RFC: 2083
Statut : Proposition de norme

PORTABLE NETWORKS GRAPHIC 1.0

SPECIFICATION

Crédits : Thomas Boutell / Boutell Com Inc
Traduction : V.G. FREMAUX/ Ingénieur professeur / EISTI

Note du Traducteur :

Le texte suivant est une traduction partielle de la spécification PNG. Les parties retenues ont été transcrites telles qu'éditées par les auteurs originaux du protocole, sans ajouts, commentaires, ni omissions. Ce document n'est qu'une proposition et n'a pas valeur de "standard".

Afin de permettre au lecteur de compléter son information, nous précisons ci-dessous les parties que nous avons omises, et qui seront transcrites dans le futur. De plus, nous donnons la table des matières complète du document original.

Sont omises :

- *Les recommandations concernant la mise en œuvre des codeurs*
  (Cela regroupe des discussions détaillées sur la manière de traiter tel ou tel aspect particulier. Ce texte est en partie traduit, mais d'autres urgences nous ont arrêté en cours.)
- *Les recommandations concernant la mise en œuvre des décodeurs*
  (Cette section sera traduite ultérieurement, quand nous reprendrons la précédente.)
- *Les "motivations"*
  (Elles donnent un certain nombre d'explications sur les choix techniques retenus. Ce texte n'est pas d'une importance capitale pour utiliser et exploiter le format PNG. Nous traduirons néanmoins cette section dans la mesure où elle apporte une réflexion intéressante sur les formats graphiques en général.)

Statut de ce mémoire

Ce mémoire est à destination de la communauté Internet. Il ne constitue en aucun cas un standard officiel. La distribution de ce mémo est illimitée.

Note de l'IESG :

L'IESG ne prend pas position sur les indications de Propriété Intellectuelle mentionnées dans ce document.

Contexte

Ce document décrit le format PNG (Portable Network Graphics), un format de fichier extensible pour le stockage non destructif, portable, et convenablement compressé d'images. Le format PNG constitue une alternative libre de droits du GIF et peut également remplacer le format TIFF dans certaines de ses applications. Sont supportées les images en couleurs indexées, niveaux de gris, et vraies couleurs, avec éventuellement un canal alpha supplémentaire. La profondeur des pixels peut varier de 1 à 16 bits .

Le format PNG est particulièrement adapté aux applications d'édition graphique "en ligne", telles que le World Wide Web, et de ce fait est tramé avec option d'affichage en résolution progressive. Le format PNG est robuste, contenant les mécanismes de vérification d'intégrité du fichier dans son ensemble, ainsi que de détection des erreurs simples de transmission. De plus, le format PNG peut enregistr 153 er le gamma et les données de calibrage en chromie afin d'obtenir un rendu précis et conforme en milieu hétérogène. La présente définit le type Mime Internet image/png.

TABLE DES MATIERES

1. Introduction

Le format PNG présente un standard de codage d'images en "champs de bits", portable, sans pertes, bien compressé, et exhaustivement spécifié. Bien que la motivation initiale ayant conduit au développement du format PNG fut de remplacer le format GIF, ce nouveau format donnera l'occasion d'introduire de nouvelles fonctions, absentes du codage GIF, et pour un coût minime en termes de développement.

Les caractéristiques GIF conservées dans le format PNG incluent:

- Images en couleurs indexées jusqu'à 256 couleurs.
- Fluxabilité: les fichiers peuvent être lus et écrits séquentiellement, permettant ainsi à ce format de fichiers une utilisation en tant que protocole de communication pour la génération et l'affichage d'images "au vol".
- Affichage progressif: un fichier image correctement encodé peut être affiché au fur et à mesure qu'il est reçu à travers une liaison de communication, affichant très rapidement une version en très basse résolution suivie d'une amélioration progressive des détails.
- Transparence: des portions de l'image peuvent être spécifiées comme transparentes, créant un effet d'image non rectangulaire.
- Informations annexes: des commentaires textuels et d'autres données auxiliaires peuvent être enregistrées dans le fichier image.
- Indépendance totale vis à vis du matériel et des environnements.
- Compression efficace, garantie 100% sans pertes.

Les caractéristiques nouvelles importantes du format PNG, absentes du format GIF, incluent:

- Images en vraie couleur jusqu'à 48 bits par pixel.
- Images en niveaux de gris jusqu'à 16 bits par pixel.
- Canal alpha complet (masque général de transparence).
- Informations gamma de l'image, qui permet la correction automatique d'affichage d'images avec une luminosité et un contraste correct quelle que soient les machines ou ont été créée l'image, et celles où elles sont diffusées.
- Détection fiable et rapide d'erreurs dans le fichier.
- Premier affichage plus rapide dans le mode progressif.

PNG se veut:

- **Simple et portable :** les développeurs devront pouvoir implémenter PNG facilement.
- **Libre de droits :** selon le voeu et à la connaissance des auteurs du format PNG, aucun algorithme du domaine privé n'est utilisé. (Des efforts considérables ont été fait pour vérifier cette affirmation).
- **Efficacement compressé :** tant les images en couleurs indexées qu'en vraies couleurs sont compressées avec la même efficacité que dans la plupart des autres formats d'images "sans pertes" couramment utilisés, et dans bien des cas avec un gain encore supérieur.
- **Interchangeable :** Tout décodeur conforme au PNG doit pouvoir lire tous les fichiers codées selon PNG.
- **Souple :** Ce format est ouvert pour des extensions futures, voire des implémentations privées, sans compromettre le principe d'interchangeabilité ci-avant.
- **Robuste :** son design permet d'effectuer un test d'intégrité complet du fichier, ainsi que la détection en temps réel d'erreurs simples de transmission.

La majeure partie de ce document donne la spécification complète du format PNG, et défini les comportements attendus de tout encodeur ou décodeur PNG. Un appendice apporte les éléments complémentaires donnant la motivation de nombreux choix techniques. Bien que ces motivations ne

fassent pas partie intégrante de la specification formelle, leur le 435 cture peut aider les développeurs pour la compréhension du standard. Des références croisées, dans le texte principal, renvoient aux différentes parties des motivations. Des appendices supplémentaires, toujours en dehors de la spécification formelle, apportent des éléments additionnels d'information sur la théorie chromatique et le gamma.

Dans cette spécification, le mot "doit" indique une implémentation obligatoire pour la conformité au standard, tandis que "pourrait" indique un comportement attendu, mais seulement conseillé. *Cf. Motivations(*) : Pourquoi un nouveau format? (Section 12.1), Pourquoi ces caractéristiques? (Section 12.2), Pourquoi pas celles-là? (Section 12.3), Pourquoi ne pas utiliser le format X? (Section 12.4). (*) Les articles de la section "Motivations" seront traduits ultérieurement.*

Prononciation

PNG se prononce "ping".

2. Représentation des données

Ce chapitre expose la représentation des données de base utilisée dans les fichiers PNG, ainsi que la représentation attendue pour les données de l'image.

2.1. Ordre des octets et entiers

Tout entier nécessitant un codage sur plus d'un octet doit présenter ces octets dans l'ordre du réseau: l'octet le plus signifiant est envoyé en premier, puis les octets suivants dans l'ordre décroissant de poids (MSB LSB pour des entiers sur deux octets, B3 B2 B1 B0 pour des entiers sur 4 octets). Le bit de poids fort (valeur 128) est toujours appelé le bit 7; celui de plus faible poids (valeur 1) est numéroté 0. Les valeurs sont non signées, sauf mention explicite contraire. Les valeurs déclarées explicitement comme signées sont inscrites en complément à 2. *Cf. Motivations(*): Ordre des octets (Section 12.5).*

(*) seront traduites ultérieurement

2.2. Valeurs de couleur

Les couleurs peuvent être représentées soit par une échelle de gris, soit par un triolet d'entiers codant les trois couleurs primaires RVB (rouge, vert, bleu). L'échelle de gris représente la luminance; les données RGB représentent une information de couleur calibrée (si le segment cHRM est présent) ou non calibrée et donc dépendant de l'environnement (si cHRM est absent). Toutes les valeurs de couleur sont comprises entre 0 (représentant le noir, ou l'absence de lumière) et la valeur d'intensité maximale pour la plus grande valeur codable. Notez que la plus grande valeur possible pour une "profondeur" d'échantillonnage donnée en bits est (2^profondeur)-1, et non pas 2^profondeur.

La plage de valeur ainsi définie n'est pas nécessairement linéaire; le segment gAMA spécifie les caractéristiques gamma de l'environnement source, et les clients de visualisation pourront utiliser ce segment pour effectuer les compensations nécessaires. *Cf. correction Gamma (Section 2.7).*

Des données source d'une précision non supportée par PNG (par exemple, couleur vraie sur 5 bit/pixel) doivent être converties au codage immédiatement supérieur supporté par PNG. Cette mise à l'échelle est réversible et doit se faire sans perte d'information, et permet de réduire le nombre de cas que les décodeurs doivent implémenter. *Cf. Recommandations pour les encodeurs: Conversion de profondeur de couleur (Section 9.1) et Recommandations pour les Décodeurs (Section 10.4).*

2.3. Codage de l'image

Du point de vue du concept, une image PNG est un tableau rectangulaire de pixels, ceux-ci apparaissant de gauche à droite à l'intérieur de chaque ligne de scan, et les lignes apparaissant de haut en bas. (Pour réaliser la fonction d'affichage progressif, les données sont en fait transmises dans un autre ordre, *Cf. Entrelacement des données (Section 2.6.)* La taille de chaque pixel est donnée par la profondeur d'échantillonnage en bits, correspondant au nombre de bits nécessaires pour coder un pixel de l'image source. Trois types de pixels sont supportés:

- Un pixel indexé en couleur est représenté par un seul échantillon contenant un index numérique sur une palette de couleurs prédéfinies. La profondeur de codage est dans ce cas le nombre d'entrées différentes dans la palette, et non la précision de couleu 1000 r dans la palette elle-même.
- Un pixel en niveaux de gris est représenté par un échantillon unique donnant numériquement une luminance du pixel, laquelle valant zéro au noir et la plus grande valeur admise par le codage au blanc.
- Un pixel en "vraie couleur" est représenté par trois échantillons: rouge (zéro = noir, max = rouge) en premier, puis vert (zéro = noir, max = vert), puis enfin bleu (zéro = noir, max = bleu). La profondeur représente dans ce cas le codage d'une couleur de base, et non la taille totale occupée par le pixel.

Optionnellement, les modèles en niveaux de gris et en vraies couleurs peuvent disposer d'un échantillon supplémentaire pour le canal alpha, décrit dans la section suivante.

