColorSpace`

`lpDevCharacter`

`lpTargetDevCharacter`

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 10 Build 20348
Serveur minimal pris en charge	Windows 10 Build 20348
En-tête	icm.h

Commentaires

Fonction CMCreateTransformExt (icm.h)

Article26/02/2024

Crée une transformation de couleur qui mappe d'une entrée **LOGCOLORSPACEA** à un espace cible facultatif, puis à un appareil de sortie, à l'aide d'un ensemble d'indicateurs qui définissent la façon dont la transformation doit être créée.

Syntaxe

```
C++

HCMTRANSFORM CMCreateTransformExt(
  LPLOGCOLORSPACEA lpColorSpace,
  LPDEVCHARACTER   lpDevCharacter,
  LPDEVCHARACTER   lpTargetDevCharacter,
  DWORD            dwFlags
);
```

Paramètres

`lpColorSpace`

Pointeur vers une structure d'espace de couleurs logique d'entrée.

`lpDevCharacter`

Pointeur vers un profil d'appareil mappé en mémoire.

`lpTargetDevCharacter`

Pointeur vers un profil cible mappé en mémoire.

`dwFlags`

Spécifie les indicateurs pour utiliser la création de contrôle de la transformation. Pour plus d'informations, consultez Indicateurs de création de transformation CMM.

Valeur retournée

Si cette fonction réussit, la valeur de retour est une transformation de couleur dans la plage 256 à 65 535. Étant donné que seul le **mot** faible de la transformation est conservé, les transformations valides ne peuvent pas dépasser cette plage.

Si cette fonction échoue, la valeur de retour est un code d'erreur dont la valeur est inférieure à 256. Lorsque la valeur de retour est inférieure à 256, signalant une erreur, le CMM doit utiliser **SetLastError** pour définir la dernière erreur sur une valeur d'erreur valide, comme défini dans Winerror.h.

Remarques

L'équivalent Unicode de **CMCreateTransformExt** est CMCreateTransformExtW.

Chaque CMM est nécessaire pour exporter cette fonction.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- CMCreateTransformExtW

Commentaires

Fonction CMCreateTransformExtW (icm.h)

Article24/08/2023

Crée une transformation de couleur qui mappe d'une entrée **LOGCOLORSPACEW** à un espace cible facultatif, puis à un appareil de sortie, à l'aide d'un ensemble d'indicateurs qui définissent la façon dont la transformation doit être créée.

Syntaxe

```
C++

HCMTRANSFORM CMCreateTransformExtW(
  LPLOGCOLORSPACEW lpColorSpace,
  LPDEVCHARACTER   lpDevCharacter,
  LPDEVCHARACTER   lpTargetDevCharacter,
  DWORD            dwFlags
);
```

Paramètres

`lpColorSpace`

Pointeur vers une structure d'espace de couleurs logique d'entrée.

`lpDevCharacter`

Pointeur vers un profil d'appareil mappé en mémoire.

`lpTargetDevCharacter`

Pointeur vers un profil cible mappé en mémoire.

`dwFlags`

Spécifie les indicateurs pour utiliser la création de contrôle de la transformation. Pour plus d'informations, consultez Indicateurs de création de transformation CMM.

Valeur retournée

Si cette fonction réussit, la valeur de retour est une transformation de couleur dans la plage 256 à 65 535. Étant donné que seul le **mot** faible de la transformation est conservé, les transformations valides ne peuvent pas dépasser cette plage.

Si cette fonction échoue, la valeur de retour est un code d'erreur dont la valeur est inférieure à 256. Lorsque la valeur de retour est inférieure à 256, signalant une erreur, le CMM doit utiliser **SetLastError** pour définir la dernière erreur sur une valeur d'erreur valide, comme défini dans Winerror.h.

Notes

Chaque CMM est nécessaire pour exporter cette fonction.

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- CMCreateTransformExtW

Commentaires

Fonction CMCreateTransformW (icm.h)

Article27/02/2024

Action déconseillée. Il n'existe pas d'API de remplacement, car celle-ci n'était plus utilisée. Les développeurs d'autres modules CMM ne sont pas obligés de l'implémenter.

Syntaxe

```
C++

HCMTRANSFORM CMCreateTransformW(
  LPLOGCOLORSPACEW lpColorSpace,
  LPDEVCHARACTER   lpDevCharacter,
  LPDEVCHARACTER   lpTargetDevCharacter
);
```

Paramètres

`lpColorSpace`

`lpDevCharacter`

`lpTargetDevCharacter`

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 10 Build 20348
Serveur minimal pris en charge	Windows 10 Build 20348
En-tête	icm.h

Commentaires

Cette page a-t-elle été utile ? Yes No

CmDeleteTransform, fonction (icm.h)

Article02/03/2024

Supprime une transformation de couleur spécifiée et libère toute la mémoire qui lui est associée.

Syntaxe

```
C++

BOOL CMDeleteTransform(
  HCMTRANSFORM hcmTransform
);
```

Paramètres

`hcmTransform`

Identifie la transformation de couleur à supprimer.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si la fonction échoue, la valeur de retour est **FALSE**. Si la fonction **CMDeleteTransform** échoue, la CMM doit appeler **SetLastError** pour définir la dernière erreur sur une valeur d'erreur valide définie dans Winerror.h.

Remarques

Chaque CMM est nécessaire pour exporter cette fonction.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

CmGetInfo, fonction (icm.h)

Article24/08/2023

Récupère diverses informations sur le module de gestion des couleurs (CMM).

Chaque CMM est nécessaire pour exporter cette fonction.

Syntaxe

```
C++

DWORD CMGetInfo(
  DWORD dwInfo
);
```

Paramètres

`dwInfo`

Spécifie les informations à récupérer. Ce paramètre peut prendre l'une des valeurs constantes suivantes.

Constant	Importance de la valeur de retour de la fonction
CMM_DESCRIPTION	Chaîne de texte qui décrit le module de gestion des couleurs.
CMM_DLL_VERSION	Numéro de version de la DLL CMM.
CMM_DRIVER_LEVEL	Informations de compatibilité des pilotes.
CMM_IDENT	Signature d'identification CMM enregistrée auprès de l'International Color Consortium (ICC).
CMM_LOGOICON	Icône de logo pour cette CMM.
CMM_VERSION	Version de Windows prise en charge.
CMM_WIN_VERSION	Compatibilité descendante avec Windows 95.

Valeur retournée

Si cette fonction réussit, la valeur de retour est la même valeur différente de zéro qui a été transmise via le paramètre *dwInfo* . Si la fonction échoue, la valeur de retour est

égale à zéro.

Notes

La fonction **CMGetInfo** peut être appelée directement par les applications pour obtenir des informations sur la CMM. Les applications ne doivent pas appeler d'autres fonctions CMM directement. Pour obtenir des informations CMM, récupérez le chemin d'accès à la CMM à partir du Registre. Appelez la fonction d'API Windows GetModuleHandle et transmettez le nom de fichier de la CMM comme valeur de son paramètre. Appelez la fonction **CMGetInfo** et passez-lui la constante CMM_DESCRIPTION comme valeur de son paramètre. Appelez la fonction LoadString . Transmettez le handle de module comme premier paramètre et la valeur de retour de la fonction **CMGetInfo** comme valeur du deuxième paramètre.

Les machines virtuelles qui ne s'exécutent pas sur Windows 95 doivent retourner 0x0050000 pour CMM_WIN_VERSION.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

CmGetNamedProfileInfo, fonction (icm.h)

Article24/08/2023

Récupère des informations sur le profil de couleur nommé spécifié.

Syntaxe

```
C++
```

```
BOOL CMGetNamedProfileInfo(
  HPROFILE            hProfile,
  PNAMED_PROFILE_INFO pNamedProfileInfo
);
```

Paramètres

`hProfile`

Handle du profil à partir duquel les informations seront récupérées.

`pNamedProfileInfo`

Pointeur vers une structure **NAMED_PROFILE_INFO** .

Valeur retournée

Si cette fonction réussit, la valeur de retour est TRUE.

Si cette fonction échoue, la valeur de retour est FALSE. Dans ce cas, la CMM doit appeler **SetLastError** pour définir la dernière erreur sur une valeur d'erreur valide définie dans Winerror.h.

Notes

Cette fonction est requise dans la CMM par défaut. Elle est facultative pour toutes les autres machines virtuelles.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- NAMED_PROFILE_INFO

Commentaires

Fonction CMGetPS2ColorRenderingDictionary (icm.h)

Article01/03/2024

Syntaxe

```
C++

BOOL CMGetPS2ColorRenderingDictionary(
  HPROFILE hProfile,
  DWORD    dwIntent,
  LPBYTE   lpBuffer,
  LPDWORD  lpcbSize,
  LPBOOL   lpbBinary
);
```

Paramètres

`hProfile`

`dwIntent`

`lpBuffer`

`lpcbSize`

`lpbBinary`

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 10 Build 20348
Serveur minimal pris en charge	Windows 10 Build 20348
En-tête	icm.h

Commentaires

Cette page a-t-elle été utile ? Yes No

Indiquer des commentaires sur le produit | Obtenir de l'aide sur Microsoft Q&A

CmGetPS2ColorRenderingIntent, fonction (icm.h)

Article26/02/2024

Récupère l'intention de rendu des couleurs PostScript niveau 2 à partir d'un profil.

Syntaxe

```
C++

BOOL CMGetPS2ColorRenderingIntent(
  HPROFILE hProfile,
  DWORD    dwIntent,
  LPBYTE   lpBuffer,
  LPDWORD  lpcbSize
);
```

Paramètres

`hProfile`

Spécifie le profil à utiliser.

`dwIntent`

Spécifie l'intention de rendu souhaitée à récupérer. Peut avoir l'une des valeurs suivantes :

INTENT_PERCEPTUAL
INTENT_SATURATION
INTENT_RELATIVE_COLORIMETRIC
INTENT_ABSOLUTE_COLORIMETRIC

Pour plus d'informations, consultez Rendu des intentions.

`lpBuffer`

Pointe vers une mémoire tampon dans laquelle l'intention de rendu des couleurs doit être placée. Si le pointeur a la valeur NULL, la fonction retourne la taille requise pour cette mémoire tampon dans **lpcbSize*.

`lpcbSize`

Pointe vers une variable spécifiant la taille de la mémoire tampon. Au retour, la variable contient le nombre d'octets réellement copiés dans la mémoire tampon.

Valeur retournée

Si cette fonction réussit, la valeur de retour est TRUE. Elle retourne également TRUE si elle est appelée avec *lpBuffer* définie sur NULL et si la taille de la mémoire tampon requise est copiée dans *lpcbSize*.

Si cette fonction échoue, la valeur de retour est FALSE. Dans ce cas, la CMM doit appeler **SetLastError** pour définir la dernière erreur sur une valeur d'erreur valide définie dans Winerror.h.

Remarques

Cette fonction est facultative pour toutes les machines virtuelles.

Si une CMM ne prend pas en charge cette fonction, Windows utilise la CMM par défaut pour obtenir l'intention de rendu des couleurs.

Si la balise n'est pas présente dans le profil indiqué par *hProfile*, la CMM la crée. L'intention de rendu résultante peut être utilisée comme opérande pour l'opérateur **findcolorrendering** PostScript Level 2.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

Cette page a-t-elle été utile ? Yes No

Indiquer des commentaires sur le produit | Obtenir de l'aide sur Microsoft Q&A

CmGetPS2ColorSpaceArray, fonction (icm.h)

Article28/02/2024

Syntaxe

```
C++

BOOL CMGetPS2ColorSpaceArray(
  HPROFILE hProfile,
  DWORD    dwIntent,
  DWORD    dwCSAType,
  LPBYTE   lpBuffer,
  LPDWORD  lpcbSize,
  LPBOOL   lpbBinary
);
```

Paramètres

`hProfile`

`dwIntent`

`dwCSAType`

`lpBuffer`

`lpcbSize`

`lpbBinary`

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 10 Build 20348
Serveur minimal pris en charge	Windows 10 Build 20348

Condition requise	Valeur
En-tête	icm.h

Commentaires

Cette page a-t-elle été utile ? Yes No

Indiquer des commentaires sur le produit | Obtenir de l'aide sur Microsoft Q&A

Fonction CMIsProfileValid (icm.h)

Article24/08/2023

Indique si le profil donné est un profil ICC valide qui peut être utilisé pour la gestion des couleurs.

Syntaxe

```
C++

BOOL CMIsProfileValid(
  HPROFILE hProfile,
  LPBOOL   lpbValid
);
```

Paramètres

`hProfile`

Spécifie le profil à case activée.

`lpbValid`

Pointeur vers une variable définie à la sortie sur TRUE si le profil est un profil ICC valide, ou FALSE si ce n'est pas le cas.

Valeur retournée

Si cette fonction réussit, la valeur de retour est TRUE.

Si cette fonction échoue, la valeur de retour est FALSE. Si la fonction échoue, le CMM doit appeler **SetLastError** pour définir la dernière erreur sur une valeur d'erreur valide définie dans Winerror.h.

Notes

Seule la gestion CMM par défaut de Windows est requise pour exporter cette fonction ; elle est facultative pour toutes les autres machines virtuelles.

Si une gestion CMM ne prend pas en charge cette fonction, Windows utilise la gestion CMM par défaut pour valider le profil.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

Fonction CMTranslateColors (icm.h)

Article26/02/2024

Traduit un tableau de couleurs d'un espace de couleurs source en espace de couleur de destination à l'aide d'une transformation de couleur.

Syntaxe

```
C++

BOOL CMTranslateColors(
  HCMTRANSFORM hcmTransform,
  LPCOLOR      lpaInputColors,
  DWORD        nColors,
  COLORTYPE    ctInput,
  LPCOLOR      lpaOutputColors,
  COLORTYPE    ctOutput
);
```

Paramètres

`hcmTransform`

Spécifie la transformation de couleur à utiliser.

`lpaInputColors`

Pointe vers un tableau de structures **COLOR** à traduire.

`nColors`

Spécifie le nombre d'éléments du tableau.

`ctInput`

Spécifie le type de couleur de l'entrée.

`lpaOutputColors`

Pointe vers une mémoire tampon dans laquelle un tableau de structures **COLOR** traduites doit être placé.

`ctOutput`

Spécifie le type de couleur de sortie.

Valeur retournée

Si cette fonction réussit, la valeur de retour est TRUE.

Si cette fonction échoue, la valeur de retour est FALSE. Le CMM doit appeler **SetLastError** pour définir la dernière erreur sur une valeur d'erreur valide définie dans Winerror.h.

Remarques

Chaque CMM est nécessaire pour exporter cette fonction.

Si les types de couleurs d'entrée et de sortie ne sont pas compatibles avec la transformation de couleur, cette fonction doit échouer.

Notez que cette fonction doit prendre en charge la traduction sur place. Autrement dit, chaque fois que l'empreinte mémoire de la sortie est inférieure ou égale à l'empreinte mémoire de l'entrée, cette fonction doit être en mesure de traduire les couleurs bitmap même si les mémoires tampons source et de destination sont identiques.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

Cette page a-t-elle été utile ? Yes No

Indiquer des commentaires sur le produit | Obtenir de l'aide sur Microsoft Q&A

Fonction CMTranslateRGB (icm.h)

Article24/08/2023

Convertit un RGBQuad fourni par l'application en espace de couleurs de l'appareil.

Syntaxe

```
C++

BOOL CMTranslateRGB(
  HCMTRANSFORM hcmTransform,
  COLORREF     ColorRef,
  LPCOLORREF   lpColorRef,
  DWORD        dwFlags
);
```

Paramètres

`hcmTransform`

Spécifie la transformation à utiliser.

`ColorRef`

RGBQuad à traduire.

`lpColorRef`

Pointe vers une mémoire tampon dans laquelle placer la traduction.

`dwFlags`

Spécifie la façon dont la transformation doit être utilisée pour effectuer la traduction. Ce paramètre peut prendre l'une des significations suivantes.

Valeur	Signification
CMS_FORWARD	Utiliser la transformation vers l'avant
CMS_BACKWARD	Utiliser la transformation inverse

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Le CMM doit appeler **SetLastError** pour définir la dernière erreur sur une valeur d'erreur valide définie dans Winerror.h.

Notes

Chaque CMM est nécessaire pour exporter cette fonction.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

CmTranslateRGBs, fonction (icm.h)

Article29/02/2024

[**CMTranslateRGBs** n'est plus disponible pour une utilisation à partir de Windows Vista.]

Convertit une bitmap d'un espace de couleurs à un autre à l'aide d'une transformation de couleur.

Syntaxe

```
C++

BOOL CMTranslateRGBs(
  HCMTRANSFORM hcmTransform,
  LPVOID       lpSrcBits,
  BMFORMAT     bmInput,
  DWORD        dwWidth,
  DWORD        dwHeight,
  DWORD        dwStride,
  LPVOID       lpDestBits,
  BMFORMAT     bmOutput,
  DWORD        dwTranslateDirection
);
```

Paramètres

`hcmTransform`

Spécifie la transformation de couleur à utiliser.

`lpSrcBits`

Pointe vers l'image bitmap à traduire.

`bmInput`

Spécifie le format bitmap d'entrée.

`dwWidth`

Spécifie le nombre de pixels par ligne de balayage dans la bitmap d'entrée.

`dwHeight`

Spécifie le nombre de lignes d'analyse dans la bitmap d'entrée.

`dwStride`

Spécifie le nombre d'octets entre le début d'une ligne d'analyse et le début de la suivante dans la bitmap d'entrée. Si *dwStride* est défini sur zéro, la CMM doit supposer que les lignes d'analyse sont remplies de façon à être alignées sur **DWORD**.

`lpDestBits`

Pointe vers une mémoire tampon de destination dans laquelle placer la bitmap traduite.

`bmOutput`

Spécifie le format bitmap de sortie.

`dwTranslateDirection`

Spécifie la direction de la transformation utilisée pour la traduction. Ce paramètre doit prendre l'une des valeurs suivantes.

Agrandir le tableau

Valeur	Signification
CMS_FORWARD	Utiliser la transformation vers l'avant
CMS_BACKWARD	Utiliser la transformation inverse

Valeur retournée

À compter de Windows Vista, la CMM par défaut (Icm32.dll) retourne **FALSE** et GetLastError signale ERROR_NOT_SUPPORTED.

Windows Server 2003, Windows XP et Windows 2000 :

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Si la fonction échoue, la CMM doit appeler SetLastError pour définir la dernière erreur sur une valeur d'erreur valide définie dans Winerror.h.

Remarques

À compter de Windows Vista, les implémenteurs CMM ne sont plus nécessaires pour implémenter cette méthode.

Windows Server 2003, Windows XP et Windows 2000 :

Chaque CMM est nécessaire pour exporter cette fonction.

Lors de l'écriture dans la mémoire tampon de destination, la CMM doit s'assurer que les lignes d'analyse sont remplies pour être alignées sur **DWORD**.

Si les formats d'entrée et de sortie ne sont pas compatibles avec la transformation de couleur, cette fonction échoue.

Si les formats bitmap d'entrée et de sortie sont à 3 canaux, 4 octets par pixel comme dans le cas de BM_xRGBQUADS, le 4e octet doit être conservé et copié dans la mémoire tampon de sortie.

Notez que cette fonction doit prendre en charge la traduction sur place. Autrement dit, chaque fois que l'empreinte mémoire de la sortie est inférieure ou égale à l'empreinte mémoire de l'entrée, cette fonction doit être en mesure de traduire les couleurs bitmap même si les mémoires tampons source et de destination sont identiques.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

Cette page a-t-elle été utile ? Yes No

Indiquer des commentaires sur le produit | Obtenir de l'aide sur Microsoft Q&A

CMTranslateRGBsExt, fonction (icm.h)

Article24/08/2023

Convertit une bitmap d'un format défini dans un autre format défini et appelle régulièrement une fonction de rappel, le cas échéant, pour signaler la progression et permettre à l'application appelante d'arrêter la traduction.

Syntaxe

```
C++
```

```
BOOL CMTranslateRGBsExt(
  HCMTRANSFORM   hcmTransform,
  LPVOID         lpSrcBits,
  BMFORMAT       bmInput,
  DWORD          dwWidth,
  DWORD          dwHeight,
  DWORD          dwInputStride,
  LPVOID         lpDestBits,
  BMFORMAT       bmOutput,
  DWORD          dwOutputStride,
  LPBMCALLBACKFN lpfnCallback,
  LPARAM         ulCallbackData
);
```

Paramètres

`hcmTransform`

Spécifie la transformation de couleur à utiliser.

`lpSrcBits`

Pointeur vers la bitmap à traduire.

`bmInput`

Spécifie le format bitmap d'entrée.

`dwWidth`

Spécifie le nombre de pixels par ligne de balayage dans la bitmap d'entrée.

`dwHeight`

Spécifie le nombre de lignes d'analyse dans la bitmap d'entrée.

`dwInputStride`

Spécifie le nombre d'octets entre le début d'une ligne d'analyse et le début de la suivante dans la bitmap d'entrée. Si *dwInputStride* est défini sur zéro, la CMM doit supposer que les lignes de balayage sont remplies de façon à être alignées sur **DWORD**.

`lpDestBits`

Pointe vers une mémoire tampon de destination dans laquelle placer la bitmap traduite.

`bmOutput`

Spécifie le format bitmap de sortie.

`dwOutputStride`

Spécifie le nombre d'octets entre le début d'une ligne d'analyse et le début de la suivante dans la bitmap d'entrée. Si *dwOutputStride* est défini sur zéro, la CMM doit remplir les lignes de balayage afin qu'elles soient alignées sur **DWORD**.

`lpfnCallback`

Pointeur vers une fonction de rappel fournie par l'application appelée régulièrement par **CMTranslateRGBsExt** pour signaler la progression et permettre au processus appelant d'annuler la traduction. (Voir **ICMProgressProcCallback**.)

`ulCallbackData`

Données renvoyées à la fonction de rappel, par exemple pour identifier la traduction qui signale la progression.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE** et la CMM doit appeler **SetLastError** pour définir la dernière erreur sur une valeur d'erreur valide définie dans Winerror.h.

Notes

Chaque CMM est nécessaire pour exporter cette fonction.

Lors de l'écriture dans la mémoire tampon de destination, la CMM doit s'assurer que les lignes d'analyse sont remplies pour être alignées sur **DWORD**.

Si les formats d'entrée et de sortie ne sont pas compatibles avec la transformation de couleur, cette fonction échoue.

Si les formats bitmap d'entrée et de sortie sont à 3 canaux, 4 octets par pixel comme dans le cas de BM_xRGBQUADS, les quatrièmes octets doivent être conservés et copiés dans la mémoire tampon de sortie.

Si la fonction de rappel retourne zéro, le traitement doit être annulé et **CMTranslateRGBsExt** doit retourner zéro pour indiquer l'échec ; la mémoire tampon de sortie peut être partiellement remplie.

Notez que cette fonction doit prendre en charge la traduction sur place. Autrement dit, chaque fois que l'empreinte mémoire de la sortie est inférieure ou égale à l'empreinte mémoire de l'entrée, cette fonction doit être en mesure de traduire les couleurs bitmap même si les mémoires tampons source et de destination sont identiques.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Voir aussi

- Concepts de base de la gestion des couleurs)
- Fonctions

Commentaires

Fonction ColorCorrectPalette (wingdi.h)

Article27/08/2023

La fonction **ColorCorrectPalette** corrige les entrées d'une palette à l'aide des paramètres WCS 1.0 dans le contexte d'appareil spécifié.

Syntaxe

```
C++

BOOL ColorCorrectPalette(
  HDC      hdc,
  HPALETTE hPal,
  DWORD    deFirst,
  DWORD    num
);
```

Paramètres

`hdc`

Spécifie un contexte d'appareil dont les paramètres WCS doivent être utilisés.

`hPal`

Spécifie le handle de la palette à corriger.

`deFirst`

Spécifie la première entrée de la palette à corriger.

`num`

Spécifie le nombre d'entrées à colorer correctement.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**.

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
Plateforme cible	Windows
En-tête	wingdi.h
Bibliothèque	Gdi32.lib
DLL	Gdi32.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

Fonction ColorMatchToTarget (wingdi.h)

Article27/08/2023

La fonction **ColorMatchToTarget** vous permet d'afficher un aperçu des couleurs telles qu'elles s'affichent sur l'appareil cible.

Syntaxe

```
C++

BOOL ColorMatchToTarget(
  HDC   hdc,
  HDC   hdcTarget,
  DWORD action
);
```

Paramètres

`hdc`

Spécifie le contexte de l'appareil pour la préversion, généralement l'écran.

`hdcTarget`

Spécifie le contexte de l'appareil cible, généralement une imprimante.

`action`

Constante qui peut avoir l'une des valeurs suivantes.

Valeur	Signification
CS_ENABLE	Mapper les couleurs à la gamme de couleurs de l'appareil cible. Cela permet la vérification des couleurs. Toutes les commandes de dessin suivantes sur le contrôleur de domaine affichent les couleurs telles qu'elles apparaissent sur l'appareil cible.
CS_DISABLE	Désactivez la vérification des couleurs.
CS_DELETE_TRANSFORM	Si la gestion des couleurs est activée pour le profil cible, désactivez-la et supprimez la transformation concaténée.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**.

Remarques

ColorMatchToTarget peut être utilisé pour vérifier les couleurs d'un appareil de sortie de couleur sur un autre appareil de sortie de couleur. Si vous définissez le paramètre *uiAction* sur CS_ENABLE, toutes les commandes de dessin suivantes sur le contrôleur de domaine affichent les couleurs telles qu'elles apparaissent sur l'appareil cible. Si *uiAction* a la valeur CS_DISABLE, la vérification est désactivée. Toutefois, la transformation de couleur actuelle n'est pas supprimée du contrôleur de domaine. Il est juste inactif.

Lorsque **ColorMatchToTarget** est appelé, la transformation de couleur de l'appareil cible est effectuée en premier, puis la transformation vers l'appareil en préversion est appliquée aux résultats de la première transformation. Il est principalement utilisé pour vérifier les conditions de mappage de gamut. Avant d'utiliser cette fonction, vous devez activer WCS pour les deux contextes d'appareil.

Cette fonction ne peut pas être mise en cascade. Bien que le mappage des couleurs à la cible soit activé en définissant *uiAction* sur CS_ENABLE, les modifications apportées à l'espace de couleurs ou à la méthode de mappage de gamuts sont ignorées. Ces modifications prennent ensuite effet lorsque le mappage de couleurs à la cible est désactivé.

> **Note** Une fuite de mémoire ne se produit pas si une application ne supprime pas de transformation à l'aide de CS_DELETE_TRANSFORM. La transformation est supprimée lorsque le contexte de l'appareil (DC) est fermé ou lorsque l'espace de couleur de l'application est supprimé. Toutefois, si la transformation ne va pas être utilisée à nouveau, ou si l'application n'effectue plus de correspondance des couleurs sur le contrôleur de domaine, elle doit supprimer explicitement la transformation pour libérer la mémoire qu'elle occupe.

Le paramètre *uiAction* doit être défini sur CS_DELETE_TRANSFORM uniquement si la gestion des couleurs est activée avant l'appel de la fonction **ColorMatchToTarget** .

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
Plateforme cible	Windows
En-tête	wingdi.h
Bibliothèque	Gdi32.lib
DLL	Gdi32.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

ColorProfileAddDisplayAssociation, fonction (icm.h)

Article24/08/2023

Associe un profil de couleurs installé à un affichage spécifié dans l'étendue donnée.

Syntaxe

C++

```
HRESULT ColorProfileAddDisplayAssociation(
  WCS_PROFILE_MANAGEMENT_SCOPE scope,
  PCWSTR                       profileName,
  LUID                         targetAdapterID,
  UINT32                       sourceID,
  BOOL                         setAsDefault,
  BOOL                         associateAsAdvancedColor
);
```

Paramètres

`scope`

Spécifie l'association à l'échelle du système ou de l'utilisateur actuel.

`profileName`

Identifie le profil installé à associer.

`targetAdapterID`

Identificateur attribué à l'adaptateur (par exemple, GPU) de l'affichage cible. Pour plus d'informations, consultez Remarques .

`sourceID`

Identificateur affecté à la source de l'affichage. Pour plus d'informations, consultez Remarques .

`setAsDefault`

Indique s'il faut ou non définir le profil nouvellement associé comme profil par défaut.

`associateAsAdvancedColor`

Spécifie à quelle liste d'association le nouveau profil est ajouté.

Valeur retournée

S_OK de réussite ou d'une valeur **HRESULT** d'échec

Notes

Pour plus d'informations sur les ID de carte d'affichage et les ID sources, consultez Connexion et configuration des affichages .

Spécifications

Client minimal pris en charge	Windows 10 Build 20348
Serveur minimal pris en charge	Windows 10 Build 20348
En-tête	icm.h

Voir aussi

Connexion et configuration des affichages

Commentaires

ColorProfileGetDisplayDefault, fonction (icm.h)

Article26/02/2024

Obtient le profil de couleur par défaut pour un affichage donné dans l'étendue spécifiée.

Syntaxe

```
C++

HRESULT ColorProfileGetDisplayDefault(
  WCS_PROFILE_MANAGEMENT_SCOPE scope,
  LUID                         targetAdapterID,
  UINT32                       sourceID,
  COLORPROFILETYPE             profileType,
  COLORPROFILESUBTYPE          profileSubType,
  LPWSTR                       *profileName
);
```

Paramètres

`scope`

Spécifie l'association à l'échelle du système ou de l'utilisateur actuel.

`targetAdapterID`

Identificateur attribué à l'adaptateur (par exemple, GPU) de l'affichage cible. Pour plus d'informations, consultez Remarques .

`sourceID`

Identificateur affecté à la source de l'affichage. Pour plus d'informations, consultez Remarques .

`profileType`

Type de profil de couleur à retourner (actuellement, seul CPT_ICC est pris en charge).

`profileSubType`

Sous-type du profil de couleur à retourner.

`profileName`

Reçoit un pointeur vers le nom de profil de couleur par défaut, qui doit être libéré avec LocalFree.

Remarques

Pour plus d'informations sur les ID de carte d'affichage et les ID sources, consultez Connexion et configuration des affichages .

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 10 Build 20348
Serveur minimal pris en charge	Windows 10 Build 20348
En-tête	icm.h

Voir aussi

Connexion et configuration des affichages

Commentaires

Fonction ColorProfileGetDisplayList (icm.h)

Article24/08/2023

Récupère la liste des profils associés à un affichage donné dans l'étendue spécifiée.

Syntaxe

```
C++

HRESULT ColorProfileGetDisplayList(
  WCS_PROFILE_MANAGEMENT_SCOPE scope,
  LUID                         targetAdapterID,
  UINT32                       sourceID,
  LPWSTR                       **profileList,
  PDWORD                       profileCount
);
```

Paramètres

`scope`

Spécifie l'association comme étant à l'échelle du système ou l'utilisateur actuel.

`targetAdapterID`

Identificateur attribué à l'adaptateur (par exemple, GPU) de l'affichage cible. Pour plus d'informations, consultez Remarques .

`sourceID`

Identificateur attribué à la source de l'affichage. Pour plus d'informations, consultez Remarques .

`profileList`

Le pointeur vers une mémoire tampon où les noms de profil sont placés doit être libéré avec LocalFree.

`profileCount`

Reçoit le nombre de noms de profils copiés dans profileList.

Valeur retournée

S_OK de réussite ou d'une valeur **HRESULT d'échec**

Notes

Pour plus d'informations sur les ID d'adaptateur d'affichage et les ID sources, consultez Connexion et configuration des affichages .

Spécifications

Client minimal pris en charge	Windows 10 Build 20348
Serveur minimal pris en charge	Windows 10 Build 20348
En-tête	icm.h

Voir aussi

Connexion et configuration des affichages

Commentaires

ColorProfileGetDisplayUserScope, fonction (icm.h)

Article26/02/2024

Obtient l'étendue de profil de couleur actuellement sélectionnée de l'affichage fourni (utilisateur ou système).

Syntaxe

```
C++

HRESULT ColorProfileGetDisplayUserScope(
  LUID                          targetAdapterID,
  UINT32                        sourceID,
  WCS_PROFILE_MANAGEMENT_SCOPE *scope
);
```

Paramètres

`targetAdapterID`

Identificateur attribué à l'adaptateur (par exemple, GPU) de l'affichage cible. Pour plus d'informations, consultez Remarques .

`sourceID`

Identificateur affecté à la source de l'affichage. Pour plus d'informations, consultez Remarques .

`scope`

Retourne l'étendue du profil de couleur actuellement sélectionné : l'utilisateur ou le système actuel.

Valeur retournée

S_OK de réussite ou d'une valeur **HRESULT** d'échec

Remarques

Pour plus d'informations sur les ID de carte d'affichage et les ID sources, consultez Connexion et configuration des affichages .

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 10 Build 20348
Serveur minimal pris en charge	Windows 10 Build 20348
En-tête	icm.h

Voir aussi

Connexion et configuration des affichages

Commentaires

Cette page a-t-elle été utile ? Yes No

Indiquer des commentaires sur le produit | Obtenir de l'aide sur Microsoft Q&A

Fonction ColorProfileRemoveDisplayAssociation (icm.h)

Article26/02/2024

Dissocie un profil de couleur installé d'un affichage spécifié dans l'étendue donnée.

Syntaxe

```
C++

HRESULT ColorProfileRemoveDisplayAssociation(
  WCS_PROFILE_MANAGEMENT_SCOPE scope,
  PCWSTR                       profileName,
  LUID                         targetAdapterID,
  UINT32                       sourceID,
  BOOL                         dissociateAdvancedColor
);
```

Paramètres

`scope`

Spécifie l'association comme étant à l'échelle du système ou l'utilisateur actuel.

`profileName`

Identifie le profil installé à associer.

`targetAdapterID`

Identificateur attribué à l'adaptateur (par exemple, GPU) de l'affichage cible. Pour plus d'informations, consultez Remarques .

`sourceID`

Identificateur attribué à la source de l'affichage. Pour plus d'informations, consultez Remarques .

`dissociateAdvancedColor`

Spécifie à quelle liste d'association le nouveau profil est ajouté.

Valeur retournée

S_OK de réussite ou d'une valeur **HRESULT d'échec**

Remarques

Pour plus d'informations sur les ID d'adaptateur d'affichage et les ID sources, consultez Connexion et configuration des affichages .

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 10 Build 20348
Serveur minimal pris en charge	Windows 10 Build 20348
En-tête	icm.h

Voir aussi

Connexion et configuration des affichages

Commentaires

ColorProfileSetDisplayDefaultAssociation, fonction (icm.h)

Article24/08/2023

Définit un profil de couleur installé comme profil par défaut pour un affichage spécifié dans l'étendue donnée.

Syntaxe

```
C++

HRESULT ColorProfileSetDisplayDefaultAssociation(
  WCS_PROFILE_MANAGEMENT_SCOPE scope,
  PCWSTR                       profileName,
  COLORPROFILETYPE             profileType,
  COLORPROFILESUBTYPE          profileSubType,
  LUID                         targetAdapterID,
  UINT32                       sourceID
);
```

Paramètres

`scope`

Spécifie l'association à l'échelle du système ou de l'utilisateur actuel.

`profileName`

Identifie le profil installé à associer.

`profileType`

Type de profil de couleur à définir par défaut (actuellement, seul CPT_ICC est pris en charge).

`profileSubType`

Sous-type du profil de couleur à définir comme valeur par défaut.

`targetAdapterID`

Identificateur attribué à l'adaptateur (par exemple, GPU) de l'affichage cible. Pour plus d'informations, consultez Remarques .

`sourceID`

Identificateur affecté à la source de l'affichage. Pour plus d'informations, consultez Remarques .

Valeur retournée

S_OK de réussite ou d'une valeur **HRESULT** d'échec

Notes

Pour plus d'informations sur les ID de carte d'affichage et les ID sources, consultez Connexion et configuration des affichages .

Spécifications

Client minimal pris en charge	Windows 10 Build 20348
Serveur minimal pris en charge	Windows 10 Build 20348
En-tête	icm.h

Voir aussi

Connexion et configuration des affichages

Commentaires

Fonction ConvertColorNameToIndex (icm.h)

Article24/08/2023

Convertit les noms de couleurs d'un espace de couleurs nommé en numéros d'index dans un profil de couleur ICC (International Color Consortium).

Syntaxe

```
C++
```

```
BOOL ConvertColorNameToIndex(
  HPROFILE    hProfile,
  PCOLOR_NAME paColorName,
  PDWORD      paIndex,
  DWORD       dwCount
);
```

Paramètres

`hProfile`

Handle d'un profil de couleur nommé ICC.

`paColorName`

Pointeur vers un tableau de structures de noms de couleur.

`paIndex`

Pointeur vers un tableau de **DWORDS** que cette fonction remplit avec les index. Les index commencent par un, et non par zéro.

`dwCount`

Nombre de noms de couleurs à convertir.

Valeur retournée

Si cette fonction réussit avec la conversion, la valeur de retour est **TRUE**.

Si la fonction de conversion échoue, la valeur de retour est **FALSE**.

Notes

Cette fonction échoue si *hProfile* n'est pas un profil ICC valide.

Cette fonction ne prend pas en charge les profils WINDOWS Color System (WCS) CAMP, DMP et GMMP ; car les profils nommés sont des types de profils ICC explicites.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

Fonction ConvertIndexToColorName (icm.h)

Article24/08/2023

Transforme les index d'un espace de couleurs en tableau de noms dans un espace de couleurs nommé.

Syntaxe

```
C++
```

```
BOOL ConvertIndexToColorName(
  HPROFILE    hProfile,
  PDWORD      paIndex,
  PCOLOR_NAME paColorName,
  DWORD       dwCount
);
```

Paramètres

`hProfile`

Handle d'un profil d'espace de couleur ICC (International Color Consortium).

`paIndex`

Pointeur vers un tableau de numéros d'index d'espace de couleur. Les indices commencent par un, et non par zéro.

`paColorName`

Pointeur vers un tableau de structures de noms de couleur.

`dwCount`

Nombre d'index à convertir.

Valeur retournée

Si cette fonction de conversion réussit, la valeur de retour est **TRUE**.

Si cette fonction de conversion échoue, la valeur de retour est **FALSE**.

Notes

Cette fonction échoue si *hProfile* n'est pas un profil ICC valide.

Cette fonction ne prend pas en charge les profils WINDOWS Color System (WCS) CAMP, DMP et GMMP ; car les profils nommés sont des types de profils ICC explicites.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

CreateColorSpaceA, fonction (wingdi.h)

Article27/08/2023

La fonction **CreateColorSpace** crée un espace de couleurs logique.

Syntaxe

```
C++

HCOLORSPACE CreateColorSpaceA(
  LPLOGCOLORSPACEA lplcs
);
```

Paramètres

`lplcs`

Pointeur vers la structure de données LOGCOLORSPACE .

Valeur retournée

Si cette fonction réussit, la valeur de retour est un handle qui identifie un espace de couleurs.

Si cette fonction échoue, la valeur de retour est **NULL**.

Remarques

Lorsque l'espace de couleur n'est plus nécessaire, utilisez **DeleteColorSpace** pour le supprimer.

Windows 95/98/Me : CreateColorSpaceW est pris en charge par la couche Microsoft pour Unicode. Pour cela, vous devez ajouter certains fichiers à votre application, comme indiqué dans Microsoft Layer pour Unicode sur Windows 95/98/Me Systems .

> **Notes**
>
> L'en-tête wingdi.h définit CreateColorSpace comme un alias qui sélectionne automatiquement la version ANSI ou Unicode de cette fonction en fonction de la

définition de la constante de préprocesseur UNICODE. Le mélange de l'utilisation de l'alias neutre en encodage avec du code qui n'est pas neutre en encodage peut entraîner des incompatibilités qui entraînent des erreurs de compilation ou d'exécution. Pour plus d'informations, consultez **Conventions pour les prototypes de fonction**.

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
Plateforme cible	Windows
En-tête	wingdi.h
Bibliothèque	Gdi32.lib
DLL	Gdi32.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- DeleteColorSpaceW

Commentaires

CreateColorTransformA, fonction (icm.h)

Article29/02/2024

Crée une transformation de couleur que les applications peuvent utiliser pour effectuer la gestion des couleurs.

Syntaxe

```
C++

HTRANSFORM CreateColorTransformA(
  LPLOGCOLORSPACEA pLogColorSpace,
  HPROFILE         hDestProfile,
  HPROFILE         hTargetProfile,
  DWORD            dwFlags
);
```

Paramètres

`pLogColorSpace`

Pointeur vers l'entrée **LOGCOLORSPACEA**.

`hDestProfile`

Gérez le profil de l'appareil de destination. La fonction détermine si le HPROFILE contient des informations de profil ICC (International Color Consortium) ou Windows Color System (WCS).

`hTargetProfile`

Gérez le profil de l'appareil cible. La fonction détermine si le fichier HPROFILE contient des informations de profil ICC ou WCS.

`dwFlags`

Spécifie les indicateurs pour utiliser la création de contrôle de la transformation. Consultez la section Notes.

Valeur de retour

Si cette fonction réussit, la valeur de retour est un handle de la transformation de couleur.

Si cette fonction échoue, la valeur de retour est **NULL**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Remarques

Si le profil cible est **NULL**, la transformation va de l'espace de couleurs logique source au profil de destination. Si le profil cible est donné, la transformation passe de l'espace de couleurs logique source au profil cible, puis au profil de destination. Cela permet d'afficher un aperçu de la sortie destinée à l'appareil cible sur l'appareil de destination.

Les valeurs dans *dwFlags* sont destinées uniquement à des indicateurs. Le module de gestion des couleurs doit déterminer la meilleure façon de les utiliser.

Windows Vista : trois nouveaux indicateurs ont été ajoutés et peuvent être utilisés avec *dwFlags* :

Agrandir le tableau

Indicateur	Description
PRESERVEBLACK	Si ce bit est défini, le moteur de transformation insère le GMMP de génération noire approprié comme dernier GMMP dans la séquence de transformation. Cet indicateur fonctionne uniquement dans une transformation WCS pure.
SEQUENTIAL_TRANSFORM	Si ce bit est défini, chaque étape du pipeline de traitement WCS est effectuée pour chaque pixel de l'image et aucune transformation de couleur optimisée n'est générée. Cet indicateur fonctionne uniquement dans une transformation WCS pure. **Restrictions** : une transformation créée avec le jeu d'indicateurs SEQUENTIAL_TRANSFORM ne peut être utilisée que dans le thread sur lequel elle a été créée et uniquement pour un seul appel de traduction de couleur à la fois. COM doit être initialisé avant de créer la transformation séquentielle et doit rester initialisé pendant toute la durée de vie de l'objet transform.
WCS_ALWAYS	Si ce bit est défini, même les transformations toutes ICC utilisent le chemin de code WCS.

Notes

SEQUENTIAL_TRANSFORM a été omis par inadvertance de l'en-tête icm.h dans le Kit de développement logiciel (SDK) Windows Vista. Si vous souhaitez utiliser l'indicateur **SEQUENTIAL_TRANSFORM** , définissez-le dans votre application comme suit :#define SEQUENTIAL_TRANSFORM 0x80800000

Pour plus d'informations, consultez Indicateurs de création de transformation CMM. Tous les indicateurs mentionnés ici sont pris en charge pour tous les types de transformations, à l'exception de FAST_TRANSLATE, qui fonctionne uniquement dans une transformation ICC-to-ICC pure.

La fonction **CreateColorTransform** est utilisée en dehors d'un contexte d'appareil. Les couleurs peuvent changer lors de la transformation d'un profil de couleur vers le même profil de couleur. Cela est dû à des erreurs de précision. Par conséquent, une transformation de couleur ne doit pas être effectuée dans ces circonstances.

Les balises B2Ax sont requises pour tout profil qui est la cible d'une transformation.

La prise en charge de la transformation WCS pour les profils Icc ColorSpace est limitée aux profils d'espace de couleurs RVB. Les types de profils ICC suivants ne peuvent pas être utilisés dans une transformation traitée CITE, soit une transformation MIXTE WCS/ICC, soit une transformation icc avec **WCS_ALWAYS** défini :

- Profils ColorSpace non RVB
- Profils NamedColor
- profils n-canal (où n8 >)
- Profils DeviceLink
- Profils abstraits

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib

Condition requise	Valeur
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

CreateColorTransformW, fonction (icm.h)

Article29/02/2024

Crée une transformation de couleur que les applications peuvent utiliser pour effectuer la gestion des couleurs.

Syntaxe

```
C++

HTRANSFORM CreateColorTransformW(
  LPLOGCOLORSPACEW pLogColorSpace,
  HPROFILE         hDestProfile,
  HPROFILE         hTargetProfile,
  DWORD            dwFlags
);
```

Paramètres

`pLogColorSpace`

Pointeur vers l'entrée **LOGCOLORSPACEA**.

`hDestProfile`

Gérez le profil de l'appareil de destination. La fonction détermine si le HPROFILE contient des informations de profil ICC (International Color Consortium) ou WCS (Windows Color System).

`hTargetProfile`

Gérez le profil de l'appareil cible. La fonction détermine si le fichier HPROFILE contient des informations de profil ICC ou WCS.

`dwFlags`

Spécifie les indicateurs à utiliser pour la création de contrôle de la transformation. Consultez la section Notes.

Valeur de retour

Si cette fonction réussit, la valeur de retour est un handle de la transformation de couleur.

Si cette fonction échoue, la valeur de retour est **NULL**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Remarques

Si le profil cible est **NULL**, la transformation passe de l'espace de couleur logique source au profil de destination. Si le profil cible est donné, la transformation va de l'espace de couleur logique source au profil cible, puis au profil de destination. Cela permet d'afficher un aperçu de la sortie destinée à l'appareil cible sur l'appareil de destination.

Les valeurs dans *dwFlags* sont destinées à servir uniquement d'indicateurs. Le module de gestion des couleurs doit déterminer la meilleure façon de les utiliser.

Windows Vista : Trois nouveaux indicateurs qui peuvent être utilisés avec *dwFlags* ont été ajoutés :

Agrandir le tableau

Indicateur	Description
PRESERVEBLACK	Si ce bit est défini, le moteur de transformation insère le GMMP de génération noire approprié comme dernier GMMP dans la séquence de transformation. Cet indicateur fonctionne uniquement dans une transformation WCS pure.
SEQUENTIAL_TRANSFORM	Si ce bit est défini, chaque étape du pipeline de traitement WCS est effectuée pour chaque pixel de l'image et aucune transformation de couleur optimisée n'est générée. Cet indicateur fonctionne uniquement dans une transformation WCS pure. **Restrictions** : une transformation créée avec l'indicateur de SEQUENTIAL_TRANSFORM ne peut être utilisée que dans le thread sur lequel elle a été créée et uniquement pour un appel de traduction de couleur à la fois. COM doit être initialisé avant la création de la transformation séquentielle et doit rester initialisé pendant toute la durée de vie de l'objet de transformation.
WCS_ALWAYS	Si ce bit est défini, même les transformations toutes icc utilisent le chemin de code WCS.

Notes

SEQUENTIAL_TRANSFORM a été omis par inadvertance de l'en-tête icm.h dans le Kit de développement logiciel (SDK) Windows Vista. Si vous souhaitez utiliser l'indicateur **SEQUENTIAL_TRANSFORM** , définissez-le dans votre application comme suit :#define SEQUENTIAL_TRANSFORM 0x80800000

Pour plus d'informations, consultez Indicateurs de création de transformation CMM. Tous les indicateurs mentionnés ici sont pris en charge pour tous les types de transformations, à l'exception de FAST_TRANSLATE, qui fonctionne uniquement dans une transformation icc-à-ICC pure.

La fonction **CreateColorTransform** est utilisée en dehors d'un contexte d'appareil. Les couleurs peuvent changer lors de la transformation d'un profil de couleur vers le même profil de couleur. Cela est dû à des erreurs de précision. Par conséquent, une transformation de couleur ne doit pas être effectuée dans ces circonstances.

Les balises B2Ax sont requises pour tout profil qui est la cible d'une transformation.

La prise en charge des transformations WCS pour les profils Icc ColorSpace est limitée aux profils d'espace de couleurs RVB. Les types de profils ICC suivants ne peuvent pas être utilisés dans une transformation traitée cite, soit une transformation WCS/ICC mixte, soit une transformation de tous les profils ICC avec **WCS_ALWAYS** définie :

- Profils ColorSpace non RVB
- Profils NamedColor
- profils n-canal (où n > 8)
- Profils DeviceLink
- Profils abstraits

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib

Condition requise	Valeur
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

CreateDeviceLinkProfile, fonction (icm.h)

Article24/08/2023

Crée un *profil de liaison d'appareil* ICC (International Color Consortium) à partir d'un ensemble de profils de couleur, à l'aide des intentions spécifiées.

Syntaxe

```
C++

BOOL CreateDeviceLinkProfile(
  PHPROFILE hProfile,
  DWORD     nProfiles,
  PDWORD    padwIntent,
  DWORD     nIntents,
  DWORD     dwFlags,
  PBYTE     *pProfileData,
  DWORD     indexPreferredCMM
);
```

Paramètres

`hProfile`

Pointeur vers un tableau de handles des profils de couleur à utiliser. La fonction détermine si les HPROFILEs contiennent des informations de profil ICC et, si c'est le cas, elle les traite de manière appropriée.

`nProfiles`

Spécifie le nombre de profils dans le tableau pointé vers *hProfile*.

`padwIntent`

Pointeur vers un tableau de **DWORDS** contenant les intentions à utiliser. Consultez Intentions de rendu.

`nIntents`

Nombre d'intentions dans le tableau pointé vers par *padwIntent*.

`dwFlags`

Spécifie les indicateurs pour utiliser la création de contrôle de la transformation. Pour plus d'informations, consultez Indicateurs de création de transformation CMM.

`pProfileData`

Pointeur vers un pointeur vers une mémoire tampon. Si elle réussit, cette fonction alloue la mémoire tampon, place son adresse dans **pProfileData* et la remplit avec un profil de lien d'appareil. Si la fonction réussit, l'application appelante doit libérer la mémoire tampon une fois qu'elle n'est plus nécessaire.

`indexPreferredCMM`

Spécifie l'index de base unique du profil de couleur qui indique le module de gestion des couleurs (CMM) à utiliser. Le développeur d'applications peut autoriser Windows à choisir la gestion des applications en définissant ce paramètre sur INDEX_DONT_CARE. Consultez Utilisation de modules de gestion des couleurs (CMM).

Valeur retournée

Si cette fonction réussit, la valeur de retour est une valeur différente de zéro.

Si cette fonction échoue, la valeur de retour est zéro. Pour obtenir des informations d'erreur étendues, appelez GetLastError.

Notes

Pour les HPROFILEs qui contiennent des informations de profil WCS, les HPROFILEs sont convertis en handles de profil ICC valides, puis ces handles de profil ICC sont utilisés pour créer le profil de liaison d'appareil.

Le premier et le dernier profil du tableau doivent être des profils d'appareil. Les autres profils peuvent être des espaces de couleurs ou des profils abstraits.

L'espace de couleur de sortie de chaque profil doit être l'espace de couleur d'entrée du profil suivant.

L'application appelante doit libérer la mémoire tampon allouée par cette fonction et pointée vers le paramètre *pProfileData* . La fonction **GlobalFree** doit être utilisée pour libérer la mémoire tampon.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- GlobalFree

Commentaires

CreateMultiProfileTransform, fonction (icm.h)

Article24/08/2023

Accepte un tableau de profils ou un profil de lien d'appareil unique et crée une transformation de couleur que les applications peuvent utiliser pour effectuer le mappage des couleurs.

Syntaxe

```
C++

HTRANSFORM CreateMultiProfileTransform(
  PHPROFILE pahProfiles,
  DWORD     nProfiles,
  PDWORD    padwIntent,
  DWORD     nIntents,
  DWORD     dwFlags,
  DWORD     indexPreferredCMM
);
```

Paramètres

`pahProfiles`

Pointeur vers un tableau de handles vers les profils à utiliser. La fonction détermine si les HPROFILEs contiennent des informations de profil ICC (International Color Consortium) ou Windows Color System (WCS) et les traite de manière appropriée. Lorsque des profils WCS valides sont retournés par **OpenColorProfileW** et **WcsOpenColorProfileW**, ces handles de profil contiennent la combinaison de profils DMP, CAMP et GMMP.

`nProfiles`

Spécifie le nombre de profils dans le tableau. Le maximum est 10.

`padwIntent`

Pointeur vers un tableau d'intentions à utiliser. Chaque intention est l'une des valeurs suivantes :

INTENT_PERCEPTUAL

INTENT_SATURATION

INTENT_RELATIVE_COLORIMETRIC

INTENT_ABSOLUTE_COLORIMETRIC

Les GMMPs sont une généralisation des intentions. Il existe deux sources possibles d'intentions : le profil « destination » et le paramètre de liste d'intentions pour **CreateMultiProfileTransform**. Le terme « destination » n'est pas utilisé, car tous les profils du paramètre de liste de profils, sauf deux, serviront de première destination, puis de source.

Pour plus d'informations, consultez Rendu des intentions.

`nIntents`

Spécifie le nombre d'éléments dans le tableau d'intentions : peut être 1 ou la même valeur que *nProfiles*. Pour les tableaux de profils qui contiennent des profils WCS, la première intention de rendu est ignorée et seuls *les éléments nProfiles* -1 sont utilisés pour ces tableaux de profils. Le nombre maximal de *nIntents* est de 10.

`dwFlags`

Spécifie les indicateurs utilisés pour contrôler la création de la transformation. Consultez la section Notes.

`indexPreferredCMM`

Spécifie l'index de base unique du profil de couleur qui indique le module de gestion des couleurs (CMM) à utiliser. Le développeur d'applications peut autoriser Windows à choisir la gestion des applications en définissant ce paramètre sur INDEX_DONT_CARE. Consultez Utilisation de modules de gestion des couleurs (CMM) Les machines virtuelles tierces sont uniquement disponibles pour les flux de travail ICC. Les tableaux de profils contenant des profils WCS ignorent cet indicateur. Il est également ignoré lorsque seuls les profils ICC sont utilisés et lorsque l'indicateur WCS_ALWAYS est utilisé.

Valeur retournée

Si cette fonction réussit, la valeur de retour est un handle de la transformation de couleur.

Si cette fonction échoue, la valeur de retour est **NULL**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Notes

Si un profil de lien d'appareil est utilisé, la fonction échoue si *nProfiles n'est* pas défini sur 1.

Le tableau d'intentions spécifie la façon dont les profils doivent être combinés. La *n*ième intention est utilisée pour combiner le *nième*profil dans le tableau. Si une seule intention est spécifiée, elle est utilisée pour le premier profil et tous les autres profils sont combinés à l'aide de l'intention de correspondance.

Les valeurs dans *dwFlags* sont destinées uniquement à des indicateurs. Le module de gestion des couleurs doit déterminer la meilleure façon de les utiliser.

Windows Vista : trois nouveaux indicateurs ont été ajoutés et peuvent être utilisés avec *dwFlags* :

Indicateur	Description
PRESERVEBLACK	Si ce bit est défini, le moteur de transformation insère le GMMP de génération noire approprié comme dernier GMMP dans la séquence de transformation. Cet indicateur fonctionne uniquement dans une transformation WCS pure.
SEQUENTIAL_TRANSFORM	Si ce bit est défini, chaque étape du pipeline de traitement WCS est effectuée pour chaque pixel de l'image et aucune transformation de couleur optimisée n'est générée. Cet indicateur fonctionne uniquement dans une transformation WCS pure. **Restrictions** : une transformation créée avec le jeu d'indicateurs SEQUENTIAL_TRANSFORM ne peut être utilisée que dans le thread sur lequel elle a été créée et uniquement pour un seul appel de traduction de couleur à la fois. COM doit être initialisé avant de créer la transformation séquentielle et doit rester initialisé pendant toute la durée de vie de l'objet transform.
WCS_ALWAYS	Si ce bit est défini, même les transformations toutes ICC utilisent le chemin de code WCS.

> ⓘ **Notes**
>
> SEQUENTIAL_TRANSFORM a été omis par inadvertance de l'en-tête icm.h dans le Kit de développement logiciel (SDK) Windows Vista. Si vous souhaitez utiliser l'indicateur SEQUENTIAL_TRANSFORM, définissez-le dans votre application comme suit :
>
> #define SEQUENTIAL_TRANSFORM 0x80800000

Pour plus d'informations, consultez Indicateurs de création de transformation CMM. Tous les indicateurs mentionnés sont pris en charge pour tous les types de transformations, à l'exception des FAST_TRANSLATE et des USE_RELATIVE_COLORIMETRIC, qui fonctionnent uniquement dans une transformation ICC-to-ICC pure.

La fonction **CreateMultiProfileTransform** est utilisée en dehors d'un contexte d'appareil. Les couleurs peuvent changer lors de la transformation d'un profil de couleur vers le même profil de couleur. Cela est dû à des erreurs de précision. Par conséquent, une transformation de couleur ne doit pas être effectuée dans ces circonstances.

Nous recommandons qu'il n'y ait qu'un seul GMMP entre un DMP source et de destination. Les descriptions de limites de gamut (GBD) sont créées à partir des combinaisons DMP/CAMP. Les GMMP suivants utilisent les GDB antérieurs dans la chaîne de traitement jusqu'à ce qu'il existe un GBD DMP/CAMP suivant dans la séquence à utiliser. Par exemple, supposons une séquence DMP1, CAMP1, GMMP1, GMMP2, GMMP3, DMP2, CAMP2, GMMP4, GMMP5, CAMP3, DMP3. Ensuite, GMMP1, GMMP2 utilisent GBD1 comme source et destination. Ensuite, GMMP3 utilise GBD1 comme source et GBD2 comme destination. Ensuite, GMMP4 utilise GBD2 comme source et destination. Enfin, GMMP5 utilise GBD2 comme source et GBD3 comme destination. Cela suppose qu'aucun GMMP n'est identique à un gmmp à côté.

Pour les profils WCS, nous vous recommandons de définir les intentions de rendu sur DWORD_MAX afin d'utiliser le GMMP dans le handle de profil WCS. En effet, le tableau d'intentions de rendu est prioritaire sur les intentions de rendu ou les modèles de mappage de gamut spécifiés ou contenus dans les profils spécifiés par les HPROFILEs. Le tableau d'intentions de rendu fait référence au GMMP par défaut pour ces intentions de rendu. Dans l'idéal, un seul mappage de gamuts est effectué entre un appareil source et un appareil de destination en définissant l'un ou l'autre GMMP sur **NULL** lors de la création du fichier HPROFILE avec les informations de profil WCS. Toute application héritée qui utilise un DMP WCS appelle une séquence de GMMPs. Les gdBs sont choisis en fonction des DPM et des CAMPs. Pour les limites de gamut GMMP intermédiaires, les GBD source et de destination sont utilisés.

En résumé, si *nIntents* == 1, le premier GMM est défini en fonction du GMMP défini par défaut* pour la valeur *padwIntent* , sauf si cette valeur est DWORD_MAX, auquel cas les informations GMM incorporées du deuxième profil sont utilisées (les informations GMM incorporées sont soit un GMMP, soit, dans le cas d'un profil ICC, GMM de base correspondant à** l'intention de l'en-tête de profil). Le reste des gmms sont définis en fonction du GMMP défini comme valeur par défaut* pour RelativeColorimetric.

Si *nIntents* = *nProfiles* -1, chaque GMM est défini en fonction du GMMP défini comme valeur par défaut* pour la valeur dans le tableau *padwIntent* à l'index correspondant, sauf si les valeurs *padwIntent* sont DWORD_MAX. Pour les valeurs du tableau *padwIntent* qui sont DWORD_MAX, les MMM aux positions correspondantes sont définies en fonction des informations GMM incorporées du deuxième des deux profils dont les gamuts sont mappés par le GMM. (Là encore, les informations GMM incorporées sont un GMMP ou, dans le cas d'un profil ICC, le GMM de base correspondant à** l'intention de l'en-tête du profil).

Si *nIntents* = *nProfiles*, la première intention est ignorée et la fonction se comporte comme dans le cas où *nIntents* = *nProfiles* -1.

Toute autre combinaison de *padwIntents* et *nIntents* retourne une erreur.

* « set as default » signifie que le GMMP par défaut est interrogé à l'aide **de WcsGetDefaultColorProfile** avec son paramètre *profileManagementScope* défini sur WCS_PROFILE_MANAGEMENT_SCOPE_CURRENT_USER. Cela peut retourner des valeurs par défaut utilisateur actuel ou système, comme décrit dans la documentation de **WcsGetDefaultColorProfile**.

** « GMM correspondant à » ne signifie pas « GMM du GMMP défini par défaut pour ». Au lieu de cela, cela signifie « une association constante entre les intentions de profil ICC et les algorithmes GMM de base ».

La prise en charge de la transformation WCS pour les profils Icc ColorSpace est limitée aux profils d'espace de couleurs RVB. Les types de profils ICC suivants ne peuvent pas être utilisés dans une transformation traitée CITE, soit une transformation MIXTE WCS/ICC, soit une transformation icc avec **WCS_ALWAYS** défini :

- Profils ColorSpace non RVB
- Profils NamedColor
- profils n-canal (où n8 >)
- Profils DeviceLink
- Profils abstraits

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en	Windows 2000 Server [applications de bureau uniquement]

charge	
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- COLOR, structure**
- DeleteColorTransform

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

CreateProfileFromLogColorSpaceA, fonction (icm.h)

Article24/08/2023

Convertit un espace de couleur logique en profil d'appareil.

Syntaxe

```
C++

BOOL CreateProfileFromLogColorSpaceA(
  LPLOGCOLORSPACEA pLogColorSpace,
  PBYTE            *pProfile
);
```

Paramètres

`pLogColorSpace`

Pointeur vers une structure d'espace de couleurs logique. Pour plus **d'informations, consultez LOGCOLORSPACEA** . Le membre **lcsFilename** [0] de la structure doit être défini sur le caractère **null** ('\0') sinon cet appel de fonction échoue avec la valeur de retour de INVALID_PARAMETER.

`pProfile`

Pointeur vers un pointeur vers une mémoire tampon où le profil d'appareil sera créé. Cette fonction alloue la mémoire tampon et la remplit d'informations de profil si elle réussit. Si ce n'est pas le cas, le pointeur a la valeur **NULL**. L'appelant est responsable de libérer cette mémoire tampon quand elle n'est plus nécessaire.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**.

Si le membre **lcsFilename** [0] si la structure **LOGCOLORSPACEA** pointée par *pLogColorSpace* n'est pas « \0 », cette fonction retourne INVALID_PARAMETER.

Notes

Cette fonction peut être utilisée avec des chaînes ASCII ou Unicode. La mémoire tampon créée par cette fonction doit être libérée par l'appelant quand elle n'est plus nécessaire, sinon il y aura une fuite de mémoire. La fonction GlobalFree doit être utilisée pour libérer cette mémoire tampon.

Cette fonction ne prend pas en charge les profils WINDOWS Color System (WCS) CAMP, DMP et GMMP.

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- GlobalFree

Commentaires

CreateProfileFromLogColorSpaceW, fonction (icm.h)

Article02/03/2024

Convertit un espace de couleur logique en profil d'appareil.

Syntaxe

```
C++

BOOL CreateProfileFromLogColorSpaceW(
  LPLOGCOLORSPACEW pLogColorSpace,
  PBYTE            *pProfile
);
```

Paramètres

`pLogColorSpace`

Pointeur vers une structure d'espace de couleurs logique. Pour plus **d'informations, consultez LOGCOLORSPACEA** . Le membre **lcsFilename** [0] de la structure doit être défini sur le caractère **null** ('\0') sinon cet appel de fonction échoue avec la valeur de retour de INVALID_PARAMETER.

`pProfile`

Pointeur vers un pointeur vers une mémoire tampon où le profil d'appareil sera créé. Cette fonction alloue la mémoire tampon et la remplit d'informations de profil si elle réussit. Si ce n'est pas le cas, le pointeur a la valeur **NULL**. L'appelant est responsable de libérer cette mémoire tampon quand elle n'est plus nécessaire.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**.

Si le membre **lcsFilename** [0] si la structure **LOGCOLORSPACEA** pointée par *pLogColorSpace* n'est pas « \0 », cette fonction retourne INVALID_PARAMETER.

Remarques

Cette fonction peut être utilisée avec des chaînes ASCII ou Unicode. La mémoire tampon créée par cette fonction doit être libérée par l'appelant quand elle n'est plus nécessaire, sinon il y aura une fuite de mémoire. La fonction GlobalFree doit être utilisée pour libérer cette mémoire tampon.

Cette fonction ne prend pas en charge les profils WINDOWS Color System (WCS) CAMP, DMP et GMMP.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- GlobalFree

Commentaires

DeleteColorSpace, fonction (wingdi.h)

Article27/08/2023

La fonction **DeleteColorSpace** supprime et détruit un espace de couleurs spécifié.

Syntaxe

```
C++

BOOL DeleteColorSpace(
  HCOLORSPACE hcs
);
```

Paramètres

`hcs`

Spécifie le handle dans un espace de couleurs à supprimer.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**.

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
Plateforme cible	Windows
En-tête	wingdi.h
Bibliothèque	Gdi32.lib
DLL	Gdi32.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

DeleteColorTransform, fonction (icm.h)

Article24/08/2023

Supprime une transformation de couleur donnée.

Syntaxe

```
C++

BOOL DeleteColorTransform(
  HTRANSFORM hxform
);
```

Paramètres

`hxform`

Identifie la transformation de couleur à supprimer.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- Structure COLOR
- CreateMultiProfileTransform

Commentaires

DisassociateColorProfileFromDeviceA, fonction (icm.h)

Article24/08/2023

Dissocie un profil de couleur spécifié avec un appareil spécifié sur un ordinateur spécifié.

> ⓘ **Notes**
>
> Cette API ne prend pas en charge les profils de couleur avancés pour les moniteurs HDR. Utilisez **ColorProfileRemoveDisplayAssociation** pour gérer les profils de couleurs avancés.

Syntaxe

```
C++

BOOL DisassociateColorProfileFromDeviceA(
  PCSTR pMachineName,
  PCSTR pProfileName,
  PCSTR pDeviceName
);
```

Paramètres

`pMachineName`

Réservé. Doit être **NULL**. Ce paramètre est destiné à pointer vers le nom de l'ordinateur sur lequel dissocier le profil et l'appareil spécifiés. Un pointeur **NULL** indique l'ordinateur local.

`pProfileName`

Pointeur vers le nom de fichier du profil à dissocier.

`pDeviceName`

Pointeur vers le nom de l'appareil à dissocier.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Notes

Si plusieurs profils sont associés à un appareil, WCS utilise le dernier profil associé comme profil par défaut. Autrement dit, si votre application associe séquentiellement trois profils à un appareil, WCS utilise le dernier associé comme profil par défaut. Si votre application appelle ensuite la fonction **DisassociateColorProfileFromDevice** pour dissocier le troisième profil (qui est la valeur par défaut dans cet exemple), wcs utilise le deuxième profil comme profil par défaut.

Si votre application dissocie tous les profils d'un appareil, WCS utilise le profil sRGB comme profil par défaut.

DisassociateColorProfileFromDevice supprime toujours le profil spécifié de la liste d'association de profils par utilisateur de l'utilisateur actuel pour l'appareil spécifié. Avant de supprimer le profil de la liste, **DisassociateColorProfileFromDevice** détermine si l'utilisateur a précédemment exprimé le désir d'utiliser une liste d'association de profil par utilisateur pour l'appareil. Dans ce cas, **DisassociateColorProfileFromDevice** supprime simplement le profil spécifié de la liste d'association de profil par utilisateur existante pour l'appareil. Si ce n'est pas le cas, **DisassociateColorProfileFromDevice** crée une liste d'association de profil par utilisateur pour l'appareil en copiant la liste d'association à l'échelle du système pour cet appareil. Il supprime ensuite le profil spécifié de la liste par utilisateur. À partir de ce point, l'utilisateur actuel utilise une liste d'association de profil par utilisateur pour l'appareil spécifié, comme si WcsSetUsePerUserProfiles avait été appelé pour *pDevice* avec le paramètre *usePerUserProfiles* défini sur **TRUE**.

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- AssociateColorProfileWithDeviceA

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

Fonction DisassociateColorProfileFromDeviceW (icm.h)

Article24/08/2023

Dissocie un profil de couleur spécifié avec un appareil spécifié sur un ordinateur spécifié.

> **Notes**
>
> Cette API ne prend pas en charge les profils de « couleur avancée » pour les moniteurs HDR. Utilisez **ColorProfileRemoveDisplayAssociation** pour gérer les profils de couleurs avancés.

Syntaxe

```
C++

BOOL DisassociateColorProfileFromDeviceW(
  PCWSTR pMachineName,
  PCWSTR pProfileName,
  PCWSTR pDeviceName
);
```

Paramètres

`pMachineName`

Réservé. Doit avoir la **valeur NULL**. Ce paramètre est destiné à pointer vers le nom de l'ordinateur sur lequel dissocier le profil et l'appareil spécifiés. Un pointeur **NULL** indique l'ordinateur local.

`pProfileName`

Pointeur vers le nom de fichier du profil à dissocier.

`pDeviceName`

Pointeur vers le nom de l'appareil à dissocier.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Notes

Si plusieurs profils sont associés à un appareil, WCS utilise le dernier associé comme profil par défaut. Autrement dit, si votre application associe séquentiellement trois profils à un appareil, WCS utilise le dernier associé comme valeur par défaut. Si votre application appelle ensuite la fonction **DisassociateColorProfileFromDevice** pour dissocier le troisième profil (qui est la valeur par défaut dans cet exemple), le WCS utilise le deuxième profil comme profil par défaut.

Si votre application dissocie tous les profils d'un appareil, WCS utilise le profil sRGB comme profil par défaut.

DisassociateColorProfileFromDevice supprime toujours le profil spécifié de la liste d'associations de profils par utilisateur de l'utilisateur actuel pour l'appareil spécifié. Avant de supprimer le profil de la liste, **DisassociateColorProfileFromDevice** détermine si l'utilisateur a déjà exprimé le désir d'utiliser une liste d'associations de profils par utilisateur pour l'appareil. Si c'est le cas, **DisassociateColorProfileFromDevice** supprime simplement le profil spécifié de la liste d'associations de profils par utilisateur existante pour l'appareil. Si ce n'est pas le cas, **DisassociateColorProfileFromDevice** crée une liste d'associations de profils par utilisateur pour l'appareil en copiant la liste d'associations à l'échelle du système pour cet appareil. Il supprime ensuite le profil spécifié de la liste par utilisateur. À partir de ce point, l'utilisateur actuel utilise une liste d'association de profils par utilisateur pour l'appareil spécifié, comme si **WcsSetUsePerUserProfiles** avait été appelé pour *pDevice* avec le paramètre *usePerUserProfiles* défini sur **TRUE**.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]

En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- AssociateColorProfileWithDeviceW

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

Fonction EnumColorProfilesA (icm.h)

Article24/08/2023

Énumère tous les profils qui satisfont aux critères d'énumération donnés.

Syntaxe

```
C++

BOOL EnumColorProfilesA(
  PCSTR      pMachineName,
  PENUMTYPEA pEnumRecord,
  PBYTE      pEnumerationBuffer,
  PDWORD     pdwSizeOfEnumerationBuffer,
  PDWORD     pnProfiles
);
```

Paramètres

`pMachineName`

Réservé. Doit avoir la **valeur NULL**. Ce paramètre est destiné à pointer vers le nom de l'ordinateur sur lequel énumérer les profils. Un pointeur **NULL** indique l'ordinateur local.

`pEnumRecord`

Pointeur vers une structure spécifiant les critères d'énumération.

`pEnumerationBuffer`

Pointeur vers une mémoire tampon dans laquelle les profils doivent être énumérés. Une MULTI_SZ chaîne de noms de profil répondant aux critères spécifiés dans **pEnumRecord* sera placée dans cette mémoire tampon.

`pdwSizeOfEnumerationBuffer`

Pointeur vers une variable contenant la taille de la mémoire tampon pointée par *pBuffer*. Au retour, **pdwSize* contient la taille de la mémoire tampon réellement utilisée ou nécessaire.

`pnProfiles`

Pointeur vers une variable qui contiendra, au retour, le nombre de noms de profil réellement copiés dans la mémoire tampon.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, appelez GetLastError.

Notes

Plusieurs profils sont généralement associés aux imprimantes, en fonction des types de papier et d'encre. Il existe un profil par défaut pour chaque appareil. Pour les profils ICC (International Color Consortium), GDI sélectionne le meilleur parmi les profils icc associés lorsque votre application crée un contexte d'appareil (DC).

N'essayez pas d'utiliser **EnumColorProfiles** pour déterminer le profil par défaut d'un appareil. Au lieu de cela, créez un contexte d'appareil pour l'appareil, puis appelez la fonction **GetICMProfile** . Sur Windows Vista et Windows 7, la fonction **WcsGetDefaultColorProfile** peut également être utilisée pour déterminer le profil de couleur par défaut d'un appareil.

Si le membre **dwFields** de la structure de type **ENUMTYPE** pointé par le paramètre *pEnumRecord* est défini sur ET_DEVICENAME, cette fonction énumère tous les profils de couleur associés à tous les types d'appareils attachés à l'ordinateur de l'utilisateur, quelle que soit la classe d'appareil. Si le membre **dwFields** de la structure pointée vers le paramètre *pEnumRecord* est défini sur ET_DEVICENAME ou ET_DEVICECLASS et qu'une classe d'appareil est spécifiée dans le membre **dwDeviceClass** de la structure, cette fonction énumère uniquement les profils associés à la classe d'appareil spécifiée. Si le membre **dwFields** est défini uniquement sur ET_DEVICECLASS, la fonction **EnumColorProfiles** énumère tous les profils qui peuvent être associés à ce type d'appareil.

Chaque fois **qu'EnumColorProfiles** examine les profils associés à un appareil spécifique, les résultats varient selon que l'utilisateur a choisi d'utiliser la liste à l'échelle du système des profils associés à cet appareil ou sa propre liste (« par utilisateur »). L'appel **de WcsSetUsePerUserProfiles** avec son paramètre *usePerUserProfiles* défini sur **TRUE** entraîne les appels futurs à **EnumColorProfiles** à examiner uniquement la liste des associations de profils par utilisateur de l'utilisateur actuel pour l'appareil spécifié. L'appel **de WcsSetUsePerUserProfiles** avec son paramètre *usePerUserProfiles* défini sur

FALSE entraîne les appels futurs à **EnumColorProfiles** à examiner la liste à l'échelle du système des associations de profils pour l'appareil spécifié. Si **WcsSetUsePerUserProfiles** n'a jamais été appelé pour l'utilisateur actuel, **EnumColorProfiles** examine la liste à l'échelle du système.

Votre application peut utiliser **EnumColorProfiles** pour obtenir la taille de la mémoire tampon dans laquelle les profils sont énumérés. Il doit appeler la fonction **EnumColorProfiles** avec le paramètre *pBuffer* défini sur **NULL**. Lorsque la fonction retourne, le paramètre *pdwSize* contient la taille de mémoire tampon requise en octets. Votre programme peut utiliser ces informations pour allouer la mémoire tampon d'énumération. Il peut ensuite appeler **à nouveau EnumColorProfiles** avec le paramètre *pBuffer* défini sur l'adresse de la mémoire tampon.

Cette fonction fournit les informations permettant de convertir des informations DMP spécifiques à WCS en l'enregistrement EnumType hérité dans activer l'énumération de profil cohérente. Les valeurs par défaut seront identiques à ICC si ces informations ne sont pas présentes.

Prise en charge par utilisateur/LUA

L'énumération est spécifique à l'utilisateur actuel. Les associations d'appareils utilisateur actuelles et à l'échelle du système sont prises en compte. Pour la configuration du profil par défaut, les paramètres utilisateur actuels remplacent ceux à l'échelle du système.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- GetICMProfile
- ENUMTYPEW

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l’aide sur Microsoft Q&A

EnumColorProfilesW, fonction (icm.h)

Article24/08/2023

Énumère tous les profils répondant aux critères d'énumération donnés.

Syntaxe

C++

```
BOOL EnumColorProfilesW(
  PCWSTR     pMachineName,
  PENUMTYPEW pEnumRecord,
  PBYTE      pEnumerationBuffer,
  PDWORD     pdwSizeOfEnumerationBuffer,
  PDWORD     pnProfiles
);
```

Paramètres

`pMachineName`

Réservé. Doit être **NULL**. Ce paramètre est destiné à pointer vers le nom de l'ordinateur sur lequel énumérer les profils. Un pointeur **NULL** indique l'ordinateur local.

`pEnumRecord`

Pointeur vers une structure spécifiant les critères d'énumération.

`pEnumerationBuffer`

Pointeur vers une mémoire tampon dans laquelle les profils doivent être énumérés. Une MULTI_SZ chaîne de noms de profil répondant aux critères spécifiés dans **pEnumRecord* sera placée dans cette mémoire tampon.

`pdwSizeOfEnumerationBuffer`

Pointeur vers une variable contenant la taille de la mémoire tampon pointée par *pBuffer*. Au retour, **pdwSize* contient la taille de la mémoire tampon réellement utilisée ou nécessaire.

`pnProfiles`

Pointeur vers une variable qui contiendra, au retour, le nombre de noms de profil réellement copiés dans la mémoire tampon.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, appelez GetLastError.

Notes

Plusieurs profils sont généralement associés aux imprimantes, en fonction des types de papier et d'encre. Il existe un profil par défaut pour chaque appareil. Pour les profils ICC (International Color Consortium), GDI sélectionne le meilleur parmi les profils associés à l'ICC lorsque votre application crée un contexte d'appareil (DC).

N'essayez pas d'utiliser **EnumColorProfiles** pour déterminer le profil par défaut d'un appareil. Au lieu de cela, créez un contexte d'appareil pour l'appareil, puis appelez la fonction **GetICMProfile** . Sur Windows Vista et Windows 7, la fonction **WcsGetDefaultColorProfile** peut également être utilisée pour déterminer le profil de couleur par défaut d'un appareil.

Si le membre **dwFields** de la structure de type **ENUMTYPE** vers laquelle pointe le paramètre *pEnumRecord* est défini sur ET_DEVICENAME, cette fonction énumère tous les profils de couleurs associés à tous les types d'appareils attachés à l'ordinateur de l'utilisateur, quelle que soit la classe d'appareil. Si le membre **dwFields** de la structure pointé par le paramètre *pEnumRecord* est défini sur ET_DEVICENAME ou ET_DEVICECLASS et qu'une classe d'appareil est spécifiée dans le membre **dwDeviceClass** de la structure, cette fonction énumère uniquement les profils associés à la classe d'appareil spécifiée. Si le membre **dwFields** est défini uniquement sur ET_DEVICECLASS, la fonction **EnumColorProfiles** énumère tous les profils qui peuvent être associés à ce type d'appareil.

Chaque fois **qu'EnumColorProfiles** examine les profils associés à un appareil spécifique, les résultats varient selon que l'utilisateur a choisi d'utiliser la liste à l'échelle du système des profils associés à cet appareil, ou sa propre liste (« par utilisateur »). L'appel **de WcsSetUsePerUserProfiles** avec son paramètre *usePerUserProfiles* défini sur **TRUE** entraîne les appels futurs à **EnumColorProfiles** pour examiner uniquement la liste des associations de profil par utilisateur de l'utilisateur actuel pour l'appareil spécifié. L'appel **de WcsSetUsePerUserProfiles** avec son paramètre *usePerUserProfiles* défini sur **FALSE**

entraîne les futurs appels à **EnumColorProfiles** d'examiner la liste à l'échelle du système des associations de profils pour l'appareil spécifié. Si **WcsSetUsePerUserProfiles** n'a jamais été appelé pour l'utilisateur actuel, **EnumColorProfiles** examine la liste à l'échelle du système.

Votre application peut utiliser **EnumColorProfiles** pour obtenir la taille de la mémoire tampon dans laquelle les profils sont énumérés. Il doit appeler la fonction **EnumColorProfiles** avec le paramètre *pBuffer* défini sur **NULL**. Lorsque la fonction est retournée, le paramètre *pdwSize* contient la taille de mémoire tampon requise en octets. Votre programme peut utiliser ces informations pour allouer la mémoire tampon d'énumération. Il peut ensuite appeler **à nouveau EnumColorProfiles** avec le paramètre *pBuffer* défini sur l'adresse de la mémoire tampon.

Cette fonction fournit les informations permettant de convertir des informations DMP spécifiques à WCS en enregistrement EnumType hérité dans activer l'énumération de profil cohérente. Les valeurs par défaut sont identiques à icc si ces informations ne sont pas présentes.

Prise en charge par utilisateur/LUA

L'énumération est spécifique à l'utilisateur actuel. Les associations d'appareils utilisateur actuels et à l'échelle du système sont prises en compte. Pour la configuration de profil par défaut, les paramètres utilisateur actuels remplacent ceux à l'échelle du système.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- GetICMProfile
- ENUMTYPEW

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l’aide sur Microsoft Q&A

EnumICMProfilesA, fonction (wingdi.h)

Article27/08/2023

La fonction **EnumICMProfiles** énumère les différents profils de couleur de sortie pris en charge par le système pour un contexte d'appareil donné.

Syntaxe

```
C++

int EnumICMProfilesA(
  HDC          hdc,
  ICMENUMPROCA proc,
  LPARAM       param
);
```

Paramètres

`hdc`

Spécifie le contexte de l'appareil.

`proc`

Spécifie la procédure instance adresse d'une fonction de rappel définie par l'application. (Voir EnumICMProfilesProcCallback.)

`param`

Données fournies par l'application qui sont passées à la fonction de rappel, ainsi que les informations de profil de couleur.

Valeur retournée

Cette fonction retourne zéro si l'application a interrompu l'énumération. La valeur de retour est -1 s'il n'existe aucun profil de couleur à énumérer. Sinon, la valeur de retour est la dernière valeur retournée par la fonction de rappel.

Remarques

La fonction **EnumICMProfiles** retourne une liste de profils associés à un contexte d'appareil (DC) et dont les paramètres correspondent à ceux du contrôleur de domaine. Il est possible qu'un contexte d'appareil contienne des profils d'appareil qui ne sont pas associés à des périphériques matériels particuliers, ou des profils d'appareil qui ne correspondent pas aux paramètres du contrôleur de domaine. Le profil sRGB en est un exemple. La fonction SetICMProfile est utilisée pour associer ces types de profils à un contrôleur de domaine. La fonction GetICMProfile peut être utilisée pour récupérer un profil qui n'est pas énuméré par la fonction **EnumICMProfiles** .

Windows 95/98/Me:EnumICMProfilesW est pris en charge par Microsoft Layer pour Unicode. Pour l'utiliser, vous devez ajouter certains fichiers à votre application, comme indiqué dans Microsoft Layer pour Unicode sur les systèmes Windows 95/98/Me .

> **Notes**
>
> L'en-tête wingdi.h définit EnumICMProfiles en tant qu'alias qui sélectionne automatiquement la version ANSI ou Unicode de cette fonction en fonction de la définition de la constante de préprocesseur UNICODE. La combinaison de l'utilisation de l'alias neutre en encodage avec du code qui n'est pas neutre en encodage peut entraîner des incompatibilités qui entraînent des erreurs de compilation ou d'exécution. Pour plus d'informations, consultez **Conventions pour les prototypes de fonction**.

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
Plateforme cible	Windows
En-tête	wingdi.h
Bibliothèque	Gdi32.lib
DLL	Gdi32.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- DeleteColorSpaceW
- Fonction de rappel ICMENUMPROCA
- GetICMProfileW
- SetICMProfileW

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

GetCMMInfo, fonction (icm.h)

Article26/02/2024

Récupère diverses informations sur le module de gestion des couleurs (CMM) qui a créé la transformation de couleur spécifiée.

Syntaxe

```
C++

DWORD GetCMMInfo(
  HTRANSFORM hColorTransform,
  DWORD      unnamedParam2
);
```

Paramètres

`hColorTransform`

Identifie la transformation pour laquelle rechercher des informations sur la gestion des applications de gestion des applications cloud.

`unnamedParam2`

Spécifie les informations à récupérer. Ce paramètre peut prendre l'une des valeurs constantes suivantes.

Agrandir le tableau

Valeur	Signification
CMM_WIN_VERSION	Récupère la version de Windows ciblée par le module de gestion des couleurs (CMM).
CMM_DLL_VERSION	Récupère le numéro de version de la CMM.
CMM_IDENT	Récupère la signature CMM enregistrée auprès de l'International Color Consortium (ICC).

Valeur retournée

Si cette fonction réussit, la valeur de retour correspond aux informations spécifiées dans *dwInfo*.

Si cette fonction échoue, la valeur de retour est zéro.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

Cette page a-t-elle été utile ? Yes No

Indiquer des commentaires sur le produit | Obtenir de l'aide sur Microsoft Q&A

GetColorDirectoryA, fonction (icm.h)

Article11/03/2023

> **Notes**
>
> Cette API peut ne pas être disponible dans les versions ultérieures. Nous encourageons les logiciels nouveaux et existants à utiliser d'autres API pour les interactions de profil de couleur. Reportez-vous au tableau ci-dessous pour obtenir quelques exemples.

Scénario	Mechanism
Énumération de tous les profils installés	Utiliser **WcsEnumColorProfilesSize** et **WcsEnumColorProfiles**, ou **EnumColorProfilesA**
Installation/désinstallation des profils de couleur	Utiliser **InstallColorProfileA**/**UninstallColorProfileA**
Ouverture directe d'un fichier de profil de couleur	Utilisez **OpenColorProfileA** avec dwType=PROFILE_FILENAME dans le paramètre de struct PROFILE. Ou utilisez **WcsOpenColorProfileA**. **Icm.h** contient de nombreuses API qui acceptent le fichier HPROFILE retourné pour la manipulation du profil de couleur

Récupère le chemin d'accès du répertoire Windows COLOR sur un ordinateur spécifié.

Syntaxe

C++

```
BOOL GetColorDirectoryA(
  PCSTR  pMachineName,
  PSTR   pBuffer,
  PDWORD pdwSize
);
```

Paramètres

`pMachineName`

Réservés au; doit être **NULL**. Ce paramètre est destiné à pointer vers le nom de l'ordinateur sur lequel le profil doit être installé. Un pointeur **NULL** indique l'ordinateur local.

`pBuffer`

Pointe vers la mémoire tampon dans laquelle le chemin du répertoire de couleurs doit être placé.

`pdwSize`

Pointe vers une variable contenant la taille en octets de la mémoire tampon pointée par *pBuffer*. Au retour, la variable contient la taille de la mémoire tampon réellement utilisée ou nécessaire.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Remarques

Prise en charge par utilisateur/LUA

Le répertoire de couleurs est toujours à l'échelle du système. Cette fonction est exécutable dans le contexte LUA.

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

GetColorDirectoryW, fonction (icm.h)

Article29/02/2024

> **Notes**
>
> Cette API peut ne pas être disponible dans les versions ultérieures. Nous encourageons les logiciels nouveaux et existants à utiliser d'autres API pour les interactions de profil de couleur. Reportez-vous au tableau ci-dessous pour obtenir quelques exemples.

Agrandir le tableau

Scénario	Mécanisme
Énumération de tous les profils installés	Utiliser WcsEnumColorProfilesSize et WcsEnumColorProfiles, ou EnumColorProfilesW
Installation/désinstallation des profils de couleur	Utiliser InstallColorProfileW/UninstallColorProfileW
Ouverture directe d'un fichier de profil de couleur	Utilisez OpenColorProfileW avec dwType=PROFILE_FILENAME dans le paramètre de struct PROFILE. Ou utilisez WcsOpenColorProfileW. Icm.h contient de nombreuses API qui acceptent le fichier HPROFILE retourné pour la manipulation du profil de couleur

Récupère le chemin d'accès du répertoire Windows COLOR sur un ordinateur spécifié.

Syntaxe

C++

```
BOOL GetColorDirectoryW(
  PCWSTR pMachineName,
  PWSTR  pBuffer,
  PDWORD pdwSize
);
```

Paramètres

`pMachineName`

Réservés au; doit être **NULL**. Ce paramètre est destiné à pointer vers le nom de l'ordinateur sur lequel le profil doit être installé. Un pointeur **NULL** indique l'ordinateur local.

`pBuffer`

Pointe vers la mémoire tampon dans laquelle le chemin du répertoire de couleurs doit être placé.

`pdwSize`

Pointe vers une variable contenant la taille en octets de la mémoire tampon pointée par *pBuffer*. Au retour, la variable contient la taille de la mémoire tampon réellement utilisée ou nécessaire.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Remarques

Prise en charge par utilisateur/LUA

Le répertoire de couleurs est toujours à l'échelle du système. Cette fonction est exécutable dans le contexte LUA.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]

Condition requise	Valeur
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

GetColorProfileElement, fonction (icm.h)

Article24/08/2023

Copie les données d'un élément de profil balisé spécifié d'un profil de couleur spécifié dans une mémoire tampon.

Syntaxe

```
C++

BOOL GetColorProfileElement(
  HPROFILE hProfile,
  TAGTYPE  tag,
  DWORD    dwOffset,
  PDWORD   pcbElement,
  PVOID    pElement,
  PBOOL    pbReference
);
```

Paramètres

`hProfile`

Spécifie un handle pour le profil de couleur ICC (International Color Consortium) en question.

`tag`

Identifie l'élément balisé à partir duquel copier.

`dwOffset`

Spécifie le décalage par rapport au premier octet des données de l'élément balisé à partir duquel commencer la copie.

`pcbElement`

Pointeur vers une variable spécifiant le nombre d'octets à copier. Au retour, la variable contient le nombre d'octets réellement copiés.