Les pixels sont toujours organisés en lignes, successivement, et sans bits perdus entre chaque pixel. Les pixels codés sur un nombre de bits inférieur à un octet ne peuvent être rangés à cheval sur deux octets; Ils sont placés dans l'octet à partir du bit de poids fort. Les profondeurs autorisées et les modèles de pixel sont limités de telle sorte que l'organisation des octets reste simple et efficace.

PNG autorise des modèles de pixels multi-échantillons de 8- et 16-bits. De ce fait, ces pixels ne peuvent tenir dans un seul octet. Des échantillons 16-bits sont stockés selon l'ordre de transmission réseau (MSB en premier).

Les lignes commencent toujours sur le début d'un octet. Lorsque le codage du pixel est inférieur à 8 bits et la largeur de ligne n'est pas divisible par le nombre de pixels par octet, les bits de poids faible du dernier octet de la ligne sont perdus. La valeur de ces bits perdus n'est pas précisée.

Un octet spécial de "type de filtre" est ajouté en début de chaque ligne (voir Filtrage Section 2.5). L'octet de type de filtre n'est pas considéré comme faisant part de données image, mais il est inclus dans le flux de données envoyé à l'étage de compression.

2.4. Canal Alpha

Un canal alpha, représentant la transparence sur la base du pixel, peut être incluse dans des images PNG en niveaux de gris ou en vraies couleurs.

Une valeur alpha de zéro attribuée à un pixel représente la transparence totale, et une valeur de (2^profondeur)-1 indique que le pixel est complètement opaque. Les valeurs intermédiaires indiquent une transparence partielle du pixel, combinant la couleur de celui-ci avec celle du fond de façon à former une image composite. (Ainsi, la couche alpha est réellement le degré d'opacité du pixel. Cependant, nombreux sont ceux pour lesquels la couche alpha transporte une information de transparence, plutôt que d'opacité. Nous continuerons dans ce sens).

Un canal Alpha peut être intégré dans un image de profondeur 8 ou 16 bits par échantillon, mais pas dans celles de profondeur inférieure à 8 bits. Les échantillons Alpha sont représentés avec la même profondeur que les autres canaux (ceux des couleurs de base). L'échantillon alpha de chaque pixel est stocké

immédiatement après la valeur de luminance pour le modèle en "niveau de gris" ou après l'échantillon bleu pour le modèle RVB.

Les valeurs d'échantillons de couleur du pixel ne sont pas affectées par la valeur de l'échantillon alpha du même pixel. Il s'agit d'un modèle appelé parfois "non associé" ou alpha "non-premultiplié". (Une autre technique courante est d'enregistrer les échantillons de couleur une fois multipliés par la valeur du canal alpha; en effet, cela revient à avoir composé à l'avance l'image finale sur un fond noir. PNG n'utilise pas la technique de pré-multiplication alpha).

Le contrôle de transparence est possible sans payer le coût (en termes de poids d'image) d'un canal alpha complet. Dans une image en couleurs indexées, une valeur d'alpha peut être définie pour chaque entrée de palette. Dans des images en niveaux de gris ou en vraies couleurs, il sera possible d'affecter la transparence à une valeur particulière de couleur des pixels. Ces techniques sont contrôlées par le segment auxiliaire tRNS.

Si aucun canal alpha ni s 1000 egment tRNS ne sont présents, tous les pixels de l'image seront considérés comme opaques.

Les navigateurs pourront supporter le traitement de la transparence, entièrement ou en partie seulement. *Cf. Motivations: Alpha Non-prémultiplié (Section 12.8), Recommandations pour les Encodeurs: Création de canal Alpha (Section 9.4), et Recommandations pour les Décodeurs: Traitement du canal Alpha (Section 10.8).*

2.5. Filtrage

Le format PNG permet le filtrage des données de l'image avant compression. Ce filtrage peut dans certains cas améliorer le gain de compression à suivre. L'étage de filtrage lui-même ne réduit pas la taille des données. Tous les filtres PNG sont strictement non destructifs.

Le format PNG définit divers algorithmes de filtrage, dont "None" pour lequel aucune modification n'est apportée aux données. L'algorithme de filtrage utilisé est spécifié pour chaque ligne par un octet dit de "type de filtre" précédant le flux de données d'image pour chaque ligne devant être envoyée à l'étage de compression. Un encodeur intelligent pourrait commuter le type de filtre d'une ligne d'image à l'autre. La méthode pour choisir quel est le filtrage à appliquer est du ressort de l'encodeur. *Cf. Algorithmes de Filtrage (Chapitre 6) et Motivations: Filtrage (Section 12.9).*

2.6. Entrelacement des données

Une image PNG peut être enregistrée selon un ordre entrelacé afin d'en permettre un affichage par définition progressive. L'idée de ce mécanisme est de pouvoir afficher une ébauche de l'image le plus tôt possible lorsqu'elle est consultée via un réseau, sa définition augmentant au fur et à mesure que l'image est chargée. L'entrelacement augmente légèrement la taille totale de l'image, mais donne une impression de plus grande rapidité de chargement à l'utilisateur final. Notez que les décodeurs actuels sont sensés lire toute image entrelacée, même s'ils ne peuvent gérer l'affichage progressif pour l'utilisateur final.

Selon la méthode d'entrelacement notée 0, les pixels sont enregistrés séquenciellement de gauche à droite, et les lignes enregistrées séquenciellement de haut en bas (non entrelacé).

Selon la méthode d'entrelacement notée 1, appelée aussi Adam7 d'après le nom de son auteur, (Adam M. Costello), les pixels sont organisés de façon à permettre 7 passes d'affichage de l'image. Chaque passe transmet un sous-ensemble de pixels de l'image. Pour savoir quels pixels sont à envoyer dans la passe courante, il suffit de répéter le schéma suivant sur l'image, en partant du coin gauche supérieur. Seront à transmettre les bits correspondants au numéro de la passe sur le masque suivant:

```
          1 6 4 6 2 6 4 6
          7 7 7 7 7 7 7 7
          5 6 5 6 5 6 5 6
          7 7 7 7 7 7 7 7
          3 6 4 6 3 6 4 6
          7 7 7 7 7 7 7 7
          5 6 5 6 5 6 5 6
          7 7 7 7 7 7 7 7
```

La transmission de chaque passe suit le diagramme d'entrelacement ci-dessus, toujours de gauche à droite et de haut en bas. Par exemple, la passe 2 envoie les pixels 4, 12, 20, etc. des lignes 0, 8, 16, etc. (0,0 correspond au coin gauche supérieur de l'image). La dernière passe envoie les lignes entières 1, 3, 5, etc.

Les données dans chaque passe sont disposées comme s'il s'agissait d'une image de dimensions appropriées. Par exemple, si l'image complète fait 16 par 16 pixels, la passe 3 contiendrait deux lignes, chacune transmettant quatre pixels. Si les pixels sont codés sur moins de 8 bits, chaque ligne est complétée de bits de bourrage jusqu'à faire un nombre entier d'octets (*Cf. Codage de l'image, Section 2.3*). L'image réduite ainsi obtenue est filtrée comme toute image normale, et un octet de type de filtrage est ajouté à chacune de ses lignes (*Cf. Algorithmes de Filtrage, Chapitre 6*). Remarquez que l'ordre de transm d88 ission est défini de telle sorte que toutes les lignes transmises dans la même passe soient composées du même nombre de pixels; Ceci est rendu nécessaire pour l'application correcte de certains filtres.

Attention : Si l'image contient moins de cinq colonnes ou cinq rangées, certaines passes seront complètement ignorées. Les encodeurs et décodeurs doivent attentivement implémenter ce cas. En particulier, les octets de type de filtre ne sont ajoutés que pour des lignes non vides; ils ne sont pas ajoutés dans les lignes d'une passe vide. *Cf. Motivations: Entrelacement (Section 12.6) et Recommandations pour les Décodeurs: affichage progressif (Section 10.9).*

2.7. Correction de Gamma

Les images PNG peuvent spécifier, via le segment gAMA, les caractéristiques gamma de l'image selon les conditions de sa création. Nous encourageons les programmes amenés à afficher ces images d'utiliser ces informations, plus l'information indiquant le type de moniteur ayant servi à la création d'image, ainsi que les conditions de lumière ambiante lors de cette création. Le but étant d'exploiter ces informations pour diffuser l'image sur le destinataire avec un rendu de contraste et de qualité la plus proche possible de l'intention originale du créateur. *Cf. Introduction au Gamma (Chapitre 13) si vous n'êtes pas familier de ces techniques*.

La correction gamma n'est pas appliquée au canal alpha, s'il existe. Les échantillons alpha représentent en effet un codage toujours linéaire de l'opacité.

Pour des applications de haute précision, le chromatisme exact des données RVB peut être spécifié par le segment cHRM dans l'image PNG, permettant une correspondance de couleurs encore plus serrée que ce que la simple correction gamma peut générer. *Cf. Introduction aux Couleurs (Chapitre 14) si vous n'êtes pas familier des techniques de représentation de couleurs.*

Cf. Motivations: Pourquoi le gamma? (Section 12.7), Recommandations pour les Encodeurs: Insertion des données gamma (Section 9.2), et Recommandations pour les Décodeurs: Gestion du gamma (Section 10.5).

2.8. Données texte

Un fichier PNG peut enregistrer du texte associé à l'image, tel qu'une description de l'image, ou une mention de copyright. Des mots-clef sont prévus pour indiquer la signification de chaque texte. L'usage de la table de caractères ISO 8859-1 (Latin-1) est recommandé pour ces données [ISO-8859]. Il s'agit d'un surensemble de l'ASCII 7- bits.

Les codes de caractères non définis dans l'ensemble Latin-1 devront être évités, car leur signification

dépend alors de l'environnement. Si un caractère non-Latin-1 apparaissait dans une chaîne de texte d'un fichier PNG, son interprétation pourra varier selon les plates-formes et les décodeurs. Certains systèmes pourront même être incapables d'afficher tous les caractères de l'ensemble Latin-1, bien que la plupart des systèmes actuels le fasse sans problème.

Des fonctions de stockage de texte sous forme compressé sont prévus. *Cf. Motivations: Chaînes de texte (Section 12.10).*

3. Structure du fichier

Un fichier PNG consiste en une signature PNG suivi d'une série de blocs. Ce chapitre définit la signature ainsi que les propriétés générales des blocs. Le détail de chaque bloc est précisé dans les chapitres suivants.

3.1.Signature d'un fichier PNG

Les huit premiers octets d'un fichier PNG contiennent toujours les valeurs suivantes (en décimal) :

```
137 80 78 71 13 10 26 10
```

Cette signature indique que le reste du fichier contient une seule image codée selon en PNG, consistant en une série de blocs dont le premier est un bloc IHDR et le dernier un bloc IEND. *Cf Motivations: Signature de fichier PNG (Section 12.11).*

3.2. Structure de bloc

Chaque bloc est constitué de quatre parties:

Longueur

Un entier non-signé sur 4 octets donnant le nombre d'octets de la partie données du bloc. Cette longueur n'est calculée que sur la partie "données", ne se compte pas elle-même, ni les parties "type", ni le CRC. Zéro est donc une longueur possible. Bien que les encodeurs et décodeurs soient sensés interpréter ce paramètre comme un entier non signé, on limitera sa valeur à (2^31)-1.