`pElement`

Pointeur vers une mémoire tampon dans laquelle les données d'élément balisées doivent être copiées. La mémoire tampon doit contenir au moins autant d'octets que spécifié par la variable pointée par *pcbSize*. Si le pointeur *pBuffer* est défini sur **NULL**, la taille de l'ensemble des données d'élément balisées en octets est retournée à l'emplacement de mémoire pointé par *pcbSize,* et *dwOffset* est ignoré. Dans ce cas, la fonction retourne **FALSE**.

`pbReference`

Pointe vers une valeur booléenne qui a la valeur **TRUE** si plusieurs balises dans le profil de couleur font référence aux mêmes données que la balise spécifiée, ou **FALSE** si ce n'est pas le cas.

Valeur retournée

Si cette fonction réussit, la valeur de retour est différente de zéro.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Notes

Cette fonction échoue si *hProfile* n'est pas un profil ICC (International Color Consortium) valide.

Si le pointeur *pBuffer* est défini sur **NULL**, la taille de la totalité des données d'élément balisées en octets est retournée dans la variable pointée par *pcbSize,* et *dwOffset* est ignoré.

Cette fonction ne prend pas en charge les profils WINDOWS Color System (WCS) CAMP, DMP et GMMP ; Étant donné que les éléments de profil sont implicitement associés aux types de balises ICC et codés en dur, et qu'il existe de nombreuses bibliothèques d'analyse XML robustes.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]

Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

GetColorProfileElementTag, fonction (icm.h)

Article24/08/2023

Récupère le nom de balise spécifié par *dwIndex* dans la table de balises d'un profil de couleur ICC (International Color Consortium) donné, où *dwIndex* est un index de base unique dans cette table.

Syntaxe

```
C++

BOOL GetColorProfileElementTag(
  HPROFILE hProfile,
  DWORD    dwIndex,
  PTAGTYPE pTag
);
```

Paramètres

`hProfile`

Spécifie un handle pour le profil de couleur ICC en question.

`dwIndex`

Spécifie l'index de base unique de la balise à récupérer.

`pTag`

Pointeur vers une variable dans laquelle le nom de la balise doit être placé.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Notes

Cette fonction échoue si *hProfile* n'est pas un profil ICC valide.

GetColorProfileElementTag peut être utilisé pour énumérer toutes les balises d'un profil après avoir obtenu le nombre de balises dans le profil à l'aide **de GetCountColorProfileElements**.

Cette fonction ne prend pas en charge les profils WINDOWS Color System (WCS) CAMP, DMP et GMMP ; Étant donné que les éléments de profil sont implicitement associés aux types de balises ICC et codés en dur, et qu'il existe de nombreuses bibliothèques d'analyse XML robustes.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

GetColorProfileFromHandle, fonction (icm.h)

Article24/08/2023

Avec un handle pour un profil de couleur ouvert, la fonction **GetColorProfileFromHandle** copie le contenu du profil dans une mémoire tampon fournie par l'application. Si le handle est un handle WCS (Windows Color System), le DMP est retourné et les valeurs CAMP et GMMP associées au HPROFILE sont ignorées.

Syntaxe

```
C++

BOOL GetColorProfileFromHandle(
  HPROFILE hProfile,
  PBYTE    pProfile,
  PDWORD   pcbProfile
);
```

Paramètres

`hProfile`

Gérer dans un profil de couleur ouvert. La fonction détermine si le fichier HPROFILE contient des informations de profil ICC ou WCS.

`pProfile`

Pointeur vers la mémoire tampon pour recevoir des données de profil ICC ou DMP brutes. Peut être **NULL**. Si tel est le cas, la taille requise pour la mémoire tampon sera stockée dans l'emplacement de mémoire pointé par *pcbSize*. La mémoire tampon peut être allouée à la taille appropriée, et cette fonction est appelée à nouveau avec *pBuffer* contenant l'adresse de la mémoire tampon.

`pcbProfile`

Pointeur vers un **DWORD** qui contient la taille de la mémoire tampon pointée vers *pBuffer*. Au retour, il est rempli avec la taille de la mémoire tampon qui a été réellement utilisée si la fonction réussit. Si cette fonction est appelée avec *pBuffer* défini sur **NULL**, ce paramètre contient la taille de la mémoire tampon requise.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**. Il retourne **FALSE** si le paramètre *pBuffer* a la valeur **NULL** et que la taille requise pour la mémoire tampon est copiée dans *pcbSize*.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

GetColorProfileHeader, fonction (icm.h)

Article26/02/2024

Récupère ou dérive la structure d'en-tête ICC à partir d'un profil de couleur ICC ou d'un profil XML WCS. Les pilotes et les applications doivent supposer que le retour **de TRUE** indique uniquement qu'un en-tête correctement structuré est retourné. Chaque balise doit toujours être validée indépendamment à l'aide d'API ICM2 héritées ou d'API de schéma XML.

Syntaxe

```
C++

BOOL GetColorProfileHeader(
  HPROFILE       hProfile,
  PPROFILEHEADER pHeader
);
```

Paramètres

`hProfile`

Spécifie un handle pour le profil de couleur en question.

`pHeader`

Pointe vers une variable dans laquelle la structure d'en-tête ICC doit être placée.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Cette fonction échoue si un profil ICC ou WCS XML non valide est référencé dans le paramètre hProfile. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Remarques

Pour déterminer si l'en-tête est dérivé d'un handle de profil ICC ou DMP, case activée la signature d'en-tête (octets d'en-tête 36 à 39). Si la signature est « acsp » (big endian), un

profil ICC a été utilisé. Si la signature est « cdmp » (big-endian), un DMP a été utilisé.

Les caractéristiques distinctives qui identifient un en-tête comme ayant été « synthétisé » pour un DMP WCS sont les suivantes :

pIcmProfileHeader-phSignature> = 'pmdc' (little endian = big endian 'cdmp')

pIcmProfileHeader-phCMMType> = '1scw' (little endian = big endian 'wcs1').

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- PROFILEHEADER

Commentaires

GetColorSpace, fonction (wingdi.h)

Article27/08/2023

La fonction **GetColorSpace** récupère le handle dans l'espace de couleur d'entrée à partir d'un contexte d'appareil spécifié.

Syntaxe

```
C++

HCOLORSPACE GetColorSpace(
  HDC hdc
);
```

Paramètres

`hdc`

Spécifie un contexte d'appareil dont le handle d'espace de couleur d'entrée doit être récupéré.

Valeur retournée

Si la fonction réussit, la valeur de retour est le handle d'espace de couleur d'entrée actuel.

Si cette fonction échoue, la valeur de retour est **NULL**.

Remarques

GetColorSpace obtient le handle de l'espace de couleur d'entrée, que la gestion des couleurs soit activée ou non pour le contexte de l'appareil.

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]

Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
Plateforme cible	Windows
En-tête	wingdi.h
Bibliothèque	Gdi32.lib
DLL	Gdi32.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

GetCountColorProfileElements, fonction (icm.h)

Article24/08/2023

Récupère le nombre d'éléments balisés dans un profil de couleur donné.

Syntaxe

```
C++

BOOL GetCountColorProfileElements(
  HPROFILE hProfile,
  PDWORD   pnElementCount
);
```

Paramètres

`hProfile`

Spécifie un handle pour le profil en question.

`pnElementCount`

Pointeur vers une variable dans laquelle placer le nombre d'éléments balisés dans le profil.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Notes

Cette fonction échoue si *hProfile* n'est pas un profil ICC valide.

Cette fonction ne prend pas en charge les profils WINDOWS Color System (WCS) CAMP, DMP et GMMP.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

GetDeviceGammaRamp, fonction (wingdi.h)

Article27/08/2023

La fonction **GetDeviceGammaRamp** obtient la rampe gamma sur les tableaux d'affichage couleur directs avec des pilotes qui prennent en charge les rampes gamma téléchargeables dans le matériel.

> **ⓘ Important**
>
> Nous vous recommandons vivement de ne pas utiliser cette API. L'utilisation de cette API est soumise à des limitations majeures. Pour plus d'informations, consultez **SetDeviceGammaRamp** .

Syntaxe

```
C++

BOOL GetDeviceGammaRamp(
  HDC    hdc,
  LPVOID lpRamp
);
```

Paramètres

`hdc`

Spécifie le contexte de l'appareil du tableau d'affichage en couleur directe en question.

`lpRamp`

Pointe vers une mémoire tampon où la fonction peut placer la rampe gamma actuelle de la carte d'affichage couleur. La rampe gamma est spécifiée dans trois tableaux de 256 éléments **WORD** chacun, qui contiennent le mappage entre les valeurs RVB dans la mémoire tampon de trame et les valeurs DAC (digital-analog-converter). La séquence des tableaux est rouge, vert, bleu.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**.

Exemple

```
C++
```

```
WORD gArray[3][256];
GetDeviceGammaRamp(handle, gArray);
// `handle` is the device context. See GetDC for more details.
// `gArray` will hold the gamma array values in a 2-D array
```

Remarques

Les modes d'affichage des couleurs directes n'utilisent pas de tables de choix de couleurs et sont généralement 16, 24 ou 32 bits. Les tableaux vidéo en couleur directe ne prennent pas tous en charge les rampes gamma chargeables. **GetDeviceGammaRamp** réussit uniquement pour les appareils avec des pilotes qui prennent en charge les rampes gamma téléchargeables dans le matériel.

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
Plateforme cible	Windows
En-tête	wingdi.h
Bibliothèque	Gdi32.lib
DLL	Gdi32.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l’aide sur Microsoft Q&A

GetICMProfileA, fonction (wingdi.h)

Article27/08/2023

La fonction **GetICMProfile** récupère le nom de fichier du profil de couleur de sortie actuel pour un contexte d'appareil spécifié.

Syntaxe

```
C++

BOOL GetICMProfileA(
  HDC     hdc,
  LPDWORD pBufSize,
  LPSTR   pszFilename
);
```

Paramètres

`hdc`

Spécifie un contexte d'appareil à partir duquel récupérer le profil de couleur.

`pBufSize`

Pointeur vers un **DWORD** qui contient la taille de la mémoire tampon pointée par *lpszFilename*. Pour la version ANSI de cette fonction, la taille est en octets. Pour la version Unicode, la taille est en WCHAR. Si cette fonction réussit, au retour, ce paramètre contient la taille de la mémoire tampon réellement utilisée. Toutefois, si la mémoire tampon n'est pas suffisamment grande, cette fonction retourne **FALSE**. Dans ce cas, la fonction **GetLastError()** retourne ERROR_INSUFFICIENT_BUFFER et le **DWORD** pointé par ce paramètre contient la taille nécessaire pour la mémoire tampon *lpszFilename* .

`pszFilename`

Pointe vers la mémoire tampon qui reçoit le nom du chemin d'accès du profil.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**. Elle retourne également **TRUE** si le paramètre *lpszFilename* a la valeur **NULL** et que la taille requise pour la mémoire

tampon est copiée dans *lpcbName*.

Si cette fonction échoue, la valeur de retour est **FALSE**.

Remarques

GetICMProfile obtient le nom de fichier du profil de sortie actuel, que la gestion des couleurs soit activée ou non pour le contexte de l'appareil.

Dans un contexte d'appareil, **GetICMProfile** génère, via le paramètre *lpszFilename*, le nom du chemin d'accès du fichier contenant le profil de couleur actuellement utilisé par le contexte de l'appareil. Il génère également, via le paramètre *lpcbName*, la longueur de la chaîne contenant le nom du chemin d'accès.

Il est possible que le nom de profil retourné par **GetICMProfile** ne figure pas dans la liste des profils retournés par EnumICMProfiles. La fonction **EnumICMProfiles** retourne tous les profils d'espace de couleur associés à un contexte d'appareil (DC) dont les paramètres correspondent à ceux du contrôleur de domaine. Si la fonction SetICMProfile est utilisée pour définir le profil actuel, un profil peut être associé au contrôleur de domaine qui ne correspond pas à ses paramètres. Par instance, la fonction **SetICMProfile** peut être utilisée pour associer le profil sRGB indépendant de l'appareil à un contrôleur de domaine. Ce profil sera utilisé comme profil WCS actuel pour ce contrôleur de domaine, et les appels à **GetICMProfile** retourneront son nom de fichier. Toutefois, le profil n'apparaît pas dans la liste des profils retournés par **EnumICMProfiles**.

Si cette fonction est appelée avant tout appel à la fonction **SetICMProfile** , elle peut être utilisée pour obtenir le profil par défaut d'un contexte d'appareil.

Windows 95/98/Me : GetICMProfileW est pris en charge par Microsoft Layer pour Unicode. Pour l'utiliser, vous devez ajouter certains fichiers à votre application, comme indiqué dans Microsoft Layer pour Unicode sur les systèmes Windows 95/98/Me .

> ⓘ **Notes**
>
> L'en-tête wingdi.h définit GetICMProfile en tant qu'alias qui sélectionne automatiquement la version ANSI ou Unicode de cette fonction en fonction de la définition de la constante de préprocesseur UNICODE. La combinaison de l'utilisation de l'alias neutre en encodage avec du code qui n'est pas neutre en encodage peut entraîner des incompatibilités qui entraînent des erreurs de

compilation ou d'exécution. Pour plus d'informations, consultez **Conventions pour les prototypes de fonction**.

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
Plateforme cible	Windows
En-tête	wingdi.h
Bibliothèque	Gdi32.lib
DLL	Gdi32.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- DeleteColorSpaceW
- Fonction de rappel ICMENUMPROCA
- EnumICMProfilesW
- SetICMProfileW

Commentaires

GetLogColorSpaceA, fonction (wingdi.h)

Article27/08/2023

La fonction **GetLogColorSpace** récupère la définition de l'espace de couleurs identifiée par un handle spécifié.

Syntaxe

```
C++

BOOL GetLogColorSpaceA(
  HCOLORSPACE      hColorSpace,
  LPLOGCOLORSPACEA lpBuffer,
  DWORD            nSize
);
```

Paramètres

`hColorSpace`

Spécifie le handle dans un espace de couleurs.

`lpBuffer`

Pointe vers une mémoire tampon pour recevoir la structure LOGCOLORSPACE .

`nSize`

Spécifie la taille maximale de la mémoire tampon.

Valeur retournée

Si cette fonction réussit, la valeur de retour est TRUE.

Si cette fonction échoue, la valeur de retour est **FALSE**.

Remarques

Windows 95/98/Me : GetLogColorSpaceW est pris en charge par la couche Microsoft pour Unicode. Pour cela, vous devez ajouter certains fichiers à votre application, comme indiqué dans Microsoft Layer pour Unicode sur Windows 95/98/Me Systems .

Notes

L'en-tête wingdi.h définit GetLogColorSpace comme un alias qui sélectionne automatiquement la version ANSI ou Unicode de cette fonction en fonction de la définition de la constante de préprocesseur UNICODE. Le mélange de l'utilisation de l'alias neutre en encodage avec du code qui n'est pas neutre en encodage peut entraîner des incompatibilités qui entraînent des erreurs de compilation ou d'exécution. Pour plus d'informations, consultez **Conventions pour les prototypes de fonction**.

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
Plateforme cible	Windows
En-tête	wingdi.h
Bibliothèque	Gdi32.lib
DLL	Gdi32.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

GetNamedProfileInfo, fonction (icm.h)

Article24/08/2023

Récupère des informations sur le profil de couleur nommé ICC (International Color Consortium) spécifié dans le premier paramètre.

Syntaxe

```
C++

BOOL GetNamedProfileInfo(
  HPROFILE            hProfile,
  PNAMED_PROFILE_INFO pNamedProfileInfo
);
```

Paramètres

`hProfile`

Handle du profil ICC à partir duquel les informations seront récupérées.

`pNamedProfileInfo`

Pointeur vers une structure **NAMED_PROFILE_INFO** .

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**.

Notes

Cette fonction échoue si *hProfile* n'est pas un profil ICC valide.

Cette fonction ne prend pas en charge les profils WINDOWS Color System (WCS) CAMP, DMP et GMMP ; car les profils nommés sont des types de profils ICC explicites.

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Gdi32.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- NAMED_PROFILE_INFO

Commentaires

GetPS2ColorRenderingDictionary, fonction (icm.h)

Article26/02/2024

Récupère le dictionnaire de rendu de couleur PostScript Niveau 2 à partir du profil de couleur ICC spécifié.

Syntaxe

```
C++

BOOL GetPS2ColorRenderingDictionary(
  HPROFILE hProfile,
  DWORD    dwIntent,
  PBYTE    pPS2ColorRenderingDictionary,
  PDWORD   pcbPS2ColorRenderingDictionary,
  PBOOL    pbBinary
);
```

Paramètres

`hProfile`

Spécifie un handle pour le profil de couleur ICC en question.

`dwIntent`

Spécifie l'intention de rendu souhaitée pour le dictionnaire de rendu des couleurs. Les valeurs autorisées sont :

- INTENT_PERCEPTUAL
- INTENT_SATURATION
- INTENT_RELATIVE_COLORIMETRIC
- INTENT_ABSOLUTE_COLORIMETRIC

Pour plus d'informations, consultez Intentions de rendu.

`pPS2ColorRenderingDictionary`

Pointeur vers une mémoire tampon dans laquelle le dictionnaire de rendu des couleurs doit être placé. Si le pointeur *pBuffer* a la valeur **NULL**, la taille de mémoire tampon requise est retournée dans **pcbSize*.

`pcbPS2ColorRenderingDictionary`

Pointeur vers une variable contenant la taille de la mémoire tampon en octets. Au retour, la variable contient le nombre d'octets réellement copiés.

`pbBinary`

Pointeur vers une variable booléenne. Si **la valeur est TRUE**, le dictionnaire de rendu des couleurs peut être copié sous forme binaire. Si **la valeur est FALSE**, le dictionnaire est encodé sous ASCII85 forme. En retour, cette variable booléenne indique si le dictionnaire était réellement binaire (**TRUE**) ou ASCII85 (**FALSE**).

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**. Elle retourne également **TRUE** si le paramètre *pBuffer* a la valeur **NULL** et si la taille requise pour la mémoire tampon est copiée dans *pcbSize*.

Si cette fonction échoue, la valeur de retour est **FALSE**.

Remarques

Si le dictionnaire n'est pas disponible dans le profil, la fonction **GetPS2ColorRenderingDictionary** en génère un à l'aide du contenu du profil. Ce dictionnaire peut ensuite être utilisé comme opérande pour l'opérateur **setcolorrendering de** niveau PostScript 2.

Cette méthode ne prend pas en charge les profils WCS.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Condition requise	Valeur
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

Cette page a-t-elle été utile ? Yes No

Indiquer des commentaires sur le produit | Obtenir de l'aide sur Microsoft Q&A

Fonction GetPS2ColorRenderingIntent (icm.h)

Article29/02/2024

Récupère l'intention de rendu des couleurs PostScript Niveau 2 à partir d'un profil de couleur ICC.

Syntaxe

```
C++

BOOL GetPS2ColorRenderingIntent(
  HPROFILE hProfile,
  DWORD    dwIntent,
  PBYTE    pBuffer,
  PDWORD   pcbPS2ColorRenderingIntent
);
```

Paramètres

`hProfile`

Spécifie un handle pour le profil de couleur ICC en question.

`dwIntent`

Spécifie l'intention de rendu souhaitée à récupérer. Les valeurs autorisées sont :

INTENT_PERCEPTUAL

INTENT_SATURATION

INTENT_RELATIVE_COLORIMETRIC

INTENT_ABSOLUTE_COLORIMETRIC

Pour plus d'informations, consultez Rendu des intentions.

`pBuffer`

Pointe vers une mémoire tampon dans laquelle l'intention de rendu des couleurs doit être placée. Si le pointeur *pBuffer* a la valeur **NULL**, la taille de mémoire tampon requise est retournée dans **pcbSize*.

`pcbPS2ColorRenderingIntent`

Pointe vers une variable contenant la taille de la mémoire tampon en octets. Au retour, cette variable contient le nombre d'octets réellement copiés.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**. Si cette fonction réussit, la valeur de retour est **TRUE**. Elle retourne également **TRUE** si le paramètre *pBuffer* a la valeur **NULL** et si la taille requise pour la mémoire tampon est copiée dans *pcbSize*.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Remarques

L'intention de rendu retournée par **GetPS2ColorRenderingIntent** peut être utilisée comme opérande pour l'opérateur findcolorrendering de niveau PostScript 2.

Cette méthode ne prend pas en charge les profils WCS.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

Cette page a-t-elle été utile ? Yes No

Indiquer des commentaires sur le produit | Obtenir de l'aide sur Microsoft Q&A

GetPS2ColorSpaceArray, fonction (icm.h)

Article24/08/2023

Récupère le tableau d'espaces de couleurs PostScript Niveau 2 à partir d'un profil de couleur ICC.

Syntaxe

```
C++

BOOL GetPS2ColorSpaceArray(
  HPROFILE hProfile,
  DWORD    dwIntent,
  DWORD    dwCSAType,
  PBYTE    pPS2ColorSpaceArray,
  PDWORD   pcbPS2ColorSpaceArray,
  PBOOL    pbBinary
);
```

Paramètres

`hProfile`

Spécifie un handle pour le profil ICC à partir duquel récupérer le tableau d'espace de couleur PostScript Level 2.

`dwIntent`

Spécifie l'intention de rendu souhaitée pour le tableau d'espaces de couleurs. Ce champ peut prendre l'une des valeurs suivantes :

INTENT_PERCEPTUAL

INTENT_SATURATION

INTENT_RELATIVE_COLORIMETRIC

INTENT_ABSOLUTE_COLORIMETRIC

Pour plus d'informations, consultez Rendu des intentions.

`dwCSAType`

Spécifie le type de tableau d'espaces de couleurs. Consultez Identificateurs de type d'espace de couleur.

`pPS2ColorSpaceArray`

Pointeur vers une mémoire tampon dans laquelle le tableau d'espaces de couleurs doit être placé. Si le pointeur *pBuffer* a la valeur **NULL**, la fonction retourne la taille requise de la mémoire tampon dans l'emplacement de mémoire pointé par *pcbSize*.

`pcbPS2ColorSpaceArray`

Pointeur vers une variable contenant la taille de la mémoire tampon en octets. Au retour, il contient le nombre d'octets copiés dans la mémoire tampon.

`pbBinary`

Pointeur vers une variable booléenne. Si la valeur **est TRUE**, les données copiées peuvent être binaires. Si la valeur **est FALSE**, les données doivent être encodées en ASCII85. Au retour, l'emplacement de mémoire pointé vers *pbBinary* indique si les données retournées sont binaires (**TRUE**) ou ASCII85 (**FALSE**).

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**. Elle retourne également **TRUE** si le paramètre *pBuffer* a la valeur **NULL** et si la taille requise pour la mémoire tampon est copiée dans *pcbSize*.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Notes

Si le tableau d'espaces de couleurs n'est pas disponible dans le profil, la fonction **GetPS2ColorSpaceArray** génère un tableau d'espaces de couleurs PostScript Niveau 2 à l'aide du contenu du profil. Ce tableau peut ensuite être utilisé comme opérande pour l'opérateur setcolorspace PostScript Level2.

Cette méthode ne prend pas en charge les profils WCS.

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

GetStandardColorSpaceProfileA, fonction (icm.h)

Article24/08/2023

Récupère le profil de couleur inscrit pour l'espace de couleurs standard spécifié.

Syntaxe

C++

```
BOOL GetStandardColorSpaceProfileA(
  PCSTR  pMachineName,
  DWORD  dwSCS,
  PSTR   pBuffer,
  PDWORD pcbSize
);
```

Paramètres

`pMachineName`

Réservé. Doit avoir la **valeur NULL**. Ce paramètre est destiné à pointer vers le nom de l'ordinateur sur lequel obtenir un profil d'espace de couleurs standard. Un pointeur **NULL** indique l'ordinateur local.

`dwSCS`

Spécifie la valeur d'ID de l'espace de couleurs standard pour lequel récupérer le profil. Les seules valeurs valides pour ce paramètre sont LCS_sRGB et LCS_WINDOWS_COLOR_SPACE.

`pBuffer`

Pointeur vers la mémoire tampon dans laquelle le nom du profil doit être placé. Si **la** valeur est NULL, l'appel retourne **TRUE** et la taille requise de la mémoire tampon est placée dans *pdwSize*.

`pcbSize`

Pointeur vers une variable contenant la taille en octets de la mémoire tampon pointée par *pProfileName*. Au retour, la variable contient la taille de la mémoire tampon

réellement utilisée ou nécessaire.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Notes

Si la mémoire tampon pointée par *pProfileName* doit être allouée dynamiquement par une application, l'application peut appeler la fonction **GetStandardColorSpaceProfile** pour récupérer la taille requise pour la mémoire tampon. Si **GetStandardColorSpaceProfile** est appelé avec *pProfileName* défini sur **NULL**, il retourne **FALSE** et le **DWORD** pointé vers *pdwSize* contient le nombre d'octets nécessaires pour la mémoire tampon pointée vers par *pProfileName*. L'application peut ensuite allouer la mémoire tampon et appeler **à nouveau GetStandardColorSpaceProfile** avec *pProfileName* défini sur l'adresse de la mémoire tampon.

Cette fonction prend en charge les profils de modèle d'appareil (DPM) windows Color System (WCS) en plus des profils ICC (International Color Consortium). Il ne prend pas en charge les profils WCS CAMP ou GMMP et retourne une erreur si ces profils sont utilisés.

Vue d'ensemble des fonctionnalités spécifiques de Windows Vista

Cela prend en charge les DPM WCS en plus des profils ICC. Il ne prend pas en charge les profils WCS CAMP ou GMMP et retourne une erreur si ces profils sont utilisés avec cette API.

Prise en charge par utilisateur/LUA

Cela récupère le profil de couleur inscrit pour l'espace de couleur standard donné pour l'utilisateur actuel. S'il n'existe aucun paramètre de ce type pour l'utilisateur actuel, il récupère le paramètre à l'échelle du système.

Cela utilise **WcsGetDefaultColorProfile** avec WCS_PROFILE_MANAGEMENT_SCOPE_CURRENT_USER.

Il s'agit d'un fichier exécutable dans le contexte LUA.

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- SetStandardColorSpaceProfile

Commentaires

GetStandardColorSpaceProfileW, fonction (icm.h)

Article24/08/2023

Récupère le profil de couleur inscrit pour l'espace de couleurs standard spécifié.

Syntaxe

```
C++

BOOL GetStandardColorSpaceProfileW(
  PCWSTR pMachineName,
  DWORD  dwSCS,
  PWSTR  pBuffer,
  PDWORD pcbSize
);
```

Paramètres

`pMachineName`

Réservé. Doit être **NULL**. Ce paramètre est destiné à pointer vers le nom de l'ordinateur sur lequel obtenir un profil d'espace de couleurs standard. Un pointeur **NULL** indique l'ordinateur local.

`dwSCS`

Spécifie la valeur d'ID de l'espace de couleurs standard pour lequel récupérer le profil. Les seules valeurs valides pour ce paramètre sont LCS_sRGB et LCS_WINDOWS_COLOR_SPACE.

`pBuffer`

Pointeur vers la mémoire tampon dans laquelle le nom du profil doit être placé. Si **la** valeur est NULL, l'appel retourne **TRUE** et la taille requise de la mémoire tampon est placée dans *pdwSize*.

`pcbSize`

Pointeur vers une variable contenant la taille en octets de la mémoire tampon pointée par *pProfileName*. Au retour, la variable contient la taille de la mémoire tampon

réellement utilisée ou nécessaire.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Notes

Si la mémoire tampon pointée par *pProfileName* doit être allouée dynamiquement par une application, l'application peut appeler la fonction **GetStandardColorSpaceProfile** pour récupérer la taille requise pour la mémoire tampon. Si **GetStandardColorSpaceProfile** est appelé avec *pProfileName* défini sur **NULL**, il retourne **FALSE** et le **DWORD** pointé par *pdwSize* contient le nombre d'octets nécessaires pour la mémoire tampon pointée par *pProfileName*. L'application peut ensuite allouer la mémoire tampon et appeler **à nouveau GetStandardColorSpaceProfile** avec *pProfileName* défini sur l'adresse de la mémoire tampon.

Cette fonction prend en charge les profils de modèle d'appareil (DPM) windows Color System (WCS) en plus des profils ICC (International Color Consortium). Il ne prend pas en charge les profils WCS CAMP ou GMMP et retourne une erreur si ces profils sont utilisés.

Vue d'ensemble des fonctionnalités spécifiques de Windows Vista

Cela prendra en charge les DPM WCS en plus des profils ICC. Il ne prend pas en charge les profils WCS CAMP ou GMMP et retourne une erreur si ces profils sont utilisés avec cette API.

Prise en charge par utilisateur/LUA

Cette opération récupère le profil de couleur inscrit pour l'espace de couleur standard donné pour l'utilisateur actuel. S'il n'existe aucun paramètre de ce type pour l'utilisateur actuel, il récupère le paramètre à l'échelle du système.

Cela utilise **WcsGetDefaultColorProfile** avec WCS_PROFILE_MANAGEMENT_SCOPE_CURRENT_USER.

Il s'agit d'un exécutable dans le contexte LUA.

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- SetStandardColorSpaceProfile

Commentaires

InstallColorProfileA, fonction (icm.h)

Article24/08/2023

Installe un profil donné à utiliser sur un ordinateur spécifié. Le profil est également copié dans le répertoire COLOR.

Syntaxe

```
C++

BOOL InstallColorProfileA(
  PCSTR pMachineName,
  PCSTR pProfileName
);
```

Paramètres

`pMachineName`

Réservé. Doit être **NULL**. Ce paramètre est destiné à pointer vers le nom de l'ordinateur sur lequel le profil doit être installé. Un pointeur **NULL** indique l'ordinateur local.

`pProfileName`

Pointeur vers le nom de chemin complet du profil à installer.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]

Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

InstallColorProfileW, fonction (icm.h)

Article24/08/2023

Installe un profil donné à utiliser sur un ordinateur spécifié. Le profil est également copié dans le répertoire COLOR.

Syntaxe

```
C++

BOOL InstallColorProfileW(
  PCWSTR pMachineName,
  PCWSTR pProfileName
);
```

Paramètres

`pMachineName`

Réservé. Doit être **NULL**. Ce paramètre est destiné à pointer vers le nom de l'ordinateur sur lequel le profil doit être installé. Un pointeur **NULL** indique l'ordinateur local.

`pProfileName`

Pointeur vers le nom de chemin complet du profil à installer.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]

Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

IsColorProfileTagPresent, fonction (icm.h)

Article04/03/2024

Indique si une balise ICC (International Color Consortium) spécifiée est présente dans le profil de couleur spécifié.

Syntaxe

```
C++
```

```
BOOL IsColorProfileTagPresent(
  HPROFILE hProfile,
  TAGTYPE  tag,
  PBOOL    pbPresent
);
```

Paramètres

`hProfile`

Spécifie un handle pour le profil ICC en question.

`tag`

Spécifie la balise ICC à case activée.

`pbPresent`

Pointeur vers une variable qui a la valeur **TRUE** au retour si la balise ICC spécifiée est présente, ou **FALSE** si ce n'est pas le cas.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Remarques

Cette fonction échoue si *hProfile* n'est pas un profil ICC valide.

Cette fonction ne prend pas en charge les profils WINDOWS Color System (WCS) CAMP, DMP et GMMP ; parce que les éléments de profil sont implicitement associés à et codés en dur en types de balises ICC et qu'il existe de nombreuses bibliothèques d'analyse XML robustes.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

Cette page a-t-elle été utile ? Yes No

Indiquer des commentaires sur le produit | Obtenir de l'aide sur Microsoft Q&A

Fonction IsColorProfileValid (icm.h)

Article24/08/2023

Vous permet de déterminer si le profil spécifié est un profil ICC (International Color Consortium) valide ou un handle de profil Windows Color System (WCS) valide qui peut être utilisé pour la gestion des couleurs. La validation de profil WCS n'appelle pas les modèles d'appareil sous-jacents, mais valide simplement par rapport au schéma XML et aux limites de plage d'éléments de schéma.

Syntaxe

```
C++

BOOL IsColorProfileValid(
  HPROFILE hProfile,
  PBOOL    pbValid
);
```

Paramètres

`hProfile`

Spécifie un handle pour le profil à valider. La fonction détermine si le fichier HPROFILE contient des informations de profil ICC ou WCS.

`pbValid`

Pointeur vers une variable qui a la valeur **TRUE** au retour si l'opération réussit et si le profil est un profil ICC ou WCS valide. Si l'opération échoue ou si le profil n'est pas valide, la variable est **FALSE**.

Valeur retournée

Si cette fonction réussit et que le profil est valide, la valeur de retour est **TRUE**.

Si cette fonction échoue (ou réussit et que le profil n'est pas valide), la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

OpenColorProfileA, fonction (icm.h)

Article24/08/2023

Crée un handle pour un profil de couleur spécifié. Le handle peut ensuite être utilisé dans d'autres fonctions de gestion des profils.

Syntaxe

```
C++

HPROFILE OpenColorProfileA(
  PPROFILE pProfile,
  DWORD    dwDesiredAccess,
  DWORD    dwShareMode,
  DWORD    dwCreationMode
);
```

Paramètres

`pProfile`

Pointeur vers une structure de profil de couleur spécifiant le profil. Le pointeur *pProfile* peut être libéré dès que le handle est créé.

`dwDesiredAccess`

Spécifie comment accéder au profil donné. Ce paramètre doit prendre l'une des valeurs constantes suivantes.

Valeur	Signification
PROFILE_READ	Ouvre le profil pour l'accès en lecture.
PROFILE_READWRITE	Ouvre le profil pour l'accès en lecture et en écriture. N'a aucun effet pour les profils XML WCS.

`dwShareMode`

Spécifie comment le profil doit être partagé, si le profil est contenu dans un fichier. La valeur zéro empêche le partage du profil. Le paramètre peut contenir une ou les deux constantes suivantes (combinées par addition ou PAR LOGIQUE).

Valeur	Signification
FILE_SHARE_READ	D'autres opérations d'ouverture peuvent être effectuées sur le profil pour l'accès en lecture.
FILE_SHARE_WRITE	D'autres opérations d'ouverture peuvent être effectuées sur le profil pour l'accès en écriture. N'a aucun effet pour les profils XML WCS.

`dwCreationMode`

Spécifie les actions à effectuer sur le profil lors de son ouverture, s'il est contenu dans un fichier. Ce paramètre doit prendre l'une des valeurs constantes suivantes.

Valeur	Signification
CREATE_NEW	Crée un profil. Échoue si le profil existe déjà.
CREATE_ALWAYS	Crée un profil. Remplace le profil s'il existe.
OPEN_EXISTING	Ouvre le profil. Échoue s'il n'existe pas
OPEN_ALWAYS	Ouvre le profil s'il existe. Pour les profils ICC, si le profil n'existe pas, crée le profil. Pour les profils XML WCS, si le profil n'existe pas, retourne une erreur.
TRUNCATE_EXISTING	Ouvre le profil et le tronque à zéro octet, renvoyant un profil ICC vide. Échoue si le profil n'existe pas.

Valeur retournée

Si cette fonction réussit, la valeur de retour est le handle du profil de couleur ouvert. Pour les profils ICC et WCS, un CAMP et un GMMP sont fournis par la fonction en fonction des valeurs CAMP et GMMP par défaut actuelles dans le registre.

Quand OpenColorProfile rencontre un profil ICC avec un profil WCS incorporé, et si le membre dwType au sein de la structure profile ne prend pas la valeur DONT_USE_EMBEDDED_WCS_PROFILES, il doit extraire et utiliser le ou les profils WCS contenus dans ce WcsProfilesTag. Le fichier HPROFILE retourné est un HPROFILE WCS.

Si cette fonction échoue, la valeur de retour est **NULL**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Notes

Si les données de profil ne sont pas spécifiées à l'aide d'un nom de fichier, *dwShareMode* et *dwCreationMode* sont ignorés.

Les indicateurs *dwCreationMode* CREATE_NEW, CREATE_ALWAYS et TRUNCATE_EXISTING retournent toujours des HPROFILEs ICC vides. Si *d'autres indicateurs dwCreationMode* sont présents, InternalOpenColorProfile est appelé (à l'aide des indicateurs fournis par l'API) pour déterminer si le profil est ICC ou WCS XML.

Dans le chemin de code ICC, un fichier HPROFILE ICC est retourné à l'aide des indicateurs de partage, d'accès et de création demandés, comme indiqué dans les tableaux ci-dessus.

Dans le chemin WCS, l'indicateur *dwCreationMode* OPEN_ALWAYS échoue si le profil n'existe pas, car les profils WCS ne peuvent pas être créés ou modifiés dans l'architecture WCS (ils doivent être modifiés en dehors de celui-ci, à l'aide de MSXML6). Pour la même raison, *l'indicateur dwShareMode* FILE_SHARE_WRITE et l'indicateur *dwDesiredAccess* PROFILE_READWRITE sont ignorés dans le chemin WCS.

Lorsque la fonction ouvre le profil ICC, elle recherche un *WcsProfilesTag* et, le cas échéant, elle extrait et utilise les profils WCS d'origine qu'il contient. (Voir **WcsCreateIccProfile**.)

Un fichier HPROFILE avec des informations de profil WCS est dérivé d'un DMP en acquérant le CAMP par défaut et le GMMP par défaut à partir du Registre. Un HPROFILE est une composition d'un DMP, CAMP et GMMP.

Une fois le handle du profil de couleur créé, toutes les informations utilisées pour créer ce handle peuvent être supprimées.

Utilisez la fonction **CloseColorProfile** pour fermer un handle d'objet retourné par **OpenColorProfile**.

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib

DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- CloseColorProfile
- PROFIL

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

Fonction OpenColorProfileW (icm.h)

Article24/08/2023

Crée un handle dans un profil de couleur spécifié. Le handle peut ensuite être utilisé dans d'autres fonctions de gestion de profil.

Syntaxe

```
C++

HPROFILE OpenColorProfileW(
  PPROFILE pProfile,
  DWORD    dwDesiredAccess,
  DWORD    dwShareMode,
  DWORD    dwCreationMode
);
```

Paramètres

`pProfile`

Pointeur vers une structure de profil de couleur spécifiant le profil. Le pointeur *pProfile* peut être libéré dès que le handle est créé.

`dwDesiredAccess`

Spécifie comment accéder au profil donné. Ce paramètre doit prendre l'une des valeurs constantes suivantes.

Valeur	Signification
PROFILE_READ	Ouvre le profil pour l'accès en lecture.
PROFILE_READWRITE	Ouvre le profil pour l'accès en lecture et en écriture. N'a aucun effet pour les profils XML WCS.

`dwShareMode`

Spécifie la façon dont le profil doit être partagé, si le profil est contenu dans un fichier. La valeur zéro empêche le profil d'être partagé. Le paramètre peut contenir l'une ou les deux constantes suivantes (combinées par addition ou OR logique).

Valeur	Signification
FILE_SHARE_READ	D'autres opérations ouvertes peuvent être effectuées sur le profil pour l'accès en lecture.
FILE_SHARE_WRITE	D'autres opérations ouvertes peuvent être effectuées sur le profil pour l'accès en écriture. N'a aucun effet pour les profils XML WCS.

`dwCreationMode`

Spécifie les actions à effectuer sur le profil lors de son ouverture, s'il est contenu dans un fichier. Ce paramètre doit prendre l'une des valeurs constantes suivantes.

Valeur	Signification
CREATE_NEW	Crée un profil. Échoue si le profil existe déjà.
CREATE_ALWAYS	Crée un profil. Remplace le profil s'il existe.
OPEN_EXISTING	Ouvre le profil. Échoue s'il n'existe pas
OPEN_ALWAYS	Ouvre le profil s'il existe. Pour les profils ICC, si le profil n'existe pas, crée le profil. Pour les profils XML WCS, si le profil n'existe pas, retourne une erreur.
TRUNCATE_EXISTING	Ouvre le profil et le tronque à zéro octet, renvoyant un profil ICC vide. Échoue si le profil n'existe pas.

Valeur retournée

Si cette fonction réussit, la valeur de retour est le handle du profil de couleur ouvert. Pour les profils ICC et WCS, un CAMP et un GMMP sont fournis par la fonction en fonction des valeurs CAMP et GMMP par défaut actuelles dans le registre.

Quand OpenColorProfile rencontre un profil ICC avec un profil WCS incorporé, et si le membre dwType au sein de la structure Profil ne prend pas la valeur DONT_USE_EMBEDDED_WCS_PROFILES, il doit extraire et utiliser le ou les profils WCS contenus dans ce WcsProfilesTag. Le fichier HPROFILE retourné est un HPROFILE WCS.

Si cette fonction échoue, la valeur de retour est **NULL**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Notes

Si les données de profil ne sont pas spécifiées à l'aide d'un nom de fichier, *dwShareMode* et *dwCreationMode* sont ignorés.

les indicateurs *dwCreationMode* CREATE_NEW, CREATE_ALWAYS et TRUNCATE_EXISTING retournent toujours des HPROFILEs ICC vides. Si *d'autres indicateurs dwCreationMode* sont présents, InternalOpenColorProfile est appelé (à l'aide des indicateurs fournis par l'API) pour déterminer si le profil est ICC ou WCS XML.

Dans le chemin de code ICC, un HPROFILE ICC est retourné à l'aide des indicateurs de partage, d'accès et de création demandés, comme spécifié dans les tableaux ci-dessus.

Dans le chemin WCS, l'indicateur *dwCreationMode* OPEN_ALWAYS échoue si le profil n'existe pas, car les profils WCS ne peuvent pas être créés ou modifiés dans l'architecture WCS (ils doivent être modifiés en dehors de celui-ci, à l'aide de MSXML6). Pour la même raison, *l'indicateur dwShareMode* FILE_SHARE_WRITE et l'indicateur *dwDesiredAccess* PROFILE_READWRITE sont ignorés dans le chemin WCS.

Lorsque la fonction ouvre le profil ICC, elle recherche un *WcsProfilesTag* et, le cas échéant, elle extrait et utilise les profils WCS d'origine qu'il contient. (Voir **WcsCreateIccProfile**.)

Un fichier HPROFILE avec les informations de profil WCS est dérivé d'un DMP en acquérant le CAMP par défaut et le GMMP par défaut à partir du Registre. Un HPROFILE est une composition d'un DMP, d'un CAMP et d'un GMMP.

Une fois le handle du profil de couleur créé, toutes les informations utilisées pour créer ce handle peuvent être supprimées.

Utilisez la fonction **CloseColorProfile** pour fermer un handle d'objet retourné par **OpenColorProfile**.

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib

DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- CloseColorProfile
- PROFIL

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

Fonction RegisterCMMA (icm.h)

Article29/02/2024

Associe une valeur d'identification spécifiée à la bibliothèque de liens dynamiques du module de gestion des couleurs (DLL CMM) spécifiée. Lorsque cet ID apparaît dans un profil de couleur, Windows peut ensuite localiser la MMT correspondante afin de créer une transformation.

Syntaxe

```
C++
```

```
BOOL RegisterCMMA(
  PCSTR pMachineName,
  DWORD cmmID,
  PCSTR pCMMdll
);
```

Paramètres

`pMachineName`

Réservés au; doit actuellement être défini sur **NULL**, jusqu'à ce que l'inscription non locale soit prise en charge. Ce paramètre est destiné à pointer vers le nom de l'ordinateur sur lequel une DLL CMM doit être inscrite. Un pointeur **NULL** indique l'ordinateur local.

`cmmID`

Spécifie la signature d'ID du CMM enregistré auprès de l'International Color Consortium (ICC).

`pCMMdll`

Pointeur vers le nom de chemin complet de la DLL CMM.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

Cette page a-t-elle été utile ? Yes No

Indiquer des commentaires sur le produit | Obtenir de l'aide sur Microsoft Q&A

Fonction RegisterCMMW (icm.h)

Article29/02/2024

Associe une valeur d'identification spécifiée à la bibliothèque de liens dynamiques du module de gestion des couleurs (DLL CMM) spécifiée. Lorsque cet ID apparaît dans un profil de couleur, Windows peut ensuite localiser la MMT correspondante afin de créer une transformation.

Syntaxe

```
C++
```

```
BOOL RegisterCMMW(
  PCWSTR pMachineName,
  DWORD  cmmID,
  PCWSTR pCMMdll
);
```

Paramètres

`pMachineName`

Réservés au; doit actuellement être défini sur **NULL**, jusqu'à ce que l'inscription non locale soit prise en charge. Ce paramètre est destiné à pointer vers le nom de l'ordinateur sur lequel une DLL CMM doit être inscrite. Un pointeur **NULL** indique l'ordinateur local.

`cmmID`

Spécifie la signature d'ID du CMM enregistré auprès de l'International Color Consortium (ICC).

`pCMMdll`

Pointeur vers le nom de chemin complet de la DLL CMM.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

Cette page a-t-elle été utile ? Yes No

Indiquer des commentaires sur le produit | Obtenir de l'aide sur Microsoft Q&A

SelectCMM, fonction (icm.h)

Article24/08/2023

Vous permet de sélectionner le module de gestion des couleurs (CMM) préféré à utiliser.