Type de bloc

Un code de type de bloc sur 4 octets. Pour faciliter l'implémentation et l'examen de fichiers PNG, ces codes se présenteront sous forme d'un groupe de lettres ASCII (A-Z et a-z, ou 65-90 et 97-122 en décimal). Cependant, les encodeurs et décodeurs devront toujours interpréter ces lettres en tant que valeurs numériques, et jamais sous forme d'une chaîne de caractères. Par exemple, le code de type IDAT n'est pas équivalent à celui formé par les quatre lettres équivalentes en EBCDIC. Les conventions nommant les divers types de blocs sont exposées dans les chapitres suivants.

Données

Les octets de données, s'il y en a, et selon le type de bloc. Cette partie peut être vide.

CRC

Un CRC (Cyclic Redundancy Check) sur 4 octets, calculé à partir des octets précédents du même bloc, tenant compte du type et des données, mais pas de la longueur. Le CRC est tout le temps présent, même pour des blocs ne contenant aucune données. Voir l'algorithme du CRC (Section 3.4). Le nombre d'octets de données peut être quelconque tant qu'il reste inférieure à la limite supérieure; de ce fait, les programmeurs ne pourront être assurés qu'un bloc sera aligné au delà de l'octet.

Les blocs peuvent apparaître dans un ordre quelconque, hormis certaines restrictions définies pour chacun des types spécifiés. (Une restriction notable est qu'un bloc IHDR doit toujours apparaître en premier, et un bloc IEND en dernier; ainsi, le bloc IEND sert de marqueur de fin de fichier). On pourra trouver plusieurs blocs d'un type donné, mais seulement si ce type le permet. *Cf Motivations: Structure des blocs (Section 12.12).*

3.3. Conventions d'appellation des blocs

Les codes de type sont nommés de telle sorte qu'un décodeur peut déduire certaines propriétés d'un bloc même lorsque le code de type n'est pas reconnu. Ces règles sont instituées pour permettre une extension sure et souple du format PNG, en permettant a un décodeur de décider du traitement à effectuer lorsqu'un bloc de type inconnu est rencontré. Ces règles d'appellation n'ont que peu d'intérêt lor 1000 sque le décodeur reconnaît le type de bloc.

Quatre bits particuliers dans le type (le bit 5 - valeur 32- dans chacun des 4 octets) sont utilisés pour transmettre des propriétés du bloc. Ce choix permet une lecture facile des propriétés "à vue" selon si chaque lettre composant le type est majuscule (bit 5 à 0) ou minuscule (bit 5 à 1). Un décodeur devra quant à lui tester numériquement la valeur de ces bits pour déduire les propriétés d'un bloc inconnu; le test de la casse sur une interprétation caractère est insuffisant, et peut même conduire à une interprétation erronée, suivant l'implémentation locale des tables de caractères.

Il est important de noter que les bits de propriétés des blocs sont une partie indissociable du nom du bloc, et de ce fait sont fixés pour un type de bloc prédéfini. De ce fait, deux blocs TEXT et Text sont des blocs entièrement différents, et non le même bloc doté de propriétés différentes. Les décodeurs devront reconnaître les types de blocs par une comparaison littérale numérique des quatre octets du code de type; il est donc tout à fait illicite d'opérer des changements de casse sur les types de blocs.

La sémantique des bits de propriétés est la suivante :

- Marqueur Auxiliaire: bit 5 / premier octet
  0 (majuscule) = critique, 1 (minuscule) = auxiliaire.

  Les blocs qui ne sont pas absolument nécessaires à un affichage significatif de l'image sont appelés blocs "auxiliaires". Un décodeur rencontrant un bloc de type inconnu marqué du caractère "auxiliaire" (bit 5, octet 1 à 1) peut sans risque ignorer le bloc et accepter d'afficher l'image. Le bloc temps (tIME) en est un exemple.

  Les blocs contenant des données indispensables à l'affichage correct de l'image sont appelés "critiques". Un décodeur rencontrant un bloc de nature inconnue avec la propriété "critique" marquée (bit 5, octet 1 à 0) doit indiquer au client que l'image contient des informations qu'il ne peut correctement interpréter. Le bloc d'en-tête (IHDR) est un exemple de bloc "critique".

- Marqueur Privé: bit 5 / second octet
  0 (majuscule) = public, 1 (minuscule) = privé.

  Un bloc est public s'il est conforme à la spécification PNG ou s'il fait partie de la liste des blocs publics spéciaux définis par le PNG. Des applications peuvent quant à elles créer et

exploiter des blocs privés (non standards) pour leur usage propre. Le nom des blocs privés doivent impérativement avoir une minuscule en deuxième lettre, tandis qu'une deuxième lettre majuscule sera réservée aux blocs publics. Notez qu'un décodeur n'a pas besoin de tester ce bit, dans la mesure où il n'a pas de signification fonctionnelle; Il s'agit surtout d'une facilité "administrative" pour éviter tout conflit entre noms privés et publics. *Cf. Types Additionnels (Section 4.4) et Recommandations pour les Encodeurs: Utilisation de blocs privés (Section 9.8).*

- Marqueur Réservé: bit 5 / troisième octet
  Doit être à 0 (majuscule) dans tout fichier conforme à la présente version du format PNG.

  La signification du troisième bit de propriétés n'est pas encore défini et est réservé pour des développements futurs. A l'heure actuelle, tous les noms devront avoir une troisième lettre majuscule. (Les décodeurs ne devront pas nécessairement générer une faute sur réception d'un troisième caractère minuscule; cependant, comme des futures versions de la spécification PNG pourraient donner une signification à ce bit, la meilleure position à adopter lorsqu'un bloc est reçu avec une troisième lettre minuscule est de considérer ce bloc comme inconnu).

- Marqueur Sécurité à la Copie : bit 5 / quatrième octet
  0 (majuscule) = copie non sûre, 1 (minuscule) = copie sûre

  Ce bit n'a pas d'intérêt pour les simples décodeurs, mais pourra être utilisé par des éditeurs PNG (programmes modifiant les fichiers PNG). Ce bit définit le comportement à adopter lorsque des blocs inconnus sont détectés d44 dans un fichiers à modifier.

  Si le marqueur de sécurité à la copie vaut 1, le bloc pourra être copié dans un fichier PNG modifié que le bloc soit reconnu ou non par le programme applicatif, et quelque soit l'étendue des modifications.

  Si le marqueur de sécurité à la copie vaut 0, cela indique que le bloc considéré a une influence sur l'image contenue dans le fichier. Si le programme a procédé à des modifications de blocs "critiques", du type addition, modification, suppression, ou réordonnancement de blocs critiques, alors des blocs non reconnus ne doivent pas être copiés dans le fichier PNG. (Bien sûr, si le programme reconnaît le bloc, il pourra décider d'en copier une version modifiée en conséquence).

  Un éditeur PNG aura toujours la permission de copier tous les blocs non reconnus si les modifications, suppressions, ajouts ou réordonnancements ne concernent que des blocs auxiliaires. Ceci à une conséquence sur les blocs auxiliaires : ils leur est interdit de dépendre d'autre blocs auxiliaires.

  Un éditeur PNG incapable de reconnaître un bloc critique doit afficher une erreur et refuser globalement de traiter ce fichier PNG. Le mécanisme sûr/non sûr est essentiellement dédié au traitement de blocs auxiliaires. Le marqueur de sécurité à la copie sera donc toujours

4. Spécifications des blocs

Ce chapitre définit les blocs PNG prédéfinis.

4.1. Blocs critiques

Toute implémentation doit être en mesure de comprendre et correctement afficher les blocs critiques standard. Une image PNG valide doit contenir un bloc IHDR, au moins un bloc IDAT, et enfin un bloc IEND.

4.1.1. En tête image IHDR

Le bloc IHDR doit apparaître EN PREMIER. Il contient :

Largeur :	4 octets
Hauteur :	4 octets
Echantillonnage :	1 octet
Type couleur :	1 octet
Méthode de compression :	1 octet
Méthode de filtrage :	1 octet
Méthode d'entrelacement :	1 octet

La largeur et la hauteur donnent les dimensions de l'image exprimées en pixels. Elles sont représentées chacune par un entier non signé sur 4 octets. La valeur zéro est invalide. La dimension maximum est (2^31)-1 pixels dans les deux directions, de manière à s'accommoder aux langages ayant des difficultés à traiter les entiers non signés de 4 octets.

La valeur de profondeur d'échantillonnage est un entier sur un octet donnant le nombre de bits par échantillon ou par entrée de la palette (et non par pixel dans ce cas). Les valeurs valides sont 1, 2, 4, 8, et 16, toutes ces valeurs n'étant pas autorisées selon le type de codage couleur. Le type couleur est codée sur un octet et décrit le type de modèle colorimétrique utilisé pour l'image. Le code de type couleur est une somme des commutateurs suivants: 1 (palette utilisée), 2 (couleur utilisée), et 4 (canal alpha utilisé). Les valeurs valides sont 0, 2, 3, 4, et 6.

Des restrictions en profondeur de bit sont imposées pour chaque modèle de couleurs afin de simplifier l'implémentation et pour interdire des combinaisons qui pourraient poser des problèmes de compression. Les décodeurs doivent supporter toutes les combinaisons "légales" de profondeur d'échantillonnage et de type de modèle de couleur. Ces combinaisons autorisées sont :

Modèle de couleur	Profondeurs autorisées	Interprétation
0	1,2,4,8,16	Échantillons en niveaux de gris
2	8,16	Trois échantillons R,V,B.
3	1,2,4,8	Couleurs indexées sur palette; un bloc PLTE doit être présent.
4	8,16	Échantillons en niveaux de gris, suivi d'une valeur de transparence.
6	8,16	Trois échantillons R,V, B, suivis d'une valeur de transparence.