Syntaxe

```
C++

BOOL SelectCMM(
  DWORD dwCMMType
);
```

Paramètres

`dwCMMType`

Spécifie la signature de la CMM souhaitée telle qu'enregistrée auprès de l'International Color Consortium (ICC).

Windows 2000 uniquement : Si vous affectez la **valeur NULL** à ce paramètre, le système WCS sélectionne la CMM par défaut.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Notes

Pour **que SelectCMM** réussisse, la CMM spécifiée doit être inscrite auprès du système.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau

	uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

SetColorProfileElement, fonction (icm.h)

Article24/08/2023

Définit les données d'élément d'un élément de profil balisé dans un profil de couleur ICC.

Syntaxe

```
C++

BOOL SetColorProfileElement(
  HPROFILE hProfile,
  TAGTYPE  tag,
  DWORD    dwOffset,
  PDWORD   pcbElement,
  PVOID    pElement
);
```

Paramètres

`hProfile`

Spécifie un handle pour le profil ICC en question.

`tag`

Identifie l'élément balisé.

`dwOffset`

Spécifie le décalage par rapport au premier octet des données de l'élément balisé à partir duquel commencer l'écriture.

`pcbElement`

Pointeur vers une variable contenant le nombre d'octets de données à écrire. Au retour, il contient le nombre d'octets réellement écrits.

`pElement`

Pointeur vers une mémoire tampon contenant les données à écrire dans l'élément balisé dans le profil de couleur.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Notes

Cette fonction échoue si *hProfile* n'est pas un profil ICC valide.

Si le profil de couleur n'est pas ouvert pour l'autorisation de lecture/écriture, cette fonction échoue.

Si *dwOffset* dépasse la taille définie pour l'élément balisé spécifié, cette fonction échoue.

Si *dwOffset* + **pcbSize* est supérieur à la taille de l'élément spécifié, cette fonction écrit uniquement autant d'octets que la taille actuelle de l'élément.

Toutes les données existantes dans la partie spécifiée de l'élément balisé sont remplacées lorsque cette fonction réussit.

Cette fonction ne prend pas en charge les profils WINDOWS Color System (WCS) CAMP, DMP et GMMP ; parce que les éléments de profil sont implicitement associés à et codés en dur en types de balises ICC et qu'il existe de nombreuses bibliothèques d'analyse XML robustes.

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

Fonction SetColorProfileElementReference (icm.h)

Article24/08/2023

Crée dans un profil de couleur ICC spécifié une nouvelle balise qui référence les mêmes données qu'une balise existante.

Syntaxe

```
C++

BOOL SetColorProfileElementReference(
  HPROFILE hProfile,
  TAGTYPE  newTag,
  TAGTYPE  refTag
);
```

Paramètres

`hProfile`

Spécifie un handle pour le profil de couleur ICC en question.

`newTag`

Identifie la nouvelle balise à créer.

`refTag`

Identifie la balise existante dont les données doivent être référencées par la nouvelle balise.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Notes

Cette fonction échoue si *hProfile* n'est pas un profil ICC valide.

Si *newTag* existe déjà ou *si refTag* n'existe pas, **SetColorProfileElementReference** échoue.

Si le profil de couleur n'a pas été ouvert avec l'autorisation de lecture/écriture, **SetColorProfileElementReference** échoue.

Cette fonction ne prend pas en charge les profils WINDOWS Color System (WCS) CAMP, DMP et GMMP ; parce que les éléments de profil sont implicitement associés à et codés en dur sur les types de balises ICC et qu'il existe de nombreuses bibliothèques d'analyse XML robustes.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

Fonction SetColorProfileElementSize (icm.h)

Article26/02/2024

Définit la taille d'un élément étiqueté dans un profil de couleur ICC.

Syntaxe

```
C++

BOOL SetColorProfileElementSize(
  HPROFILE hProfile,
  TAGTYPE  tagType,
  DWORD    pcbElement
);
```

Paramètres

`hProfile`

Spécifie un handle pour le profil de couleur ICC en question.

`tagType`

Identifie l'élément étiqueté.

`pcbElement`

Spécifie la taille sur laquelle définir l'élément étiqueté. Si *cbSize* est égal à zéro, cette fonction supprime l'élément étiqueté spécifié. Si la balise est une référence, seule l'entrée de table de balise est supprimée, pas les données.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Remarques

Cette fonction échoue si *hProfile* n'est pas un profil ICC valide.

Pour créer un élément étiqueté dans un profil de couleur, utilisez **SetColorProfileElementSize** pour définir la taille, puis utilisez SetColorProfileElement pour définir la valeur de l'élément.

Si la balise spécifiée existe déjà dans le profil, **SetColorProfileElementSize** modifie la taille de l'élément en le tronqué ou en ajoutant des zéros à la fin selon le cas.

Si la balise spécifiée existe déjà et est une référence à une autre balise, **SetColorProfileElementSize** crée une zone de données pour la balise qui n'est pas partagée.

Cette fonction ne prend pas en charge les profils WINDOWS Color System (WCS) CAMP, DMP et GMMP ; parce que les éléments de profil sont implicitement associés à et codés en dur sur les types de balises ICC et qu'il existe de nombreuses bibliothèques d'analyse XML robustes.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- SetColorProfileElement

Commentaires

Cette page a-t-elle été utile ? Yes No

Indiquer des commentaires sur le produit | Obtenir de l'aide sur Microsoft Q&A

SetColorProfileHeader, fonction (icm.h)

Article24/08/2023

Définit les données d'en-tête dans un profil de couleur ICC spécifié.

Syntaxe

```
C++

BOOL SetColorProfileHeader(
  HPROFILE       hProfile,
  PPROFILEHEADER pHeader
);
```

Paramètres

`hProfile`

Spécifie un handle pour le profil de couleur ICC en question.

`pHeader`

Pointeur vers les données d'en-tête de profil à écrire dans le profil spécifié.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Notes

Cette fonction échoue si *hProfile* n'est pas un profil ICC valide.

Si le profil de couleur n'a pas été ouvert avec l'autorisation de lecture/écriture, **SetColorProfileHeader** échoue.

SetColorProfileHeader remplace l'en-tête actuel dans le profil ICC.

Cette fonction ne prend pas en charge les profils WINDOWS Color System (WCS) CAMP, DMP et GMMP ; parce que les éléments de profil sont implicitement associés à et codés en dur en types de balises ICC et qu'il existe de nombreuses bibliothèques d'analyse XML robustes.

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- PROFILEHEADER

Commentaires

Fonction SetColorSpace (wingdi.h)

Article04/03/2024

La fonction **SetColorSpace** définit l'espace de couleur d'entrée pour un contexte d'appareil donné.

Syntaxe

```
C++

HCOLORSPACE SetColorSpace(
  HDC         hdc,
  HCOLORSPACE hcs
);
```

Paramètres

`hdc`

Spécifie le handle dans un contexte d'appareil.

`hcs`

Identifie le handle à l'espace de couleur à définir.

Valeur retournée

Si cette fonction réussit, la valeur de retour est un handle pour le *hColorSpace* remplacé.

Si cette fonction échoue, la valeur de retour est **NULL**.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]

Condition requise	Valeur
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
Plateforme cible	Windows
En-tête	wingdi.h
Bibliothèque	Gdi32.lib
DLL	Gdi32.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

SetDeviceGammaRamp, fonction (wingdi.h)

Article27/08/2023

La fonction **SetDeviceGammaRamp** définit la rampe gamma sur les cartes d'affichage couleur directes avec des pilotes qui prennent en charge les rampes gamma téléchargeables dans le matériel.

> **ⓘ Important**
>
> Nous vous recommandons vivement de ne pas utiliser cette API. L'utilisation de cette API est soumise à des limitations majeures :
>
> - **SetDeviceGammaRamp** implémente des heuristiques pour case activée si une rampe fournie génère un écran illisible. Si une rampe enfreint ces heuristiques, la fonction échoue silencieusement (autrement dit, elle retourne **TRUE**, mais elle ne définit pas votre rampe). Pour cette raison, vous ne pouvez pas vous attendre à utiliser cette fonction pour définir *n'importe quelle* rampe gamma arbitraire. En particulier, l'heuristique empêche les rampes qui entraîneraient presque tous les pixels approchant d'une valeur unique (par exemple, noir/blanc plein écran), car cela peut empêcher un utilisateur de récupérer l'écran.
> - En raison de la nature globale de la fonction, toute autre application sur le système peut, à tout moment, remplacer n'importe quelle rampe que vous avez définie. Dans certains cas, le système d'exploitation lui-même peut réserver l'utilisation de cette fonction, ce qui entraîne le remplacement de toute rampe existante. La rampe gamma est également réinitialisée sur la plupart des événements d'affichage (connexion/déconnexion d'un moniteur, modifications de résolution, etc.). Vous ne pouvez donc pas être certain qu'une rampe que vous définissez est en vigueur.
> - Cette API a un comportement non défini en mode HDR.
> - Cette API a une interaction non définie avec les solutions d'étalonnage des couleurs intégrées et tierces.

Pour l'étalonnage des couleurs, nous vous recommandons de créer un profil ICC (International Color Consortium) et de laisser le système d'exploitation appliquer le profil. Pour les scénarios OEM avancés, il existe un modèle de pilote de périphérique que vous pouvez utiliser pour personnaliser l'étalonnage des couleurs plus directement. Pour plus d'informations sur la gestion des profils de couleur, consultez Le système de couleurs **Windows** .

Pour le filtrage de la lumière bleue, Windows fournit désormais une prise en charge intégrée appelée **Night Light** . Nous vous recommandons de diriger les utilisateurs vers cette fonctionnalité.

Pour l'adaptation des couleurs (par exemple, l'ajustement de l'étalonnage des couleurs en fonction des capteurs de lumière ambiante), Windows fournit désormais une prise en charge intégrée, que nous recommandons d'utiliser par les fabricants OEM.

Pour les effets de filtre personnalisés, il existe une variété de **filtres de couleur** d'accessibilité intégrés pour faciliter une gamme de cas.

Syntaxe

C++

```
BOOL SetDeviceGammaRamp(
  HDC    hdc,
  LPVOID lpRamp
);
```

Paramètres

`hdc`

Spécifie le contexte de l'appareil du tableau d'affichage en couleur directe en question.

`lpRamp`

Pointeur vers une mémoire tampon contenant la rampe gamma à définir. La rampe gamma est spécifiée dans trois tableaux de 256 éléments **WORD** chacun, qui contiennent le mappage entre les valeurs RVB dans la mémoire tampon de trame et les valeurs de convertisseur numérique-analogique (*DAC*). La séquence des tableaux est rouge, vert, bleu. Les valeurs RVB doivent être stockées dans les bits les plus significatifs de chaque WORD pour augmenter l'indépendance de la DAC.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**.

Remarques

Les modes d'affichage des couleurs directes n'utilisent pas de tables de choix de couleurs et sont généralement 16, 24 ou 32 bits. Toutes les cartes vidéo couleur directes ne prennent pas en charge les rampes gamma chargeables. **SetDeviceGammaRamp** réussit uniquement pour les appareils avec des pilotes qui prennent en charge les rampes gamma téléchargeables dans le matériel.

> ⓘ **Notes**
>
> L'exécution de cette API peut prendre un temps non trivial. Le retour sur du matériel peut prendre jusqu'à 200 ms.

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
Plateforme cible	Windows
En-tête	wingdi.h
Bibliothèque	Gdi32.lib
DLL	Gdi32.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l’aide sur Microsoft Q&A

Fonction SetICMMode (wingdi.h)

Article04/03/2024

La fonction **SetICMMode** permet d'activer, de désactiver ou d'interroger la gestion des couleurs d'image sur un contexte d'appareil (DC) donné.

Syntaxe

```
C++

int SetICMMode(
  HDC hdc,
  int mode
);
```

Paramètres

`hdc`

Identifie le handle dans le contexte de l'appareil.

`mode`

Active et désactive la gestion des couleurs des images. Ce paramètre peut prendre l'une des valeurs constantes suivantes.

Agrandir le tableau

Valeur	Signification
ICM_ON	Active la gestion des couleurs. Désactive la correction des couleurs de style ancien des demi-tons.
ICM_OFF	Désactive la gestion des couleurs. Active l'ancienne correction des couleurs des demi-tons.
ICM_QUERY	Interroge l'état actuel de la gestion des couleurs.
ICM_DONE_OUTSIDEDC	Désactive la gestion des couleurs à l'intérieur du contrôleur de domaine. Sous Windows 2000, désactive également la correction des couleurs de style ancien des demi-tons. Non pris en charge sous Windows 95.

Valeur retournée

Si cette fonction réussit, la valeur de retour est une valeur différente de zéro.

Si cette fonction échoue, la valeur de retour est zéro.

Si ICM_QUERY est spécifié et que la fonction réussit, la valeur différente de zéro renvoyée est ICM_ON ou ICM_OFF pour indiquer le mode actuel.

Remarques

Si le système ne trouve pas de profil de couleur ICC correspondant à l'état de l'appareil, **SetICMMode** échoue et retourne zéro.

Une fois que WCS est activé pour un contexte d'appareil (DC), les couleurs passées dans le contrôleur de domaine à l'aide de la plupart des fonctions de l'API Win32 sont mises en correspondance de couleur. Les principales exceptions sont **BitBlt** et **StretchBlt**. L'hypothèse est que lors d'un transfert de bloc de bits (blit) d'un contrôleur de domaine à l'autre, les deux contrôleurs de domaine sont déjà compatibles et ne nécessitent aucune correction de couleur. Si ce n'est pas le cas, une correction de couleur peut être effectuée. Plus précisément, si une bitmap indépendante de l'appareil (DIB) est utilisée comme source pour un blit et que la fente est effectuée dans un contrôleur de domaine sur lequel WCS est activé, la correspondance des couleurs est effectuée. Si ce n'est pas ce que vous souhaitez, désactivez WCS pour le contrôleur de domaine de destination en appelant **SetICMMode** avant d'appeler **BitBlt** ou **StretchBlt**.

Si la fonction **CreateCompatibleDC** est utilisée pour créer une bitmap dans un contrôleur de domaine, il est possible que la couleur de la bitmap soit mise en correspondance deux fois, une fois lors de sa création et une fois lorsqu'une fente est effectuée. La raison en est qu'une bitmap dans un contrôleur de domaine créé par la fonction **CreateCompatibleDC** acquiert le pinceau, les stylets et la palette actuels du contrôleur de domaine source. Toutefois, WCS sera désactivé par défaut pour le nouveau contrôleur de domaine. Si WCS est activé ultérieurement pour le nouveau contrôleur de domaine à l'aide de la fonction **SetICMMode** , une correction de couleur est effectuée. Pour empêcher les corrections de double couleur à l'aide de la fonction **CreateCompatibleDC** , utilisez la fonction **SetICMMode** pour désactiver WCS pour le contrôleur de domaine source avant l'appel de la fonction **CreateCompatibleDC** .

Lorsqu'un contrôleur de domaine compatible est créé à partir du contrôleur de domaine d'une imprimante (consultez **CreateCompatibleDC**), la correspondance des couleurs par défaut est toujours effectuée si elle est activée pour le contrôleur de domaine de l'imprimante. Le profil de couleur par défaut de l'imprimante est utilisé lorsqu'une fente

est effectuée dans le contrôleur de domaine de l'imprimante à l'aide de **SetDIBitsToDevice** ou **StretchDIBits**. Si ce n'est pas ce que vous voulez, désactivez WCS pour le contrôleur de domaine de l'imprimante en appelant **SetICMMode** avant d'appeler **SetDIBitsToDevice** ou **StretchDIBits**.

En outre, lors de l'impression sur le contrôleur de domaine d'une imprimante avec WCS activé, la fonction **SetICMMode** doit être appelée après chaque appel à la fonction **StartPage** pour réactiver WCS. La fonction **StartPage** appelle les fonctions **RestoreDC** et **SaveDC** , ce qui entraîne la désactivation de WCS pour le contrôleur de domaine de l'imprimante.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
Plateforme cible	Windows
En-tête	wingdi.h
Bibliothèque	Gdi32.lib
DLL	Gdi32.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- BitBlt
- CreateCompatibleDC
- SetDIBitsToDevice
- StartPage
- StretchBlt
- StretchDIBits

Commentaires

Cette page a-t-elle été utile ? Yes No

Indiquer des commentaires sur le produit | Obtenir de l'aide sur Microsoft Q&A

SetICMProfileA, fonction (wingdi.h)

Article04/03/2024

La fonction **SetICMProfile** définit un profil de couleur spécifié comme profil de sortie pour un contexte d'appareil (DC) spécifié.

Syntaxe

```
C++

BOOL SetICMProfileA(
  HDC   hdc,
  LPSTR lpFileName
);
```

Paramètres

`hdc`

Spécifie un contexte d'appareil dans lequel définir le profil de couleur.

`lpFileName`

Spécifie le nom du chemin d'accès du profil de couleur à définir.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**.

Remarques

SetICMProfile associe un profil de couleur à un contexte d'appareil. Il devient le profil de sortie pour ce contexte d'appareil. Le profil de couleur n'a pas besoin d'être associé à un appareil particulier. Les profils indépendants de l'appareil, tels que sRGB, peuvent également être utilisés. Si le profil de couleur n'est pas associé à un périphérique matériel, il est retourné par GetICMProfile, mais pas par EnumICMProfiles.

Notez que sous Windows 95 ou version ultérieure, le pilote de périphérique PostScript pour les imprimantes suppose un modèle de couleur CMJN. Par conséquent, toutes les imprimantes PostScript doivent utiliser un profil de couleur CMJN. Windows 2000 n'a pas cette limitation.

SetICMProfile prend en charge uniquement les profils RVB dans les contrôleurs de domaine compatibles.

Windows 95/98/Me : SetICMProfileW est pris en charge par Microsoft Layer pour Unicode. Pour l'utiliser, vous devez ajouter certains fichiers à votre application, comme indiqué dans Microsoft Layer pour Unicode sur les systèmes Windows 95/98/Me.

> **Notes**
>
> L'en-tête wingdi.h définit SetICMProfile comme alias qui sélectionne automatiquement la version ANSI ou Unicode de cette fonction en fonction de la définition de la constante de préprocesseur UNICODE. La combinaison de l'utilisation de l'alias neutre en encodage avec du code qui n'est pas neutre en encodage peut entraîner des incompatibilités qui entraînent des erreurs de compilation ou d'exécution. Pour plus d'informations, consultez Conventions pour les prototypes de fonction.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
Plateforme cible	Windows
En-tête	wingdi.h
Bibliothèque	Gdi32.lib
DLL	Gdi32.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- EnumICMProfilesW
- GetICMProfileW

Commentaires

Cette page a-t-elle été utile ? Yes No

Indiquer des commentaires sur le produit | Obtenir de l'aide sur Microsoft Q&A

SetStandardColorSpaceProfileA, fonction (icm.h)

Article24/08/2023

Inscrit un profil spécifié pour un espace de couleurs standard donné. Le profil peut être interrogé à l'aide de GetStandardColorSpaceProfileW.

Syntaxe

```
C++

BOOL SetStandardColorSpaceProfileA(
  PCSTR pMachineName,
  DWORD dwProfileID,
  PCSTR pProfilename
);
```

Paramètres

`pMachineName`

Réservé. Doit être **NULL**. Ce paramètre est destiné à pointer vers le nom de l'ordinateur sur lequel définir un profil d'espace de couleurs standard. Un pointeur **NULL** indique l'ordinateur local.

`dwProfileID`

Spécifie la valeur d'ID de l'espace de couleurs standard que représente le profil donné. Il s'agit d'une valeur d'ID personnalisée utilisée pour identifier de manière unique le profil d'espace de couleurs au sein de votre application.

`pProfilename`

Pointe vers un chemin d'accès complet au fichier de profil.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, appelez GetLastError.

Notes

Le profil doit déjà être installé sur le système avant de pouvoir être inscrit pour un espace de couleurs standard.

Cette fonction prend en charge les profils de modèle d'appareil (DPM) windows Color System (WCS) en plus des profils ICC (International Color Consortium). Il ne prend pas en charge les profils WCS CAMP ou GMMP et retourne une erreur si ces profils sont utilisés.

Prise en charge par utilisateur/LUA

Cela permet d'inscrire un profil spécifié pour un espace de couleur standard donné uniquement pour l'utilisateur actuel.

Cela utilise **WcsSetDefaultColorProfile** avec WCS_PROFILE_MANAGEMENT_SCOPE_CURRENT_USER.

Il s'agit d'un exécutable dans le contexte LUA si le profil est déjà installé. Sinon, l'accès est refusé, car l'installation est à l'échelle du système et nécessite des privilèges d'administrateur.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs

- Fonctions
- SetStandardColorSpaceProfileW

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

Fonction SetStandardColorSpaceProfileW (icm.h)

Article24/08/2023

Inscrit un profil spécifié pour un espace de couleurs standard donné. Le profil peut être interrogé à l'aide de GetStandardColorSpaceProfileW.

Syntaxe

```
C++

BOOL SetStandardColorSpaceProfileW(
  PCWSTR pMachineName,
  DWORD  dwProfileID,
  PCWSTR pProfileName
);
```

Paramètres

`pMachineName`

Réservé. Doit avoir la **valeur NULL**. Ce paramètre est destiné à pointer vers le nom de l'ordinateur sur lequel définir un profil d'espace de couleurs standard. Un pointeur **NULL** indique l'ordinateur local.

`dwProfileID`

Spécifie la valeur d'ID de l'espace de couleurs standard que représente le profil donné. Il s'agit d'une valeur d'ID personnalisée utilisée pour identifier de manière unique le profil d'espace de couleurs au sein de votre application.

`pProfileName`

Pointe vers un chemin complet vers le fichier de profil.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, appelez GetLastError.

Notes

Le profil doit déjà être installé sur le système avant de pouvoir être inscrit pour un espace de couleurs standard.

Cette fonction prend en charge les profils de modèle d'appareil (DPM) windows Color System (WCS) en plus des profils ICC (International Color Consortium). Il ne prend pas en charge les profils WCS CAMP ou GMMP et retourne une erreur si ces profils sont utilisés.

Prise en charge par utilisateur/LUA

Cela inscrit un profil spécifié pour un espace de couleur standard donné uniquement pour l'utilisateur actuel.

Cela utilise **WcsSetDefaultColorProfile** avec WCS_PROFILE_MANAGEMENT_SCOPE_CURRENT_USER.

Il s'agit d'un fichier exécutable dans le contexte LUA si le profil est déjà installé. Sinon, l'accès est refusé, car l'installation est à l'échelle du système et nécessite des privilèges d'administrateur.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs

- Fonctions
- SetStandardColorSpaceProfileW

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

SetupColorMatchingA, fonction (icm.h)

Article26/02/2024

Crée une boîte de dialogue Gestion des couleurs qui permet à l'utilisateur d'activer la gestion des couleurs et, si c'est le cas, fournit un contrôle sur les profils de couleurs utilisés et sur l'intention de rendu.

Syntaxe

```
C++
```

```
BOOL SetupColorMatchingA(
  PCOLORMATCHSETUPA pcms
);
```

Paramètres

`pcms`

Pointeur vers une structure **COLORMATCHSETUPW** qui, sur l'entrée, contient des informations utilisées pour initialiser la boîte de dialogue.

Lorsque **SetupColorMatching** retourne, si l'utilisateur a cliqué sur le bouton OK, cette structure contient des informations sur la sélection de l'utilisateur. Sinon, si une erreur s'est produite ou si l'utilisateur a annulé la boîte de dialogue, la structure reste inchangée.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE** , ce qui indique qu'aucune erreur ne s'est produite et que l'utilisateur a cliqué sur le bouton OK.

Si cette fonction échoue, la valeur de retour est **FALSE** , ce qui indique qu'une erreur s'est produite ou que la boîte de dialogue a été annulée. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Icmui.lib
DLL	Icmui.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

Fonction SetupColorMatchingW (icm.h)

Article24/08/2023

Crée une boîte de dialogue Gestion des couleurs qui permet à l'utilisateur d'activer la gestion des couleurs et, si c'est le cas, fournit un contrôle sur les profils de couleurs utilisés et sur l'intention de rendu.

Syntaxe

```
C++
```

```
BOOL SetupColorMatchingW(
  PCOLORMATCHSETUPW pcms
);
```

Paramètres

`pcms`

Pointeur vers une structure **COLORMATCHSETUPW** qui, sur l'entrée, contient des informations utilisées pour initialiser la boîte de dialogue.

Lorsque **SetupColorMatching** retourne, si l'utilisateur a cliqué sur le bouton OK, cette structure contient des informations sur la sélection de l'utilisateur. Sinon, si une erreur s'est produite ou si l'utilisateur a annulé la boîte de dialogue, la structure reste inchangée.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE** , ce qui indique qu'aucune erreur ne s'est produite et que l'utilisateur a cliqué sur le bouton OK.

Si cette fonction échoue, la valeur de retour est **FALSE** , ce qui indique qu'une erreur s'est produite ou que la boîte de dialogue a été annulée. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Icmui.lib
DLL	Icmui.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

Fonction TranslateBitmapBits (icm.h)

Article24/08/2023

Traduit les couleurs d'une bitmap ayant un format défini afin de produire une autre bitmap dans un format demandé.

Syntaxe

```
C++

BOOL TranslateBitmapBits(
  HTRANSFORM    hColorTransform,
  PVOID         pSrcBits,
  BMFORMAT      bmInput,
  DWORD         dwWidth,
  DWORD         dwHeight,
  DWORD         dwInputStride,
  PVOID         pDestBits,
  BMFORMAT      bmOutput,
  DWORD         dwOutputStride,
  PBMCALLBACKFN pfnCallBack,
  LPARAM        ulCallbackData
);
```

Paramètres

`hColorTransform`

Identifie la transformation de couleur à utiliser.

`pSrcBits`

Pointeur vers la bitmap à traduire.

`bmInput`

Spécifie le format de l'image bitmap d'entrée. Doit être défini sur l'une des valeurs du type énuméré **BMFORMAT** .

> ⓘ **Notes**
>
> Cette fonction ne prend pas en charge **BM_XYZTRIPLETS** ou BM_YxyTRIPLETS en tant **qu'entrées** .

`dwWidth`

Spécifie le nombre de pixels par ligne d'analyse dans la bitmap d'entrée.

`dwHeight`

Spécifie le nombre de lignes d'analyse dans l'image bitmap d'entrée.

`dwInputStride`

Spécifie le nombre d'octets entre le début d'une ligne d'analyse et le début de la suivante dans la bitmap d'entrée ; si la valeur est égale à zéro, la fonction suppose que les lignes d'analyse sont rembourrées afin d'être alignées sur **DWORD**.

`pDestBits`

Pointeur vers la mémoire tampon dans laquelle placer l'image bitmap traduite.

`bmOutput`

Spécifie le format de l'image bitmap de sortie. Doit être défini sur l'une des valeurs du type énuméré **BMFORMAT** .

`dwOutputStride`

Spécifie le nombre d'octets entre le début d'une ligne d'analyse et le début de la suivante dans la bitmap de sortie ; si la valeur est égale à zéro, la fonction suppose que les lignes d'analyse doivent être complétées pour être alignées sur **DWORD**.

`pfnCallBack`

Pointeur vers une fonction de rappel appelée régulièrement par **TranslateBitmapBitBits** pour signaler la progression et permettre au processus d'appel d'annuler la traduction. (Voir **ICMProgressProcCallback**)

`ulCallbackData`

Données renvoyées à la fonction de rappel, par exemple, pour identifier la traduction qui signale la progression.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Notes

Si les formats d'entrée et de sortie ne sont pas compatibles avec la transformation de couleur, cette fonction échoue.

Lorsque l'un des BMFORMAT à virgule flottante, BM_32b_scARGB ou BM_32b_scRGB est utilisé, les données de couleur en cours de traduction ne doivent pas contenir de NaN ou d'infini. NaN et infinity ne sont pas considérés comme représentant des valeurs de composants de couleur légitimes, et le résultat de la traduction de pixels contenant naN ou infini n'a aucun sens en termes de couleur. Les valeurs NaN ou infini dans les données de couleur en cours de traitement seront gérées en mode silencieux et aucune erreur ne sera retournée.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- ICMProgressProcCallback
- Structures d'en-tête bitmap Windows
- BMFORMAT

Commentaires

Obtenir de l’aide sur Microsoft Q&A

Fonction TranslateColors (icm.h)

Article24/08/2023

Traduit un tableau de couleurs de l'espace de couleur source en espace de couleur de destination tel que défini par une transformation de couleur.

Syntaxe

```
C++

BOOL TranslateColors(
  HTRANSFORM hColorTransform,
  PCOLOR     paInputColors,
  DWORD      nColors,
  COLORTYPE  ctInput,
  PCOLOR     paOutputColors,
  COLORTYPE  ctOutput
);
```

Paramètres

`hColorTransform`

Identifie la transformation de couleur à utiliser.

`paInputColors`

Pointeur vers un tableau de structures *nColors***COLOR** à traduire.

`nColors`

Contient le nombre d'éléments dans les tableaux pointés par *paInputColors* et *paOutputColors*.

`ctInput`

Spécifie le type de couleur d'entrée.

`paOutputColors`

Pointeur vers un tableau de structures *nColors***COLOR** qui reçoivent les couleurs traduites.

`ctOutput`

Spécifie le type de couleur de sortie.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Notes

Si les types de couleurs d'entrée et de sortie ne sont pas compatibles avec la transformation de couleur, cette fonction échoue.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

UninstallColorProfileA, fonction (icm.h)

Article09/03/2023

Supprime un profil de couleur spécifié d'un ordinateur spécifié. Les fichiers associés sont éventuellement supprimés du système.

Syntaxe

```
C++

BOOL UninstallColorProfileA(
  PCSTR pMachineName,
  PCSTR pProfileName,
  BOOL  bDelete
);
```

Paramètres

`pMachineName`

Réservé. Doit être **NULL**. Ce paramètre est destiné à pointer vers le nom de l'ordinateur à partir duquel désinstaller le profil spécifié. Un pointeur **NULL** indique l'ordinateur local.

`pProfileName`

Pointe vers le nom de fichier du profil à désinstaller.

`bDelete`

Si la valeur **est TRUE**, la fonction supprime le profil du répertoire COLOR. Si la valeur **est FALSE**, cette fonction n'a aucun effet.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, appelez **GetLastError**.

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

UninstallColorProfileW, fonction (icm.h)

Article24/08/2023

Supprime un profil de couleur spécifié d'un ordinateur spécifié. Les fichiers associés sont éventuellement supprimés du système.

Syntaxe

```
C++

BOOL UninstallColorProfileW(
  PCWSTR pMachineName,
  PCWSTR pProfileName,
  BOOL   bDelete
);
```

Paramètres

`pMachineName`

Réservé. Doit être **NULL**. Ce paramètre est destiné à pointer vers le nom de l'ordinateur à partir duquel désinstaller le profil spécifié. Un pointeur **NULL** indique l'ordinateur local.

`pProfileName`

Pointe vers le nom de fichier du profil à désinstaller.

`bDelete`

Si la valeur **est TRUE**, la fonction supprime le profil du répertoire COLOR. Si la valeur **est FALSE**, cette fonction n'a aucun effet.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

UnregisterCMMA, fonction (icm.h)

Article02/03/2024

Dissocie une valeur d'ID spécifiée d'une bibliothèque de liens dynamiques de module de gestion des couleurs (DLL CMM).

Syntaxe

```
C++

BOOL UnregisterCMMA(
  PCSTR pMachineName,
  DWORD cmmID
);
```

Paramètres

`pMachineName`

Réservés au; doit actuellement être défini sur **NULL**, jusqu'à ce que l'inscription non locale soit prise en charge. Ce paramètre est destiné à pointer vers le nom de l'ordinateur sur lequel une inscription DLL CMM doit être supprimée. Un pointeur **NULL** indique l'ordinateur local.

`cmmID`

Spécifie la valeur d'ID identifiant la CMM dont l'inscription doit être supprimée. Il s'agit de la signature de la CMM enregistrée auprès de l'International Color Consortium (ICC).

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Remarques

Si le profil pour la création d'une transformation spécifie ultérieurement cet ID, la CMM Par défaut Windows est utilisée plutôt que cette DLL CMM.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

Annuler l’inscriptionCMMW, fonction (icm.h)

Article24/08/2023

Dissocie une valeur d’ID spécifiée d’une bibliothèque de liens dynamiques de module de gestion des couleurs (DLL CMM).

Syntaxe

```
C++
```

```
BOOL UnregisterCMMW(
  PCWSTR pMachineName,
  DWORD  cmmID
);
```

Paramètres

`pMachineName`

Réservés au; doit actuellement être défini sur **NULL**, jusqu’à ce que l’inscription non locale soit prise en charge. Ce paramètre est destiné à pointer vers le nom de l’ordinateur sur lequel une inscription DLL CMM doit être supprimée. Un pointeur **NULL** indique l’ordinateur local.

`cmmID`

Spécifie la valeur d’ID identifiant la CMM dont l’inscription doit être supprimée. Il s’agit de la signature de la CMM enregistrée auprès de l’International Color Consortium (ICC).

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d’erreur étendues, **appelez GetLastError**.

Notes

Si le profil pour la création d'une transformation spécifie ultérieurement cet ID, la CMM Par défaut Windows est utilisée plutôt que cette DLL CMM.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

WcsAssociateColorProfileWithDevice, fonction (icm.h)

Article24/08/2023

Associe un profil de couleur WCS spécifié à un appareil spécifié.

> ⓘ **Notes**
>
> Cette API ne prend pas en charge les profils de couleur avancés pour les moniteurs HDR. Utilisez **ColorProfileAddDisplayAssociation** pour gérer les profils de couleurs avancés.

Syntaxe

C++

```
BOOL WcsAssociateColorProfileWithDevice(
  WCS_PROFILE_MANAGEMENT_SCOPE scope,
  PCWSTR                       pProfileName,
  PCWSTR                       pDeviceName
);
```

Paramètres

`scope`

Valeur **WCS_PROFILE_MANAGEMENT_SCOPE** qui spécifie l'étendue de cette opération de gestion de profil, qui peut être à l'échelle du système ou pour l'utilisateur actuel.

`pProfileName`

Pointeur vers le nom de fichier du profil à associer.

`pDeviceName`

Pointeur vers le nom de l'appareil auquel le profil doit être associé.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Notes

La fonction **WCSAssociateColorProfileWithDevice** échoue si le profil n'a pas été installé sur l'ordinateur à l'aide de la fonction **InstallColorProfileW** .

Si le paramètre *profileManagementScope* est WCS_PROFILE_MANAGEMENT_SCOPE_SYSTEM_WIDE, l'association de profil est à l'échelle du système et s'applique à tous les utilisateurs. Si *profileManagementScope* est WCS_PROFILE_MANAGEMENT_SCOPE_CURRENT_USER, l'association concerne uniquement l'utilisateur actuel.

Cette fonction est exécutable dans Least-Privileged contexte de compte d'utilisateur (LUA) si *profileManagementScope* est WCS_PROFILE_MANAGEMENT_SCOPE_CURRENT_USER. Sinon, des privilèges d'administration sont requis.

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- Schémas et algorithmes windows Color System
- WcsDisassociateColorProfileFromDevice**

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l’aide sur Microsoft Q&A

WcsCheckColors, fonction (icm.h)

Article29/02/2024

Détermine si les couleurs d'un tableau se trouvent dans la gamme de sortie d'une transformation de couleur WCS spécifiée.

Syntaxe

```
C++

BOOL WcsCheckColors(
  HTRANSFORM    hColorTransform,
  DWORD         nColors,
  DWORD         nInputChannels,
  COLORDATATYPE cdtInput,
  DWORD         cbInput,
  PVOID         pInputData,
  PBYTE         paResult
);
```

Paramètres

`hColorTransform`

Handle de la transformation de couleur WCS spécifiée.

`nColors`

Nombre d'éléments dans le tableau pointé vers *pInputData* et *paResult*.

`nInputChannels`

Nombre de canaux par élément dans le tableau pointé vers *pInputData*.

`cdtInput`

Type de données de couleur COLORDATATYPE d'entrée.

`cbInput`

Taille de la mémoire tampon de *pInputData*.

`pInputData`

Pointeur vers un tableau de couleurs d'entrée. Les couleurs de ce tableau correspondent à l'espace de couleurs du profil source. La taille de la mémoire tampon pour ce tableau sera le nombre d'octets indiqué par *cbInput*.

`paResult`

Pointeur vers un tableau *d'octets nColors* qui reçoit les résultats du test.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Remarques

Si les types de données d'entrée et de couleur de sortie ne sont pas compatibles avec la transformation de couleur, cette fonction convertit les données de couleur d'entrée en fonction des besoins.

Cette fonction échoue si vous utilisez une transformation ICC (International Color Consortium) utilisée.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- Schémas et algorithmes du système de couleurs Windows

Commentaires

Cette page a-t-elle été utile ? Yes No

Indiquer des commentaires sur le produit | Obtenir de l'aide sur Microsoft Q&A

WcsCreateIccProfile, fonction (icm.h)

Article24/08/2023

Convertit un profil WCS en profil ICC (International Color Consortium).

Syntaxe

```
C++

HPROFILE WcsCreateIccProfile(
  HPROFILE hWcsProfile,
  DWORD    dwOptions
);
```

Paramètres

`hWcsProfile`

Handle dans le profil de couleur WCS qui est converti. Consultez la section Notes.

`dwOptions`

Valeur d'indicateur qui spécifie les options de conversion de profil.

Par défaut, les profils WCS d'origine utilisés pour la conversion sont incorporés dans le profil ICC de sortie dans une balise privée Microsoft, *WcsProfilesTag* (avec la signature « MS000 ». Cela produit un profil ICC compatible avec le logiciel ICC, tout en conservant les données de profil WCS d'origine disponibles pour le code conçu pour l'analyser.

Les valeurs possibles de ce paramètre sont les suivantes. Tous les bits non définis dans cette liste sont réservés et doivent être définis sur zéro :

Valeur	Description
WCS_DEFAULT	Spécifie que le nouveau profil ICC contient le profil WCS d'origine dans un WcsProfilesTag privé.
WCS_ICCONLY	Spécifie que le nouveau profil ICC ne contient ni le WcsProfilesTag ni le profil WCS d'origine.

Valeur retournée

Si cette fonction réussit, la valeur de retour est le handle du nouveau profil de couleur.

Si cette fonction échoue, la valeur de retour est **NULL**. Pour obtenir des informations d'erreur étendues, appelez GetLastError.

Notes

Cette fonction peut être utilisée avec des chaînes ASCII ou Unicode.

La fonction **CloseColorProfile** doit être utilisée pour fermer le handle HPROFILE retourné lorsqu'il n'est plus nécessaire.

Les fichiers DMP, CAMP et GMMP du HPROFILE sont incorporés dans une balise privée dans le profil ICC créé.

Le profil ICC créé à l'aide de cette API verra sa balise de description de profil construite à partir des éléments ProfileName des profils WCS selon le modèle suivant : « Créé par Microsoft WCS à partir de DMP:[le DMP ProfileName], CAMP:[le PROFILNAME CAMP], GMMP:[le profil GMMPName] »

Lorsque WCS rencontre ce profil ICC (via **OpenColorProfileW** ou **WcsOpenColorProfileW**), il extrait et utilise le ou les profils WCS contenus dans le *WcsProfilesTag*.

Les informations hors gamut dans les étiquettes gamut créées dans WCS utilisent la distance de couleur perceptive dans CIECAM02, qui est la racine carrée moyenne dans l'espace Jab CIECAM02. La distance dans les balises de gamut de profil ICC héritées est la racine carrée moyenne dans l'espace CIELAB. Il est recommandé d'utiliser l'espace CIECAM02 lorsqu'il est disponible pour fournir des métriques de distance plus précises.

WCS extrait et utilise le profil WCS d'origine au moyen d'un profil XML explicitement associé à un appareil, ou d'un profil ICC doté d'un*WcsProfilesTag*.

WcsProfilesTag est une balise de profil ICC privée Microsoft utilisée dans **les profils créés par WcsCreateIccProfile** pour contenir l'entrée des profils WCS dans **WcsCreateIccProfile**. Cette balise est conforme aux exigences de profil ICC pour les balises de profil. Les composants non XML de la balise doivent être dans l'ordre d'octets « Big-Endian », qui est standard pour les profils ICC. En outre, les données de balise doivent être alignées sur une limite de 4 octets (mesurées à partir du début du profil ICC). La structure de la balise est définie par le **WcsProfilesTagType** ci-dessous. Notez que les composants XML de la balise, les profils WCS contenus dans WcsProfileTag, sont laissés dans leur ordre d'octet natif, qui peut être soit petit-endien soit big-endian, car les analyseurs XML traitent correctement l'un ou l'autre.

La signature WcsProfilesTag est « MS00 ». Il s'agit de la signature de balise qui apparaîtra dans la table de balises de profils ICC pour le WcsProfilesTag.

La **structure WcsProfilesTagType** a la structure suivante :

Décalage d'octet	Contenu
0-3	Signature de type MS10.
4-7	Réservé, doit être défini sur 0 (tradition ICC).
8-11	Décalage d'octet entre le début de la balise et les données CDMP.
12-15	Taille des données CDMP en octets.
16-19	Décalage d'octet entre le début de la balise et les données CAMP.
20-23	Taille des données CAMP en octets.
24-27	Décalage d'octet entre le début de la balise et les données GMMP.
28-31	Décalage d'octet entre le début de la balise et les données GMMP.
31-n	Séquence d'octets (taille d'élément -32) [où la taille de l'élément correspond à la taille de balise enregistrée dans l'entrée de la table de balise de profil ICC pour cette balise.]

Il s'agit des profils XML WCS utilisés par **WcsCreateIccProfile** pour créer ce profil ICC. Les profils WCS sont triés : d'abord le DMP (obligatoire), suivi du CAMP (le cas échéant), suivi du GMMP (le cas échéant).

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- Schémas et algorithmes du système de couleurs Windows

Commentaires

WcsDisassociateColorProfileFromDevice, fonction (icm.h)

Article24/08/2023

Dissocie un profil de couleur WCS spécifié d'un appareil spécifié sur un ordinateur.

> ⓘ **Notes**
>
> Cette API ne prend pas en charge les profils de « couleur avancée » pour les moniteurs HDR. Utilisez **ColorProfileRemoveDisplayAssociation** pour gérer les profils de couleurs avancés.

Syntaxe

```
C++

BOOL WcsDisassociateColorProfileFromDevice(
  WCS_PROFILE_MANAGEMENT_SCOPE scope,
  PCWSTR                       pProfileName,
  PCWSTR                       pDeviceName
);
```

Paramètres

`scope`

Valeur **WCS_PROFILE_MANAGEMENT_SCOPE** qui spécifie l'étendue de cette opération de gestion de profil, qui peut être à l'échelle du système ou pour l'utilisateur actuel.

`pProfileName`

Pointeur vers le nom de fichier du profil à dissocier.

`pDeviceName`

Pointeur vers le nom de l'appareil à partir duquel dissocier le profil.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Notes

Le profil de couleur WCS doit être installé. En outre, vous devez utiliser la même valeur *profileManagementScope* que lorsque l'appareil était associé au profil. Consultez **WcsAssociateColorProfileWithDevice**.

Si *profileManagementScope* est WCS_PROFILE_MANAGEMENT_SCOPE_SYSTEM_WIDE, la dissociation de profil est à l'échelle du système et s'applique à tous les utilisateurs. Si *profileManagementScope* est WCS_PROFILE_MANAGEMENT_SCOPE_CURRENT_USER, la dissociation concerne uniquement l'utilisateur actuel.

Si plusieurs profils de couleur sont associés à un appareil, WCS utilise le dernier profil associé comme profil par défaut. Par exemple, si votre application associe séquentiellement trois profils à un appareil, WCS utilise le dernier profil associé comme profil par défaut. Si votre application appelle ensuite la fonction **WcsDisassociateColorProfileFromDevice pour dissocier** le troisième profil (qui est la valeur par défaut dans cet exemple), WCS utilise le deuxième profil comme profil par défaut.

Si votre application dissocie tous les profils d'un appareil, WCS utilise le profil sRGB comme profil par défaut.

Si *profileManagementScope* est WCS_PROFILE_MANAGEMENT_SCOPE_CURRENT_USER, cette fonction est exécutable dans Least-Privileged contexte compte d'utilisateur (LUA). Dans le cas contraire, des privilèges d'administration sont requis.

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib

DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- Schémas et algorithmes du système de couleurs Windows
- WcsAssociateColorProfileWithDevice

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

WcsEnumColorProfiles, fonction (icm.h)

Article24/08/2023

Énumère les profils de couleur associés à n'importe quel appareil, dans l'étendue spécifiée.

> ⓘ **Notes**
>
> Cette API ne prend pas en charge les profils de couleur avancés pour les moniteurs HDR. Utilisez **ColorProfileGetDisplayList** pour gérer les profils de couleurs avancés.

Syntaxe

C++

```
BOOL WcsEnumColorProfiles(
  WCS_PROFILE_MANAGEMENT_SCOPE scope,
  PENUMTYPEW                   pEnumRecord,
  PBYTE                        pBuffer,
  DWORD                        dwSize,
  PDWORD                       pnProfiles
);
```

Paramètres

`scope`

Valeur **WCS_PROFILE_MANAGEMENT_SCOPE** spécifiant l'étendue de cette opération de gestion de profil.