La profondeur d'échantillonnage est toujours identique au nombre de bits par échantillon (d'un canal de couleur élémentaire) sauf dans le cas du type 3, pour laquelle l'échantillonnage est toujours sur 8 bits. La méthode de compression est donnée sous forme d'un entier sur un octet qui indique quelle méthode est

utilisée pour compresser les données image. A ce jour, seule la méthode de compression 0 (déflation/inflation avec fenêtre glissante de 32K) est définie. Toutes les images standard PNG doivent être compressées par cette méthode. Ce champ n'est implanté que pour des développements futurs ou pour son utilisation par des implémentations propriétaires. Les décodeurs doivent vérifier cet octet et afficher une erreur si un code non valide est détecté. *Cf. Compression par Déflation/Inflation (Chapitre 5).*

La méthode de filtrage est représentée par un entier sur un octet et indique la méthode de pré-traitement appliquée à l'image image avant la compression. A l'heure actuelle, seule la méthode de filtrage 0 (filtrage adaptatif par cinq filtres prédéfinis) est définie. Tout comme pour la méthode de compression, les décodeurs devront afficher une erreur en cas de code non valide. *Cf. Algorithmes de Filtrage (Chapitre 6).*

La méthode d'entrelacement est donnée sous forme d'un entier sur un octet indiquant l'ordre de transmission des données de l'image. Deux valeurs sont possibles actuellement: 0 (pas d'entrelacement) ou 1 (méthode Adam7). *Cf. Entrelacement des données (Section 2.6).*

4.1.2. Palette : bloc PLTE

Le bloc PLTE contient de 1 à 256 entrées de palette, chacune codée par trois octets donnant la valeur des canaux :

Rouge :	1 octet (0 = noir, 255 = rouge 100%)
Vert :	1 octet (0 = noir, 255 = vert 100%)
Bleu :	1 octet (0 = noir, 255 = bleu 100%)

Le nombre total d'entrées est déterminé à partir de la taille du bloc palette. Une longueur des données de ce bloc qui ne serait pas divisible par 3 indique une erreur. Ce bloc apparaît dans le cas d'un modèle de couleur de type 3, peut apparaître pour les modèles 2 et 6, et est absent dans le cas des modèles 0 et 4. En cas de présence, il doit précéder le premier bloc IDAT. Il ne peut y avoir qu'un bloc PLTE dans un ficher.

Pour le modèle de couleur 3 (couleurs indexées), le bloc PLTE est indispensable. La première entrée de la palette PLTE code la valeur notée 0 de pixel, la seconde entrée la valeur notée 1, etc. Le nombre d'entrées de palette ne peut excéder la profondeur bit définie pour les pixels (par exemple, 2^4 = 16 pour une profondeur bit de 4). Le cas où moins d'entrées de palette que ce que la profondeur bit permet est admis. Dans ce cas, tout pixel dont l'index sort de la plage définie par la palette indique une erreur.

Pour les modèles de couleur 2 et 6 (couleur vraie avec ou sans alpha), le bloc PLTE est facultatif. S'il est présent, il définit l'ensemble de 1 à 256 couleurs le plus convenable dans le cas où l' 1000 utilisateur de l'image ne peut afficher celle-ci en vraies couleurs (palette préférentielle). Si PLTE n'est pas présente, le client devra choisir ces couleurs de lui-même, mais il sera largement préférable que ce choix ait été fait auparavant par l'encodeur. *Cf. Recommandations pour les Encodeurs: Palettes Préférentielles, Section 9.5.*

Notez que la palette utilise des couleurs codées en 8 bits (1 octet) par canal chromatique indépendamment de la profondeur bit de chaque pixel. En particulier, la palette a toujours une profondeur de 8 bits même s'il s'agit d'une quantification suggérée d'une image en vraies couleurs sur 16-bits. Il n'y a aucune obligation que toutes les couleurs soient définies, ni utilisées dans l'image. Il n'y en a pas plus que toutes les couleurs y soient différentes.

4.1.3. Données image: bloc IDAT

Le bloc IDAT contient les données de l'image contenue dans le fichier. Pour créer ces données :

- Commencer par les lignes de pixels représentés par la description donnée dans le chapitre Codage

de l'image (Section 2.3); la disposition et la taille totale de ces données brutes sont déterminées par les champs du bloc IHDR.
- Filtrer l'image selon la méthode spécifiée dans le bloc IHDR. (Notez qu'avec la méthode 0, la seule actuellement définie, il est nécessaire de faire précéder chaque ligne d'un octet indiquant le type de filtrage).
- Compresser l'image filtrée avec la méthode spécifiée dans le bloc IHDR.

Les blocs IDAT contiennent les données obtenues en sortie de l'algorithme de compression. Pour lire l'image, il suffit d'inverser le processus.

Il peut exister plusieurs blocs IDAT; si c'est le cas, ils devront apparaître consécutivement sans autre blocs intermédiaires. Les données compressées seront alors la concaténation du contenu de tous les blocs IDAT. L'encodeur pourra diviser les données compressées en plusieurs blocs IDAT à sa convenance. (Cette multiplicité des blocs IDAT est permise pour que les encodeurs puissent travailler avec une quantité de mémoire fixée; typiquement, la taille de chaque bloc IDAT dépendra de la taille mémoire du tampon d'image attribué à l'encodeur). Il est important de noter que les limites des blocs IDAT n'ont aucune signification d'un point de vue sémantique et peuvent découper les données compressées à n'importe quel endroit du flux. A la limite, un fichier PNG dont les blocs IDAT ne contiendraient qu'un seul octet est parfaitement valide, bien qu'extrêmement dispendieux en place. (A ce sujet, même des blocs IDAT à zéro octets sont admis, la perte de place étant encore plus grande).*Cf. Algorithmes de Filtrage (Chapitre 6) et Compression par Inflation/Déflation (Chapitre 5).*

4.1.4. Bloc de fin d'image IEND

Le bloc IEND doit apparaître EN DERNIER. Il marque la fin du flux de données transportée par le PNG. La partie données de ce bloc est vide.

4.2. Blocs auxiliaires

Tous les blocs auxiliaires sont optionnels, en ce sens que certains encodeurs pourront les omettre, et les décodeurs pourront les ignorer. Cependant, nous encourageons le développement d'encodeurs qui écriront les blocs auxiliaires standards dans la mesure où l'information est disponible, et des décodeurs qui interprètent ces blocs dès que possible ou approprié.

Les blocs auxiliaires sont listés ci-après par ordre alphabétique. L'ordre dans lequel ils apparaissent dans un fichier est sans importance.

4.2.1. Couleur de fond : bKGD

Le bloc bKGD définit une couleur de fond "par défaut" pour l'affichage de l'image. Notez que les navigateurs ne sont aucunement tenus de respecter cette couleur; un navigateur pourra utiliser une couleur de son choix.

Pour le modèle de couleur 3 (couleurs indexées), le bloc bKGD contient:

Index 1000 de palette : 1 octet

La valeur donnée est l'index de la couleur dans la palette contenue dans le fichier. Pour les modèles de couleur 0 et 4 (niveaux de gris, avec ou sans canal alpha), le bloc bKGD contient :

Gris : 2 octets, gamme 0 .. (2^profondeur_bit)-1

(Par mesure de simplification, 2 octets sont utilisés quelle que soit la définition de l'image). La valeur

donnée est le niveau de gris utilisée pour afficher le fond. Pour les modèles 2 et 6 (vraies couleurs, avec ou sans canal alpha), le bloc bKGD contient :

Rouge :	2 octets, gamme 0 .. (2^profondeur_bit)-1
Vert :	2 octets, gamme 0 .. (2^profondeur_bit)-1
Bleu :	2 octets, gamme 0 .. (2^profondeur_bit)-1

(Par mesure de simplification, 2 octets sont utilisés quelle que soit la définition de l'image). La valeur codée est la couleur RVB utilisée pour le fond.

Lorsqu'il est présent, le bloc bKGD devra précéder le premier bloc IDAT, et devra passer après le bloc PLTE, s'il existe.*Cf Recommendations pour les Décodeurs: Couleur de Fond (Section 10.7).*

4.2.2. Chromatisme primaire et point blanc : cHRM

Certaines applications d'affichage dépendant du matériel, et nécessitant une définition explicite des couleurs primaires pourront la trouver dans un fichier PNG. Il s'agira d'un bloc cHRM qui spécifie les coordonnées x,y du modèle chromatique CIE 1931 pour les couleurs rouge, vert, et bleu primaires, ainsi que la référence du point blanc. *Cf.* *Modèles colorimétriques (Chapitre 14).*

Le bloc cHRM contient :

Point blanc x:	4 octets
Point blanc y :	4 octets
Rouge x :	4 octets
Rouge y :	4 octets
Vert x :	4 octets
Vert y :	4 octets
Bleu x :	4 octets
Bleu y :	4 octets

Chaque valeur est codée selon un entier de 4 octets non signé, représentant les coordonnées x ou y multipliées par un facteur 100000. Par exemple, une coordonnée de 0,3127 sera enregistrée comme un entier de valeur 31270. Le bloc cHRM est autorisé dans tout fichier PNG, bien qu'il n'ait pas d'utilité pour les images en niveaux de gris. Si l'encodeur ne connaît pas ou ne peut obtenir les données chromatiques du système source, le bloc cHRM ne devra pas être écrit dans le fichier; l'absence d'un bloc cHRM au décodage indique que les couleurs primaires de l'image dépendent du système utilisé.

Si le bloc cHRM est présent, il devra précéder le premier bloc IDAT, et devra précéder un bloc PLTE s'il existe.*Cf. Recommandations pour les Encodeurs: Gestion des Couleurs (Section 9.3), et Recomm 1000 andations pour les Décodeurs: Gestion des couleurs (Section 10.6).*

4.2.3. Gamma de l'image : gAMA

Le bloc gAMA code le gamma de la caméra (réelle ou virtuelle) qui a produit l'image, et donc le gamme de l'image originale. Plus précisément, le bloc gAMA code la valeur *file_gamma*, selon la définition du codage Gamma (Chapitre 13).

Le bloc gAMA contient :

Gamma image :	4 octets

La valeur est codée selon un entier non signé sur 4 octets, représentant la valeur gamma multipliée par 100000. Par exemple, un gamma de 0.45 sera enregistré comme un entier valant 45000.

Si l'encodeur est incapable de connaître le gamma de l'image originale, il ne devra pas écrire le bloc gAMA; l'absence d'un tel bloc au décodage indique que le gamma est inconnu.

Si le bloc gAMA est présent, il devra précéder le premier bloc IDAT, ainsi que le bloc PLTE s'il existe. *Cf. Correction de Gamma (Section 2.7), Recommandations pour les Encodeurs: Gestion du gamma (Section 9.2), et Recommandations pour les Décodeurs: Gestion du Gamma (Section 10.5).*

4.2.4. Histogramme de l'image : hIST

Le bloc hIST donne la fréquence approximative d'utilisation de chaque couleur de la palette. Un tel bloc ne peut être présent que si un bloc palette PLTE est défini. Si un client est incapable de reproduire toutes les couleurs listées dans la palette, l'histogramme peut alors aider à déterminer quel sous ensemble de couleurs rendra l'image au mieux.

Le bloc hIST contient une série d'entiers non signés sur deux octets (16 bits). On devra trouver autant d'entrées d'histogramme que de couleurs définies dans le bloc PLTE. Chaque entrée code de façon proportionnelle le nombre de pixels de l'image qui sont de la couleur associée. Le facteur d'échelle est choisi par l'encodeur.

La valeur de la fréquence de couleur est approximative, sauf pour la valeur zéro qui indique avec certitude que la couleur correspondante de la palette n'est jamais utilisée dans l'ensemble de l'image. Si au moins un pixel de l'image est d'une couleur donnée, il est important que sa fréquence associée ne soit pas nulle (au moins 1).

Lorsque la palette représente la quantification suggérée d'une image enregistrée en vraies couleurs, l'histogramme est d'autant plus approximatif, dans la mesure où il n'est pas garanti que l'encodeur et le décodeur attribuent des pixels à des entrées de palette de la même manière. Dans ce cas, aucune entrée ne doit être nulle.

Si un bloc hIST est présent, il devra suivre un bloc PLTE, et devra précéder le premier bloc IDAT. *Cf. Motivation: Histogramme de palette (Section 12.14), et Recommandations pour les Décodeurs: Palettes suggérées et utilisation des histogrammes (Section 10.10).*

4.2.5. Dimensions physiques de pixel : pHYs

Le bloc pHYs spécifie la taille du pixel et son rapport dimensionnel. Il contient :

Pixels par unité, axe X :	4 octets (entier non signé)
Pixels par unité, axe Y :	4 octets (entier non signé)

Unité :	1 octet

Les valeurs suivantes sont valides pour l'unité :

0 :	unité inconnue
1 :	mètre

Lorsque le code d'unité est 0, le bloc pHYs ne définit qu 1000 e le rapport dimensionnel du pixel. La taille effective du pixel reste non spécifiée.

Note: un pouce vaut 0,0254 m.

Si ce bloc auxiliaire est absent, on considérera que l'on a affaire a des pixels carrés, de taille inconnue. S'il est présent, ce bloc doit précéder le premier bloc IDAT.*Cf. Recommandations pour les Décodeurs: Dimensions des Pixels (Section 10.2).*

4.2.6. Bits significatifs : sBIT

Pour simplifier les décodeurs, le format PNG spécifie les profondeurs de bits qui peuvent être utilisées, et demande que les échantillons soient recalibrés pour utiliser la plage complète définie par cette profondeur de bits. Le bloc sBIT est défini pour garder une trace du nombre de bits significatifs d'origine. Ceci permet aux décodeurs de pouvoir retrouver les données originales de l'image, sans pertes, même si l'échantillonnage initial n'est pas supporté par le format PNG. Nous recommandons l'émission d'un bloc sBIT dans le cas où les données ont été calibrées à partir d'une définition inférieure. Pour le modèle de couleurs 0 (niveaux de gris), le bloc sBIT contiendra un octet unique, indiquant le nombre de bits significatifs par pixel de l'image originale.

Pour le modèle de couleurs 2 (vraies couleurs), le bloc sBIT contiendra trois octets, indiquant le nombre de bits significatifs pour chaque canal primaire de l'image originale (R, V, B).

Pour le modèle 3 (couleurs indexées), le bloc sBIT contient trois octets, indiquant le nombre de bits significatifs utilisés dans le codage des couleurs désignées dans la palette, et ce, pour chaque couleur primaire. Pour le modèle 4 (niveaux de gris avec canal alpha), le bloc sBIT contient deux octets, indiquant le nombre de bits significatifs pour chaque canal de l'image originale (luminosité et transparence). Pour le modèle 6 (vraies couleurs avec canal alpha), le bloc sBIT contiendra 4 octets, indiquant les valeurs du modèle 2, plus le nombre de bits significatifs du canal de transparence.

Chacune des profondeurs explicitées dans le bloc sBIT doit être supérieure à zéro et inférieure ou égale à la profondeur en bits définie dans le fichier PNG (8 pour des images en couleurs indexées, la valeur mentionnée dans le bloc IHDR pour les autres modèles de couleurs).

Un décodeur pourra ignorer un bloc sBIT: l'image enregistrée après calibrage reste un fichier PNG valide de profondeur bit indiquée dans le bloc IHDR. Cependant, si le décodeur souhaite pouvoir reproduire l'image dans sa définition originale, ceci pourra être fait en décalant vers la droite les échantillons (ou les entrées de palette, pour les images en couleurs indexées). Le calibrage fait par l'encodeur "poussera" vers les poids forts les valeurs originales des échantillons.

Si un bloc sBIT est présent, il doit précéder le premier bloc IDAT, ainsi que le bloc PLTE s'il existe. *Cf. Recommandations pour les Encodeurs: Calibrage des échantillons (Section 9.1) et Recommandations pour les Décodeurs: Restitution de définition (Section 10.4).*

4.2.7. Texte : tEXt

Des information textuelles que l'encodeur souhaîte enregistrer avec l'image pourront l'être dans des blocs tEXt. Chaque bloc tEXt contiendra un mot-clef et une chaîne de caractères, selon le format :

Mot-clef :	1-79 octets (chaîne de caractères)
Séparateur (Null) :	1 octet
Texte :	n octets (chaîne de caractères)

Le mot-clef et la chaîne de caractères sont séparés par un caractère nul (octet de valeur 0). Par conséquent, ni le mot-clef, ni la chaîne de caractères ne peuvent contenir un caractère nul. Notez que la chaîne de caractère n'est pas terminée par un Nu 1000 ll (la longueur de bloc est une information suffisante pour déterminer la fin de la chaîne). Le mot-clef doit au moins contenir un caractère, et au plus 80. La taille de la chaîne peut être quelconque, de 0 caractères à la taille maximale admissible par le bloc, moins la taille du mot-clef moins un (le séparateur).

Un nombre quelconque de blocs tEXt peuvent être présents, et plusieurs blocs de même mot-clef sont admis.

Le mot-clef définit la nature de l'information contenue dans la chaîne qui suit. Les mot-clef suivant ont été prédéfinis et pourront être utilisés si le besoin l'exige :

Title	Titre (court) de l'image
Author	Nom du créateur
Description	Description de l'image (peut être longue)
Copyright	Mention de propriété intellectuelle
Creation Time	Date de création originale
Software	Logiciel utilisé pour la création de l'image
Disclaimer	Editeur
Warning	Avertissement sur le contenu
Source	Plateforme
Comment	Commentaire, conversion des commentaires GIF

Pour renseigner le mot-clef Creation Time, le format de date défini dans le paragraphe 5.2.14 de la RFC 1123 est suggéré, mais non imposé [RFC-1123]. Les décodeurs devront pouvoir assumer n'importe quel format libre d'information, pour tous les champs textes.

D'autres mots-clefs pourront être "inventés" à d'autre fins. Les mots-clef ayant un intérêt général pourront être "enregistrés" par l'équipe de maintenance de la spécification PNG. Cependant, l'usage de mots-clefs propriétaires non enregistrés reste permis (nous recommandons simplement de choisir des mots-clefs suffisamment explicites, afin de minimiser les risques de voir le même mot utilisé par deux opérateurs distincts avec une signification divergente).

Les mots-clefs et le texte seront interprétés selon la table de caractères définie par l'ISO 8859-1 (Latin-1) [ISO-8859]. La chaîne de caractères pourra contenir tout caractère Latin-1. Un saut de ligne sera représenté par un caractère 'linefeed' unique (10 en décimal, ou \n en notation C); l'utilisation d'autres caractères de contrôle dans cette chaîne est déconseillé.

Les mots-clefs ne peuvent être constitués que de caractères imprimables Latin-1 ou espaces; soient les caractères de code 32 à 126 et 161 à 255. Pour réduire toute erreur de lecture (humaine) des mots-clefs, il est interdit déplacer des espaces au début et à la fin du mot-clef. De même, plusieurs espaces consécutifs sont illégaux. Notez aussi que l'espace non sécable (code 160) n'est pas permis dans les mots-clef, car on ne peut le distinguer visuellement d'un espace normal.

Les mots-clefs doivent être écrits tels qu'ils ont été enregistrés, permettant ainsi aux décodeurs de procéder par comparaison numérique pour leur reconnaissance. En particulier, les mots-clefs sont considérés comme sensibles à la casse. *Cf. Recommandations pour Encodeurs: Association de textes (Section 9.7) et Recommandations pour les Décodeurs: Association de textes (Section 10.11).*

4.2.8. Dernière modification le : tIME

Le bloc tIME donne la date et l'heure de la dernière édition (et non celles de la création originale). Il contient :

Année :	2 octets (complète; par exemple, 1995, et non 95)
Mois :	1 octet (1-12)
Jour :	1 octet (1-31)
Heures :	1 octet (0-23)
Minutes :	1 octet (0-59)
Secondes :	1 octet (0-60) (60, pour des raisons de bords)

On préférera le Temps Universel (UTC, ou encore GMT) plutôt que l'heure locale. Le bloc tIME est prévu pour être utilisé en tant qu'étiquette temporelle automatique, remise à jour chaque fois que l'image a changé. Il est recommandé que ce bloc ne soit pas édité par des éditeurs PNG qui n'auront apporté aucune modification à l'image. *Cf. mot-clef Creation Time du bloc tEXt pouvant servir de marqueur temporel utilisateur.*

4.2.9. Transparence : tRNS

Le bloc tRNS spécifie su l'image utilise la transparence simple : soit des valeurs alpha associées à chaque entrée de palette (pour des images en couleurs indexées) ou une couleur transparente unique (pour les images en niveaux de gris ou vraies couleurs). Bien que la transparence simple ne soit pas aussi élégante qu'un canal alpha complet, elle nécessite beaucoup moins d'espace et suffit dans la grande majorité des cas. Pour le modèle de couleurs 3 (couleurs indexées), le bloc tRNS contient une série de valeurs alpha sur un octet, correspondant aux entrées du bloc PLTE :

Alpha pour l'entrée 0 de palette :	1 octet
Alpha pour l'entrée 1 de palette :	1 octet

... etc ...

Chaque entrée indique que tout pixel telle couleur devra être traité avec la valeur de transparence spécifiée. Les valeurs Alpha ont alors la même interprétation que dans un canal alpha complet classique sur 8 bits : 0 indique une transparence totale, 255 une opacité totale, quelle que soit la profondeur bit du codage de l'image. Le bloc tRNS ne peut comporter plus d'entrées alpha que d'entrées de palette, l'inverse est par contre possible. Dans un tel cas, les valeurs alpha de toutes les couleurs restantes seront considérées comme valant 255. Ceci permet, par exemple, dans le cas où une seule couleur de palette (l'entrée 0) est donnée comme couleur transparente, de pouvoir réduire le bloc tRNS à un octet (nul), et ainsi gagner de la place.