`pEnumRecord`

Pointeur vers une structure spécifiant les critères d'énumération.

`pBuffer`

Pointeur vers une mémoire tampon dans laquelle les noms de profil doivent être énumérés. La fonction **WcsEnumColorProfiles** place, dans cette mémoire tampon, une chaîne MULTI_SZ qui se compose de noms de profil qui répondent aux critères spécifiés dans **pEnumRecord*.

`dwSize`

Variable qui contient la taille, en octets, de la mémoire tampon pointée par *pBuffer*. Consultez **Remarques**.

`pnProfiles`

Pointeur facultatif vers une variable qui reçoit le nombre de noms de profil copiés dans la mémoire tampon vers laquelle *pBuffer* pointe. Peut avoir la valeur **NULL** si ces informations ne sont pas nécessaires.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Notes

Utilisez la fonction **WcsEnumColorProfilesSize** pour récupérer la valeur du paramètre *dwSize* , qui est la taille, en octets, de la mémoire tampon vers laquelle pointe le paramètre *pBuffer* .

Si le paramètre *profileManagementScope* est WCS_PROFILE_MANAGEMENT_SCOPE_SYSTEM_WIDE, seules les associations de profils à l'échelle du système à l'appareil sont prises en compte. Si *profileManagementScope* est WCS_PROFILE_MANAGEMENT_SCOPE_CURRENT_USER, seules les associations par utilisateur pour l'utilisateur actuel sont prises en compte. Si **WcsSetUsePerUserProfiles** n'a jamais été appelé pour cet utilisateur, ou si **WcsSetUsePerUserProfiles** a été récemment appelé pour cet utilisateur avec son paramètre *usePerUserProfiles* défini sur **FALSE**, **WCSEnumColorProfiles** retourne une liste vide.

Si WCS_PROFILE_MANAGEMENT_SCOPE_CURRENT_USER (le paramètre utilisateur actuel) est présent, il remplace la valeur par défaut à l'échelle du système pour le paramètre *profileManagementScope* .

Cette fonction est exécutable dans Least-Privileged contexte de compte d'utilisateur (LUA).

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- Schémas et algorithmes windows Color System
- WCS_PROFILE_MANAGEMENT_SCOPE
- WcsEnumColorProfilesSize

Commentaires

WcsEnumColorProfilesSize, fonction (icm.h)

Article24/08/2023

Retourne la taille, en octets, de la mémoire tampon requise par la fonction WcsEnumColorProfiles pour énumérer les profils de couleurs.

> ⓘ Notes
>
> Cette API ne prend pas en charge les profils de couleur avancés pour les moniteurs HDR. Utilisez **ColorProfileGetDisplayList** pour gérer les profils de couleurs avancés.

Syntaxe

C++

```
BOOL WcsEnumColorProfilesSize(
  WCS_PROFILE_MANAGEMENT_SCOPE scope,
  PENUMTYPEW                   pEnumRecord,
  PDWORD                       pdwSize
);
```

Paramètres

`scope`

Valeur WCS_PROFILE_MANAGEMENT_SCOPE qui spécifie l'étendue de l'opération de gestion de profil effectuée par cette fonction.

`pEnumRecord`

Pointeur vers une structure qui spécifie les critères d'énumération.

`pdwSize`

Pointeur vers une variable qui reçoit la taille de la mémoire tampon requise pour recevoir tous les noms de profil énumérés. Cette valeur est utilisée par le paramètre *dwSize* de la fonction WcsEnumColorProfiles .

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Notes

Cette fonction est exécutable dans Least-Privileged contexte de compte d'utilisateur (LUA).

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- Schémas et algorithmes windows Color System
- WcsEnumColorProfiles
- WCS_PROFILE_MANAGEMENT_SCOPE

Commentaires

WcsGetCalibrationManagementState, fonction (icm.h)

Article26/02/2024

Détermine si la gestion système de l'état d'étalonnage de l'affichage est activée.

Syntaxe

```
C++
```

```
BOOL WcsGetCalibrationManagementState(
  BOOL *pbIsEnabled
);
```

Paramètres

`pbIsEnabled`

TRUE si la gestion système de l'état d'étalonnage de l'affichage est activée ; sinon , **FALSE**.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Condition requise	Valeur
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

Cette page a-t-elle été utile ? Yes No

Indiquer des commentaires sur le produit | Obtenir de l'aide sur Microsoft Q&A

WcsGetDefaultColorProfile, fonction (icm.h)

Article24/08/2023

Récupère le profil de couleur par défaut pour un appareil ou pour une valeur par défaut indépendante de l'appareil si l'appareil n'est pas spécifié.

> ⓘ **Notes**
>
> Cette API ne prend pas en charge les profils de « couleur avancée » pour les moniteurs HDR. Utilisez **ColorProfileGetDisplayDefault** pour gérer les profils de couleurs avancés.

Syntaxe

C++

```
BOOL WcsGetDefaultColorProfile(
  WCS_PROFILE_MANAGEMENT_SCOPE scope,
  PCWSTR                       pDeviceName,
  COLORPROFILETYPE             cptColorProfileType,
  COLORPROFILESUBTYPE          cpstColorProfileSubType,
  DWORD                        dwProfileID,
  DWORD                        cbProfileName,
  LPWSTR                       pProfileName
);
```

Paramètres

`scope`

Valeur **WCS_PROFILE_MANAGEMENT_SCOPE** spécifiant l'étendue de cette opération de gestion de profil.

`pDeviceName`

Pointeur vers le nom de l'appareil pour lequel le profil de couleur par défaut est obtenu. Si **la valeur est NULL**, un profil par défaut indépendant de l'appareil est obtenu.

`cptColorProfileType`

Valeur **COLORPROFILETYPE** spécifiant le type de profil de couleur.

`cpstColorProfileSubType`

Valeur **COLORPROFILESUBTYPE** spécifiant le sous-type de profil de couleur.

`dwProfileID`

ID de l'espace de couleurs que représente le profil de couleur.

`cbProfileName`

Taille de la mémoire tampon, en octets, de la mémoire tampon pointée par *pProfileName*.

`pProfileName`

Pointeur vers une mémoire tampon pour recevoir le nom du profil de couleur. La taille de la mémoire tampon, en octets, sera indiquée par *cbProfileName*.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Notes

Utilisez la fonction **WcsGetDefaultColorProfileSize** pour obtenir la taille requise de la mémoire tampon pointée par le paramètre *pProfileName* .

Si WCS_PROFILE_MANAGEMENT_SCOPE_CURRENT_USER est présent, il remplace la valeur par défaut à l'échelle du système pour *profileManagementScope*.

Cette fonction est exécutable dans Least-Privileged contexte de compte d'utilisateur (LUA).

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau

	uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Schémas et algorithmes du système de couleurs Windows
- Fonctions
- COLORPROFILESUBTYPE
- COLORPROFILETYPE
- WCS_PROFILE_MANAGEMENT_SCOPE
- WcsGetDefaultColorProfileSize

Commentaires

WcsGetDefaultColorProfileSize, fonction (icm.h)

Article26/02/2024

Retourne la taille, en octets, du nom de profil de couleur par défaut (y compris la terminaison **NULL**) d'un appareil.

> ⓘ **Notes**
>
> Cette API ne prend pas en charge les profils de « couleur avancée » pour les moniteurs HDR. Utilisez **ColorProfileGetDisplayDefault** pour gérer les profils de couleurs avancés.

Syntaxe

```
C++

BOOL WcsGetDefaultColorProfileSize(
  WCS_PROFILE_MANAGEMENT_SCOPE scope,
  PCWSTR                       pDeviceName,
  COLORPROFILETYPE             cptColorProfileType,
  COLORPROFILESUBTYPE          cpstColorProfileSubType,
  DWORD                        dwProfileID,
  PDWORD                       pcbProfileName
);
```

Paramètres

`scope`

Valeur **WCS_PROFILE_MANAGEMENT_SCOPE** qui spécifie l'étendue de cette opération de gestion de profil.

`pDeviceName`

Pointeur vers le nom de l'appareil pour lequel le profil de couleur par défaut doit être obtenu. Si **la valeur est NULL**, un profil par défaut indépendant de l'appareil est utilisé.

`cptColorProfileType`

Valeur **COLORPROFILETYPE** spécifiant le type de profil de couleur.

`cpstColorProfileSubType`

Valeur **COLORPROFILESUBTYPE** spécifiant le sous-type de profil de couleur.

`dwProfileID`

ID de l'espace de couleurs que représente le profil de couleur.

`pcbProfileName`

Pointeur vers un emplacement qui reçoit la taille, en octets, du nom du chemin d'accès du profil de couleur par défaut, y compris la terminaison **NULL** .

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Remarques

Utilisez cette fonction pour renvoyer la taille requise de la mémoire tampon pointée par le paramètre *pProfileName* dans la fonction **WcsGetDefaultColorProfile** .

Cette fonction est exécutable dans Least-Privileged contexte de compte d'utilisateur (LUA).

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib

Condition requise	Valeur
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Schémas et algorithmes du système de couleurs Windows
- Fonctions
- COLORPROFILESUBTYPE
- COLORPROFILETYPE
- WCS_PROFILE_MANAGEMENT_SCOPE
- WcsGetDefaultColorProfile

Commentaires

WcsGetDefaultRenderingIntent, fonction (icm.h)

Article24/08/2023

Récupère l'intention de rendu par défaut dans l'étendue de gestion de profil spécifiée.

Syntaxe

```
C++
```

```
BOOL WcsGetDefaultRenderingIntent(
  WCS_PROFILE_MANAGEMENT_SCOPE scope,
  PDWORD                       pdwRenderingIntent
);
```

Paramètres

`scope`

Étendue de gestion des profils pour cette opération, qui peut être à l'échelle du système ou de l'utilisateur actuel uniquement.

`pdwRenderingIntent`

Pointeur vers la variable qui contiendra l'intention de rendu.

Pour plus d'informations, consultez Intentions de rendu.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Notes

Cette fonction ne revient pas à l'étendue à l'échelle du système si vous ne définissez pas l'intention de rendu par défaut par utilisateur. Au lieu de cela, il échoue, ce qui permet au processus appelant de faire la distinction entre le paramètre par utilisateur et le

paramètre à l'échelle du système. Si l'intention de rendu par utilisateur ne peut pas être récupérée, appelez à nouveau cette fonction à l'échelle du système.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Schémas et algorithmes windows Color System
- Fonctions

Commentaires

WcsGetDefaultRenderingIntent, fonction (icm.h)

Article24/08/2023

Récupère l'intention de rendu par défaut dans l'étendue de gestion de profil spécifiée.

Syntaxe

```
C++
```

```
BOOL WcsGetDefaultRenderingIntent(
  WCS_PROFILE_MANAGEMENT_SCOPE scope,
  PDWORD                       pdwRenderingIntent
);
```

Paramètres

`scope`

Étendue de gestion des profils pour cette opération, qui peut être à l'échelle du système ou de l'utilisateur actuel uniquement.

`pdwRenderingIntent`

Pointeur vers la variable qui contiendra l'intention de rendu.

Pour plus d'informations, consultez Intentions de rendu.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Notes

Cette fonction ne revient pas à l'étendue à l'échelle du système si vous ne définissez pas l'intention de rendu par défaut par utilisateur. Au lieu de cela, il échoue, ce qui permet au processus appelant de faire la distinction entre le paramètre par utilisateur et le

paramètre à l'échelle du système. Si l'intention de rendu par utilisateur ne peut pas être récupérée, appelez à nouveau cette fonction à l'échelle du système.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Schémas et algorithmes windows Color System
- Fonctions

Commentaires

WcsOpenColorProfileA, fonction (icm.h)

Article24/08/2023

Crée un handle pour un profil de couleur spécifié.

Syntaxe

```
C++
```

```
HPROFILE WcsOpenColorProfileA(
  PPROFILE pCDMPProfile,
  PPROFILE pCAMPProfile,
  PPROFILE pGMMPProfile,
  DWORD    dwDesireAccess,
  DWORD    dwShareMode,
  DWORD    dwCreationMode,
  DWORD    dwFlags
);
```

Paramètres

`pCDMPProfile`

Pointeur vers un DMP WCS ou une structure de profil de couleur ICC spécifiant le profil. Vous pouvez libérer le pointeur *pCDMPProfile* après avoir créé le handle. Si le profil est ICC et que son membre **dwType** est défini sur DONT_USE_EMBEDDED_WCS_PROFILES, **WcsOpenColorProfile** ignore tout profil WCS incorporé dans le profil ICC.

`pCAMPProfile`

Pointeur vers une structure de profil qui spécifie un profil de modèle d'apparence de couleur WCS (CAMP). Vous pouvez libérer le pointeur *pCAMPProfile* après avoir créé le handle. Si **la** valeur est NULL, la valeur CAMP par défaut est utilisée et le paramètre utilisateur actuel, WCS_PROFILE_MANAGEMENT_SCOPE_CURRENT_USER, est utilisé lors de l'interrogation du CAMP par défaut.

`pGMMPProfile`

Pointeur vers une structure de profil qui spécifie un profil GMMP (Gamut Map Model Profile) WCS. Vous pouvez libérer le pointeur *pGMMPProfile* après avoir créé le handle. Si **la** valeur est NULL, la valeur GMMP par défaut de l'intention de rendu par défaut est utilisée et le paramètre utilisateur actuel,

WCS_PROFILE_MANAGEMENT_SCOPE_CURRENT_USER, est utilisé lors de l'interrogation du GMMP par défaut. Pour obtenir une description des intentions de rendu, consultez Intentions de rendu.

`dwDesireAccess`

Valeur d'indicateur qui spécifie comment accéder au profil de couleur spécifié. Ce paramètre doit prendre l'une des valeurs suivantes :

Valeur	Description
PROFILE_READ	Spécifie que le profil de couleurs s'ouvre pour un accès en lecture seule.
PROFILE_READWRITE	Spécifie que le profil de couleurs s'ouvre pour l'accès en lecture et en écriture. La valeur de cet indicateur est ignorée si le profil est un profil WCS.

`dwShareMode`

Valeur d'indicateur qui spécifie les actions à effectuer lors de l'ouverture d'un profil de couleurs contenu dans un fichier. Ce paramètre doit prendre l'une des valeurs suivantes, qui sont définies dans *winnt.h* :

Valeur	Description
FILE_SHARE_READ	Spécifie que vous pouvez effectuer d'autres opérations d'ouverture (pour l'accès en lecture) sur le profil.
FILE_SHARE_WRITE	Spécifie que vous pouvez effectuer d'autres opérations d'ouverture (pour l'accès en écriture) sur le profil. Cette valeur d'indicateur est ignorée lorsqu'un profil WCS est ouvert.

`dwCreationMode`

Valeur d'indicateur qui spécifie les actions à effectuer lors de l'ouverture d'un profil de couleurs s'il est contenu dans un fichier. Ce paramètre doit prendre l'une des valeurs suivantes, qui sont définies dans *winbase.h* :

Valeur	Description
CREATE_NEW	Spécifie qu'un nouveau profil est créé. Cette fonction échoue si le profil existe déjà.
CREATE_ALWAYS	Spécifie qu'un nouveau profil est créé. Si un profil existe déjà, il est remplacé.

Valeur	Description
OPEN_EXISTING	Spécifie que le profil est ouvert. Cette fonction échoue si le profil n'existe pas.
OPEN_ALWAYS	Spécifie que le profil doit être ouvert s'il existe un fichier ICC (International Color Consortium). S'il n'existe pas de profil ICC, WCS crée un profil ICC. La fonction échoue pour les profils WCS si cet indicateur est défini et qu'aucun profil WCS n'existe.
TRUNCATE_EXISTING	Spécifie que le profil doit être ouvert et tronqué à zéro octet. La fonction échoue si le profil n'existe pas.

`dwFlags`

Valeur d'indicateur qui spécifie s'il faut utiliser le profil WCS incorporé. Ce paramètre n'a aucun effet, sauf si *pCDMProfile* spécifie un profil ICC qui contient un profil WCS incorporé.

Ce paramètre prend l'une des valeurs suivantes :

Valeur	Description
0	Spécifie que le profil WCS incorporé sera utilisé et que le profil ICC spécifié par pCDMPProfile sera ignoré.
DONT_USE_EMBEDDED_WCS_PROFILES	Spécifie que le profil ICC spécifié par pCDMPProfile sera utilisé et que le profil WCS incorporé sera ignoré.

Valeur retournée

Si cette fonction réussit, la valeur de retour est le handle du profil de couleur ouvert.

Si cette fonction échoue, la valeur de retour est **NULL**.

Notes

Cette API accepte un ensemble de DMP, CAMP et GMMP et retourne un handle de profil WCS. **Les valeurs NULL** pour GMMP sont valides. Une valeur **NULL** pour CAMP est remplacée par la valeur CAMP par défaut.

Cette API accepte également les profils ICC. L'utilisation d'un profil ICC ne garantit pas le traitement par le moteur CITE wcs. Le moteur WCS ne sera utilisé que s'il est passé au moins un profil WCS. Les flux de travail ICC purs seront cohérents avec le comportement hérité.

Vous pouvez utiliser le handle retourné par cette fonction dans d'autres fonctions de gestion des profils de couleurs.

Les indicateurs *dwCreationMode* CREATE_NEW, CREATE_ALWAYS et TRUNCATE_EXISTING retournent toujours des HPROFILEs ICC vides. Si *d'autres indicateurs dwCreationMode* sont présents, la fonction détermine si le profil est ICC ou WCS XML.

Dans le chemin de code ICC, un fichier HPROFILE ICC est retourné à l'aide des indicateurs de partage, d'accès et de création demandés, comme spécifié dans les tableaux ci-dessus.

Dans le chemin d'accès WCS, l'indicateur *dwCreationMode* OPEN_ALWAYS échoue si le profil n'existe pas, car les profils WCS ne peuvent pas être créés ou modifiés dans l'architecture WCS (ils doivent être modifiés en dehors de celui-ci, à l'aide de MSXML6). Pour la même raison, *l'indicateur dwShareMode* FILE_SHARE_WRITE et l'indicateur *dwDesiredAccess* PROFILE_READWRITE sont ignorés dans le chemin WCS.

Une fois le handle du profil de couleur créé, toutes les informations utilisées pour créer ce handle peuvent être supprimées.

Utilisez la fonction **CloseColorProfile** pour fermer un handle d'objet retourné par **WcsOpenColorProfile**.

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Schémas et algorithmes windows Color System
- Fonctions

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l’aide sur Microsoft Q&A

WcsOpenColorProfileW, fonction (icm.h)

Article24/08/2023

Crée un handle pour un profil de couleur spécifié.

Syntaxe

```
C++
```

```
HPROFILE WcsOpenColorProfileW(
  PPROFILE pCDMPProfile,
  PPROFILE pCAMPProfile,
  PPROFILE pGMMPProfile,
  DWORD    dwDesireAccess,
  DWORD    dwShareMode,
  DWORD    dwCreationMode,
  DWORD    dwFlags
);
```

Paramètres

`pCDMPProfile`

Pointeur vers un DMP WCS ou une structure de profil de couleur ICC spécifiant le profil. Vous pouvez libérer le pointeur *pCDMPProfile* après avoir créé le handle. Si le profil est ICC et que son membre **dwType** est défini sur DONT_USE_EMBEDDED_WCS_PROFILES, **WcsOpenColorProfile** ignore tout profil WCS incorporé dans le profil ICC.

`pCAMPProfile`

Pointeur vers une structure de profil qui spécifie un profil de modèle d'apparence de couleur WCS (CAMP). Vous pouvez libérer le pointeur *pCAMPProfile* après avoir créé le handle. Si **la** valeur est NULL, la valeur CAMP par défaut est utilisée et le paramètre utilisateur actuel, WCS_PROFILE_MANAGEMENT_SCOPE_CURRENT_USER, est utilisé lors de l'interrogation du CAMP par défaut.

`pGMMPProfile`

Pointeur vers une structure de profil qui spécifie un profil GMMP (Gamut Map Model Profile) WCS. Vous pouvez libérer le pointeur *pGMMPProfile* après avoir créé le handle. Si **la** valeur est NULL, la valeur GMMP par défaut de l'intention de rendu par défaut est utilisée et le paramètre utilisateur actuel,

WCS_PROFILE_MANAGEMENT_SCOPE_CURRENT_USER, est utilisé lors de l'interrogation du GMMP par défaut. Pour obtenir une description des intentions de rendu, consultez Intentions de rendu.

`dwDesireAccess`

Valeur d'indicateur qui spécifie comment accéder au profil de couleur spécifié. Ce paramètre doit prendre l'une des valeurs suivantes :

Valeur	Description
PROFILE_READ	Spécifie que le profil de couleurs s'ouvre pour un accès en lecture seule.
PROFILE_READWRITE	Spécifie que le profil de couleurs s'ouvre pour l'accès en lecture et en écriture. La valeur de cet indicateur est ignorée si le profil est un profil WCS.

`dwShareMode`

Valeur d'indicateur qui spécifie les actions à effectuer lors de l'ouverture d'un profil de couleurs contenu dans un fichier. Ce paramètre doit prendre l'une des valeurs suivantes, qui sont définies dans *winnt.h* :

Valeur	Description
FILE_SHARE_READ	Spécifie que vous pouvez effectuer d'autres opérations d'ouverture (pour l'accès en lecture) sur le profil.
FILE_SHARE_WRITE	Spécifie que vous pouvez effectuer d'autres opérations d'ouverture (pour l'accès en écriture) sur le profil. Cette valeur d'indicateur est ignorée lorsqu'un profil WCS est ouvert.

`dwCreationMode`

Valeur d'indicateur qui spécifie les actions à effectuer lors de l'ouverture d'un profil de couleurs s'il est contenu dans un fichier. Ce paramètre doit prendre l'une des valeurs suivantes, qui sont définies dans *winbase.h* :

Valeur	Description
CREATE_NEW	Spécifie qu'un nouveau profil est créé. Cette fonction échoue si le profil existe déjà.
CREATE_ALWAYS	Spécifie qu'un nouveau profil est créé. Si un profil existe déjà, il est remplacé.

Valeur	Description
OPEN_EXISTING	Spécifie que le profil est ouvert. Cette fonction échoue si le profil n'existe pas.
OPEN_ALWAYS	Spécifie que le profil doit être ouvert s'il existe un fichier ICC (International Color Consortium). S'il n'existe pas de profil ICC, WCS crée un profil ICC. La fonction échoue pour les profils WCS si cet indicateur est défini et qu'aucun profil WCS n'existe.
TRUNCATE_EXISTING	Spécifie que le profil doit être ouvert et tronqué à zéro octet. La fonction échoue si le profil n'existe pas.

`dwFlags`

Valeur d'indicateur qui spécifie s'il faut utiliser le profil WCS incorporé. Ce paramètre n'a aucun effet, sauf si *pCDMProfile* spécifie un profil ICC qui contient un profil WCS incorporé.

Ce paramètre prend l'une des valeurs suivantes :

Valeur	Description
0	Spécifie que le profil WCS incorporé sera utilisé et que le profil ICC spécifié par pCDMPProfile sera ignoré.
DONT_USE_EMBEDDED_WCS_PROFILES	Spécifie que le profil ICC spécifié par pCDMPProfile sera utilisé et que le profil WCS incorporé sera ignoré.

Valeur retournée

Si cette fonction réussit, la valeur de retour est le handle du profil de couleur ouvert.

Si cette fonction échoue, la valeur de retour est **NULL**.

Notes

Cette API accepte un ensemble de DMP, CAMP et GMMP et retourne un handle de profil WCS. **Les valeurs NULL** pour GMMP sont valides. Une valeur **NULL** pour CAMP est remplacée par la valeur CAMP par défaut.

Cette API accepte également les profils ICC. L'utilisation d'un profil ICC ne garantit pas le traitement par le moteur CITE wcs. Le moteur WCS ne sera utilisé que s'il est passé au moins un profil WCS. Les flux de travail ICC purs seront cohérents avec le comportement hérité.

Vous pouvez utiliser le handle retourné par cette fonction dans d'autres fonctions de gestion des profils de couleurs.

Les indicateurs *dwCreationMode* CREATE_NEW, CREATE_ALWAYS et TRUNCATE_EXISTING retournent toujours des HPROFILEs ICC vides. Si *d'autres indicateurs dwCreationMode* sont présents, la fonction détermine si le profil est ICC ou WCS XML.

Dans le chemin de code ICC, un fichier HPROFILE ICC est retourné à l'aide des indicateurs de partage, d'accès et de création demandés, comme spécifié dans les tableaux ci-dessus.

Dans le chemin d'accès WCS, l'indicateur *dwCreationMode* OPEN_ALWAYS échoue si le profil n'existe pas, car les profils WCS ne peuvent pas être créés ou modifiés dans l'architecture WCS (ils doivent être modifiés en dehors de celui-ci, à l'aide de MSXML6). Pour la même raison, *l'indicateur dwShareMode* FILE_SHARE_WRITE et l'indicateur *dwDesiredAccess* PROFILE_READWRITE sont ignorés dans le chemin WCS.

Une fois le handle du profil de couleur créé, toutes les informations utilisées pour créer ce handle peuvent être supprimées.

Utilisez la fonction **CloseColorProfile** pour fermer un handle d'objet retourné par **WcsOpenColorProfile**.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Schémas et algorithmes windows Color System
- Fonctions

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l’aide sur Microsoft Q&A

WcsSetCalibrationManagementState, fonction (icm.h)

Article26/02/2024

Active ou désactive la gestion système de l'état d'étalonnage de l'affichage.

Syntaxe

```
C++

BOOL WcsSetCalibrationManagementState(
  BOOL bIsEnabled
);
```

Paramètres

`bIsEnabled`

TRUE pour activer la gestion système de l'état d'étalonnage de l'affichage. **FALSE** pour désactiver la gestion système de l'état d'étalonnage de l'affichage.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**.

Remarques

Cette fonction doit être appelée avec des autorisations élevées pour qu'elle réussisse.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau

Condition requise	Valeur
	uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions

Commentaires

WcsSetDefaultColorProfile, fonction (icm.h)

Article24/08/2023

Définit le nom du profil de couleur par défaut pour le type de profil spécifié dans l'étendue de gestion de profil spécifiée.

> ⓘ **Notes**
>
> Cette API ne prend pas en charge les profils de « couleur avancée » pour les moniteurs HDR. Utilisez **ColorProfileSetDisplayDefaultAssociation** pour gérer les profils de couleur avancés.

Syntaxe

```
C++

BOOL WcsSetDefaultColorProfile(
  WCS_PROFILE_MANAGEMENT_SCOPE scope,
  PCWSTR                       pDeviceName,
  COLORPROFILETYPE             cptColorProfileType,
  COLORPROFILESUBTYPE          cpstColorProfileSubType,
  DWORD                        dwProfileID,
  LPCWSTR                      pProfileName
);
```

Paramètres

`scope`

Valeur **WCS_PROFILE_MANAGEMENT_SCOPE** qui spécifie l'étendue de cette opération de gestion de profil.

`pDeviceName`

Pointeur vers le nom de l'appareil pour lequel le profil de couleur par défaut doit être défini. Si **la valeur est NULL**, un profil par défaut indépendant de l'appareil est utilisé.

`cptColorProfileType`

Valeur **COLORPROFILETYPE** qui spécifie le type de profil de couleur.

`cpstColorProfileSubType`

Valeur **COLORPROFILESUBTYPE** qui spécifie le sous-type de profil de couleur.

`dwProfileID`

ID de l'espace de couleurs que représente le profil de couleur. Il s'agit d'une valeur d'ID personnalisée utilisée pour identifier de manière unique le profil d'espace de couleurs au sein de votre application.

`pProfileName`

Pointeur vers une mémoire tampon qui contient le nom du profil de couleur. Consultez la section Notes.

Valeur de retour

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, appelez GetLastError.

Notes

Si le paramètre *pProfileName* a la valeur **NULL** et que le paramètre *profileManagementScope* est WCS_PROFILE_MANAGEMENT_SCOPE_CURRENT_USER, les appels suivants à **WcsSetDefaultColorProfile** retournent le profil par défaut à l'échelle du système.

Si *profileManagementScope* est WCS_PROFILE_MANAGEMENT_SCOPE_CURRENT_USER, cette fonction est exécutable dans Least-Privileged contexte compte d'utilisateur (LUA). Dans le cas contraire, des privilèges d'administration sont requis. Le profil spécifié doit déjà être installé, mais il n'est peut-être pas encore associé à l'appareil spécifié dans l'étendue de gestion de profil spécifiée.

Si *profileManagementScope* est WCS_PROFILE_MANAGEMENT_SCOPE_CURRENT_USER, cette fonction ne fonctionnera pas correctement si elle est lancée à partir du contexte système et non d'un compte d'utilisateur.

Lorsque WcsSetDefaultColorProfile est appelé pour définir un profil de modèle d'appareil DMP comme profil par défaut pour l'espace de travail RVB ou personnalisé, seul un profil DMP de type RGBVirtualDevice, LCD ou CRT est valide ; toutes les autres ne sont pas valides.

Lorsque WcsSetDefaultColorProfile est appelé pour définir un profil ICC (International Color Consortium) comme profil par défaut pour l'espace de travail RVB ou personnalisé, seul un profil ICC avec la classe « spac » ou « disp » et l'espace de couleur « RVB » est valide ; toutes les autres ne sont pas valides.

Consultez les notes sur les combinaisons valides de type/sous-type de profil.

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Schémas et algorithmes du système de couleurs Windows
- Fonctions
- COLORPROFILESUBTYPE
- COLORPROFILETYPE
- WCS_PROFILE_MANAGEMENT_SCOPE
- WcsGetDefaultColorProfileSize

Commentaires

WcsSetDefaultRenderingIntent, fonction (icm.h)

Article26/02/2024

Définit l'intention de rendu par défaut dans l'étendue de gestion de profil spécifiée.

Syntaxe

```
C++

BOOL WcsSetDefaultRenderingIntent(
  WCS_PROFILE_MANAGEMENT_SCOPE scope,
  DWORD                        dwRenderingIntent
);
```

Paramètres

`scope`

Étendue de gestion des profils pour cette opération, qui peut être à l'échelle du système ou de l'utilisateur actuel uniquement.

`dwRenderingIntent`

Intention de rendu. Il peut être défini sur l'une des valeurs suivantes :

INTENT_PERCEPTUAL

INTENT_RELATIVE_COLORIMETRIC

INTENT_SATURATION

INTENT_ABSOLUTE_COLORIMETRIC

DWORD_MAX

Si *dwRenderingIntent* est DWORD_MAX et que *l'étendue* est WCS_PROFILE_MANAGEMENT_SCOPE_CURRENT_USER, l'intention de rendu par défaut de l'utilisateur actuel revient à la valeur par défaut à l'échelle du système.

Pour plus d'informations, consultez Intentions de rendu.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Schémas et algorithmes du système de couleurs Windows
- Fonctions

Commentaires

WcsSetUsePerUserProfiles, fonction (icm.h)

Article26/02/2024

Permet à un utilisateur de spécifier s'il faut ou non utiliser une liste d'association de profils par utilisateur pour l'appareil spécifié.

Syntaxe

```
C++

BOOL WcsSetUsePerUserProfiles(
  LPCWSTR pDeviceName,
  DWORD   dwDeviceClass,
  BOOL    usePerUserProfiles
);
```

Paramètres

`pDeviceName`

Pointeur vers une chaîne qui contient le nom convivial de l'appareil.

`dwDeviceClass`

Valeur d'indicateur qui spécifie la classe de l'appareil. Ce paramètre doit prendre l'une des valeurs suivantes :

Agrandir le tableau

Valeur	Description
CLASS_MONITOR	Spécifie un périphérique d'affichage.
CLASS_PRINTER	Spécifie une imprimante.
CLASS_SCANNER	Spécifie un appareil de capture d'image.

`usePerUserProfiles`

Valeur booléenne **true** si l'utilisateur souhaite utiliser une liste d'association de profils par utilisateur pour l'appareil spécifié ; sinon , **FALSE**.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Remarques

Si *usePerUserProfiles* a la valeur **TRUE** et que l'utilisateur n'utilise pas déjà une liste d'associations de profils par utilisateur pour *pDeviceName*, la liste d'associations de profils par utilisateur est initialisée en effectuant une copie de la liste d'association de profils à l'échelle du système pour le même appareil. À partir de là, les modifications apportées à la liste à l'échelle du système ne sont pas incluses dans la liste par utilisateur.

Cette fonction est exécutable dans Least-Privileged contexte de compte d'utilisateur (LUA).

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Schémas et algorithmes du système de couleurs Windows
- Fonctions
- WcsGetUsePerUserProfiles

Commentaires

Cette page a-t-elle été utile ? Yes No

Indiquer des commentaires sur le produit | Obtenir de l'aide sur Microsoft Q&A

WcsTranslateColors, fonction (icm.h)

Article26/02/2024

Traduit un tableau de couleurs de l'espace de couleur source en espace de couleur de destination tel que défini par une transformation de couleur.

Syntaxe

```
C++

BOOL WcsTranslateColors(
  HTRANSFORM    hColorTransform,
  DWORD         nColors,
  DWORD         nInputChannels,
  COLORDATATYPE cdtInput,
  DWORD         cbInput,
  PVOID         pInputData,
  DWORD         nOutputChannels,
  COLORDATATYPE cdtOutput,
  DWORD         cbOutput,
  PVOID         pOutputData
);
```

Paramètres

`hColorTransform`

Handle pour la transformation de couleur WCS.

`nColors`

Nombre d'éléments dans le tableau sur lesquels *pInputData* et *pOutputData* pointent.

`nInputChannels`

Nombre de canaux par élément dans le tableau vers lequel *pointe pInputData* .

`cdtInput`

Type **de données de couleur COLORDATATYPE** d'entrée.

`cbInput`

Taille de la mémoire tampon, en octets, de *pInputData*.

`pInputData`

Pointeur vers un tableau de couleurs d'entrée. La taille de la mémoire tampon pour ce tableau, en octets, est la valeur **DWORD** de *cbInput*.

`nOutputChannels`

Nombre de canaux par élément dans le tableau vers lequel *pOutputData* pointe.

`cdtOutput`

Sortie **COLORDATATYPE** qui a spécifié le type de données de couleur.

`cbOutput`

Taille de la mémoire tampon, en octets, de *pOutputData*.

`pOutputData`

Pointeur vers un tableau de couleurs qui reçoit les résultats de la traduction de couleurs. La taille de la mémoire tampon pour ce tableau, en octets, est la valeur **DWORD** de *cbOutput*.

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Remarques

Si les types de données d'entrée et de couleur de sortie ne sont pas compatibles avec la transformation de couleur, cette fonction échoue. Cette fonction échoue si une transformation ICC est utilisée.

Configuration requise

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h
Bibliothèque	Mscms.lib
DLL	Mscms.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Schémas et algorithmes du système de couleurs Windows
- Fonctions

Commentaires

Fonctions WCS obsolètes

Article • 13/06/2023

Cette section délimite les fonctions obsolètes dans WCS 1.0. Ils sont actuellement conservés à des fins de compatibilité descendante. Toutefois, ils seront supprimés de l'API dans les versions ultérieures de WCS.

Fonction	Description
UpdateICMRegKey	*Obsolète* : gère les profils de couleur et les MCM.

Commentaires

UpdateICMRegKeyA, fonction (wingdi.h)

Article27/08/2023

(Obsolète ; conservé pour la compatibilité descendante)

La fonction **UpdateICMRegKey** gère les profils de couleurs et les modules de gestion des couleurs dans le système.

Syntaxe

```
C++

BOOL UpdateICMRegKeyA(
  DWORD reserved,
  LPSTR lpszCMID,
  LPSTR lpszFileName,
  UINT  command
);
```

Paramètres

`reserved`

Réservé, doit être défini sur zéro.

`lpszCMID`

Pointe vers une chaîne qui spécifie l'identificateur de profil ICC pour la DLL de gestion des couleurs à utiliser avec le profil.

`lpszFileName`

Pointe vers un nom de fichier de profil de couleur ICC complet ou vers une structure **DEVMODE** .

`command`

Spécifie une fonction à exécuter. Il peut avoir l'une des valeurs suivantes.

Valeur	Signification
ICM_ADDPROFILE	Installe le profil ICC dans le système.

ICM_DELETEPROFILE	Désinstalle le profil ICC du système, mais ne supprime pas le fichier.
ICM_QUERYPROFILE	Détermine si le profil est déjà installé dans le système.
ICM_SETDEFAULTPROFILE	Place le profil en premier parmi les égaux.
ICM_REGISTERICMATCHER	Inscrit une CMM dans le système. Le paramètre *pszFileName* pointe vers un chemin d'accès complet pour la DLL CMM. Le paramètre *lpszCMID* pointe vers un **DWORD** identifiant la CMM.
ICM_UNREGISTERICMATCHER	Annule l'inscription de la CMM du système. Le paramètre *lpszCMID* pointe vers un **DWORD** identifiant la CMM.
ICM_QUERYMATCH	Détermine si un profil existe en fonction de la structure **DEVMODE** pointée par le paramètre *pszFileName* .

Valeur retournée

Si cette fonction réussit, la valeur de retour est **TRUE**.

Si cette fonction échoue, la valeur de retour est **FALSE**.

Remarques

Tous les paramètres ne sont pas utilisés par toutes les fonctions. Le paramètre *nCommand* spécifie la fonction à exécuter.

Cette fonction est conservée à des fins de compatibilité descendante et peut être supprimée dans les versions ultérieures d'ICM.

Windows 95/98/Me : UpdateICMRegKeyW est pris en charge par Microsoft Layer pour Unicode. Pour l'utiliser, vous devez ajouter certains fichiers à votre application, comme indiqué dans Microsoft Layer pour Unicode sur les systèmes Windows 95/98/Me .

> **Notes**
>
> L'en-tête wingdi.h définit UpdateICMRegKey comme alias qui sélectionne automatiquement la version ANSI ou Unicode de cette fonction en fonction de la définition de la constante de préprocesseur UNICODE. La combinaison de l'utilisation de l'alias neutre en encodage avec du code qui n'est pas neutre en encodage peut entraîner des incompatibilités qui entraînent des erreurs de

compilation ou d'exécution. Pour plus d'informations, consultez **Conventions pour les prototypes de fonction**.

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
Plateforme cible	Windows
En-tête	wingdi.h
Bibliothèque	Gdi32.lib
DLL	Gdi32.dll

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions WCS obsolètes

Commentaires

Énumérations WCS 1.0

Article • 13/06/2023

Cette section de la référence contient une liste alphabétique des énumérations les plus importantes utilisées par WCS 1.0 :

- BMFORMAT
- COLORDATATYPE
- COLORPROFILESUBTYPE
- COLORPROFILETYPE
- COLORTYPE
- WCS_PROFILE_MANAGEMENT_SCOPE

Commentaires

Énumération BMFORMAT (icm.h)

Article29/02/2024

Les valeurs du type énuméré **BMFORMAT** sont utilisées par plusieurs fonctions WCS pour indiquer le format dans lequel se trouvent des bitmaps particulières.

Syntax

C++

```
typedef enum {
  BM_x555RGB = 0x0000,
  BM_x555XYZ = 0x0101,
  BM_x555Yxy,
  BM_x555Lab,
  BM_x555G3CH,
  BM_RGBTRIPLETS = 0x0002,
  BM_BGRTRIPLETS = 0x0004,
  BM_XYZTRIPLETS = 0x0201,
  BM_YxyTRIPLETS,
  BM_LabTRIPLETS,
  BM_G3CHTRIPLETS,
  BM_5CHANNEL,
  BM_6CHANNEL,
  BM_7CHANNEL,
  BM_8CHANNEL,
  BM_GRAY,
  BM_xRGBQUADS = 0x0008,
  BM_xBGRQUADS = 0x0010,
  BM_xG3CHQUADS = 0x0304,
  BM_KYMCQUADS,
  BM_CMYKQUADS = 0x0020,
  BM_10b_RGB = 0x0009,
  BM_10b_XYZ = 0x0401,
  BM_10b_Yxy,
  BM_10b_Lab,
  BM_10b_G3CH,
  BM_NAMED_INDEX,
  BM_16b_RGB = 0x000A,
  BM_16b_XYZ = 0x0501,
  BM_16b_Yxy,
  BM_16b_Lab,
  BM_16b_G3CH,
  BM_16b_GRAY,
  BM_565RGB = 0x0001,
  BM_32b_scRGB = 0x0601,
  BM_32b_scARGB = 0x0602,
  BM_S2DOT13FIXED_scRGB = 0x0603,
  BM_S2DOT13FIXED_scARGB = 0x0604,
  BM_R10G10B10A2 = 0x0701,
  BM_R10G10B10A2_XR = 0x0702,
  BM_R16G16B16A16_FLOAT = 0x0703
} BMFORMAT;
```

Constantes

Agrandir le tableau

`BM_x555RGB` Valeur : *0x0000* 16 bits par pixel. Espace de couleur RVB. 5 bits par canal. Le bit le plus significatif est ignoré.
`BM_x555XYZ` Valeur : *0x0101* 16 bits par pixel. Espace de couleur XYZ indépendant de l'appareil CIE. 5 bits par canal. Le bit le plus significatif est ignoré.
`BM_x555Yxy` 16 bits par pixel. Espace de couleur Yxy. 5 bits par canal. Le bit le plus significatif est ignoré.
`BM_x555Lab`

16 bits par pixel. Espace de couleur L*a*b. 5 bits par canal. Le bit le plus significatif est ignoré.
`BM_x555G3CH` 16 bits par pixel. Espace de couleurs G3CH. 5 bits par canal. Le bit le plus significatif est ignoré.
`BM_RGBTRIPLETS` Valeur : *0x0002* 24 bits par pixel maximum. Pour trois couleurs de canal, telles que Rouge, Vert, Bleu, la taille totale est de 24 bits par pixel. Pour les couleurs à canal unique, telles que le gris, la taille totale est de 8 bits par pixel.
`BM_BGRTRIPLETS` Valeur : *0x0004* 24 bits par pixel maximum. Pour trois couleurs de canal, telles que Rouge, Vert, Bleu, la taille totale est de 24 bits par pixel. Pour les couleurs à canal unique, telles que le gris, la taille totale est de 8 bits par pixel.
`BM_XYZTRIPLETS` Valeur : *0x0201* 24 bits par pixel maximum. Pour trois canaux, X, Y et Z, la taille totale est de 24 bits par pixel. Pour l'échelle gris à canal unique, la taille totale est de 8 bits par pixel. **ⓘ Notes** La fonction **TranslateBitmapBitBits** ne prend pas en charge **BM_XYZTRIPLETS** en tant qu'entrée.
`BM_YxyTRIPLETS` 24 bits par pixel maximum. Pour les valeurs de trois canaux, Y, x et y, la taille totale est de 24 bits par pixel. Pour l'échelle gris à canal unique, la taille totale est de 8 bits par pixel. **ⓘ Notes** La fonction **TranslateBitmapBitBits** ne prend pas en charge **BM_YxyTRIPLETS** en tant qu'entrée.
`BM_LabTRIPLETS` 24 bits par pixel maximum. Pour les valeurs de trois canaux, L, a et b, la taille totale est de 24 bits par pixel. Pour l'échelle gris à canal unique, la taille totale est de 8 bits par pixel.
`BM_G3CHTRIPLETS` 24 bits par pixel maximum. Pour trois valeurs de canal, la taille totale est de 24 bits par pixel. Pour l'échelle gris à canal unique, la taille totale est de 8 bits par pixel.
`BM_5CHANNEL` 40 bits par pixel. 8 bits chacun sont utilisés pour chaque canal.
`BM_6CHANNEL` 48 bits par pixel. 8 bits chacun sont utilisés pour chaque canal.
`BM_7CHANNEL` 56 bits par pixel. 8 bits chacun sont utilisés pour chaque canal.
`BM_8CHANNEL` 64 bits par pixel. 8 bits chacun sont utilisés pour chaque canal.
`BM_GRAY` 32 bits par pixel. Seule la valeur d'échelle grise 8 bits est utilisée.
`BM_xRGBQUADS` Valeur : *0x0008* 32 bits par pixel. 8 bits sont utilisés pour chaque canal de couleur. L'octet le plus significatif est ignoré.
`BM_xBGRQUADS` Valeur : *0x0010* 32 bits par pixel. 8 bits sont utilisés pour chaque canal de couleur. L'octet le plus significatif est ignoré.
`BM_xG3CHQUADS` Valeur : *0x0304* 32 bits par pixel. 8 bits sont utilisés pour chaque canal de couleur. L'octet le plus significatif est ignoré.
`BM_KYMCQUADS` 32 bits par pixel. 8 bits sont utilisés pour chaque canal de couleur.