Dans le modèle 0 (niveaux de gris), le bloc tRNS contient une seule valeur indiquant un niveau de gris, enregistrées sous le format :

Gris : 2 octets, gamme 0 .. (2^profondeur_bit)-1

(Par mesure de simplification, 2 octets sont utilisés quelle que soit la définition de l'image). Les pixels de ce niveau de gris seront traités comme des pixels transparents (équivalents à une valeur alpha de 0); tous les autres pixels sont traités comme totalement opaques (valeur alpha (2^profondeur_bit)-1). Dans le modèle de couleur 2 (vraies couleurs), le bloc tRNS contient une couleur RVB, enregistrée sous le format 1000 :

Rouge : 2 octets, gamme 0 .. (2^profondeur_bit)-1

Vert : 2 octets, gamme 0 .. (2^profondeur_bit)-1

Bleu : 2 octets, gamme 0 .. (2^profondeur_bit)-1

(Par mesure de simplification, 2 octets sont utilisés quelle que soit la définition de l'image). Les pixels de couleur égale à la couleur spécifiée dans le bloc tRNS sont traités comme totalement transparents (valeur alpha à 0); tous les autres pixels sont traités comme totalement opaques (valeur alpha (2^profondeur_bit)-1).

Un bloc tRNS est interdit dans les modèles de couleur 4 et 6, car ces modes supposent l'existence d'un véritable canal alpha. Note: lorsque l'on travaille en niveaux de gris 16 bits ou en vraies couleurs, il est important de comparer les deux octets de l'échantillon pour déterminer si un pixel est transparent ou non. Lorsque que certains décodeurs éliminent certains bits des poids faibles pour générer l'affichage, il faudra prendre la précaution de tester auparavant la transparence. Par exemple, si le niveau de gris 0x0001 est assigné à la fonction de transparence, il serait incorrect de ne comparer que les poids forts des échantillons et déduire qu'une valeur 0x0002 est aussi transparente.

Lorsqu'un bloc tRNS est présent, il doit précéder le premier bloc IDAT, et passer après le bloc PLTE, s'il existe.

4.2.10. Données texte compressées : zTXt

Un bloc zTXt contient des données textuelles, tout comme un bloc tEXt mais en tirant le bénéfice de la compression. Les blocs zTXt et tEXt sont équivalents d'un point de vue sémantique, mais le type zTXt est recommandé pour enregistrer des textes longs.

Un bloc zTXt contient :

Mot-clef : 1-79 octets (chaîne de caractères)

Séparateur (Null) :	1 octet
Méthode de compression :	1 octet
Texte compressé :	n octets

Le mot-clef et le séparateur nul sont à l'identique du bloc tEXt. Notez que le mot-clef n'est pas compressé. L'octet méthode de compression permet de dissocier plusieurs méthodes de compression du texte. Actuellement, seule la valeur 0 est définie (compression par déflation/inflation). L'octet indiquant la méthode de compression est suivi de la chaîne compressée. Avec la méthode 0, le flux compressé adhère à la spécification zlib (Cf. Compression par déflation/inflation, Chapitre 5). La décompression de ce flux produit un texte en Latin-1 identique au texte qui aurait été enregistré dans un bloc tEXt.

Un nombre quelconque de blocs zTXt et tEXt pet être présent dans un fichier PNG. Cf. la définition précédente du bloc tEXt pour les mots-clefs prédéfinis et les conseils sur le format du texte. *Cf. Recommandations pour les Encodeurs: Traitement des blocs texte (Section 9.7), et Recommandations pour les Décodeurs: Traitement des blocs texte (Section 10.11).*

4.3. Résumé à propos des blocs prédéfinis

La table suivante résume les caractéristiques des principaux blocs standards :

Blocs critiques (ordre indispensable, PLTE est optionnel) :

Noms	Multiple	Contraintes d'ordre
IHDR	Non	En premier obligatoirement
PLTE	Non	Avant IDAT
IDAT	Oui	IDAT multiples consécutifs
IEND	Non	En dernier obligatoirement

Blocs auxiliaires (ordre non significatif, excepté spécifications) :

Noms	Multiple	Contraintes d'ordre
cHRM	Non	Avant PLTE et IDAT
gAMA	Non	Avant PLTE et IDAT
sBIT	Non	Avant PLTE et IDAT
bKGD	Non	Après PLTE; avant IDAT
hIST	Non	Après PLTE; avant IDAT
tRNS	Non	Après PLTE; avant IDAT
pHYs	Non	Avant IDAT

tIME	Non	Aucune
tEXt	Oui	Aucune
zTXt	Oui	Aucune

Mots clefs standards pour les blocs tEXt et zTXt :

Title	Mention ou titre court de l'image
Author	Nom du créateur de l'image
Description	Description de l'image (peut être longue)
Copyright	Notice de droits d'exploitation
Creation Time	Date de la création originale
Software	Logiciel ayant généré l'image
Disclaimer	Propriétaire légal
Warning	Avertissement sur le contenu
Source	Plate forme de création 7ac
Comment	Commentaires; conversion de commentaires GIF

4.4. Autres types de blocs

D'autres blocs PNG publics sont définis dans le document "PNG Special-Purpose Public Chunks" [PNG-EXTENSIONS]. Les blocs qui y sont décrits sont d'un usage moins fréquent que ceux définis dans cette spécification. Malgré cela, nous demandons aux auteurs d'applications utilisant le format PNG de vérifier si ces blocs additionnels ne peuvent suffire à leurs besoins, avant de créer un nouveau type de bloc. De nouveaux blocs peuvent être ajoutés à cette liste sur simple proposition en contactant l'équipe de maintenance de la spécification PNG à l'adresse png-info@uunet.uu.net ou png-group@w3.org.

Des nouveaux types de blocs ne seront enregistrés que s'ils peuvent avoir une utilité pour d'autres développeurs et s'ils ne violent pas la philosophie générale de construction des PNGs. La validation de ces nouveaux blocs n'est pas systématique, mais nous sommes conscients de l'intérêt de diffuser une nouvelle spécification de type lorsque nous sentons que son usage peut être généralisé. Notez cependant que la création de nouveaux types de blocs critiques est déconseillée, sauf en cas de nécessité absolue. Des applications pourront mettre en œuvre des blocs de type privés pour un usage propriétaire. *Cf. Recommandations pour les Encodeurs: Usage des blocs privés (Section 9.8).*

Les décodeurs doivent être prêts à recevoir des codes de types privés ou publics inconnus. Ces blocs devront être traités comme spécifié dans la section Conventions d'appellation des blocs (Section 3.3).

5. Compression par déflation/inflation

La méthode de compression 0, supportée par PNG (la seule actuellement disponible) définit une compression par déflation/inflation utilisant une fenêtre glissante de 32K. La compression par déflation est un dérivé du LZ77 utilisé dans les programmes zip, gzip, pkzip et assimilés. Notre groupe de travail a mené des recherches exhaustives pour garantir la liberté de droits du procédé. Des implémentations en langage C entièrement portables sont disponibles gratuitement.

Des flux compressés par déflation sont enregistrés sous le format "zlib", selon la structure suivante :

Méthode de compression/flags :	1 octet
flags additionnels / bits de vérification :	1 octet
Données compressées :	n octet
Vérification :	4 octets

Les détails de ce codage sont donnés dans la spécification zlib [RFC-1950].

Dans la méthode de compression PNG 0, le code méthode de compression/flags doit préciser la méthode 8 ("déflation") et une taille de fenêtre LZ77 au plus de 32K. Notez que le code de méthode "zlib" et le code compression PNG n'ont pas de relation. La position des flags additionnels ne doivent pas spécifier un quelconque dictionnaire.

Les données compressées dans les données zlib sont organisées sous forme d'une série de segments, chacun pouvant représenter des données brutes (non compressées), des données compressées selon un LZ77 encodé par codes Huffman standards, ou compressées selon un LZ77 encodé par codes Huffman personnalisés. Un bit marqueur dans le dernier segment signale la fin du flux à l'intention du décodeur. Les détails du processus de compression et d'encodage sont donnés par la spécification RFC-1951.

La valeur de vérification enregistré à la fin du flux zlib est calculé sur les données non compressées incluses dans le flux. Notez que cette valeur n'est pas calculé selon l'algorithme de CRC utilisé pour la vérification de blocs PNG. Cette valeur de vérification zlib est surtout utilisée pour vérifier que les implémentations de la déflation et de l'inflation sont compatibles et cohérentes. Le CRC de bloc PNG, quant à lui, permet de s'assurer que le bloc PNG a été transmis correctement. Dans un fichier PNG, la concaténation de tous les segments de données transportés par la série de blocs IDAT donne le flux zlib décrit ci-dessus. Ce flux, décompressé, produit l'image filtrée selon le procédé décrit dans ce document.

Il est important d'insister sur le fait que les blocs IDAT "découpent" ce flux tout à fait arbitrairement, et que la fin des données d'un bloc IDAT peut intervenir n'importe où dans le flux zlib. Il y aura peu de chance d'avoir une quelconque corrélation entre le découpage en blocs IDAT et le découpage selon les blocs "déflatés". Par exemple, il sera possible que la valeur de vérification d'un segment zlib se retrouve à cheval entre deux blocs IDAT.

Dans le même ordre d'idées, il n'y a aucune corrélation entre la structure de l'image (c'est à dire, les limites des lignes) et les 4ac limites de segments de données compressées ni de blocs IDAT. L'image est représentée par l'intégralité du flux zlib enregistré sous forme d'un certain nombre de blocs IDAT; un décodeur qui prendrait toute considération autre serait incorrect. (Bien sûr, certaines implémentations d'encodeurs pourraient introduire ce genre de corrélation. Mais les décodeurs ne doivent pas compter dessus).

Le format PNG utilise aussi la technique zlib dans les blocs zTXt. Dans un tel bloc, la fin du bloc qui suit

l'octet codant le type de compression est un flux zlib tel que spécifié ci-dessus. Ce flux, une fois décompressé, contient le texte en clair associé au mot-clef. Contrairement aux données de l'image, le texte ne peut être partagé entre plusieurs blocs zTXt; chaque bloc zTXt contient un flux zlib autonome et indissociable. Une documentation complémentaire et une implémentation source en C pour la déflation et l'inflation sont disponibles dans les archives Info-zip.

6. Algorithmes de filtrage

Ce chapitre décrit les algorithmes de filtrage utilisés avant la compression de l'image. Le but de ce filtrage est de préparer l'image en vue d'une compression optimale.