`BM_CMYKQUADS` Valeur : *0x0020* 32 bits par pixel. 8 bits sont utilisés pour chaque canal de couleur.
`BM_10b_RGB` Valeur : *0x0009* 32 bits par pixel. 10 bits sont utilisés pour chaque canal de couleur. Les 2 bits les plus significatifs sont ignorés.
`BM_10b_XYZ` Valeur : *0x0401* 32 bits par pixel. 10 bits sont utilisés pour chaque canal de couleur. Les 2 bits les plus significatifs sont ignorés.
`BM_10b_Yxy` 32 bits par pixel. 10 bits sont utilisés pour chaque canal de couleur. Les 2 bits les plus significatifs sont ignorés.
`BM_10b_Lab` 32 bits par pixel. 10 bits sont utilisés pour chaque canal de couleur. Les 2 bits les plus significatifs sont ignorés.
`BM_10b_G3CH` 32 bits par pixel. 10 bits sont utilisés pour chaque canal de couleur. Les 2 bits les plus significatifs sont ignorés.
`BM_NAMED_INDEX` 32 bits par pixel. Index de couleur nommés. La numérotation de l'index commence à 1.
`BM_16b_RGB` Valeur : *0x000A* 48 bits par pixel. Chaque canal utilise 16 bits.
`BM_16b_XYZ` Valeur : *0x0501* 48 bits par pixel. Chaque canal utilise 16 bits.
`BM_16b_Yxy` 48 bits par pixel. Chaque canal utilise 16 bits.
`BM_16b_Lab` 48 bits par pixel. Chaque canal utilise 16 bits.
`BM_16b_G3CH` 48 bits par pixel. Chaque canal utilise 16 bits.
`BM_16b_GRAY` 16 bits par pixel.
`BM_565RGB` Valeur : *0x0001* 16 bits par pixel. 5 bits sont utilisés pour le rouge, 6 pour le vert et 5 pour le bleu.
`BM_32b_scRGB` Valeur : *0x0601* 96 bits par pixel, 32 bits par canal IEEE à virgule flottante.
`BM_32b_scARGB` Valeur : *0x0602* 128 bits par pixel, 32 bits par canal IEEE à virgule flottante.
`BM_S2DOT13FIXED_scRGB` Valeur : *0x0603* 48 bits par pixel, entier à point fixe compris entre -4 et +4 avec un bit de signe et un exposant 2 bits et une mantisse de 13 bits.
`BM_S2DOT13FIXED_scARGB` Valeur : *0x0604* 64 bits par pixel, entier à point fixe compris entre -4 et +4 avec un bit de signe et un exposant 2 bits et une mantisse de 13 bits.
`BM_R10G10B10A2` Valeur : *0x0701* 32 bits par pixel. 10 bits sont utilisés pour chaque canal de couleur. Les deux bits les plus significatifs sont alpha.
`BM_R10G10B10A2_XR` Valeur : *0x0702* 32 bits par pixel. 10 bits sont utilisés pour chaque canal de couleur. Les 10 bits de chaque canal de couleur ont un point fixe de 2,8 avec un biais de -0,75, ce qui donne une plage de [-0,76 .. 1.25]. Cette plage correspond à [-0.5 .. 1,5] dans un gamma = 1 espace. Les deux bits les plus significatifs sont conservés pour alpha. Cela utilise un espace de couleur SRGB de plage étendue (XR). Il a les mêmes primaires RVB, point blanc et gamma que sRGB.

`BM_R16G16B16A16_FLOAT` Valeur : *0x0703* 64 bits par pixel. Chaque canal est un float 16 bits. Le dernier MOT est alpha.

Remarques

Tableau des formats bitmap

Le tableau suivant indique, pour chacun des formats, le nombre de bits par pixel, le nombre de canaux, l'ordre des canaux et la structure bit par bit de chaque octet. Vous devrez peut-être faire défiler vers la droite pour afficher toutes les colonnes de la table.

Agrandir le tableau

Format	Bits par pixel	Nombre de canaux	Ordre des canaux	Octet 0	Octet 1	Octet 2
BM_GRAY	8	1		K7K6K5K4K3K2K1K0		
BM_565RGB	16	3	BVR	G2G1G0B4B3B2B1B0	R4R3R2R1R0G5G4G3	
BM_x555RGB	16	3	BVR	G2G1G0B4B3B2B1B0	xR4R3R2R1R0G4G3	
BM_x555XYZ	16	3	ZYX	Y2Y1Y0Z4Z3Z2Z1Z0	xX4X3X2X1X0Y4Y3	
BM_x555Yxy	16	3	yxY	x2x1x0y4y3y2y1y0	xY4Y3Y2Y1Y0x4x3	
BM_x555Lab	16	3	baL	a2a1a0b4b3b2b1b0	xL4L3L2L1L0a4a3	
BM_x555G3CH	16	3	123	xC14C13C12C11C10C24C23	C22C21C20C34C33C32C31C30	
BM_16b_GRAY	16	1	K	K7K6K5K4K3K2K1K0	K15K14K13K12K11K10K9K8	
BM_RGBTRIPLETS	24	3	BVR	B7B6B5B4B3B2B1B0	G7G6G5G4G3G2G1G0	R7R6R5R4R3R2R1R0
BM_BGRTRIPLETS	24	3	RGB	R7R6R5R4R3R2R1R0	G7G6G5G4G3G2G1G0	B7B6B5B4B3B2B1B0
BM_XYZTRIPLETS	24	3	XYZ	X7X6X5X4X3X2X1X0	Y7Y6Y5Y4Y3Y2Y1Y0	Z7Z6Z5Z4Z3Z2Z1Z0
BM_YxyTRIPLETS	24	3	Yxy	Y7Y6Y5Y4Y3Y2Y1Y0	x7x6x5x4x3x2x1x0	y7y6y5y4y3y2y1y0
BM_LabTRIPLETS	24	3	Laboratoire	L7L6L5L4L3L2L1L0	a7a6a5a4a3a2a1a0	b7b6b5b4b3b2b1b0
BM_G3CHTRIPLETS	24	3	123	C17C16C15C14C13C12C11C10	C27C26C25C24C23C22C21C20	C37C36C35C34C33C32C31C
BM_xRGBQUADS	32	3	BGRx	B7B6B5B4B3B2B1B0	G7G6G5G4G3G2G1G0	R7R6R5R4R3R2R1R0
BM_xBGRQUADS	32	3	RGBx	R7R6R5R4R3R2R1R0	G7G6G5G4G3G2G1G0	B7B6B5B4B3B2B1B0
BM_xG3CHQUADS	32	3	123x	C17C16C15C14C13C12C11C10	C27C26C25C24C23C22C21C20	C37C36C35C34C33C32C31C
BM_CMYKQUADS	32	4	KYMC	K7K6K5K4K3K2K1K0	Y7Y6Y5Y4Y3Y2Y1Y0	M7M6M5M4M3M2M1M0
BM_KYMCQUADS	32	4	CMJN	C7C6C5C4C3C2C1C0	M7M6M5M4M3M2M1M0	Y7Y6Y5Y4Y3Y2Y1Y0
BM_10b_RGB	32	3	BVR	B7B6B5B4B3B2B1B0	G5G4G3G2G1G0B9B8	R3R2R1R0G9G8G7G6
BM_10b_XYZ	32	3	ZYX	Z7Z6Z5Z4Z3Z2Z1Z0	Y5Y4Y3Y2Y1Y0Z9Z8	X3X2X1X0Y9Y8Y7Y6
BM_10b_Yxy	32	3	YxY	y7y6y5y4y3y2y1y0	x5x4x3x2x1x0y9y8	Y3Y2Y1Y0x9x8x7x6
BM_10b_Lab	32	3	baL	b7b6b5b4b3b2b1b0	a5a4a3a2a1a0b9b8	L3L2L1L0a9a8a7a6
BM_10b_G3CH	32	3	321	C37C36C35C34C33C32C31C30	C25C24C23C22C21C20C39C38	C13C12C11C10C29C28C27C
BM_NAMED_INDEX	32			n7n6n5n4n3n2n1n0	n15n14n13n12n11n10n9n8	n23n22n21n20n19n18n17n1
BM_5CHANNEL	40	5	12345	C17C16C15C14C13C12C11C10	C27C26C25C24C23C22C21C20	C37C36C35C34C33C32C31C
BM_6CHANNEL	48	6	123456	C17C16C15C14C13C12C11C10	C27C26C25C24C23C22C21C20	C37C36C35C34C33C32C31C
BM_16b_RGB	48	3	RGB	R7R6R5R4R3R2R1R0	R15R14R13R12R11R10R9R8	G7G6G5G4G3G2G1G0
BM_16b_XYZ	48	3	XYZ	X7X6X5X4X3X2X1X0	X15X14X13X12X11X10X9X8	Y7Y6Y5Y4Y3Y2Y1Y0

Format	Bits par pixel	Nombre de canaux	Ordre des canaux	Octet 0	Octet 1	Octet 2
BM_16b_Lab	48	3	Laboratoire	L7L6L5L4L3L2L1L0	L15L14L13L12L11L10L9L8	a7a6a5a4a3a2a1a0
BM_16b_G3CH	48	3	321	C37C36C35C34C33C32C31C30	C315C314C313C312C311C310C39C38	C27C26C25C24C23C22C21C
BM_16b_Yxy	48	3	Yxy	Y7Y6Y5Y4Y3Y2Y1Y0	Y15Y14Y13Y12Y11Y10Y9Y8	x7x6x5x4x3x2x1x0
BM_7CHANNEL	56	7	1234567	C17C16C15C14C13C12C11C10	C27C26C25C24C23C22C21C20	C37C36C35C34C33C32C31C
BM_8CHANNEL	64	8	12345678	C17C16C15C14C13C12C11C10	C27C26C25C24C23C22C21C20	C37C36C35C34C33C32C31C
BM_32b_scRGB	96	3	BVR			
BM_32b_scARGB	128	3	BGRA			
BM_S2DOT13FIXED_scRGB	48	3	BVR			
BM_S2DOT13FIXED_scARGB	64	3	BGRA			
BM_R10G10B10A2	32	3	ABGR	A7A6B5B4B3B2B1B0	B7B6B5B4G3G2G1G0	G7G6G5G4G3G2R1R0
BM_R10G10B10A2_XR	32	3	ABGR	A7A6B5B4B3B2B1B0	B7B6B5B4G3G2G1G0	G7G6G5G4G3G2R1R0
BM_R16G16B16A16_FLOAT	64	3	RGBA	R7R6R5R4R3R2R1R0	R7R6R5R4R3R2R1R0	G7G6G5G4G3G2G1G0

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Commentaires

COLORDATATYPE, énumération (icm.h)

Article24/08/2023

Utilisé par les fonctions WCS pour indiquer le type de données du contenu vectoriel.

Syntax

C++

```
typedef enum {
  COLOR_BYTE = 1,
  COLOR_WORD,
  COLOR_FLOAT,
  COLOR_S2DOT13FIXED,
  COLOR_10b_R10G10B10A2,
  COLOR_10b_R10G10B10A2_XR,
  COLOR_FLOAT16
} COLORDATATYPE;
```

Constantes

`COLOR_BYTE` Valeur : *1* Les données de couleur sont stockées sous forme de 8 bits par canal, avec une valeur comprise entre 0 et 255, inclus.
`COLOR_WORD` Les données de couleur sont stockées sous forme de 16 bits par canal, avec une valeur comprise entre 0 et 65535, inclus.
`COLOR_FLOAT` Les données de couleur sont stockées sous la forme d'une valeur de 32 bits par canal, comme défini par la norme ieee à virgule flottante 32 bits.
`COLOR_S2DOT13FIXED` Les données de couleur sont stockées sous forme de 16 bits par canal, avec une plage fixe comprise entre -4 et +4, inclus. Un format signé est utilisé, avec 1 bit pour le signe, 2 bits pour la partie entière et 13 bits pour la partie fractionnaire.
`COLOR_10b_R10G10B10A2` Les données de couleur sont stockées sous forme de 10 bits par canal. Les deux bits les plus significatifs sont alpha.

`COLOR_10b_R10G10B10A2_XR` Les données de couleur sont stockées sous la forme de 10 bits par canal, 32 bits par pixel. Les 10 bits de chaque canal de couleur sont de 2,8 points fixes avec un biais de -0,75, ce qui donne une plage de [-0,76 .. 1.25]. Cette plage correspond à [-0,5 .. 1,5] dans un gamma = 1 espace. Les deux bits les plus significatifs sont conservés pour alpha. Cela utilise un espace de couleurs sRGB de plage étendue (XR). Il a les mêmes primaires RVB, point blanc et gamma que sRGB.
`COLOR_FLOAT16` Les données de couleur sont stockées sous forme de valeur de 16 bits par canal, comme défini par la norme ieee à virgule flottante 16 bits.

Notes

Les types de données PCOLORDATATYPE et LPCOLORDATATYPE sont définis en tant que pointeurs vers l'énumération **COLORDATATYPE** :

```
typedef COLORDATATYPE *PCOLORDATATYPE, *LPCOLORDATATYPE;
```

Spécifications

Client minimal pris en charge	Windows Vista [applications de bureau uniquement]
Serveur minimal pris en charge	Windows Server 2008 [applications de bureau uniquement]
En-tête	icm.h

Commentaires

Énumération COLORPROFILESUBTYPE (icm.h)

Article26/02/2024

Spécifie le sous-type du profil de couleur.

Syntax

C++

```
typedef enum {
  CPST_PERCEPTUAL,
  CPST_RELATIVE_COLORIMETRIC,
  CPST_SATURATION,
  CPST_ABSOLUTE_COLORIMETRIC,
  CPST_NONE,
  CPST_RGB_WORKING_SPACE,
  CPST_CUSTOM_WORKING_SPACE,
  CPST_STANDARD_DISPLAY_COLOR_MODE,
  CPST_EXTENDED_DISPLAY_COLOR_MODE
} COLORPROFILESUBTYPE;
```

Constantes

Agrandir le tableau

`CPST_PERCEPTUAL` Intention de rendu perceptuel pour les profils de modèle de carte de gamut (GMMPs) définis dans WCS.
`CPST_RELATIVE_COLORIMETRIC` Intention de rendu colorimétrique relative pour les GMMP définis dans WCS.
`CPST_SATURATION` Intention de rendu de saturation pour les GMMP définis dans WCS.
`CPST_ABSOLUTE_COLORIMETRIC` Intention de rendu colorimétrique absolue pour les GMMP définis dans WCS.
`CPST_NONE` Le sous-type de profil de couleur n'est pas applicable au type de profil de couleur sélectionné.

`CPST_RGB_WORKING_SPACE` Espace de travail de couleur RVB pour les profils ICC (International Color Consortium) ou les profils de modèle d'appareil (DPM) définis dans WCS.
`CPST_CUSTOM_WORKING_SPACE` Espace de travail de couleur personnalisé.
`CPST_STANDARD_DISPLAY_COLOR_MODE` TBD
`CPST_EXTENDED_DISPLAY_COLOR_MODE` TBD

Remarques

Pour obtenir une description des intentions de rendu, consultez Intentions de rendu.

Les types de données PCOLORPROFILESUBTYPE et LPCOLORPROFILESUBTYPE sont définis en tant que pointeurs vers l'énumération **COLORPROFILESUBTYPE** :

```
typedef COLORPROFILESUBTYPE *PCOLORPROFILESUBTYPE, *LPCOLORPROFILESUBTYPE;
```

Les combinaisons de type/sous-type de profil valides sont

${ROWSPAN3}$ COLORPROFILETYPE
${REMOVE}$

COLORPROFILESUBTYPE valide

${ROWSPAN3}$ Notes
${REMOVE}$

par défaut pour un appareil

configuration globale par défaut

Utilisation prévue

Utilisation prévue

CPT_ICC

CPST_NONE

Obtenir/définir le profil ICC par défaut associé à un appareil

CPST_RGBWorkingSpace ou CPST_CustomWorkingSpace

Obtenir/définir le profil ICC en tant qu'espace de travail RVB global ou personnalisé

CPT_DMP

CPST_NONE

Obtenir/Définir le profil DMP par défaut associé à un appareil

CPST_RGBWorkingSpace ou CPST_CustomWorkingSpace

Obtenir/définir DMP en tant qu'espace de travail RVB global ou personnalisé

CPT_CAMP

CPST_NONE

Obtenir/définir le profil CAMP par défaut associé à un appareil

CPST_NONE

Obtenir/définir le profil CAMP comme profil d'apparence de couleur globale

CPT_GMMP

CPST_NONE

Obtenir/définir le profil GMMP par défaut associé à un appareil

CPST_Perceptual ou
CPST_Absolute_colorimetric ou
CPST_Relative_colorimetric ou
CPTS_Saturation

Obtenez/Définissez GMMP comme profil de modèle de carte de gamut global pour une intention de rendu spécifique, comme décrit par ce sous-type à utiliser dans l'API CreateMultiProfileTransform lors de la résolution du tableau d'intentions de rendu dans la transformation WCS.

COLORPROFILESUBTYPE La valeur par défaut globale peut être ou ?d avec WCS_DEFAULT définir ce GMMP comme valeur par défaut globale à utiliser dans OpenColorProfile ou WcsOpenColorProfile où GMMP a la valeur **NULL**.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 10 Build 20348
Serveur minimal pris en charge	Windows 10 Build 20348
En-tête	icm.h

Voir aussi

- COLORPROFILETYPE
- WcsSetDefaultColorProfile

Commentaires

Cette page a-t-elle été utile ? Yes No

Indiquer des commentaires sur le produit | Obtenir de l'aide sur Microsoft Q&A

Énumération COLORPROFILETYPE (icm.h)

Article26/02/2024

Spécifie le type de profil de couleur.

Syntax

```
C++
```

```
typedef enum {
  CPT_ICC,
  CPT_DMP,
  CPT_CAMP,
  CPT_GMMP
} COLORPROFILETYPE;
```

Constantes

Agrandir le tableau

`CPT_ICC` Un profil ICC (International Color Consortium). Si vous spécifiez cette valeur, seules les valeurs CPST_RGB_WORKING_SPACE et CPST_CUSTOM_WORKING_SPACE de **COLORPROFILESUBTYPE** sont valides.
`CPT_DMP` Un profil de modèle d'appareil (DMP) défini dans WCS. Si vous spécifiez cette valeur, seules les valeurs CPST_RGB_WORKING_SPACE et CPST_CUSTOM_WORKING_SPACE de **COLORPROFILESUBTYPE** sont valides.
`CPT_CAMP` Un profil de modèle d'apparence de couleur (CAMP) défini dans WCS. Si vous spécifiez cette valeur, seule la valeur CPST_NONE de **COLORPROFILESUBTYPE** est valide.
`CPT_GMMP` Spécifie un profil de modèle de carte de gamut WCS (GMMP). Si cette valeur est spécifiée, seules les valeurs CPST_PERCEPTUAL, CPST_SATURATION, CPST_RELATIVE_COLORIMETRIC et CPST_ABSOLUTE_COLORIMETRIC **de COLORPROFILESUBTYPE** sont valides. L'une de ces valeurs peut éventuellement être combinée (dans une opération **OR** au niveau du bit) avec CPST_DEFAULT.

Remarques

Les types de données PCOLORPROFILETYPE et LPCOLORPROFILETYPE sont définis en tant que pointeurs vers cette énumération :

```
typedef COLORPROFILETYPE *PCOLORPROFILETYPE, *LPCOLORPROFILETYPE;
```

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows Vista [applications de bureau uniquement]
Serveur minimal pris en charge	Windows Server 2008 [applications de bureau uniquement]
En-tête	icm.h

Voir aussi

COLORPROFILESUBTYPE

Commentaires

Énumération COLORTYPE (icm.h)

Article24/08/2023

Les valeurs de l'énumération **COLORTYPE** sont utilisées par plusieurs fonctions WCS. Les variables de type **COLOR** sont définies dans les espaces de couleurs énumérés par l'énumération **COLORTYPE** .

Syntax

C++

```
typedef enum {
  COLOR_GRAY = 1,
  COLOR_RGB,
  COLOR_XYZ,
  COLOR_Yxy,
  COLOR_Lab,
  COLOR_3_CHANNEL,
  COLOR_CMYK,
  COLOR_5_CHANNEL,
  COLOR_6_CHANNEL,
  COLOR_7_CHANNEL,
  COLOR_8_CHANNEL,
  COLOR_NAMED
} COLORTYPE;
```

Constantes

`COLOR_GRAY` Valeur : *1* La **couleur** se trouve dans l'espace de couleur GRAYCOLOR.
`COLOR_RGB` La **couleur** se trouve dans l'espace de couleur RGBCOLOR.
`COLOR_XYZ` La **couleur** se trouve dans l'espace de couleur XYZCOLOR.
`COLOR_Yxy` La **couleur** se trouve dans l'espace de couleur YxyCOLOR.
`COLOR_Lab` La **couleur** se trouve dans l'espace de couleur LabCOLOR.

`COLOR_3_CHANNEL` La **couleur** se trouve dans l'espace de couleur GENERIC3CHANNEL.
`COLOR_CMYK` La **couleur** se trouve dans l'espace de couleur CMYKCOLOR.
`COLOR_5_CHANNEL` La **couleur** se trouve dans un espace de couleurs à cinq canaux.
`COLOR_6_CHANNEL` La **couleur** se trouve dans un espace de couleurs à six canaux.
`COLOR_7_CHANNEL` La **couleur** se trouve dans un espace de couleurs à sept canaux.
`COLOR_8_CHANNEL` La **couleur** se trouve dans un espace de couleur à huit canaux.
`COLOR_NAMED` La **couleur** se trouve dans un espace de couleurs nommé.

Notes

En plus de gérer les espaces de couleurs communs à deux, trois et quatre canaux, WCS 1.0 est en mesure d'effectuer la gestion des couleurs avec des profils d'appareil qui contiennent cinq à huit canaux de couleur. Il peut également utiliser des espaces de couleurs nommés. Lorsque cinq, six, sept ou huit canaux de couleur sont utilisés, le fournisseur du profil d'appareil est libre de déterminer ce que représentent les canaux de couleur. Il en va de même pour les espaces de couleurs nommés. WCS 1.0 est en mesure de gérer ces espaces de couleurs tant qu'il existe un mappage dans le profil d'appareil qui mappe les canaux ou l'espace de nom dans le PCS. Le profil d'appareil doit également contenir un mappage du PCS vers l'espace de cinq canaux, la taille, sept ou huit, ou dans l'espace de couleurs nommé.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]

En-tête	icm.h

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l’aide sur Microsoft Q&A

énumération WCS_PROFILE_MANAGEMENT_SCOPE (icm.h)

Article24/08/2023

Spécifie l'étendue d'une opération de gestion de profil, telle que l'association d'un profil à un appareil.

Syntax

```
C++

typedef enum {
  WCS_PROFILE_MANAGEMENT_SCOPE_SYSTEM_WIDE,
  WCS_PROFILE_MANAGEMENT_SCOPE_CURRENT_USER
} WCS_PROFILE_MANAGEMENT_SCOPE;
```

Constantes

`WCS_PROFILE_MANAGEMENT_SCOPE_SYSTEM_WIDE` Indique que l'opération de gestion des profils affecte tous les utilisateurs.
`WCS_PROFILE_MANAGEMENT_SCOPE_CURRENT_USER` Indique que l'opération de gestion de profil affecte uniquement l'utilisateur actuel.

Spécifications

Client minimal pris en charge	Windows Vista [applications de bureau uniquement]
Serveur minimal pris en charge	Windows Server 2008 [applications de bureau uniquement]
En-tête	icm.h

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l’aide sur Microsoft Q&A

Interfaces pour le système de couleurs Windows

Article • 13/06/2023

Répertorie les interfaces les plus importantes utilisées par le système de couleurs Windows :

- IDeviceModelPlugIn
- IGamutMapModelPlugIn

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

Interface IDeviceModelPlugIn (wcsplugin.h)

Article03/03/2024

Décrit les méthodes définies pour l'interface COM (Component Object Model) **IDeviceModelPlugIn** .

- ColorimetricToDeviceColors
- DeviceToColorimetricColors
- DeviceToColorimetricColorsWithBlack
- GetGamutBoundaryMesh
- GetGamutBoundaryMeshSize
- GetNeutralAxis
- GetNeutralAxisSize
- GetNumChannels
- GetPrimarySamples
- Initialiser
- SetTransformDeviceModelInfo

Héritage

L'interface IDeviceModelPlugIn hérite de l'interface IUnknown.

Méthodes

L'interface **IDeviceModelPlugIn** utilise ces méthodes.

Agrandir le tableau

IDeviceModelPlugIn ::ColorimetricToDeviceColors Retourne les couleurs XYZ appropriées en réponse au nombre spécifié de couleurs, de canaux, de couleurs d'appareil et d'algorithmes de plug-in propriétaires. (IDeviceModelPlugIn.ColorimetricToDeviceColors)
IDeviceModelPlugIn ::ColorimetricToDeviceColorsWithBlack Retourne les couleurs d'appareil appropriées en réponse au nombre entrant de couleurs, de

canaux, d'informations noires, de couleurs de la Commission Internationale l'Eclairge XYZ (CIEXYZ) et aux algorithmes de plug-in propriétaires.
IDeviceModelPlugIn ::D eviceToColorimetricColors Retourne les couleurs XYZ appropriées en réponse au nombre spécifié de couleurs, de canaux, de couleurs d'appareil et d'algorithmes de plug-in propriétaires. (IDeviceModelPlugIn.DeviceToColorimetricColors)
IDeviceModelPlugIn ::GetGamutBoundaryMesh Retourne le maillage triangulaire du plug-in. Cette fonction est utilisée pour calculer gamutBoundaryDescription.
IDeviceModelPlugIn ::GetGamutBoundaryMeshSize Retourne les tailles de structure de données requises pour la fonction GetGamutBoundaryMesh.
IDeviceModelPlugIn ::GetNeutralAxis IDeviceModelPlugIn ::GetNeutralAxis retourne la colorimétrie XYZ des exemples de points le long de l'axe neutre de l'appareil.
IDeviceModelPlugIn ::GetNeutralAxisSize La fonction IDeviceModelPlugIn ::GetNeutralAxisSize retourne le nombre de points de données le long de l'axe neutre retournés par la fonction GetNeutralAxis.
IDeviceModelPlugIn ::GetNumChannels Retourne le nombre de canaux d'appareil dans le paramètre pNumChannels.
IDeviceModelPlugIn ::GetPrimarySamples Retourne la couleur de mesure de l'échantillon principal.
IDeviceModelPlugIn ::Initialize Prend un pointeur vers un Stream qui contient l'ensemble du plug-in de modèle d'appareil en tant qu'entrée et initialise tous les paramètres internes requis par le plug-in.
IDeviceModelPlugIn ::SetTransformDeviceModelInfo Fournit au plug-in des paramètres pour déterminer où se produit le plug-in spécifique dans la séquence de transformation.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Plateforme cible	Windows
En-tête	wcsplugin.h

Commentaires

Cette page a-t-elle été utile ? Yes No

Indiquer des commentaires sur le produit | Obtenir de l'aide sur Microsoft Q&A

Méthode IDeviceModelPlugIn::ColorimetricToDeviceColors (wcsplugin.h)

Article27/08/2023

Retourne les couleurs XYZ appropriées en réponse au nombre spécifié de couleurs, de canaux, de couleurs d'appareil et d'algorithmes de plug-in propriétaires.

Syntaxe

```
C++

HRESULT ColorimetricToDeviceColors(
  [in]  UINT            cColors,
  [in]  UINT            cChannels,
  [in]  const XYZColorF *pXYZColors,
  [out] FLOAT           *pDeviceValues
);
```

Paramètres

`[in] cColors`

Nombre de couleurs dans les tableaux *pXYZColors* et *pDeviceValues* .

`[in] cChannels`

Nombre de canaux de couleur dans les tableaux *pDeviceValues* .

`[in] pXYZColors`

Pointeur vers le tableau de xyzColors entrants à convertir en couleurs d'appareil.

`[out] pDeviceValues`

Pointeur vers un tableau qui contient les valeurs de couleur d'appareil sortant résultantes.

Valeur retournée

Si cette fonction réussit, la valeur de retour est S_OK.

Si cette fonction échoue, la valeur de retour est E_FAIL.

Remarques

Si *cColors* ou *cChannels* est égal à zéro, la valeur de retour est E_FAIL. S'il existe des couleurs non valides ou hors gamme (ce qui peut se produire s'il y a eu un traitement antérieur du modèle de mappage de gamut), la valeur de retour est E_FAIL.

Configuration requise

Client minimal pris en charge	Windows Vista [applications de bureau uniquement]
Serveur minimal pris en charge	Windows Server 2008 [applications de bureau uniquement]
Plateforme cible	Windows
En-tête	wcsplugin.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- IDeviceModelPlugIn

Commentaires

IDeviceModelPlugIn::ColorimetricToDeviceColorsWithBlack, méthode (wcsplugin.h)

Article27/08/2023

Retourne les couleurs d'appareil appropriées en réponse au nombre entrant de couleurs, de canaux, d'informations noires, aux couleurs de la Commission Internationale l'Eclairge XYZ (CIEXYZ) et aux algorithmes de plug-in propriétaires.

Syntaxe

C++

```
HRESULT ColorimetricToDeviceColorsWithBlack(
  [in]  UINT                  cColors,
  [in]  UINT                  cChannels,
  [out] const XYZColorF       *pXYZColors,
  [in]  const BlackInformation *pBlackInformation,
  [in]  FLOAT                 *pDeviceValues
);
```

Paramètres

`[in] cColors`

Nombre de couleurs dans les tableaux *pXYZColors* et *pDeviceValues* .

`[in] cChannels`

Nombre de canaux de couleur dans les tableaux *pDeviceValues* .

`[out] pXYZColors`

Pointeur vers le tableau de structures XYZColorF sortantes.

`[in] pBlackInformation`

Pointeur vers blackInformation.

`[in] pDeviceValues`

Pointeur vers le tableau des couleurs d'appareil entrantes qui doivent être converties en structures XYZColorF .

Valeur retournée

Si cette fonction réussit, la valeur de retour est S_OK.

Si cette fonction échoue, la valeur de retour est E_FAIL. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Remarques

Si *cColors* ou *cChannels* est égal à zéro, la valeur de retour est E_FAIL.

Configuration requise

Client minimal pris en charge	Windows Vista [applications de bureau uniquement]
Serveur minimal pris en charge	Windows Server 2008 [applications de bureau uniquement]
Plateforme cible	Windows
En-tête	wcsplugin.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- IDeviceModelPlugIn

Commentaires

Méthode IDeviceModelPlugIn ::D eviceToColorimetricColors (wcsplugin.h)

Article03/03/2024

Retourne les couleurs XYZ appropriées en réponse au nombre spécifié de couleurs, de canaux, de couleurs d'appareil et d'algorithmes de plug-in propriétaires.

Syntaxe

```
C++

HRESULT DeviceToColorimetricColors(
  [in]  UINT        cColors,
  [in]  UINT        cChannels,
  [in]  const FLOAT *pDeviceValues,
  [out] XYZColorF   *pXYZColors
);
```

Paramètres

`[in] cColors`

Nombre de couleurs dans les tableaux *pXYZColors* et *pDeviceValues* .

`[in] cChannels`

Nombre de canaux de couleur dans les tableaux *pDeviceValues* .

`[in] pDeviceValues`

Pointeur vers le tableau de xyzColors sortants.

`[out] pXYZColors`

Pointeur vers le tableau des couleurs d'appareil entrantes à convertir en XYZColors.

Valeur retournée

Si cette fonction réussit, la valeur de retour est S_OK.

Si cette fonction échoue, la valeur de retour est E_FAIL. Pour obtenir des informations d'erreur étendues, **appelez GetLastError**.

Remarques

Si *cColors* ou *cChannels* est égal à zéro, la valeur de retour est E_FAIL.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows Vista [applications de bureau uniquement]
Serveur minimal pris en charge	Windows Server 2008 [applications de bureau uniquement]
Plateforme cible	Windows
En-tête	wcsplugin.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- IDeviceModelPlugIn

Commentaires

Méthode IDeviceModelPlugIn::GetGamutBoundaryMesh (wcsplugin.h)

Article27/08/2023

Retourne le maillage triangulaire du plug-in. Cette fonction est utilisée pour calculer gamutBoundaryDescription.

Syntaxe

```
C++

HRESULT GetGamutBoundaryMesh(
  [in]  UINT               cChannels,
  [in]  UINT               cVertices,
  [in]  UINT               cTriangles,
  [out] FLOAT              *pVertices,
  [out] GamutShellTriangle *pTriangles
);
```

Paramètres

`[in] cChannels`

Le nombre de canaux.

`[in] cVertices`

Nombre de sommets.

`[in] cTriangles`

Nombre de triangles.

`[out] pVertices`

Pointeur vers le tableau de sommets dans l'interpréteur de commandes gamut du modèle de plug-in.

`[out] pTriangles`

Pointeur vers le tableau de triangles dans l'interpréteur de commandes gamut du modèle de plug-in.

Valeur retournée

Si cette fonction réussit, la valeur de retour est S_OK.

Si cette fonction échoue, la valeur de retour est E_FAIL.

Remarques

Cette fonction est appelée par la fonction CreateMultiProfileTransform .

Configuration requise

Client minimal pris en charge	Windows Vista [applications de bureau uniquement]
Serveur minimal pris en charge	Windows Server 2008 [applications de bureau uniquement]
Plateforme cible	Windows
En-tête	wcsplugin.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- IDeviceModelPlugIn

Commentaires

Méthode IDeviceModelPlugIn::GetGamutBoundaryMeshSize (wcsplugin.h)

Article27/08/2023

Retourne les tailles de structure de données requises pour la fonction GetGamutBoundaryMesh .

Syntaxe

```
C++

HRESULT GetGamutBoundaryMeshSize(
  [out] UINT *pNumVertices,
  [out] UINT *pNumTriangles
);
```

Paramètres

`[out] pNumVertices`

Nombre requis de sommets.

`[out] pNumTriangles`

Nombre requis de triangles.

Valeur retournée

Si cette fonction réussit, la valeur de retour est S_OK.

Si le modèle d'appareil de plug-in souhaite que WCS calcule son maillage de limite de gamut, la valeur de retour est S_FALSE.

Si cette fonction échoue, la valeur de retour est E_FAIL.

Remarques

Cette fonction est appelée par la fonction CreateMultiProfileTransform .

Configuration requise

Client minimal pris en charge	Windows Vista [applications de bureau uniquement]
Serveur minimal pris en charge	Windows Server 2008 [applications de bureau uniquement]
Plateforme cible	Windows
En-tête	wcsplugin.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- IDeviceModelPlugIn

Commentaires

Méthode IDeviceModelPlugIn::GetNeutralAxis (wcsplugin.h)

Article27/08/2023

IDeviceModelPlugIn::GetNeutralAxis retourne la colorimétrie XYZ des exemples de points le long de l'axe neutre de l'appareil.

Syntaxe

```
C++

HRESULT GetNeutralAxis(
  [in]  UINT      cColors,
  [out] XYZColorF *pXYZColors
);
```

Paramètres

`[in] cColors`

Nombre de points retournés.

`[out] pXYZColors`

Pointeur vers un tableau de structures XYZColorF .

Valeur retournée

Si cette fonction réussit, la valeur de retour est S_OK.

Si cette fonction échoue, la valeur de retour est E_FAIL.

Remarques

Vous devez définir l'« axe neutre » d'une manière appropriée pour votre appareil. En règle générale, il s'agit de points créés par les valeurs grises de l'appareil. Il peut s'agir de R=G=B ou C=M=Y=0 et de n'importe quelle valeur de K. Pour certains appareils, le

gris le plus agréable peut être celui qui utilise une combinaison différente de colorants, comme M=Y=0 et C=K. Le plug-in est chargé de déterminer la colorimétrie d'un échantillonnage des valeurs d'axe neutre et de les retourner. L'échantillonnage peut être aussi épars que deux points (blanc et noir) ou aussi dense que vous le souhaitez.

Il n'est pas nécessaire que les échantillons soient uniformément espadés dans un espace de couleur.

Configuration requise

Client minimal pris en charge	Windows Vista [applications de bureau uniquement]
Serveur minimal pris en charge	Windows Server 2008 [applications de bureau uniquement]
Plateforme cible	Windows
En-tête	wcsplugin.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- IDeviceModelPlugIn

Commentaires

Méthode IDeviceModelPlugIn ::GetNeutralAxisSize (wcsplugin.h)

Article03/03/2024

La fonction IDeviceModelPlugIn ::GetNeutralAxisSize retourne le nombre de points de données le long de l'axe neutre retournés par la fonction GetNeutralAxis . Il est fourni afin qu'un module de gestion des couleurs (CMM) puisse allouer une mémoire tampon de taille appropriée.

Syntaxe

```
C++

HRESULT GetNeutralAxisSize(
  [out] UINT *pcColors
);
```

Paramètres

`[out] pcColors`

Nombre de points qui seront retournés par un appel à GetNeutralAxis. La valeur minimale est 2 (noir et blanc).

Valeur retournée

Si cette fonction réussit, la valeur de retour est S_OK.

Si cette fonction échoue, la valeur de retour est E_FAIL.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows Vista [applications de bureau uniquement]

Condition requise	Valeur
Serveur minimal pris en charge	Windows Server 2008 [applications de bureau uniquement]
Plateforme cible	Windows
En-tête	wcsplugin.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- IDeviceModelPlugIn

Commentaires

Cette page a-t-elle été utile ? Yes No

Indiquer des commentaires sur le produit | Obtenir de l'aide sur Microsoft Q&A

Méthode IDeviceModelPlugIn ::GetNumChannels (wcsplugin.h)

Article03/03/2024

Retourne le nombre de canaux d'appareil dans le paramètre *pNumChannels*.

Syntaxe

```
C++

HRESULT GetNumChannels(
  [out] UINT *pNumChannels
);
```

Paramètres

`[out] pNumChannels`

Pointeur vers un entier non signé représentant le nombre de canaux de couleur pour votre appareil.

Valeur retournée

Si cette fonction réussit, la valeur de retour est S_OK.

Si cette fonction échoue, la valeur de retour est E_FAIL.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows Vista [applications de bureau uniquement]
Serveur minimal pris en charge	Windows Server 2008 [applications de bureau uniquement]
Plateforme cible	Windows

Condition requise	Valeur
En-tête	wcsplugin.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- IDeviceModelPlugIn

Commentaires

Cette page a-t-elle été utile ? Yes No

Indiquer des commentaires sur le produit | Obtenir de l'aide sur Microsoft Q&A

Méthode IDeviceModelPlugIn::GetPrimarySamples (wcsplugin.h)

Article27/08/2023

Retourne la couleur de mesure de l'exemple principal.

Syntaxe

```
C++

HRESULT GetPrimarySamples(
  [out] PrimaryXYZColors *pPrimaryColor
);
```

Paramètres

`[out] pPrimaryColor`

Type de couleur primaire, qui est déterminé à l'aide de l'ordre de cercle de teinte. Si le modèle d'appareil de plug-in ne prend pas en charge nativement les primaires pour le rouge, le jaune, le vert, le cyan, le bleu, le magenta, le noir et le blanc, il doit toujours retourner des données primaires virtuelles.

Valeur retournée

Si cette fonction réussit, la valeur de retour est S_OK.

Si cette fonction échoue, la valeur de retour est E_FAIL.

Configuration requise

Client minimal pris en charge	Windows Vista [applications de bureau uniquement]
Serveur minimal pris en charge	Windows Server 2008 [applications de bureau uniquement]
Plateforme cible	Windows

En-tête	wcsplugin.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- IDeviceModelPlugIn

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

IDeviceModelPlugIn::Initialize, méthode (wcsplugin.h)

Article27/08/2023

Prend un pointeur vers un stream qui contient l'ensemble du plug-in de modèle d'appareil en tant qu'entrée et initialise tous les paramètres internes requis par le plug-in.

Syntaxe

```
C++

HRESULT Initialize(
  [in] BSTR bstrXml,
  [in] UINT cNumModels,
  [in] UINT iModelPosition
);
```

Paramètres

`[in] bstrXml`

Chaîne qui contient le profil de plug-in de modèle d'appareil XML BSTR. Ce paramètre stocke les données sous forme de XML Unicode peu endian ; toutefois, il peut ne pas avoir d'octets de début pour l'étiqueter en tant que tel. En outre, l'encodage mot clé dans le code XML peut ne pas refléter le fait qu'il est mis en forme en unicode à faible endian. En outre, en raison de l'action du moteur MSXML, le fichier XML BSTR est traité et peut ne pas avoir exactement le même contenu que le fichier XML d'origine.

`[in] cNumModels`

Nombre total de modèles dans la séquence de transformation.

`[in] iModelPosition`

Position du modèle mono-base de l'autre modèle d'appareil dans le flux de travail de *uiNumModels* , comme indiqué dans la fonction **Initialize** .

Valeur retournée

Si cette fonction réussit, la valeur de retour est S_OK.

Si cette fonction échoue, la valeur de retour est E_FAIL.

Remarques

Si cette fonction est appelée plusieurs fois, les appels suivants libèrent toute la mémoire allouée et réinitialisent en fonction du nouveau paramètre *bstrXml* .

Configuration requise

Client minimal pris en charge	Windows Vista [applications de bureau uniquement]
Serveur minimal pris en charge	Windows Server 2008 [applications de bureau uniquement]
Plateforme cible	Windows
En-tête	wcsplugin.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- IDeviceModelPlugIn

Commentaires

Méthode IDeviceModelPlugIn::SetTransformDeviceModelInfo (wcsplugin.h)

Article27/08/2023

Fournit au plug-in des paramètres pour déterminer où se produit le plug-in spécifique dans la séquence de transformation.

Syntaxe

```
C++

HRESULT SetTransformDeviceModelInfo(
  [in] UINT               iModelPosition,
  [in] IDeviceModelPlugIn *pIDeviceModelOther
);
```

Paramètres

`[in] iModelPosition`

Position du modèle de base un de l'autre modèle d'appareil dans le flux de travail de *uiNumModels*, comme indiqué dans la fonction Initialize .

`[in] pIDeviceModelOther`

Pointeur vers une interface **IDeviceModelPlugIn** qui contient l'autre modèle d'appareil dans la séquence de transformation.

Valeur retournée

Si cette fonction réussit, la valeur de retour est S_OK.

Si cette fonction échoue, la valeur de retour est E_FAIL.

Remarques

Cette fonction est appelée par la fonction CreateMultiProfileTransform , qui est chargée d'appeler **AddRef** et **Release** le cas échéant. La fonction permet aux modèles d'appareils plug-ins d'échanger des informations de manière propriétaire en accédant aux fonctions d'interface de plug-in propriétaires.

Cette fonction échoue si l'autre modèle d'appareil est un modèle d'appareil de base, car ces modèles ne sont pas des plug-ins et, par conséquent, la communication entre plug-ins n'est pas prise en charge.

Configuration requise

Client minimal pris en charge	Windows Vista [applications de bureau uniquement]
Serveur minimal pris en charge	Windows Server 2008 [applications de bureau uniquement]
Plateforme cible	Windows
En-tête	wcsplugin.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- IDeviceModelPlugIn

Commentaires

Interface IGamutMapModelPlugIn (wcsplugin.h)

Article03/03/2024

Décrit les méthodes définies pour l'interface COM (Component Object Model) **IGamutMapModelPlugIn** .

- IGamutMapModelPlugIn ::Initialize
- IGamutMapModelPlugIn ::SourceToDestinationAppearanceColors

Héritage

L'interface IGamutMapModelPlugIn hérite de l'interface IUnknown.

Méthodes

L'interface IGamutMapModelPlugIn possède ces méthodes.

Agrandir le tableau

IGamutMapModelPlugIn ::Initialize Initialise un profil GMMP (Gamut Map Model Profile) à l'aide des descriptions de limites de gamut source et de destination spécifiées et des plug-ins facultatifs de modèle source et de modèle d'appareil de destination.
IGamutMapModelPlugIn ::SourceToDestinationAppearanceColors Retourne les couleurs d'apparence mappées de gamut appropriées en réponse au nombre spécifié de couleurs et aux couleurs CIEJCh.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Plateforme cible	Windows

Condition requise	Valeur
En-tête	wcsplugin.h

Commentaires

Cette page a-t-elle été utile ? Yes No

Indiquer des commentaires sur le produit | Obtenir de l'aide sur Microsoft Q&A

IGamutMapModelPlugIn ::Initialize, méthode (wcsplugin.h)

Article03/03/2024

Initialise un profil GMMP (Gamut Map Model Profile) à l'aide des descriptions de limites de gamut source et de destination spécifiées et des plug-ins facultatifs de modèle source et de modèle d'appareil de destination.

Syntaxe

```
C++

HRESULT Initialize(
  [in] BSTR                     bstrXml,
  [in] IDeviceModelPlugIn       *pSrcPlugIn,
  [in] IDeviceModelPlugIn       *pDestPlugIn,
  [in] GamutBoundaryDescription *pSrcGBD,
  [in] GamutBoundaryDescription *pDestGBD
);
```

Paramètres

`[in] bstrXml`

Chaîne qui contient le profil GMMP XML BSTR. Il s'agit d'un XML Unicode peu endian, mais sans les octets de début pour l'étiqueter en tant que tel. En outre, l'encodage mot clé dans le xml peut ne pas refléter ce format.

`[in] pSrcPlugIn`

Pointeur vers un IDeviceModelPlugIn source. Si **la valeur est NULL**, cela indique que le profil de modèle d'appareil source n'est pas un profil de plug-in.

`[in] pDestPlugIn`

Pointeur vers un IDeviceModelPlugIn de destination. Si la valeur **est NULL**, cela indique que le profil de modèle d'appareil de destination n'est pas un profil de plug-in.

`[in] pSrcGBD`

Pointeur vers une source GamutBoundaryDescription.

`[in] pDestGBD`

Pointeur vers un GamutBoundaryDescription de destination.

Valeur retournée

Si cette fonction réussit, la valeur de retour est S_OK.

Si cette fonction échoue, la valeur de retour est E_FAIL.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows Vista [applications de bureau uniquement]
Serveur minimal pris en charge	Windows Server 2008 [applications de bureau uniquement]
Plateforme cible	Windows
En-tête	wcsplugin.h

Voir aussi

- Concepts de base de la gestion des couleurs
- IGamutMapModelPlugIn

Commentaires

Méthode IGamutMapModelPlugIn ::SourceToDestinationAppearanceColors (wcsplugin.h)

Article03/03/2024

Retourne les couleurs d'apparence mappées de gamut appropriées en réponse au nombre spécifié de couleurs et aux couleurs CIEJCh .

Syntaxe

```
C++

HRESULT SourceToDestinationAppearanceColors(
  [in]  UINT            cColors,
  [in]  const JChColorF *pInputColors,
  [out] JChColorF       *pOutputColors
);
```

Paramètres

`[in] cColors`

Nombre de couleurs dans les tableaux *pXYZColors* et *pDeviceValues* .

`[in] pInputColors`

Pointeur vers le tableau des couleurs entrantes à mapper.

`[out] pOutputColors`

Pointeur vers le tableau de couleurs sortantes.

Valeur retournée

Si cette fonction réussit, la valeur de retour est S_OK.

Si cette fonction échoue, la valeur de retour est E_FAIL.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows Vista [applications de bureau uniquement]
Serveur minimal pris en charge	Windows Server 2008 [applications de bureau uniquement]
Plateforme cible	Windows
En-tête	wcsplugin.h

Voir aussi

- Concepts de base de la gestion des couleurs
- IGamutMapModelPlugIn

Commentaires

Structures WCS 1.0

Article • 13/06/2023

Cette section de la référence contient une liste des structures les plus importantes utilisées par WCS 1.0. Celles-ci sont répertoriées comme suit :

- BlackInformation
- CIEXYZ
- CIEXYZTRIPLE
- COLORMATCHSETUPW
- ENUMTYPEW
- GamutBoundaryDescription
- Structures de couleurs du modèle de carte de gamut
- GamutShell
- GamutShellTriangle
- Structures de couleurs ICM
- LOGCOLORSPACE
- NAMED_PROFILE_INFO
- PrimaryJabColors
- PrimaryXYZColors
- PROFIL
- PROFILEHEADER

Commentaires

Structure BlackInformation (wcsplugin.h)

Article27/08/2023

Contient des informations pour les modèles d'appareil qui ont un canal de couleur noire.

Syntaxe

```
C++
```

```
typedef struct _BlackInformation {
  BOOL  fBlackOnly;
  FLOAT blackWeight;
} BlackInformation;
```

Membres

`fBlackOnly`

`blackWeight`

Valeur comprise entre 0,0 et 1,0 qui indique la quantité relative de noir à utiliser dans la sortie. La valeur 0,0 signifie qu'aucun noir n'est utilisé ; la valeur 1.0 signifie que la quantité maximale de noir est utilisée.

Remarques

Si l'appareil source ne prend pas en charge un canal noir, WCS définit **bBlackOnly** sur **FALSE**.

Si **bBlackOnly** a la valeur **TRUE**, WCS génère une valeur de contrôle d'appareil de sortie où, au maximum, le canal noir est différent de zéro. Cela se produit uniquement si l'indicateur **BlackPreservation** a été défini dans WCS. Notez que dans ce cas, le modèle d'appareil peut ne pas fournir la correspondance colorimétrique la plus proche de la valeur fournie.

La conservation des noirs n'est effectuée que lorsque les appareils source et de destination prennent en charge un canal noir. Si le noir est conservé avec ces appareils, alors pour chaque valeur de contrôle d'appareil source, où tous les canaux autres que le

canal noir sont zéro, l'indicateur **bBlackOnly** est **TRUE**. Notez que cela signifie qu'une valeur où tous les canaux sont zéro définit également **bBlackOnly** sur **TRUE**.

blackWeight nous fournit des informations sur les valeurs de contrôle d'appareil utilisées dans l'appareil source.

- Pour les appareils sources avec un canal noir, **blackWeight** est défini sur la valeur noire.
- Pour les appareils sources sans canal noir, le poids noir est calculé à l'aide d'une combinaison de *pureté des couleurs* et *de légèreté relative*. *La pureté des* couleurs est définie comme (maxColorant - minColorant)/maxColorant

  La légèreté relative est définie comme (légèreté de la couleur dans l'espace d'apparence - luminosité minimale de l'appareil de destination) / (luminosité maximale de l'appareil de destination - luminosité minimale de l'appareil de destination)

  Pour les appareils RVB, blackWeight = (1 - colorPurity) * (1 - relativeLightness)

  Pour les appareils CMJN, blackWeight = colorPurity * (1 - relativeLightness)

  WCS est responsable de l'initialisation de la structure **BlackInformation** .

Si **bBlackOnly** a la valeur **FALSE**, les modèles d'appareil de base pour les appareils avec un canal noir utilisent **blackWeight** pour guider la création d'une valeur de pixel de sortie colorimétriquement appropriée. Pour les appareils CMJN, **blackWeight** fournit l'estimation initiale par WCS d'une valeur K et recherche les valeurs C, M et Y qui mèneront à la colorimétrie correcte. S'il ne trouve pas de correspondance, il ajuste la valeur K et effectue une nouvelle recherche.

Vous pouvez définir des plug-ins pour prendre en charge ou ignorer **blackInformation**.

Configuration requise

Client minimal pris en charge	Windows Vista [applications de bureau uniquement]
Serveur minimal pris en charge	Windows Server 2008 [applications de bureau uniquement]
En-tête	wcsplugin.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Structures

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

Structure CIEXYZ (wingdi.h)

Article04/03/2024

La structure **CIEXYZ** contient les coordonnées *x*, *y* et *z* d'une couleur spécifique dans un espace de couleurs spécifié.

Syntaxe

```
C++

typedef struct tagCIEXYZ {
  FXPT2DOT30 ciexyzX;
  FXPT2DOT30 ciexyzY;
  FXPT2DOT30 ciexyzZ;
} CIEXYZ;
```

Membres

`ciexyzX`

Coordonnée x dans le point de correction (2,30).

`ciexyzY`

Coordonnée y au point de correction (2,30).

`ciexyzZ`

Coordonnée z au point de correction (2,30).

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]

Condition requise	Valeur
En-tête	wingdi.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Structures

Commentaires

Cette page a-t-elle été utile ? Yes No

Indiquer des commentaires sur le produit | Obtenir de l'aide sur Microsoft Q&A

CIEXYZTRIPLE structure (wingdi.h)

Article08/03/2023

La structure **CIEXYZTRIPLE** contient les coordonnées *x*, *y* et *z* des trois couleurs qui correspondent aux points de terminaison rouge, vert et bleu pour un espace de couleur logique spécifié.

Syntaxe

```
C++
```

```
typedef struct tagICEXYZTRIPLE {
  CIEXYZ ciexyzRed;
  CIEXYZ ciexyzGreen;
  CIEXYZ ciexyzBlue;
} CIEXYZTRIPLE;
```

Membres

`ciexyzRed`

Coordonnées xyz du point de terminaison rouge.

`ciexyzGreen`

Coordonnées xyz du point de terminaison vert.

`ciexyzBlue`

Coordonnées xyz du point de terminaison bleu.

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	wingdi.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Structures

Structure CMYKCOLOR (icm.h)

Article24/08/2023

Description de la structure CMYKCOLOR.

Syntaxe

```
C++
```

```
struct CMYKCOLOR {
  WORD cyan;
  WORD magenta;
  WORD yellow;
  WORD black;
};
```

Membres

`cyan`

TBD

`magenta`

TBD

`yellow`

TBD

`black`

TBD

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]

En-tête	icm.h

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l’aide sur Microsoft Q&A

COLOR union (icm.h)

Article26/02/2024

Description de l'union COLOR.

Syntaxe

```
C++
```

```
typedef union tagCOLOR {
  struct GRAYCOLOR       gray;
  struct RGBCOLOR        rgb;
  struct CMYKCOLOR       cmyk;
  struct XYZCOLOR        XYZ;
  struct YxyCOLOR        Yxy;
  struct LabCOLOR        Lab;
  struct GENERIC3CHANNEL gen3ch;
  struct NAMEDCOLOR      named;
  struct HiFiCOLOR       hifi;
  struct {
    DWORD reserved1;
    VOID  *reserved2;
  };
} COLOR;
```

Membres

`gray`

TBD

`rgb`

TBD

`cmyk`

TBD

`XYZ`

TBD

`Yxy`

TBD

`Lab`

TBD

`gen3ch`

TBD

`named`

TBD

`hifi`

TBD

`reserved1`

TBD

`reserved2`

TBD

Remarques

Une variable de type COLOR est accessible en tant que l'une des couleurs d'espace de couleurs prises en charge en accédant au membre approprié de l'union. Pour instance, compte tenu de la déclaration de variable suivante :

```
COLOR aColor;
```

Les valeurs rouges, vertes et bleues peuvent être définies de la manière suivante :

```
aColor.rgb.red=100;
```

```
aColor.rgb.green=50;
```

```
aColor.rgb.blue=2;
```

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Commentaires

Cette page a-t-elle été utile ? Yes No

Indiquer des commentaires sur le produit | Obtenir de l'aide sur Microsoft Q&A

STRUCTURE COLORMATCHSETUPW (icm.h)

Article24/08/2023

La structure **COLORMATCHSETUP** contient des informations que la fonction SetupColorMatchingW utilise pour initialiser la boîte de dialogue **ColorManagement** . Une fois la boîte de dialogue fermée par l'utilisateur, **SetupColorMatching retourne des** informations sur la sélection de l'utilisateur dans cette structure.

Syntaxe

C++

```
typedef struct _tagCOLORMATCHSETUPW {
  DWORD          dwSize;
  DWORD          dwVersion;
  DWORD          dwFlags;
  HWND           hwndOwner;
  PCWSTR         pSourceName;
  PCWSTR         pDisplayName;
  PCWSTR         pPrinterName;
  DWORD          dwRenderIntent;
  DWORD          dwProofingIntent;
  PWSTR          pMonitorProfile;
  DWORD          ccMonitorProfile;
  PWSTR          pPrinterProfile;
  DWORD          ccPrinterProfile;
  PWSTR          pTargetProfile;
  DWORD          ccTargetProfile;
  DLGPROC        lpfnHook;
  LPARAM         lParam;
  PCMSCALLBACKW lpfnApplyCallback;
  LPARAM         lParamApplyCallback;
} COLORMATCHSETUPW, *PCOLORMATCHSETUPW, *LPCOLORMATCHSETUPW;
```

Membres

`dwSize`

Taille de la structure. Doit être défini sur **sizeof** (**COLORMATCHSETUP**).

`dwVersion`

Version de la structure **COLORMATCHSETUP** . Cette valeur doit être définie sur COLOR_MATCH_VERSION.

`dwFlags`

Ensemble d'indicateurs de bits utilisés pour initialiser la boîte de dialogue. Si la valeur est 0 lors de l'entrée, tous les contrôles supposent leur état par défaut.

Lorsque la boîte de dialogue retourne, ces indicateurs sont définis pour indiquer l'entrée de l'utilisateur.

Ce membre peut être défini à l'aide d'une combinaison des indicateurs suivants.

Indicateur	**Signification**
CMS_DISABLEICM	S'il est défini lors de l'entrée, cet indicateur indique que la zone de case activée « Activer la gestion des couleurs » est désactivée, désactivant tous les autres contrôles. Si la valeur est définie à la sortie, cela signifie que l'utilisateur ne souhaite pas que la gestion des couleurs soit effectuée.
CMS_ENABLEPROOFING	S'il est défini lors de l'entrée, cet indicateur indique que les contrôles de vérification doivent être activés et que la case Vérification case activée est cochée. Si la valeur est définie à la sortie, cela signifie que l'utilisateur souhaite effectuer la gestion des couleurs pour un appareil cible différent de l'imprimante sélectionnée.
CMS_SETRENDERINTENT	S'il est défini sur l'entrée, cet indicateur indique que le membre **dwRenderIntent** contient la valeur à utiliser pour initialiser le contrôle Intention de rendu. Sinon, la valeur par défaut du contrôle est Rendu d'image. Cet indicateur est défini lors de la sortie si WCS est activé.
CMS_SETPROOFINTENT	Ignoré, sauf si CMS_ENABLEPROOFING est également défini. S'il est défini lors de l'entrée et que CMS_ENABLEPROOFING est également défini, cet indicateur indique que le membre **dwProofingIntent** doit être utilisé pour initialiser le contrôle Intention de rendu cible. Sinon, la valeur par défaut du contrôle est Rendu d'image. Cet indicateur est défini à la sortie si la vérification linguistique est activée.
CMS_SETMONITORPROFILE	S'il est défini lors de l'entrée, cet indicateur indique que le profil de gestion des couleurs nommé dans le membre **pMonitorProfile** doit être la sélection initiale dans le contrôle de profil du moniteur. Si le profil spécifié n'est pas associé au moniteur, cet indicateur est ignoré et le profil par défaut du moniteur est utilisé.
CMS_SETPRINTERPROFILE	S'il est défini lors de l'entrée, cet indicateur indique que le profil de gestion des couleurs nommé dans le membre **pPrinterProfile** doit

Indicateur	Signification
	être la sélection initiale dans le contrôle de profil d'imprimante. Si le profil spécifié n'est pas associé à l'imprimante, cet indicateur est ignoré et le profil par défaut de l'imprimante est utilisé.
CMS_SETTARGETPROFILE	S'il est défini lors de l'entrée, cet indicateur indique que le profil de couleur nommé dans le membre **pTargetProfile** doit être la sélection initiale dans le contrôle de profil cible. Si le profil spécifié n'est pas installé, cet indicateur est ignoré et le profil par défaut de l'imprimante est utilisé. Si l'imprimante n'a pas de profil par défaut, le premier profil par ordre alphabétique s'affiche.
CMS_USEHOOK	Cet indicateur spécifie que le membre **lpfnHook** contient l'adresse d'une procédure de hook et que le membre **lParam** contient une valeur à passer à la procédure de hook lorsque le message WM_INITDIALOG est envoyé.
CMS_MONITOROVERFLOW	Cet indicateur est défini à la sortie si la gestion des couleurs doit être activée et si la taille de mémoire tampon indiquée dans **ccMonitorProfile** est insuffisante pour le nom de profil sélectionné. **GetLastError** retourne ERROR_INSUFFICIENT_BUFFER dans ce cas.
CMS_PRINTERROVERFLOW	Cet indicateur est défini à la sortie si la gestion des couleurs doit être activée et si la taille de mémoire tampon indiquée dans **ccPrinterProfile** est insuffisante pour le nom de profil sélectionné. **GetLastError** retourne ERROR_INSUFFICIENT_BUFFER dans ce cas.
CMS_TARGETOVERFLOW	Cet indicateur est défini à la sortie si la vérification linguistique doit être activée et si la taille de mémoire tampon indiquée dans **ccTargetProfile** est insuffisante pour le nom de profil sélectionné. **GetLastError** retourne ERROR_INSUFFICIENT_BUFFER dans ce cas.
CMS_USEAPPLYCALLBACK	Si défini lors de l'entrée, cet indicateur indique que la fonction **SetupColorMatching** doit appeler la fonction **PCMSCALLBACKW**. L'adresse de la fonction de rappel est contenue dans *lpfnApplyCallback*.
CMS_USEDESCRIPTION	S'il est défini lors de l'entrée, cet indicateur indique à la fonction **SetupColorMatching** de récupérer la description de profil contenue dans les étiquettes de description du profil (voir Spécification de format de profil ICC v3.4). Il les insère dans les zones **d'édition Profil du moniteur**, **Profil d'imprimante** et **Profil d'appareil émulé** de la boîte de dialogue Gestion des **couleurs** commune.

`hwndOwner`

Handle de fenêtre pour le propriétaire de la boîte de dialogue, ou **NULL** si la boîte de dialogue n'a pas de propriétaire.

`pSourceName`

Pointeur vers une chaîne spécifiée par l'application qui décrit le profil source de l'élément pour lequel la gestion des couleurs doit être effectuée. Si cette valeur est **NULL**, le contrôle Source d'image affiche le nom du profil de couleur par défaut De Windows.

`pDisplayName`

Pointe vers une chaîne nommant le moniteur à utiliser pour la gestion des couleurs. S'il ne s'agit pas du nom d'un moniteur valide, le premier moniteur énuméré est utilisé.

`pPrinterName`

Pointe vers une chaîne nommant l'imprimante sur laquelle l'image doit être affichée. S'il ne s'agit pas d'un nom d'imprimante valide, l'imprimante par défaut est utilisée et nommée dans la boîte de dialogue.

`dwRenderIntent`

Type de gestion des couleurs souhaité. Les valeurs autorisées sont :

INTENT_PERCEPTUAL

INTENT_SATURATION

INTENT_RELATIVE_COLORIMETRIC

INTENT_ABSOLUTE_COLORIMETRIC

Pour plus d'informations, consultez Intentions de rendu.

`dwProofingIntent`

Type de gestion des couleurs souhaité pour l'image avec preuve. Les valeurs autorisées sont :

INTENT_PERCEPTUAL

INTENT_SATURATION

INTENT_RELATIVE_COLORIMETRIC

INTENT_ABSOLUTE_COLORIMETRIC

Pour plus d'informations, consultez Intentions de rendu.

`pMonitorProfile`

Pointeur vers une mémoire tampon dans laquelle placer le nom du profil d'analyse sélectionné par l'utilisateur. Si l'indicateur CMS_SETMONITORPROFILE est utilisé, cet indicateur peut également être utilisé pour sélectionner un profil autre que le profil par défaut du moniteur lorsque la boîte de dialogue est affichée pour la première fois.

`ccMonitorProfile`

Taille de la mémoire tampon pointée vers le membre **pMonitorProfile** , en caractères. Si la mémoire tampon n'est pas assez grande pour contenir le nom sélectionné, le nom est tronqué à cette taille et ERROR_INSUFFICIENT_BUFFER est retourné. Une mémoire tampon de MAX_PATH taille fonctionne toujours.

`pPrinterProfile`

Pointe vers une mémoire tampon dans laquelle placer le nom du profil d'imprimante sélectionné par l'utilisateur. Si l'indicateur CMS_SETPRINTERPROFILE est utilisé, cet indicateur peut également être utilisé pour sélectionner un profil autre que celui par défaut de l'imprimante lorsque la boîte de dialogue est affichée pour la première fois.

`ccPrinterProfile`

Taille de la mémoire tampon pointée vers le membre **pPrinterProfile** , en caractères. Si la mémoire tampon n'est pas assez grande pour contenir le nom sélectionné, le nom est tronqué à cette taille et ERROR_INSUFFICIENT_BUFFER est retourné. Une mémoire tampon de MAX_PATH taille fonctionne toujours.

`pTargetProfile`

Pointe vers une mémoire tampon dans laquelle placer le nom du profil cible sélectionné par l'utilisateur pour la vérification linguistique. Si l'indicateur CMS_SETTARGETPROFILE est utilisé, cet indicateur peut également être utilisé pour sélectionner un profil autre que l'imprimante par défaut lors de la première affichage de la boîte de dialogue.

`ccTargetProfile`

Taille de la mémoire tampon pointée vers le membre **pTargetProfile** , en caractères. Si la mémoire tampon n'est pas assez grande pour contenir le nom sélectionné, le nom est tronqué à cette taille et ERROR_INSUFFICIENT_BUFFER est retourné. Une mémoire tampon de MAX_PATH taille fonctionne toujours.

`lpfnHook`

Si l'indicateur CMS_USEHOOK est défini, ce membre est l'adresse d'une procédure de boîte de dialogue (voir DialogProc) qui peut filtrer ou gérer les messages pour la boîte de dialogue. La procédure de hook ne reçoit aucun message émis avant

WM_INITDIALOG. Il est appelé sur le message WM_INITDIALOG une fois que la procédure de boîte de dialogue fournie par le système a traité le message. Sur tous les autres messages, la procédure de crochet reçoit le message avant la procédure fournie par le système. Si la procédure de hook retourne **TRUE** à ces messages, la procédure fournie par le système n'est pas appelée.

La procédure de hook peut appeler la fonction EndDialog .

`lParam`

Si l'indicateur CMS_USEHOOK est défini, ce membre est passé à la procédure de hook fournie par l'application en tant que paramètre *lParam* lorsque le message WM_INITDIALOG est traité.

`lpfnApplyCallback`

Contient un pointeur vers une fonction de rappel appelée lorsque le bouton **Appliquer** de la boîte de dialogue Gestion des couleurs est sélectionné. Si aucune fonction de rappel n'est fournie, ce membre doit être défini sur **NULL**. Consultez **PCMSCALLBACKW**.

`lParamApplyCallback`

Contient une valeur qui sera transmise à la fonction **ApplyCallbackFunction** via son paramètre *lParam* . La signification et le contenu de la valeur sont spécifiés par l'application.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Voir aussi

- Boîte de dialogue commune pour la gestion des couleurs
- DialogProc
- EndDialog

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

STRUCTURE COLORMATCHSETUPW (icm.h)

Article24/08/2023

La structure **COLORMATCHSETUP** contient des informations que la fonction SetupColorMatchingW utilise pour initialiser la boîte de dialogue **ColorManagement** . Une fois la boîte de dialogue fermée par l'utilisateur, **SetupColorMatching retourne des** informations sur la sélection de l'utilisateur dans cette structure.

Syntaxe

C++

```
typedef struct _tagCOLORMATCHSETUPW {
  DWORD          dwSize;
  DWORD          dwVersion;
  DWORD          dwFlags;
  HWND           hwndOwner;
  PCWSTR         pSourceName;
  PCWSTR         pDisplayName;
  PCWSTR         pPrinterName;
  DWORD          dwRenderIntent;
  DWORD          dwProofingIntent;
  PWSTR          pMonitorProfile;
  DWORD          ccMonitorProfile;
  PWSTR          pPrinterProfile;
  DWORD          ccPrinterProfile;
  PWSTR          pTargetProfile;
  DWORD          ccTargetProfile;
  DLGPROC        lpfnHook;
  LPARAM         lParam;
  PCMSCALLBACKW lpfnApplyCallback;
  LPARAM         lParamApplyCallback;
} COLORMATCHSETUPW, *PCOLORMATCHSETUPW, *LPCOLORMATCHSETUPW;
```

Membres

`dwSize`

Taille de la structure. Doit être défini sur **sizeof** (**COLORMATCHSETUP**).

`dwVersion`

Version de la structure **COLORMATCHSETUP** . Cette valeur doit être définie sur COLOR_MATCH_VERSION.

`dwFlags`

Ensemble d'indicateurs de bits utilisés pour initialiser la boîte de dialogue. Si la valeur est 0 lors de l'entrée, tous les contrôles supposent leur état par défaut.

Lorsque la boîte de dialogue retourne, ces indicateurs sont définis pour indiquer l'entrée de l'utilisateur.

Ce membre peut être défini à l'aide d'une combinaison des indicateurs suivants.

Indicateur	**Signification**
CMS_DISABLEICM	S'il est défini lors de l'entrée, cet indicateur indique que la zone de case activée « Activer la gestion des couleurs » est désactivée, désactivant tous les autres contrôles. Si la valeur est définie à la sortie, cela signifie que l'utilisateur ne souhaite pas que la gestion des couleurs soit effectuée.
CMS_ENABLEPROOFING	S'il est défini lors de l'entrée, cet indicateur indique que les contrôles de vérification doivent être activés et que la case Vérification case activée est cochée. Si la valeur est définie à la sortie, cela signifie que l'utilisateur souhaite effectuer la gestion des couleurs pour un appareil cible différent de l'imprimante sélectionnée.
CMS_SETRENDERINTENT	S'il est défini sur l'entrée, cet indicateur indique que le membre **dwRenderIntent** contient la valeur à utiliser pour initialiser le contrôle Intention de rendu. Sinon, la valeur par défaut du contrôle est Rendu d'image. Cet indicateur est défini lors de la sortie si WCS est activé.
CMS_SETPROOFINTENT	Ignoré, sauf si CMS_ENABLEPROOFING est également défini. S'il est défini lors de l'entrée et que CMS_ENABLEPROOFING est également défini, cet indicateur indique que le membre **dwProofingIntent** doit être utilisé pour initialiser le contrôle Intention de rendu cible. Sinon, la valeur par défaut du contrôle est Rendu d'image. Cet indicateur est défini à la sortie si la vérification linguistique est activée.
CMS_SETMONITORPROFILE	S'il est défini lors de l'entrée, cet indicateur indique que le profil de gestion des couleurs nommé dans le membre **pMonitorProfile** doit être la sélection initiale dans le contrôle de profil du moniteur. Si le profil spécifié n'est pas associé au moniteur, cet indicateur est ignoré et le profil par défaut du moniteur est utilisé.
CMS_SETPRINTERPROFILE	S'il est défini lors de l'entrée, cet indicateur indique que le profil de gestion des couleurs nommé dans le membre **pPrinterProfile** doit

Indicateur	Signification
	être la sélection initiale dans le contrôle de profil d'imprimante. Si le profil spécifié n'est pas associé à l'imprimante, cet indicateur est ignoré et le profil par défaut de l'imprimante est utilisé.
CMS_SETTARGETPROFILE	S'il est défini lors de l'entrée, cet indicateur indique que le profil de couleur nommé dans le membre **pTargetProfile** doit être la sélection initiale dans le contrôle de profil cible. Si le profil spécifié n'est pas installé, cet indicateur est ignoré et le profil par défaut de l'imprimante est utilisé. Si l'imprimante n'a pas de profil par défaut, le premier profil par ordre alphabétique s'affiche.
CMS_USEHOOK	Cet indicateur spécifie que le membre **lpfnHook** contient l'adresse d'une procédure de hook et que le membre **lParam** contient une valeur à passer à la procédure de hook lorsque le message WM_INITDIALOG est envoyé.
CMS_MONITOROVERFLOW	Cet indicateur est défini à la sortie si la gestion des couleurs doit être activée et si la taille de mémoire tampon indiquée dans **ccMonitorProfile** est insuffisante pour le nom de profil sélectionné. **GetLastError** retourne ERROR_INSUFFICIENT_BUFFER dans ce cas.
CMS_PRINTERROVERFLOW	Cet indicateur est défini à la sortie si la gestion des couleurs doit être activée et si la taille de mémoire tampon indiquée dans **ccPrinterProfile** est insuffisante pour le nom de profil sélectionné. **GetLastError** retourne ERROR_INSUFFICIENT_BUFFER dans ce cas.
CMS_TARGETOVERFLOW	Cet indicateur est défini à la sortie si la vérification linguistique doit être activée et si la taille de mémoire tampon indiquée dans **ccTargetProfile** est insuffisante pour le nom de profil sélectionné. **GetLastError** retourne ERROR_INSUFFICIENT_BUFFER dans ce cas.
CMS_USEAPPLYCALLBACK	Si défini lors de l'entrée, cet indicateur indique que la fonction **SetupColorMatching** doit appeler la fonction **PCMSCALLBACKW**. L'adresse de la fonction de rappel est contenue dans *lpfnApplyCallback*.
CMS_USEDESCRIPTION	S'il est défini lors de l'entrée, cet indicateur indique à la fonction **SetupColorMatching** de récupérer la description de profil contenue dans les étiquettes de description du profil (voir Spécification de format de profil ICC v3.4). Il les insère dans les zones **d'édition Profil du moniteur**, **Profil d'imprimante** et **Profil d'appareil émulé** de la boîte de dialogue Gestion des **couleurs** commune.

`hwndOwner`

Handle de fenêtre pour le propriétaire de la boîte de dialogue, ou **NULL** si la boîte de dialogue n'a pas de propriétaire.

`pSourceName`

Pointeur vers une chaîne spécifiée par l'application qui décrit le profil source de l'élément pour lequel la gestion des couleurs doit être effectuée. Si cette valeur est **NULL**, le contrôle Source d'image affiche le nom du profil de couleur par défaut De Windows.

`pDisplayName`

Pointe vers une chaîne nommant le moniteur à utiliser pour la gestion des couleurs. S'il ne s'agit pas du nom d'un moniteur valide, le premier moniteur énuméré est utilisé.

`pPrinterName`

Pointe vers une chaîne nommant l'imprimante sur laquelle l'image doit être affichée. S'il ne s'agit pas d'un nom d'imprimante valide, l'imprimante par défaut est utilisée et nommée dans la boîte de dialogue.

`dwRenderIntent`

Type de gestion des couleurs souhaité. Les valeurs autorisées sont :

INTENT_PERCEPTUAL

INTENT_SATURATION

INTENT_RELATIVE_COLORIMETRIC

INTENT_ABSOLUTE_COLORIMETRIC

Pour plus d'informations, consultez Intentions de rendu.

`dwProofingIntent`

Type de gestion des couleurs souhaité pour l'image avec preuve. Les valeurs autorisées sont :

INTENT_PERCEPTUAL

INTENT_SATURATION

INTENT_RELATIVE_COLORIMETRIC

INTENT_ABSOLUTE_COLORIMETRIC

Pour plus d'informations, consultez Intentions de rendu.

`pMonitorProfile`

Pointeur vers une mémoire tampon dans laquelle placer le nom du profil d'analyse sélectionné par l'utilisateur. Si l'indicateur CMS_SETMONITORPROFILE est utilisé, cet indicateur peut également être utilisé pour sélectionner un profil autre que le profil par défaut du moniteur lorsque la boîte de dialogue est affichée pour la première fois.

`ccMonitorProfile`

Taille de la mémoire tampon pointée vers le membre **pMonitorProfile** , en caractères. Si la mémoire tampon n'est pas assez grande pour contenir le nom sélectionné, le nom est tronqué à cette taille et ERROR_INSUFFICIENT_BUFFER est retourné. Une mémoire tampon de MAX_PATH taille fonctionne toujours.

`pPrinterProfile`

Pointe vers une mémoire tampon dans laquelle placer le nom du profil d'imprimante sélectionné par l'utilisateur. Si l'indicateur CMS_SETPRINTERPROFILE est utilisé, cet indicateur peut également être utilisé pour sélectionner un profil autre que celui par défaut de l'imprimante lorsque la boîte de dialogue est affichée pour la première fois.

`ccPrinterProfile`

Taille de la mémoire tampon pointée vers le membre **pPrinterProfile** , en caractères. Si la mémoire tampon n'est pas assez grande pour contenir le nom sélectionné, le nom est tronqué à cette taille et ERROR_INSUFFICIENT_BUFFER est retourné. Une mémoire tampon de MAX_PATH taille fonctionne toujours.

`pTargetProfile`

Pointe vers une mémoire tampon dans laquelle placer le nom du profil cible sélectionné par l'utilisateur pour la vérification linguistique. Si l'indicateur CMS_SETTARGETPROFILE est utilisé, cet indicateur peut également être utilisé pour sélectionner un profil autre que l'imprimante par défaut lors de la première affichage de la boîte de dialogue.

`ccTargetProfile`

Taille de la mémoire tampon pointée vers le membre **pTargetProfile** , en caractères. Si la mémoire tampon n'est pas assez grande pour contenir le nom sélectionné, le nom est tronqué à cette taille et ERROR_INSUFFICIENT_BUFFER est retourné. Une mémoire tampon de MAX_PATH taille fonctionne toujours.

`lpfnHook`

Si l'indicateur CMS_USEHOOK est défini, ce membre est l'adresse d'une procédure de boîte de dialogue (voir DialogProc) qui peut filtrer ou gérer les messages pour la boîte de dialogue. La procédure de hook ne reçoit aucun message émis avant

WM_INITDIALOG. Il est appelé sur le message WM_INITDIALOG une fois que la procédure de boîte de dialogue fournie par le système a traité le message. Sur tous les autres messages, la procédure de crochet reçoit le message avant la procédure fournie par le système. Si la procédure de hook retourne **TRUE** à ces messages, la procédure fournie par le système n'est pas appelée.

La procédure de hook peut appeler la fonction EndDialog .

`lParam`

Si l'indicateur CMS_USEHOOK est défini, ce membre est passé à la procédure de hook fournie par l'application en tant que paramètre *lParam* lorsque le message WM_INITDIALOG est traité.

`lpfnApplyCallback`

Contient un pointeur vers une fonction de rappel appelée lorsque le bouton **Appliquer** de la boîte de dialogue Gestion des couleurs est sélectionné. Si aucune fonction de rappel n'est fournie, ce membre doit être défini sur **NULL**. Consultez **PCMSCALLBACKW**.

`lParamApplyCallback`

Contient une valeur qui sera transmise à la fonction **ApplyCallbackFunction** via son paramètre *lParam* . La signification et le contenu de la valeur sont spécifiés par l'application.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Voir aussi

- Boîte de dialogue commune pour la gestion des couleurs
- DialogProc
- EndDialog

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

Structure ENUMTYPEA (icm.h)

Article24/08/2023

Contient des informations qui définissent les contraintes d'énumération de profil.

Syntaxe

C++

```
typedef struct tagENUMTYPEA {
  DWORD dwSize;
  DWORD dwVersion;
  DWORD dwFields;
  PCSTR pDeviceName;
  DWORD dwMediaType;
  DWORD dwDitheringMode;
  DWORD dwResolution[2];
  DWORD dwCMMType;
  DWORD dwClass;
  DWORD dwDataColorSpace;
  DWORD dwConnectionSpace;
  DWORD dwSignature;
  DWORD dwPlatform;
  DWORD dwProfileFlags;
  DWORD dwManufacturer;
  DWORD dwModel;
  DWORD dwAttributes[2];
  DWORD dwRenderingIntent;
  DWORD dwCreator;
  DWORD dwDeviceClass;
} ENUMTYPEA, *PENUMTYPEA, *LPENUMTYPEA;
```

Membres

`dwSize`

Taille de cette structure en octets.

`dwVersion`

Numéro de version de la structure **ENUMTYPE** . Doit être défini sur ENUM_TYPE_VERSION.

`dwFields`

Indique les champs de cette structure qui sont utilisés. Peut être défini sur n'importe quelle combinaison des valeurs constantes suivantes.

ET_DEVICENAME

ET_MEDIATYPE

ET_DITHERMODE

ET_RESOLUTION

ET_CMMTYPE

ET_CLASS

ET_DATACOLORSPACE

ET_CONNECTIONSPACE

ET_SIGNATURE

ET_PLATFORM

ET_PROFILEFLAGS

ET_MANUFACTURER

ET_MODEL

ET_ATTRIBUTES

ET_RENDERINGINTENT

ET_CREATOR

ET_DEVICECLASS

`pDeviceName`

Nom convivial de l'appareil.

`dwMediaType`

Indique le type de média associé au profil, tel qu'une imprimante ou un écran.

`dwDitheringMode`

Indique le style de dithering qui sera utilisé lorsqu'une image est affichée.

`dwResolution[2]`

Résolution horizontale (x) et verticale (y) en pixels de l'appareil sur lequel l'image sera affichée. La résolution x est stockée dans **dwResolution[0]** et la résolution y est conservée dans **dwResolution[1]** .

`dwCMMType`

Numéro d'identification de la gestion CMM utilisée dans le profil. Les numéros d'identification sont enregistrés auprès de l'ICC.

`dwClass`

Indique la classe de profil. Pour obtenir une description des classes de profil, consultez Utilisation de profils d'appareil avec WCS. Une classe de profil peut avoir l'une des valeurs suivantes.

Classe de profil	**Signature**
Profil d'appareil d'entrée	CLASS_SCANNER
Afficher le profil de l'appareil	CLASS_MONITOR
Profil d'appareil de sortie	CLASS_PRINTER
Profil Device Link	CLASS_LINK
Profil de conversion d'espace de couleur	CLASS_COLORSPACE
Profil abstrait	CLASS_ABSTRACT
Profil de couleur nommé	CLASS_NAMED
Profil de modèle d'apparence de couleur	CLASS_CAMP
Profil de modèle de carte de la gamme de couleurs	CLASS_GMMP

`dwDataColorSpace`

Valeur de signature qui indique l'espace de couleur dans lequel les données de profil sont définies. Il peut s'agir de n'importe quelle valeur des constantes d'espace de couleur.

`dwConnectionSpace`

Valeur de signature qui indique l'espace de couleur dans lequel l'espace de connexion de profil (PCS) est défini. Il peut s'agir de l'une des valeurs suivantes.

Classe de profil	Signature
XYZ	SPACE_XYZ
Laboratoire	SPACE_Lab

Lorsque le membre **dwClass** est défini sur CLASS_LINK, le PCS est extrait du membre **dwDataColorSpace** .

`dwSignature`

Réservé à un usage interne.

`dwPlatform`

Plateforme principale pour laquelle le profil a été créé. Le membre peut être défini sur l'une des valeurs suivantes.

Plateforme	Valeur
Apple Computer, Inc.	'APPL'
Microsoft Corp.	'MSFT'
Silicon Graphics, Inc.	'SGI'
SUN MICROSYSTEMS, INC.	'SUNW'
Taligent	'TGNT'

`dwProfileFlags`

Indicateurs de bits contenant des indicateurs que le CMM utilise pour interpréter les données de profil et peuvent être définis sur l'une des valeurs suivantes.

Constant	Signification
FLAG_EMBEDDEDPROFILE	Le profil est incorporé dans un fichier bitmap.
FLAG_DEPENDENTONDATA	Le profil ne peut pas être utilisé indépendamment des données de couleur incorporées. Utilisé pour les profils incorporés dans des fichiers bitmap.

`dwManufacturer`

Numéro d'identification du fabricant du profil d'appareil. Tous les numéros d'identification du fabricant sont enregistrés auprès de l'ICC.

`dwModel`

Numéro de modèle d'appareil du fabricant de l'appareil. Tous les numéros d'identification de modèle sont enregistrés auprès de l'ICC.

`dwAttributes[2]`

Attributs de profil qui peuvent être l'une des valeurs suivantes.

Constant	Signification
ATTRIB_TRANSPARENCY	Active la transparence. Si cet indicateur n'est pas utilisé, l'attribut est réfléchissant par défaut.
ATTRIB_MATTE	Active l'affichage mat. Si cet indicateur n'est pas utilisé, l'attribut est brillant par défaut.

`dwRenderingIntent`

Intention de rendu de profil qui peut être définie sur l'une des valeurs suivantes :

INTENT_PERCEPTUAL

INTENT_SATURATION

INTENT_RELATIVE_COLORIMETRIC

INTENT_ABSOLUTE_COLORIMETRIC

Pour plus d'informations, consultez Intentions de rendu.

`dwCreator`

Signature du logiciel qui a créé le profil. Les signatures sont enregistrées auprès de l'ICC.

`dwDeviceClass`

Indique la classe d'appareil. Une classe d'appareil peut avoir l'une des valeurs suivantes.

Classe de profil	Signature
Profil d'appareil d'entrée	CLASS_SCANNER
Afficher le profil de l'appareil	CLASS_MONITOR
Profil d'appareil de sortie	CLASS_PRINTER

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Voir aussi

- Informations supplémentaires
- Utilisation de profils d'appareil avec WCS
- Intentions de rendu

Commentaires

Structure ENUMTYPEW (icm.h)

Article24/08/2023

Contient des informations qui définissent les contraintes d'énumération de profil.

Syntaxe

C++

```
typedef struct tagENUMTYPEW {
  DWORD  dwSize;
  DWORD  dwVersion;
  DWORD  dwFields;
  PCWSTR pDeviceName;
  DWORD  dwMediaType;
  DWORD  dwDitheringMode;
  DWORD  dwResolution[2];
  DWORD  dwCMMType;
  DWORD  dwClass;
  DWORD  dwDataColorSpace;
  DWORD  dwConnectionSpace;
  DWORD  dwSignature;
  DWORD  dwPlatform;
  DWORD  dwProfileFlags;
  DWORD  dwManufacturer;
  DWORD  dwModel;
  DWORD  dwAttributes[2];
  DWORD  dwRenderingIntent;
  DWORD  dwCreator;
  DWORD  dwDeviceClass;
} ENUMTYPEW, *PENUMTYPEW, *LPENUMTYPEW;
```

Membres

`dwSize`

Taille de cette structure en octets.

`dwVersion`

Numéro de version de la structure **ENUMTYPE** . Doit être défini sur ENUM_TYPE_VERSION.

`dwFields`

Indique les champs de cette structure qui sont utilisés. Peut être défini sur n'importe quelle combinaison des valeurs constantes suivantes.

ET_DEVICENAME

ET_MEDIATYPE

ET_DITHERMODE

ET_RESOLUTION

ET_CMMTYPE

ET_CLASS

ET_DATACOLORSPACE

ET_CONNECTIONSPACE

ET_SIGNATURE

ET_PLATFORM

ET_PROFILEFLAGS

ET_MANUFACTURER

ET_MODEL

ET_ATTRIBUTES

ET_RENDERINGINTENT

ET_CREATOR

ET_DEVICECLASS

`pDeviceName`

Nom convivial de l'appareil.

`dwMediaType`

Indique le type de média associé au profil, tel qu'une imprimante ou un écran.

`dwDitheringMode`

Indique le style de dithering qui sera utilisé lorsqu'une image est affichée.

`dwResolution[2]`

Résolution horizontale (x) et verticale (y) en pixels de l'appareil sur lequel l'image sera affichée. La résolution x est stockée dans **dwResolution[0]** et la résolution y est conservée dans **dwResolution[1]** .

`dwCMMType`

Numéro d'identification de la gestion CMM utilisée dans le profil. Les numéros d'identification sont enregistrés auprès de l'ICC.

`dwClass`

Indique la classe de profil. Pour obtenir une description des classes de profil, consultez Utilisation de profils d'appareil avec WCS. Une classe de profil peut avoir l'une des valeurs suivantes.

Classe de profil	**Signature**
Profil d'appareil d'entrée	CLASS_SCANNER
Afficher le profil de l'appareil	CLASS_MONITOR
Profil d'appareil de sortie	CLASS_PRINTER
Profil Device Link	CLASS_LINK
Profil de conversion d'espace de couleur	CLASS_COLORSPACE
Profil abstrait	CLASS_ABSTRACT
Profil de couleur nommé	CLASS_NAMED
Profil de modèle d'apparence de couleur	CLASS_CAMP
Profil de modèle de carte de la gamme de couleurs	CLASS_GMMP

`dwDataColorSpace`

Valeur de signature qui indique l'espace de couleur dans lequel les données de profil sont définies. Il peut s'agir de n'importe quelle valeur des constantes d'espace de couleur.

`dwConnectionSpace`

Valeur de signature qui indique l'espace de couleur dans lequel l'espace de connexion de profil (PCS) est défini. Il peut s'agir de l'une des valeurs suivantes.

Classe de profil	Signature
XYZ	SPACE_XYZ
Laboratoire	SPACE_Lab

Lorsque le membre **dwClass** est défini sur CLASS_LINK, le PCS est extrait du membre **dwDataColorSpace** .

`dwSignature`

Réservé à un usage interne.

`dwPlatform`

Plateforme principale pour laquelle le profil a été créé. Le membre peut être défini sur l'une des valeurs suivantes.

Plateforme	Valeur
Apple Computer, Inc.	'APPL'
Microsoft Corp.	'MSFT'
Silicon Graphics, Inc.	'SGI'
SUN MICROSYSTEMS, INC.	'SUNW'
Taligent	'TGNT'

`dwProfileFlags`

Indicateurs de bits contenant des indicateurs que le CMM utilise pour interpréter les données de profil et peuvent être définis sur l'une des valeurs suivantes.

Constant	Signification
FLAG_EMBEDDEDPROFILE	Le profil est incorporé dans un fichier bitmap.
FLAG_DEPENDENTONDATA	Le profil ne peut pas être utilisé indépendamment des données de couleur incorporées. Utilisé pour les profils incorporés dans des fichiers bitmap.

`dwManufacturer`

Numéro d'identification du fabricant du profil d'appareil. Tous les numéros d'identification du fabricant sont enregistrés auprès de l'ICC.

`dwModel`

Numéro de modèle d'appareil du fabricant de l'appareil. Tous les numéros d'identification de modèle sont enregistrés auprès de l'ICC.

`dwAttributes[2]`

Attributs de profil qui peuvent être l'une des valeurs suivantes.