6.1. Types de filtres

La méthode de filtrage 0 de la spécification PNG définit cinq types de filtres de base :

Type	Nom
0	Aucun
1	Différentiel horizontal
2	Différentiel vertical
3	Prédictif par Moyenne
4	Filtre de Paeth

(Notez que cette méthode 0 spécifiée dans le bloc IHDR définit précisément cet ensemble de cinq filtres. Si dans le futur cet ensemble doit être complété, il faudra prendre un nouveau numéro de méthode, de sorte qu'un décodeur n'aura pas à attendre la décompression des données pour découvrir que d'autres filtres ont été utilisés).

L'encodeur peut choisir lequel de ces cinq filtres appliquer à chacune des lignes de l'image, et ce ligne par ligne. Les lignes d'images filtrées seront précédées du numéro de filtre utilisé, avant d'être soumises à l'étage de compression.

Les algorithmes de filtrage prennent en compte les octets, et non les pixels, quelque soit la profondeur de bit du pixel ou le modèle de couleur de l'image. Les algorithmes de filtrage s'intéressent à la séquence d'octets constituée par une ligne d'image, telle que spécifiée par le chapitre Codage de l'image (Section 2.3). Si l'image contient un canal alpha, celui-ci est filtré au même titre que les autres échantillons.

Lorsque l'image est entrelacée, chaque passe d'entrelacement est traité par le filtre comme une image indépendante. Le filtre s'intéressera alors à la séquence d'octets formée par les informations des pixels transmis pendant la passe d'entrelacement, la "ligne précédente" étant celle précédemment transmise dans la même passe, et non la ligne adjacente dans l'image originale. Notez que la "sous-image" transmise dans une passe reste rectangulaire, mais de largeur et de hauteur inférieure à celle de l'image complète. Le filtrage n'est pas appliqué lorsque la sous-image est vide.

Pour tous les filtres, les octets "à gauche du" premier octet d'une ligne doivent être considérés comme étant à zéro. Pour les filtres qui prennent en compte la ligne "précédente", on prendra pour ligne

précédente à la première ligne d'une passe (ou la première ligne d'image si celle-ci n'est pas entrelacée) une ligne fictive d'octets tous nuls.

Pour inverser l'effet d'un filtre, le décodeur devra utiliser les valeurs décodées du pixel précédent sur la même ligne, ou du pixel immédiatement au dessus (en même position dans la ligne précédente), ainsi que du pixel précédent ce dernier. Cela implique que le décodeur ait la capacité de stocker en mémoire au moins une ligne complète. Même pour les filtres qui ne se réfèrent pas à 1000 la ligne antérieure, le décodeur devra être en mesure de mémoriser la ligne courante, dans la mesure où rien empêche que le filtre utilisé à ligne suivante n'ait pas à son tour à se référer à cette ligne.

PNG n'impose aucune restriction sur le type de filtre à utiliser sur une ligne d'image. Cependant, les divers filtres n'ont pas tous la même efficacité selon le type de données. *Cf. Recommandations pour les Encodeurs: Sélection du filtre (Section 9.6).Cf. Motivations: Filtrage (Section 12.9).*

6.2. Filtre type 0: Aucun

Le filtre de type 0 est un "passe tout". Les données ne sont pas modifiées. Sa présence répond seulement à l'obligation d'ajouter un octet de type de filtre en tête de ligne.

6.3. Filtre type 1: différentiel horizontal

Le filtre dit "différentiel horizontal" ou "Sub" code la différence entre l'octet d'un pixel et l'octet correspondant du pixel précédent sur la même ligne.

Pour calculer chaque pixel, il suffit d'appliquer la formule suivante à chaque pixel de la ligne :

```
Sub(x) = Raw(x) - Raw(x-bpp)
```

où x varie de zéro jusqu'au nombre de pixels dans la ligne moins 1, Raw(x) est la valeur de l'octet à la position x dans la ligne, et bpp représente le nombre d'octets nécessaire au codage d'un pixel (un au minimum). Par exemple, pour le modèle de couleur 2 avec une profondeur de bits de 16, bpp vaut 6 (trois échantillons, 2 octets par échantillon); pour le modèle 0 avec une profondeur de bit de 2, bpp vaut 1 (arrondi); pour le modèle 4 avec une profondeur de bits de 16, bpp vaut 4 (deux octets de niveau de gris plus deux octets du canal alpha).

Notez que ce calcul est fait pour chaque octet, indépendamment de la profondeur de bits. Dans une image 16 bits, chaque MSB est "prédit" à partir du MSB précédent, et chaque LSB à partir du LSB précédent, grâce à la définition de bpp.

L'arithmétique utilisée est non signée modulo 256, de sorte que l'entrée et la sortie restent dans le codage admissible d'un seul octet. La séquence des valeurs Sub(x) correspond à la ligne filtrée.

Pour tout $x < 0$, on suppose $Raw(x) = 0$.

Pour inverser l'effet de ce filtre, il suffit de partir de 0, et d'ajouter arithmétiquement la valeur Sub(x) :

```
Raw(x) = Sub(x) + Raw(x-bpp)
```

(modulo 256), Raw(x-bpp) étant le précédent octet déjà décodé.

6.4. Filtre type 2 : différentiel vertical

Le filtre "différentiel vertical", ou "Up" fonctionne sur le même principe que le différentiel horizontal,

sauf que le "prédicteur" utilisé est le pixel situé juste au dessus, plutôt que le précédent sur la même ligne. Pour calculer ce filtre, la formule à appliquer devient :

$$Up_n(x) = Raw_n(x) - Raw_{n-1}(x)$$

où x varie de zéro jusqu'au nombre de pixels dans la ligne moins 1, $Raw_n(x)$ est la valeur de l'octet à la position x dans la ligne traitée (ligne n), et $Raw_{n-1}(x)$ la valeur de l'octet de même position dans la ligne au dessus (précédente).

Ce calcul est fait octet par octet, indépendamment de la profondeur bit. L'arithmétique utilisée est non signée modulo 256, de sorte que l'entrée et la sortie restent dans le codage admissible d'un seul octet. La séquence des valeurs Up(x) correspond à la ligne filtrée. Pour la première ligne de l'image (ou de la passe dans le cas d'une image entrelacée), On supposera que tous les $Raw_{n-1}(x) = 0$ quelque soit x.

Pour inverser l'effet du filtre, on appliquera la formule suivante :

$$Raw_n(x) = Raw_{n-1}(x) + Up_n(x)$$

(modulo 256), $Raw_{n-1}(x)$ étant l'octet déjà décodé de la ligne précédente.

6.5. Filtre type 3 : Différentiel par moyenne

Le filtre différentie 1000 l "par moyenne" (Average) exploite la moyenne arithmétique des deux prédicteurs précédents (pixel précédent sur la même ligne, même pixel sur la ligne précédente) comme base du codage du pixel.

Pour calculer ce filtre, la formule à appliquer devient :

$$Average_n(x) = Raw_n(x) - floor((Raw_n(x-bpp)+Raw_{n-1}(x))/2)$$

où x est la position du pixel à traiter (de 0 à largeur -1), et bpp le nombre d'octets par pixel tel que défini pour le filtre 1.

Ce calcul est fait octet par octet, indépendamment de la profondeur bit. La séquence des valeurs Averagen(x) correspond à la ligne filtrée. La soustraction de la moyenne (prédicteur) à la valeur de l'échantillon est faite modulo 256 de sorte à ce que les valeurs de départ et d'arrivée soient codables dans un octet. Par contre, la somme $Raw_n(x-bpp)+Raw_{n-1}(x)$ doit être exacte et sans dépassement de capacité (selon une arithmétique à au moins 9 bits). La fonction floor() arrondit le résultat de la division au premier entier inférieur, si celui-ci n'est pas entier; il s'agira donc d'une division entière, assimilable à un décalage d'un bit vers la droite.

Pour tout $x < 0$, on supposera que $Raw_n(x) = 0$. Pour la première ligne (ou la première ligne d'une passe si l'image est entrelacée), on considérera la présence d'une ligne fictive $Raw_0(x) = 0$ quelque soit x.

Pour inverser l'effet du filtre, on utilisera la formule suivante :

$$Raw_n(w) = Average_n(x) + floor((Raw_n(x-bpp)+Raw_{n-1}(x))/2)$$

dans laquelle le résultat est calculé modulo 256, et $Raw_j(x)$ correspond à une valeur déjà extraite.

6.6. Filtre type 4 : Filtre de Paeth

Le filtre de Paeth calcule une fonction prédictive se basant sur les trois pixels voisins (gauche, au-dessus, au dessus à gauche), et choisit comme prédicteur la valeur du pixel la plus proche de celle du pixel à traiter. Cette technique a été instaurée par Alan W. Paeth [PAETH]. Pour calculer ce filtre, appliquer la formule suivante à tous les octets de la ligne :

$$Paeth_n(x) = Raw_n(x) - PaethPredictor(Raw_n(x-bpp), Raw_{n-1}(x), Raw_{n-1}(x-bpp))$$

où x est la position du pixel à traiter (de 0 à largeur -1), et bpp le nombre d'octets par pixel tel que défini pour le filtre 1.

Ce calcul est fait octet par octet, indépendamment de la profondeur bit. L'arithmétique utilisée est non signée modulo 256, de sorte que l'entrée et la sortie restent dans le codage admissible d'un seul octet. La séquence des valeurs $Paeth_n(x)$ correspond à la ligne filtrée. Le prédicteur de Paeth est défini par le pseudo-code suivant ::

```
function PaethPredictor (a, b, c)
begin
     ; a = left, b = above, c = upper left
     p := a + b - c        ; initial estimate
     pa := abs(p - a)      ; distances to a, b, c
     pb := abs(p - b)
     pc := abs(p - c)
     ; return nearest of a,b,c,
     ; breaking ties in order a,b,c.
     if pa <= pb AND pa <= pc then return a
     else if pb <= pc then return b
     else return c
end
```

Les calculs dans le prédicteur de Paeth doivent être exécutés sans dépassement de capacité. L'arithmétique modulo 256 ne sera utilisée que dans l'étape finale de soustraction du prédicteur à la valeur non filtrée de l'octet.

Notez que l'ordre dans lequel les pixels prédicteurs sont testés est important et ne doit pas être altéré. L'ordre impératif est: le pixel de gauche, le pixel au dessus, et le pixel au dessus à gauche. (Cet ordre est différent de celui donné dans l'article de Paeth).

Pour tout $x < 0$, on supposera que $Raw_n(2dd\ x) = 0$. Pour la première ligne (ou la première ligne d'une passe si l'image est entrelacée), on considérera la présence d'une ligne fictive $Raw_0(x) = 0$ quelque soit x. Pour inverser l'effet du filtre, on utilisera la formule suivante :

$$Raw_n(x) = Paeth_n(x) + PaethPredictor(Raw_n(x-bpp), Raw_{n-1}(x), Raw_{n-1}(x-bpp))$$

dans laquelle le résultat est calculé modulo 256, et $Raw_j(x)$ correspond à une valeur déjà extraite.