Constant	Signification
ATTRIB_TRANSPARENCY	Active la transparence. Si cet indicateur n'est pas utilisé, l'attribut est réfléchissant par défaut.
ATTRIB_MATTE	Active l'affichage mat. Si cet indicateur n'est pas utilisé, l'attribut est brillant par défaut.

`dwRenderingIntent`

Intention de rendu de profil qui peut être définie sur l'une des valeurs suivantes :

INTENT_PERCEPTUAL

INTENT_SATURATION

INTENT_RELATIVE_COLORIMETRIC

INTENT_ABSOLUTE_COLORIMETRIC

Pour plus d'informations, consultez Intentions de rendu.

`dwCreator`

Signature du logiciel qui a créé le profil. Les signatures sont enregistrées auprès de la CPI.

`dwDeviceClass`

Indique la classe d'appareil. Une classe d'appareil peut avoir l'une des valeurs suivantes.

Profile, classe	Signature
Profil de périphérique d'entrée	CLASS_SCANNER
Afficher le profil d'appareil	CLASS_MONITOR
Profil d'appareil de sortie	CLASS_PRINTER

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Voir aussi

- Informations supplémentaires
- Utilisation de profils d'appareil avec WCS
- Intentions de rendu

Commentaires

Structure GamutBoundaryDescription (wcsplugin.h)

Article27/08/2023

Définit une limite de gamut.

Syntaxe

```
C++
```

```
typedef struct _GamutBoundaryDescription {
  PrimaryJabColors *pPrimaries;
  UINT             cNeutralSamples;
  JabColorF        *pNeutralSamples;
  GamutShell       *pReferenceShell;
  GamutShell       *pPlausibleShell;
  GamutShell       *pPossibleShell;
} GamutBoundaryDescription;
```

Membres

`pPrimaries`

`cNeutralSamples`

Nombre d'échantillons neutres.

`pNeutralSamples`

`pReferenceShell`

`pPlausibleShell`

`pPossibleShell`

Configuration requise

Client minimal pris en charge	Windows Vista [applications de bureau uniquement]
Serveur minimal pris en charge	Windows Server 2008 [applications de bureau uniquement]

En-tête	wcsplugin.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Structures
- Schémas et algorithmes windows Color System

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

Structure GamutShell (wcsplugin.h)

Article27/08/2023

Contient des informations qui définissent un interpréteur de commandes gamut, qui est représenté par une liste de triangles indexés. La mémoire tampon de vertex contient les données de sommets.

Syntaxe

C++

```
typedef struct _GamutShell {
  FLOAT              JMin;
  FLOAT              JMax;
  UINT               cVertices;
  UINT               cTriangles;
  JabColorF          *pVertices;
  GamutShellTriangle *pTriangles;
} GamutShell;
```

Membres

`JMin`

Légèreté minimale de l'interpréteur de commandes.

`JMax`

Légèreté maximale de l'interpréteur de commandes.

`cVertices`

Nombre de sommets dans l'interpréteur de commandes.

`cTriangles`

Nombre de triangles dans l'interpréteur de commandes.

`pVertices`

`pTriangles`

Configuration requise

Client minimal pris en charge	Windows Vista [applications de bureau uniquement]
Serveur minimal pris en charge	Windows Server 2008 [applications de bureau uniquement]
En-tête	wcsplugin.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Structures
- Schémas et algorithmes du système de couleurs Windows

Commentaires

Structure GamutShellTriangle (wcsplugin.h)

Article27/08/2023

Contient trois index de vertex pour accéder à une mémoire tampon de vertex.

Syntaxe

```
C++
```

```
typedef struct _GamutShellTriangle {
  UINT aVertexIndex[3];
} GamutShellTriangle;
```

Membres

`aVertexIndex[3]`

Tableau de trois index de vertex utilisés pour accéder à une mémoire tampon de vertex.

Configuration requise

Client minimal pris en charge	Windows Vista [applications de bureau uniquement]
Serveur minimal pris en charge	Windows Server 2008 [applications de bureau uniquement]
En-tête	wcsplugin.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Structures
- Schémas et algorithmes du système de couleurs Windows

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l’aide sur Microsoft Q&A

structure GENERIC3CHANNEL (icm.h)

Article24/08/2023

TBD

Syntaxe

```
C++

struct GENERIC3CHANNEL {
  WORD ch1;
  WORD ch2;
  WORD ch3;
};
```

Membres

`ch1`

TBD

`ch2`

TBD

`ch3`

TBD

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l’aide sur Microsoft Q&A

Structure GRAYCOLOR (icm.h)

Article24/08/2023

Description de la structure GRAYCOLOR.

Syntaxe

```
C++
```

```
struct GRAYCOLOR {
  WORD gray;
};
```

Membres

`gray`

TBD

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Commentaires

Structure HiFiCOLOR (icm.h)

Article24/08/2023

Description de la structure HiFiCOLOR.

Syntaxe

```
C++
```

```
struct HiFiCOLOR {
  BYTE channel[MAX_COLOR_CHANNELS];
};
```

Membres

`channel[MAX_COLOR_CHANNELS]`

TBD

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Commentaires

Structure JabColorF (wcsplugin.h)

Article27/08/2023

TBD

Syntaxe

```
C++
```

```
typedef struct _JabColorF {
  FLOAT J;
  FLOAT a;
  FLOAT b;
} JabColorF;
```

Membres

`J`

TBD

`a`

TBD

`b`

TBD

Configuration requise

En-tête	wcsplugin.h

Commentaires

Obtenir de l’aide sur Microsoft Q&A

Structure JChColorF (wcsplugin.h)

Article08/03/2023

TBD

Syntaxe

C++

```
typedef struct _JChColorF {
  FLOAT J;
  FLOAT C;
  FLOAT h;
} JChColorF;
```

Membres

`J`

TBD

`C`

TBD

`h`

TBD

Configuration requise

En-tête	wcsplugin.h

Structure LabCOLOR (icm.h)

Article29/02/2024

TBD

Syntaxe

```
C++

struct LabCOLOR {
  WORD L;
  WORD a;
  WORD b;
};
```

Membres

`L`

TBD

`a`

TBD

`b`

TBD

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Commentaires

Cette page a-t-elle été utile ? Yes No

Indiquer des commentaires sur le produit | Obtenir de l'aide sur Microsoft Q&A

STRUCTURE LOGCOLORSPACEA (wingdi.h)

Article04/03/2024

La structure **LOGCOLORSPACE** contient des informations qui définissent un espace de couleurs logique.

Syntaxe

```
C++
```

```
typedef struct tagLOGCOLORSPACEA {
  DWORD         lcsSignature;
  DWORD         lcsVersion;
  DWORD         lcsSize;
  LCSCSTYPE     lcsCSType;
  LCSGAMUTMATCH lcsIntent;
  CIEXYZTRIPLE  lcsEndpoints;
  DWORD         lcsGammaRed;
  DWORD         lcsGammaGreen;
  DWORD         lcsGammaBlue;
  CHAR          lcsFilename[MAX_PATH];
} LOGCOLORSPACEA, *LPLOGCOLORSPACEA;
```

Membres

`lcsSignature`

Signature d'espace de couleur. À l'heure actuelle, ce membre doit toujours être défini sur LCS_SIGNATURE.

`lcsVersion`

Numéro de version ; doit être 0x400.

`lcsSize`

Taille de cette structure, en octets.

`lcsCSType`

Type d'espace de couleur. Le membre peut être l'une des valeurs suivantes.

Agrandir le tableau

Valeur	Signification
LCS_CALIBRATED_RGB	Les valeurs de couleur sont des valeurs RVB étalonnées. Les valeurs sont traduites à l'aide des points de terminaison spécifiés par le membre **lcsEndpoints** avant d'être transmises à l'appareil.
LCS_sRGB	Les valeurs de couleur sont des valeurs sRGB.
LCS_WINDOWS_COLOR_SPACE	Les valeurs de couleur sont des valeurs de couleur d'espace de couleur par défaut de Windows.

Si LCS_CALIBRATED_RGB n'est pas spécifié, le membre **lcsEndpoints** est ignoré.

`lcsIntent`

Méthode de mappage de gamut. Ce membre peut être l'une des valeurs suivantes.

Agrandir le tableau

Valeur	Intentionnel	Nom ICC	Signification
LCS_GM_ABS_ COLORIMÉTRIQUE	Correspond	Colorimétrie absolue	Conservez le point blanc. Faire correspondre les couleurs à leur couleur la plus proche dans la gamme de destination.
LCS_GM_ ENTREPRISE	Graphic	Saturation	Maintenir la saturation. Utilisé pour les graphiques métier et d'autres situations dans lesquelles des couleurs non associées sont requises.
LCS_GM_ GRAPHIQUES	Proof	Colorimétrie relative	Maintenir la correspondance colorimétrique. Utilisé pour les conceptions graphiques et les couleurs nommées.
LCS_GM_ IMAGES	Image	Perception	Maintenir le contraste. Utilisé pour les photographies et les images naturelles.

`lcsEndpoints`

Points de terminaison rouges, verts et bleus.

`lcsGammaRed`

Mise à l'échelle de la coordonnée rouge.

`lcsGammaGreen`

Mise à l'échelle de la coordonnée verte.

`lcsGammaBlue`

Mise à l'échelle de la coordonnée bleue.

`lcsFilename[MAX_PATH]`

Chaîne terminée par null qui nomme un fichier de profil de couleur. Ce membre est généralement défini sur zéro, mais peut être utilisé pour définir l'espace de couleurs exactement comme spécifié par le profil de couleur. Cela est utile pour les appareils qui entrent des valeurs de couleur pour une imprimante spécifique, ou lors de l'utilisation d'un matcheur de couleurs d'image installable. Si un profil de couleur est spécifié, tous les autres membres de cette structure doivent être définis sur des valeurs raisonnables, même si les valeurs ne sont pas totalement exactes.

Remarques

Comme les palettes, mais contrairement aux stylets et aux pinceaux, un pointeur doit être passé lors de la création d'un LogColorSpace.

Si le membre **lcsCSType** est défini sur LCS_sRGB ou LCS_WINDOWS_COLOR_SPACE, les autres membres de cette structure sont ignorés et WCS utilise l'espace de couleurs sRGB. Les membres **lcsEndpoints,lcsGammaRed, lcsGammaGreen** et **lcsGammaBlue** sont utilisés pour décrire l'espace de couleurs logique. Le membre **lcsEndpoints** est un **CIEXYZTRIPLE** qui contient les valeurs x, y et z du point de terminaison RVB de l'espace de couleurs.

Le format de bits DWORD requis pour **lcsGammaRed**, **lcsGammaGreen** et **lcsGammaBlue** est un entier à point fixe 8,8 décalé vers la gauche de 8 bits. Cela signifie que 8 bits entiers sont suivis de 8 bits de fraction. Si l'on tient compte du décalage de bits, le format requis du DWORD 32 bits est le suivant :

00000000000nnnnnnff0000000000000

Chaque fois que le membre **lcsFilename** contient un nom de fichier et que le membre **lcsCSType** est défini sur LCS_CALIBRATED_RGB, WCS ignore les autres membres de cette structure. Il utilise l'espace de couleurs dans le fichier comme espace de couleurs auquel fait référence cette structure **LOGCOLORSPACE** .

La relation entre les valeurs de trois stimulus X,Y,Z et les valeurs de chromaticité x,y,z est la suivante :

x = X/(X+Y+Z)

y = Y/(X+Y+Z)

z = Z/(X+Y+Z)

Si le membre lcsCSType est défini sur LCS_sRGB ou LCS_WINDOWS_COLOR_SPACE, les autres membres de cette structure sont ignorés et ICM utilise l'espace de couleurs sRGB. Les applications doivent toujours initialiser le reste de la structure, car CreateProfileFromLogColorSpace ignore le membre lcsCSType et utilise les membres lcsEndpoints, lcsGammaRed, lcsGammaGreen et lcsGammaBlue pour créer un profil, qui ne peut pas être initialisé en cas d'espaces de couleurs LCS_sRGB ou LCS_WINDOWS_COLOR_SPACE.

> **Notes**
>
> L'en-tête wingdi.h définit LOGCOLORSPACE comme un alias qui sélectionne automatiquement la version ANSI ou Unicode de cette fonction en fonction de la définition de la constante de préprocesseur UNICODE. Le mélange de l'utilisation de l'alias neutre en encodage avec du code qui n'est pas neutre en encodage peut entraîner des incompatibilités qui entraînent des erreurs de compilation ou d'exécution. Pour plus d'informations, consultez Conventions pour les prototypes de fonction.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	wingdi.h

Voir aussi

BITMAPV4HEADER

BITMAPV5HEADER

CMJN

RGB

Commentaires

Cette page a-t-elle été utile ? Yes No

Indiquer des commentaires sur le produit | Obtenir de l'aide sur Microsoft Q&A

Structure NAMEDCOLOR (icm.h)

Article26/02/2024

TBD

Syntaxe

```
C++

struct NAMEDCOLOR {
  DWORD dwIndex;
};
```

Membres

`dwIndex`

TBD

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Commentaires

structure NAMED_PROFILE_INFO (icm.h)

Article29/02/2024

La structure **NAMED_PROFILE_INFO** est utilisée pour stocker des informations sur un profil de couleur nommé.

Syntaxe

```
C++
```

```
typedef struct tagNAMED_PROFILE_INFO {
  DWORD      dwFlags;
  DWORD      dwCount;
  DWORD      dwCountDevCoordinates;
  COLOR_NAME szPrefix;
  COLOR_NAME szSuffix;
} NAMED_PROFILE_INFO;
```

Membres

`dwFlags`

Non utilisé actuellement par la gestion CMM par défaut.

`dwCount`

Nombre total de couleurs nommées dans le profil.

`dwCountDevCoordinates`

Nombre total de coordonnées d'appareil pour chaque couleur nommée.

`szPrefix`

Pointeur vers une chaîne contenant le préfixe de chaque nom de couleur.

`szSuffix`

Pointeur vers une chaîne contenant le suffixe de chaque nom de couleur.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Commentaires

Cette page a-t-elle été utile ? Yes No

Indiquer des commentaires sur le produit | Obtenir de l'aide sur Microsoft Q&A

Structure PrimaryJabColors (wcsplugin.h)

Article27/08/2023

Cette structure contient huit couleurs primaires en coordonnées Jab.

Syntaxe

```
C++
```

```
typedef struct _PrimaryJabColors {
  JabColorF red;
  JabColorF yellow;
  JabColorF green;
  JabColorF cyan;
  JabColorF blue;
  JabColorF magenta;
  JabColorF black;
  JabColorF white;
} PrimaryJabColors;
```

Membres

`red`

Primaire rouge.

`yellow`

Primaire jaune.

`green`

Primaire verte.

`cyan`

Principal cyan.

`blue`

Primaire bleue.

`magenta`

Magenta primary.

`black`

Primaire noire.

`white`

Primaire blanche.

Configuration requise

Client minimal pris en charge	Windows Vista [applications de bureau uniquement]
Serveur minimal pris en charge	Windows Server 2008 [applications de bureau uniquement]
En-tête	wcsplugin.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Structures

Commentaires

Structure PrimaryXYZColors (wcsplugin.h)

Article03/03/2024

Cette structure contient huit couleurs primaires en coordonnées XYZ.

Syntaxe

```
C++
```

```
typedef struct _PrimaryXYZColors {
  XYZColorF red;
  XYZColorF yellow;
  XYZColorF green;
  XYZColorF cyan;
  XYZColorF blue;
  XYZColorF magenta;
  XYZColorF black;
  XYZColorF white;
} PrimaryXYZColors;
```

Membres

`red`

Primaire rouge.

`yellow`

Primaire jaune.

`green`

Primaire verte.

`cyan`

Principal cyan.

`blue`

Primaire bleue.

`magenta`

Magenta primary.

`black`

Primaire noire.

`white`

Primaire blanche.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows Vista [applications de bureau uniquement]
Serveur minimal pris en charge	Windows Server 2008 [applications de bureau uniquement]
En-tête	wcsplugin.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Structures

Commentaires

Structure PROFILE (icm.h)

Article29/02/2024

Contient des informations qui définissent un profil de couleur. Pour plus d'informations, consultez Utilisation de profils d'appareil avec WCS .

Syntaxe

```
C++
```

```
typedef struct tagPROFILE {
  DWORD dwType;
  PVOID pProfileData;
  DWORD cbDataSize;
} PROFILE;
```

Membres

`dwType`

Doit être défini sur l'une des valeurs suivantes.

Agrandir le tableau

Valeur	Signification
PROFILE_FILENAME	Indique que le membre **pProfileData** contient une chaîne terminée par null qui contient le nom d'un fichier de profil d'appareil.
PROFILE_MEMBUFFER	Indique que le membre **pProfileData** contient un pointeur vers un profil d'appareil dans une mémoire tampon.

`pProfileData`

Le contenu de ce membre est indiqué par le membre **dwTYPE** . Il s'agit soit d'un pointeur vers une chaîne terminée par un caractère Null contenant le nom de fichier du profil d'appareil, soit d'un pointeur vers une mémoire tampon contenant les données de profil d'appareil.

`cbDataSize`

Taille en octets de la mémoire tampon de données pointée par le membre **pProfileData** .

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Voir aussi

- Utilisation de profils d'appareil avec WCS

Commentaires

STRUCTURE PROFILEHEADER (icm.h)

Article24/08/2023

Contient des informations qui décrivent le contenu d'un fichier de profil d'appareil. Cet en-tête se produit au début d'un fichier de profil d'appareil.

Syntaxe

```
C++

typedef struct tagPROFILEHEADER {
  DWORD  phSize;
  DWORD  phCMMType;
  DWORD  phVersion;
  DWORD  phClass;
  DWORD  phDataColorSpace;
  DWORD  phConnectionSpace;
  DWORD  phDateTime[3];
  DWORD  phSignature;
  DWORD  phPlatform;
  DWORD  phProfileFlags;
  DWORD  phManufacturer;
  DWORD  phModel;
  DWORD  phAttributes[2];
  DWORD  phRenderingIntent;
  CIEXYZ phIlluminant;
  DWORD  phCreator;
  BYTE   phReserved[44];
} PROFILEHEADER;
```

Membres

`phSize`

Taille du profil en octets.

`phCMMType`

Numéro d'identification de la CMM utilisée dans le profil. Les numéros d'identification sont enregistrés auprès de la CPI.

`phVersion`

Numéro de version du profil. Le numéro de version est déterminé par l'ICC. Le numéro de version principale actuel est 02h. Le numéro de version secondaire actuel est 10h. Les

numéros de version principale et secondaire sont en nombre décimal codé binaire (BCD). Ils doivent être stockés au format suivant.

Nombre d'octets	Contenu
0	Numéro de version principale dans BCD.
1	Numéro de version mineur dans le grbble le plus significatif de cet octet. Numéro de version du correctif de bogue dans le grbble le moins significatif.
2	Réservé. Doit avoir la valeur 0.
3	Réservé. Doit avoir la valeur 0.

`phClass`

Indique la classe de profil. Pour obtenir une description des classes de profil, consultez Utilisation de profils d'appareil avec WCS. Une classe de profil peut avoir l'une des valeurs suivantes.

Profile, classe	Signature
Profil de périphérique d'entrée	CLASS_SCANNER
Afficher le profil d'appareil	CLASS_MONITOR
Profil d'appareil de sortie	CLASS_PRINTER
Profil Device Link	CLASS_LINK
Profil de conversion d'espace de couleur	CLASS_COLORSPACE
Profil abstrait	CLASS_ABSTRACT
Profil de couleur nommé	CLASS_NAMED
Profil de modèle d'apparence de couleur	CLASS_CAMP
Profil de modèle de carte de gamme de couleurs	CLASS_GMMP

`phDataColorSpace`

Valeur de signature qui indique l'espace de couleurs dans lequel les données de profil sont définies. Le membre peut être n'importe quelle valeur des constantes d'espace de couleur.

`phConnectionSpace`

Valeur de signature qui indique l'espace de couleurs dans lequel l'espace de connexion de profil (PCS) est défini. Le membre peut être l'une des valeurs suivantes.

Profile, classe	**Signature**
XYZ	SPACE_XYZ
Laboratoire	SPACE_Lab

Lorsque le membre **phClass** est défini sur CLASS_LINK, le PCS est extrait du membre **phDataColorSpace** .

`phDateTime[3]`

Date et heure de création du profil.

`phSignature`

Réservé à un usage interne.

`phPlatform`

Plateforme principale pour laquelle le profil a été créé. La plateforme principale peut être définie sur l'une des valeurs suivantes.

Plateforme	**Valeur**
Apple Computer, Inc.	'APPL'
Microsoft Corp.	'MSFT'
Silicon Graphics, Inc.	'SGI'
SUN MICROSYSTEMS, INC.	'SUNW'
Taligent	'TGNT'

`phProfileFlags`

Indicateurs de bits contenant des indicateurs que la CMM utilise pour interpréter les données de profil. Le membre peut être défini sur les valeurs suivantes.

Constant	**Signification**
FLAG_EMBEDDEDPROFILE	Le profil est incorporé dans un fichier bitmap.

Constant	Signification
FLAG_DEPENDENTONDATA	Le profil ne peut pas être utilisé indépendamment des données de couleur incorporées. Utilisé pour les profils incorporés dans des fichiers bitmap.

`phManufacturer`

Numéro d'identification du fabricant du profil d'appareil. Tous les numéros d'identification du fabricant sont enregistrés auprès de la CCI.

`phModel`

Numéro de modèle d'appareil du fabricant de l'appareil. Tous les numéros d'identification du modèle sont enregistrés auprès de l'ICC.

`phAttributes[2]`

Attributs du profil. Les attributs de profil peuvent être l'une des valeurs suivantes.

Constant	Signification
ATTRIB_TRANSPARENCY	Active la transparence. Si cet indicateur n'est pas utilisé, l'attribut est réfléchissant par défaut.
ATTRIB_MATTE	Active l'affichage mat. Si cet indicateur n'est pas utilisé, l'attribut est brillant par défaut.

`phRenderingIntent`

Intention de rendu du profil. Le membre peut être défini sur l'une des valeurs suivantes :

INTENT_PERCEPTUAL

INTENT_SATURATION

INTENT_RELATIVE_COLORIMETRIC

INTENT_ABSOLUTE_COLORIMETRIC

Pour plus d'informations, consultez Intentions de rendu.

`phIlluminant`

Profil illuminant.

`phCreator`

Signature du logiciel qui a créé le profil. Les signatures sont enregistrées auprès de la CPI.

`phReserved[44]`

Réservé.

Spécifications

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Voir aussi

- Informations supplémentaires
- Utilisation de profils d'appareil avec WCS
- Intentions de rendu

Commentaires

Structure RGBCOLOR (icm.h)

Article24/08/2023

TBD

Syntaxe

```
C++
```

```
struct RGBCOLOR {
  WORD red;
  WORD green;
  WORD blue;
};
```

Membres

`red`

TBD

`green`

TBD

`blue`

TBD

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l’aide sur Microsoft Q&A

Structure XYZCOLOR (icm.h)

Article01/03/2024

TBD

Syntaxe

```
C++

struct XYZCOLOR {
  WORD X;
  WORD Y;
  WORD Z;
};
```

Membres

`X`

TBD

`Y`

TBD

`Z`

TBD

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Commentaires

Cette page a-t-elle été utile ? Yes No

Indiquer des commentaires sur le produit | Obtenir de l'aide sur Microsoft Q&A

STRUCTURE XYZColorF (wcsplugin.h)

Article08/03/2023

TBD

Syntaxe

```
C++
```

```
typedef struct _XYZColorF {
  FLOAT X;
  FLOAT Y;
  FLOAT Z;
} XYZColorF;
```

Membres

`X`

TBD

`Y`

TBD

`Z`

TBD

Configuration requise

En-tête	wcsplugin.h

Structure YxyCOLOR (icm.h)

Article29/02/2024

TBD

Syntaxe

```
C++

struct YxyCOLOR {
  WORD Y;
  WORD x;
  WORD y;
};
```

Membres

`Y`

TBD

`x`

TBD

`y`

TBD

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
En-tête	icm.h

Commentaires

Cette page a-t-elle été utile ? Yes No

Indiquer des commentaires sur le produit | Obtenir de l'aide sur Microsoft Q&A

Macros pour les valeurs et couleurs CMJN

Article • 13/06/2023

Les macros suivantes sont utiles pour manipuler des valeurs CMJN.

Macro	Description
CMJN	Génère une valeur CMJN à partir de valeurs cyan, magenta, jaune et noire individuelles.
GetCValue	Récupère la valeur de couleur cyan à partir d'une valeur de couleur CMJN.
GetKValue	Récupère la valeur de couleur noire à partir d'une valeur de couleur CMJN.
GetMValue	Récupère la valeur magenta à partir d'une valeur de couleur CMJN.
GetYValue	Récupère la valeur de couleur jaune à partir d'une valeur de couleur CMJN.

Les macros suivantes sont utilisées avec la couleur.

Macro	Description
GetBValue	Obtient une valeur bleue RVB.
GetGValue	Obtient une valeur verte RVB.
GetRValue	Obtient une valeur rouge RVB.
PALETTEINDEX	Accepte un index dans une entrée de palette de couleurs logiques et retourne un spécificateur d'entrée de palette.
PALETTERGB	Accepte trois valeurs qui représentent les intensités relatives du rouge, du vert et du bleu, et retourne un spécificateur rouge, vert et bleu (RVB) relatif à la palette.
RGB	Sélectionne une couleur rouge, verte et bleue (RVB) en fonction des arguments fournis et des fonctionnalités de couleur de l'appareil de sortie.

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l’aide sur Microsoft Q&A

Macro CMJN (wingdi.h)

Article04/03/2024

La macro **CMJN** crée une valeur de couleur CMJN en combinant les valeurs cyan, magenta, jaune et noire spécifiées.

Syntaxe

```
C++
```

```
void CMYK(
   c,
   m,
   y,
   k
);
```

Paramètres

`c`

Valeur cyan de la couleur à créer.

`m`

Valeur magenta de la couleur à créer.

`y`

Valeur jaune de la couleur à créer.

`k`

Valeur noire de la couleur à créer.

Valeur de retour

None

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
Plateforme cible	Windows
En-tête	wingdi.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- GetCValue
- GetKValue
- GetMValue
- GetYValue
- Macros pour les valeurs et les couleurs CMJN

Commentaires

Macro GetCValue (wingdi.h)

Article27/08/2023

La macro **GetCValue** récupère la valeur de couleur cyan à partir d'une valeur de couleur CMJN.

Syntaxe

```
C++
```

```
void GetCValue(
   cmyk
);
```

Paramètres

`cmyk`

Valeur de couleur CMJN à partir de laquelle la valeur de couleur cyan sera récupérée.

Valeur de retour

None

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
Plateforme cible	Windows
En-tête	wingdi.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- CMJN
- GetKValue
- GetMValue
- GetYValue
- Macros pour les valeurs et les couleurs CMJN

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

Macro GetKValue (wingdi.h)

Article27/08/2023

La macro **GetKValue** récupère la valeur de couleur noire à partir d'une valeur de couleur CMJN.

Syntaxe

```
C++

void GetKValue(
   cmyk
);
```

Paramètres

`cmyk`

Valeur de couleur CMJN à partir de laquelle la valeur de couleur noire sera récupérée.

Valeur de retour

None

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
Plateforme cible	Windows
En-tête	wingdi.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- CMJN
- GetCValue
- GetMValue
- GetYValue
- Macros pour les valeurs et les couleurs CMJN

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

Macro GetMValue (wingdi.h)

Article27/08/2023

La macro **GetMValue** récupère la valeur de couleur magenta à partir d'une valeur de couleur CMJN.

Syntaxe

```
C++

void GetMValue(
   cmyk
);
```

Paramètres

`cmyk`

Valeur de couleur CMJN à partir de laquelle la valeur de couleur magenta sera récupérée.

Valeur de retour

None

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
Plateforme cible	Windows
En-tête	wingdi.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- CMJN
- GetCValue
- GetKValue
- GetYValue
- Macros pour les valeurs et les couleurs CMJN

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

Macro GetYValue (wingdi.h)

Article27/08/2023

La macro **GetYValue** récupère la valeur de couleur jaune à partir d'une valeur de couleur CMJN.

Syntaxe

```
C++

void GetYValue(
   cmyk
);
```

Paramètres

`cmyk`

Valeur de couleur CMJN à partir de laquelle la valeur de couleur jaune sera récupérée.

Valeur de retour

None

Configuration requise

Client minimal pris en charge	Windows 2000 Professionnel [applications de bureau uniquement]
Serveur minimal pris en charge	Windows 2000 Server [applications de bureau uniquement]
Plateforme cible	Windows
En-tête	wingdi.h

Voir aussi

- Concepts de base de la gestion des couleurs
- Fonctions
- CMJN
- GetCValue
- GetKValue
- Macros pour les valeurs et les couleurs CMJN

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

Constantes WCS

Article • 13/06/2023

Cette section contient des informations sur les constantes et les valeurs d'indicateur utilisées par WCS.

- Constantes des espaces de couleurs
- Indicateurs de création de transformation CMM
- Identificateurs de types d'espace de couleurs
- Messages de couleur courants
- Codes d'erreur spécifiques du WCS

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

Constantes des espaces de couleurs

Article • 13/06/2023

Valeur de signature qui indique l'espace de couleur dans lequel les données de profil sont définies. Il peut s'agir de l'une des valeurs suivantes.

Classe de profil	Signature
XYZ	SPACE_XYZ
Laboratoire	SPACE_Lab
Luv	SPACE_Luv
Ycbcr	SPACE_YCbCr
Yxy	SPACE_Yxy
RGB	SPACE_RGB
Échelle de gris	SPACE_GRAY
HSV	SPACE_HSV
HLS	SPACE_HLS
CMJN	SPACE_CMYK
CMY	SPACE_CMY
Canal 2 générique	SPACE_2_CHANNEL
3 canaux génériques	SPACE_3_CHANNEL
4 canaux génériques	SPACE_4_CHANNEL
Canal générique à 5	SPACE_5_CHANNEL
6 canaux génériques	SPACE_6_CHANNEL
Canal 7 génériques	SPACE_7_CHANNEL
Canal 8 génériques	SPACE_8_CHANNEL

Rubriques connexes

Concepts de base de la gestion des couleurs

Constantes ICM

Commentaires

Indicateurs de création de transformation CMM

Article • 13/06/2023

Les machines virtuelles de gestion utilisent des indicateurs de création de transformation comme indicateurs de création d'une transformation de couleur. C'est à la CMM de déterminer la meilleure façon d'utiliser ces indicateurs.

Toutes les fonctions qui utilisent ces indicateurs passent ou reçoivent des valeurs d'indicateur via un paramètre appelé *dwFlags*. Le **mot***d'ordre élevé de dwFlags* doit être défini sur une valeur du tableau suivant.

Constante	Description
ENABLE_GAMUT_CHECKING	Utilisez cette transformation pour la vérification de la gamme.
USE_RELATIVE_COLORIMETRIC	Ne conservez pas le point blanc. Si la gamme de sortie ne prend pas en charge une couleur donnée, utilisez la couleur prise en charge la plus proche. Consultez Rendu des intentions.
FAST_TRANSLATE	Rechercher la couleur uniquement. Ne pas interpoler la couleur.
PRESERVEBLACK	Insère le GMMP de génération noire approprié en tant que dernier GMMP dans la séquence de transformation
WCS_ALWAYS	Utilisez le chemin de code WCS même pour les transformations ICC.
SEQUENTIAL_TRANSFORM	Indicateur de création de transformation pour la création d'une transformation de couleur séquentielle (non optimisée).

Le **MOT** de bas ordre peut avoir l'une des valeurs constantes suivantes.

Constante	Description
PROOF_MODE	La transformation sera utilisée pour afficher un aperçu de l'image. Qualité d'image faible.
NORMAL_MODE	La transformation sera utilisée pour l'affichage normal de l'image. Qualité d'image moyenne.
BEST_MODE	La transformation sera utilisée pour l'affichage de l'image de la plus haute qualité possible sur l'appareil cible.

En passant de PROOF_MODE à BEST_MODE, la qualité de la sortie s'améliore généralement et la vitesse de transformation diminue.

Rubriques connexes

Concepts de base de la gestion des couleurs

Constantes ICM

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

Identificateurs de types d’espace de couleurs

Article • 13/06/2023

Ces constantes spécifient le type d’un tableau d’espaces de couleurs Postscript 2. Les valeurs suivantes sont des types de tableau d’espaces de couleurs valides.

CSA_A

Obtenez un tableau d’espaces de couleurs CIEBasedA à partir du profil monochrome.

CSA_GRAY

Obtenez un tableau d’espaces de couleurs CIEBasedA à partir du profil monochrome.

CSA_ABC

Obtenez un tableau d’espaces de couleurs CIEBasedABC à partir du profil RVB ou L*a*b.

CSA_DEF

Obtenez un tableau d’espaces de couleurs CIEBasedDEF à partir du profil RVB ou L*a*b.

CSA_RGB

Obtenez un tableau d’espaces de couleurs CIEBasedDEF suivi d’un tableau d’espaces de couleurs CIEBasedABC à partir du profil RVB. Lors de l’exécution, si l’imprimante ne prend pas en charge les espaces de couleurs CIEBasedDEF, la fonction utilise la définition CIEBasedABC.

CSA_Lab

Obtient un tableau d’espace de couleurs CIEBasedABC à partir du profil L*à*b.

CSA_DEFG

Obtenez un tableau d’espaces de couleurs CIEBasedDEFG à partir du profil CMJN.

CSA_CMYK

Obtenez un tableau d’espaces de couleurs CIEBasedCMYK à partir du profil CMJN.

Voir aussi

Concepts de base de la gestion des couleurs

Constantes ICM

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l'aide sur Microsoft Q&A

Messages de couleur courants

Article • 13/06/2023

Les messages suivants sont utilisés avec la couleur.

- WM_PALETTECHANGED
- WM_PALETTEISCHANGING
- WM_QUERYNEWPALETTE
- WM_SYSCOLORCHANGE

Rubriques connexes

Concepts de base de la gestion des couleurs

Constantes ICM

Commentaires

Codes d’erreur spécifiques du WCS

Article • 13/06/2023

Voici une liste de codes d’erreur spécifiquement liés à l’utilisation de WCS 1.0. Cela ne signifie pas que les fonctions WCS retournent uniquement ces codes d’erreur. Cela signifie que les codes du tableau ci-dessous sont retournés uniquement par les fonctions WCS. Ils peuvent également retourner d’autres codes d’erreur Win32 qui sont documentés ailleurs dans l’API Win32 du Kit de développement logiciel (SDK) de plateforme.

ERROR_DELETING_ICM_XFORM

La transformation de couleur spécifiée n’a pas pu être supprimée.

ERROR_DUPLICATE_TAG

La balise d’élément de profil de couleur spécifiée est déjà présente dans le profil de couleur.

ERROR_ICM_NOT_ENABLED

La gestion des couleurs des images n’est pas activée pour le moment.

ERROR_INVALID_CMM

Le nom de fichier spécifié ne fait pas référence à un module de gestion des couleurs valide.

ERROR_INVALID_COLORSPACE

L’espace de couleurs spécifié n’est pas valide.

ERROR_INVALID_PROFILE

Le nom de fichier spécifié ne fait pas référence à un profil de couleur valide.

ERROR_INVALID_TRANSFORM

La transformation de couleur spécifiée n’est pas une transformation de couleur valide.

ERROR_PROFILE_NOT_ASSOCIATED_WITH_DEVICE

Le profil de couleur spécifié n’est associé à aucun appareil.

ERROR_PROFILE_NOT_FOUND

Le nom de fichier de profil de couleur spécifié est introuvable. Le nom de fichier ou le chemin d'accès est incorrect.

ERROR_TAG_NOT_FOUND

La balise d'élément de profil de couleur spécifiée est introuvable dans le profil de couleur.

ERROR_TAG_NOT_PRESENT

Une balise d'élément de profil de couleur requise pour que la fonction réussisse n'a pas été passée à la fonction.

Rubriques connexes

Concepts de base de la gestion des couleurs

Constantes WCS

Commentaires

Clés de Registre WCS

Article • 13/06/2023

WCS utilise des clés de Registre pour signaler que certains événements de profil de couleur se sont produits. Les applications doivent interroger ces clés de Registre pour obtenir l'état du profil de couleur système mis à jour.

Profil de couleur actif modifié

Les applications peuvent souhaiter répondre aux événements de changement de profil de couleur pour un appareil de surveillance ; Cela garantit qu'ils disposent toujours d'informations de couleur précises pour leur cible, même si l'utilisateur ou une autre application a modifié le profil actif de l'appareil.

Applications de bureau

Les applications de bureau doivent écouter les modifications apportées au Registre pour déterminer quand des associations de profils de couleur ont été modifiées à l'aide **de RegNotifyChangeKeyValue**. Une application doit s'inscrire à la fois pour les modifications d'association de profil par utilisateur et pour les modifications à l'échelle du système.

RegNotifyChangeKeyValue doit être initialisé avec un HKEY fourni par **RegOpenKeyEx**. **RegOpenKeyEx** doit être initialisé à l'aide des emplacements d'arborescence du Registre suivants :

Associations de profils par utilisateur	**HKEY_CURRENT_USER SOFTWARE\Microsoft\Windows NT\CurrentVersion\ICM\ProfileAssociations\Display\{4d36e96e-e325-11ce-bfc1-08002be10318}**
Associations de profils à l'échelle du système	**HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Class\{4d36e96e-e325-11ce-bfc1-08002be10318}**

Lorsque l'application est informée d'une modification de clé de Registre, elle doit d'abord demander si des associations par utilisateur ou à l'échelle du système sont utilisées en appelant **WcsGetUsePerUserProfiles**. Il doit ensuite appeler **WcsGetDefaultColorProfile** avec la valeur **WCS_PROFILE_MANAGEMENT_SCOPE** droite

pour obtenir le nouveau profil de couleur actif pour le moniteur. Notez que toutes les modifications de clé de Registre ne correspondent pas à une modification réelle du profil de couleur actuellement actif ; l'application case activée si le profil retourné par **WcsGetDefaultColorProfile** a réellement changé.

Applications Windows universelles (UWP)

Les applications Windows universelles n'ont pas accès aux clés de Registre ci-dessus. Au lieu de cela, ils doivent inscrire un gestionnaire pour l'événement **DisplayInformation.ColorProfileChanged** . Cet événement se déclenche chaque fois que le profil de couleur actif du moniteur sur lequel l'application s'exécute a changé. ColorProfileChanged prend en compte si les associations de profils par utilisateur ou à l'échelle du système sont utilisées ; ces informations sont extraites des applications UWP.

Lorsque vous répondez à l'événement ColorProfileChanged, l'application doit interroger le profil actuellement actif à **l'aide de DisplayInformation.GetColorProfileAsync**.

Commentaires

Glossaire WCS 1.0

Article • 25/02/2024

ABCDGHLPRSTW

Commentaires

Un

Article • 13/06/2023

couleurs primaires additives

Couleurs qui peuvent être ajoutées au noir pour mélanger une nouvelle couleur. Les couleurs primaires additives sont le rouge, le vert et le bleu.

Commentaires

B

Article • 13/06/2023

luminosité

La luminosité d'une couleur fait référence à l'intensité de la lumière qui est réfléchie ou transmise par une image.

Commentaires

C

Article • 13/06/2023

CAMP

Le format de profil du modèle d'apparence de couleur est un format de fichier XML qui contient les conditions d'affichage requises pour décrire la relation entre les valeurs CIEXYZ DMP et les valeurs CIEJab.

CMJN

Espace de couleur Cyan, Magenta, Jaune et BlacK. Il est souvent implémenté sur les imprimantes.

colorant

Quelque chose, en particulier un pigment ou une encre, qui colore ou modifie la teinte de quelque chose d'autre.

canal de couleurs

Composant d'un espace de couleurs. Pour instance, un espace de couleur RVB comporte des canaux de couleur rouge, vert et bleu.

conversion de couleur

Processus de conversion des couleurs d'un espace de couleurs à un autre.

Module de gestion des couleurs (CMM)

Module de code qui utilise des profils d'appareil pour effectuer la conversion de couleurs et le mappage des couleurs.

mappage des couleurs

Voir correspondance des couleurs.

correspondance de couleur

Correspondance d'une couleur convertie à sa couleur visuellement la plus proche dans l'espace colorimétrique de destination.

modèle de couleur

Voir espace de couleur.

espace de couleur

Mappage de composants de couleur sur un système de coordonnées géométriques en trois dimensions.

Commentaires

D

Article • 13/06/2023

DAC

Convertisseur numérique-analogique.

profil de lien d’appareil

Fichier qui contient une concaténation de conversions de couleurs souvent utilisées.

profil d’appareil

Fichier qui contient des informations sur la façon de caractériser les capabilites de couleur d’un appareil. Les profils d’appareil ICC convertissent les couleurs de la gamme d’un appareil en PCS. Il contient également les informations nécessaires à la conversion des couleurs du PCS en gamut de l’appareil. Les profils d’appareil XML WCS fournissent les données de mesure et les références algorithmiques à convertir entre l’espace d’appareil natif et CIEXYZ.

DMP

Le format de profil de modèle d’appareil de couleur est un format de fichier XML qui contient les mesures requises pour décrire la relation entre les valeurs d’espace de couleur d’appareil natives et les valeurs CIEXYZ.

Commentaires

G

Article • 13/06/2023

correction gamma

Les caractéristiques de tonalité physique native d'un appareil sont souvent différentes sur différents types d'appareils. Le facteur de correction gamma compense ces différences.

courbe gamma

Collection de corrections gamma sur l'ensemble de la plage de couleurs. Lorsqu'elles sont tracées sur un graphique, elles forment une courbe.

rampe gamma

Voir courbe gamma.

Gamme

Ensemble de couleurs à partir d'un espace de couleurs qu'un appareil peut produire.

GMMP

Le format de profil du modèle de mappage de couleurs est un format de fichier XML qui contrôle le mappage des couleurs entre les gamuts d'appareils source et de destination.

Commentaires

H

Article • 13/06/2023

preuve matérielle

Aperçu ou vérification d'une image en imprimant une copie papier de celle-ci.

HLS

Espace de couleurs souvent utilisé par les artistes. Ses composants sont Hue, Lightness et Saturation (chroma).

HSV

Espace de couleurs souvent utilisé par les artistes. Ses composants sont Hue, Saturation (chroma) et Value (légèreté).

Teinte

Une couleur pure. Une teinte est une position spécifique sur la partie visible du spectre électromagnétique.

Commentaires

L

Article • 13/06/2023

Légèreté

Un canal de couleurs dans l’espace de couleurs HLS. Il s’agit d’une mesure de la luminosité relative de la couleur. L’augmentation de la légèreté produit des teintes d’une teinte. La diminution de la luminosité produit des nuances de teinte.

Commentaires

P

Article • 13/06/2023

couleurs primaires

Les trois couleurs essentielles dans un espace de couleurs qui peuvent être mélangées pour produire toutes les autres couleurs.

Espace de connexion de profil (PCS)

Espace de couleurs indépendant de l'appareil utilisé pour les conversions de couleurs et la correspondance des couleurs.

Commentaires

R

Article • 13/06/2023

intention de rendu

Approches du rendu de la couleur d'un espace de couleurs à un autre. Les intentions de rendu utilisées dans les profils ICC sont définies par l'International Color Consortium (ICC).

RGB

Espace de couleur Rouge, Vert et Bleu.

RVB Triple

Quantité qui contient des valeurs d'espace de couleur RVB.

Commentaires

S

Article • 13/06/2023

saturation (chroma)

En général, la couleur est la quantité de gris qui est mélangée dans une teinte. La chroma zéro produit une nuance de gris. La full chroma produit une teinte pure.

Ombre

Une nuance de couleur est faite en ajoutant du noir à sa teinte.

preuve réversible

Aperçu ou vérification d'une image sur un écran vidéo.

Srgb

Un espace de couleur RVB standard international proposé par Microsoft et Hewlett-Packard indépendant de l'appareil. Cette norme a été approuvée en tant que norme internationale officielle par les systèmes et équipements multimédiasParamètres et gestion de la couleurPartie 2-1 : Gestion des couleursDéfiniment RVB Espace colorimétrique. Il est disponible directement à partir de la IEC à l'adresse https://www.iec.ch/ .

couleurs primaires soustracives

Couleurs qui peuvent être soustraites du blanc pour mélanger une nouvelle couleur. Les couleurs primaires soustractives sont cyan, jaune et magenta.

Commentaires

Cette page a-t-elle été utile ? Yes No

Obtenir de l’aide sur Microsoft Q&A

T

Article • 13/06/2023

Teinte

Une teinte d’une couleur est créée en mélangeant une teinte avec du blanc.

Ton

Un ton d’une couleur est obtenu en mélangeant sa teinte avec le gris.

Commentaires

W

Article • 13/06/2023

point blanc

Valeur de couleur utilisée comme référence à laquelle l'utilisateur s'adapte. Des valeurs plus brillantes sont possibles dans les gammes de plages dynamiques élevées.

Commentaires