7. Règles d'ordonnancement des blocs

Dans le but de permettre l'enregistrement de nouveaux types de blocs dans le format PNG, il est nécessaire d'établir des règles d'ordonnancement pour tous les types de blocs. Autrement, un éditeur PNG pourrait ne pas savoir quoi faire d'un bloc inconnu.

Nous appellerons "Editeur PNG" un programme susceptible de modifier un fichier PNG, tout en essayant

de préserver le maximum d'informations auxiliaires qui lui sont rattachées. Deux exemples d'éditeurs PNG seraient un programme qui permet la modification des bloc textuels, et un programme qui ajoute une palette suggérée à un fichier PNG en vraies couleurs. Habituellement, des éditeurs graphiques ne seront pas forcément des éditeurs PNG car ils jettent souvent les informations auxiliaires pour ne s'intéresser qu'à l'image. (Note: nous recommandons fortement que les programmes traitant le format PNG préservent ces informations auxiliaires chaque fois que possible).

Un exemple de problème possible est le cas d'un nouveau bloc auxiliaire hypothétique, sûr à la copie, et défini comme devant apparaître après le bloc PLTE s'il existe. Si notre programme destiné à ajouter un bloc PLTE est incapable de reconnaître ce nouveau bloc, il pourrait insérer la palette PLTE au mauvais endroit, c'est à dire après le nouveau bloc. Nous pourrions prévenir ce problème en demandant à tous les éditeurs PNG d'éliminer tous les blocs non reconnus, mais cette solution est très limitative. Au lieu de cela, PNG exige qu'il n'y ait pas de restrictions de ce type sur l'ordre des blocs auxiliaires.

Pour éviter un problème de ce type, et continuer à pouvoir étendre le format de façon cohérente, nous avons introduit quelques contraintes sur les éditeurs PNG et l'ordre des blocs.

7.1. Comportement des éditeurs PNG

Les règles définies pour les éditeurs PNG sont les suivantes :

- Lorsqu'un éditeur PNG copie un bloc de type inconnu et marqué comme non sûr à la copie, il ne doit pas changer la position de ce bloc par rapport aux blocs critiques du fichier. Il pourra par contre intervertir tous les blocs auxiliaires situés entre les deux blocs critiques. (Ceci vaut tant que l'éditeur n'a pas à supprimer, ajouter ou changer l'ordre des blocs critiques et permet de préserver les blocs non sûrs).
- Lorsqu'un éditeur PNG copie un bloc auxiliaire inconnu et sûr à la copie, il ne doit pas changer sa position par rapport à un bloc IDAT. (ceci est une règle intéressante car le bloc IDAT est toujours présent). Toute autre interversion peut être opérée.
- Lorsqu'un éditeur copie un bloc auxiliaire de type connu, il n'aura qu'à honorer les règles spécifiques à ce bloc. Cependant, il pourra choisir d'appliquer les règles générales énoncées ci-dessus.
- Un éditeur PNG rencontrant un bloc critique de type inconnu doit arrêter son traitement, car il n'existe aucune garantie que la modification d'un fichier contenant un tel bloc ne résulte en un fichier valide. (Notez qu'éliminer tout bonnement le bloc inconnu n'est pas non plus une solution acceptable, ceci pouvant avoir des implications graves quant à la signification d'autres blocs critiques).

Ces règles sont édictées dans une situation où le programme modifie un fichier d'entrée pour produire un fichier de sortie (notion de copie de blocs), mais sont évidemment valables lorsque la modification est faite "en place". *Cf aussi Conventions d'appellation des blocs (Section 3.3)*.

7.2. Ordre des blocs auxiliaires

Les règles pour ordonner les blocs auxiliaires ne pourront jamais être plus strictes que les suivantes :

- Les blocs auxiliaires non sûrs à la copie peuvent être contraints par rapport à des blocs critiques.
- Des blocs sûrs à la copie peuvent être contraints par rapport aux blocs IDAT.

Voir par exemple les règles d'ordonnancement des blocs auxiliaires prédéfinis (Résumé à propos des blocs prédéfinis, Section 4.3).

Les décodeurs en aucun cas supposer quoi que ce soit de plus sur la position probable d'un bloc auxiliaire. En particulier, il sera impropre de supposer qu'un bloc auxiliaire a forcément une position définie par

rapport à un autre bloc auxiliaire. (Par exemple, il est dangereux d'affirmer que votre bloc privé se situe forcément avant le dernier bloc IEND. Même si votre application l'écrit bien toujours à cette place, rien n'interdit à un éditeur PNG d'insérer un nouveau bloc auxiliaire après le vôtre. La seule chose sûre est que votre bloc est bien située entre le dernier IDAT et le bloc IEND).

7.3. Ordre des blocs critiques

Les blocs critiques peuvent avoir des contraintes arbitraires, dans la mesure où les éditeurs PNG ne pourront effectuer aucune modification si un bloc critique de nature inconnue est détecté. Par exemple, la règle spéciale pour le bloc IHDR est qu'il doit toujours apparaître en premier. Un éditeur PNG, et bien sûr tout programme écrivant des fichiers PNG, doit connaître et suivre toutes les règles associées aux blocs critiques qu'il est sensé écrire.

8. Divers

8.1. Extension de fichier

Sur des systèmes où il est habituel que l'extension ait une signification particulière, l'extension ".png" est recommandé pour les fichiers PNG. Le suffixe ".png" en minuscule sera préféré si les noms de fichiers sont sensibles à la casse.

8.2. Type MIME

L'Internet Assigned Numbers Authority (IANA) a enregistré le type MIME "image/png" comme type officiel de média des fichiers PNG pour Internet [RFC-2045, RFC- 2048]. Par principe de robustesse, les décodeurs seront enjoints de prendre en compte le type provisoire "image/x-png" utilisé avant l'enregistrement officiel du type définitif.

8.3. PNG sur Macintosh

Sur un système Apple Macintosh, les conventions suivantes sont proposées :

- Le code de type de fichier des fichiers PNG est "PNGf". (Ce code a été agréé par Apple). Le code "créateur" dépendra de l'application ayant créé le fichier.
- Le contenu du fichier "données" doit être le fichier PNG tel que défini dans cette spécification.
- Le contenu du fichier de ressources est non défini. Il peut être vide, ou contenir des informations propres à l'application créatrice.
- Lorsque des fichiers PNG doivent être transférés du monde Macintosh PNG vers le monde non-Macintosh, seule la partie données du fichier doit être transférée (Raw data).

8.4. Extension multi-image (PNG animé ?)

PNG est un format strictement statique et destiné à enregistrer une image unique. Cependant, il peut être nécessaire d'enregistrer plusieurs images dans un fichier; par exemple, pour récupérer le contenu de certains fichiers GIF. Dans le futur, un format multi-image basé sur le format PNG sera défini. Mais il s'agira d'un format à part entière qui aura une autre signature. Les applications PNG seront libres de supporter ce format multi-images ou non. *Cf. Motivations :Pourquoi ne pas implémenter ces fonctions? (Section 12.3).*

8.5. Considérations en matière de sécurité

Un fichier ou un flux PNG est composé d'un ensemble de blocs explicitement nommés. Les blocs tels qu'ils sont définis jusqu'ici peuvent contenir à peu près n'importe quoi, y compris le code d'un virus. Mais il n'y a p

AP1. Appendice : Gamma

(Cet appendice ne fait pas partie de la spécification PNG).

Il serait idéal, pour les programmeurs d'applications graphiques, que tous les composants de la chaîne graphique soient linéaires. Dans une telle situation idyllique, la tension de sortie d'une caméra serait alors directement proportionnelle à l'intensité de la lumière (puissance) d'une scène, la lumière émise par un tube cathodique serait directement proportionnelle à sa tension d'entrée, etc. Malheureusement, les équipement réels ne se comportent pas de cette façon. Tous les écrans, presque tous les films photographiques, et de nombreuses caméras ont des caractéristiques intensité/tension ou tension/intensité non linéaires.

Malgré tout, tous ces éléments non linéaires ont néanmoins des fonctions de transfert qui peuvent être modélisées par une fonction mathématique assez simple: une fonction puissance. Cette fonction est d'équation générale :

```
output = input ^ gamma
```

où ^ symbolise la mise à la puissance, et "gamma" (souvent symbolisé par le symbole grec de même nom) est tout simplement l'exposant.

Par convention, l'entrée "input" et la sortie "output" sont toutes deux calibrées sur l'intervalle 0..1, 0 représentant le noir et 1 représentant l'intensité de lumière blanche maximale (ou rouge, ou verte, ou bleu si l'on ne considère qu'une composante primaire). Ainsi normalisée, cette fonction puissance est décrite complètement par un seul nombre : l'exposant "gamma".

Ainsi, pour un élément particulier de la chaîne, nous pouvons mesurer la sortie comme une fonction de son entrée, assimiler cette fonction à une fonction théorique de puissance, extraire l'exposant, et l'appeler gamma. Nous dirons alors "cet élément a un gamma de 2,5" comme raccourci de "cet appareil a une réponse selon une puissance d'exposant 2.5". Gamma peut être aussi vu comme une transformée mathématique, ou comme une table de transfert agissant sur le tampon graphique d'affichage, pour autant que l'entrée et la sortie de l'objet visé réponde à une fonction de type "puissance".

Comment les gamma se combinent ?

Une chaîne graphique réelle est composée de nombreux éléments, dont un certain nombre peuvent être non linéaires en réponse. Si tous les composants avaient une fonction de transfert de type puissance, alors la fonction de transfert de l'ensemble de la chaîne serait une fonction de type puissance. L'exposant (gamma) de la chaîne est alors le produit de tous les exposants (gammas) de chaque élément séparé du système.

Dans ce contexte, les éléments linéaires s'intègrent sans problème, dans ce sens que l'exposant d'un système linéaire est 1,0 , cas particulier de notre fonction de puissance.

Ainsi, tant que notre système n'est composé que d'éléments linéaires, ou non linéaires à loi de type puissance, alors il est raisonnable de parler de gamma du système, obtenu par produit de tous les gamma de chaque composant.

Quelle valeur devrait prendre le gamma d'un système complet ?

Si le gamma global d'un système graphique vaut 1, sa restitution est une fonction linéaire de son "excitation" (ou sa sortie est une fonction linéaire de son entrée). C'est à dire que les rapports d'intensité entre toutes les zones d'une image sont conservées comme dans la scène originale. Il semb 1000 lerait que ce but devrait être celui de tout système graphique : celui de reproduire avec exactitude la luminosité. Hélas, ce n'est pas le cas.

Lorsque l'image à reproduire est visualisée dans un environnement "clair", où les autres objets de la pièce ont à peu près la même luminosité que le blanc de l'image, alors, un gamma global de 1 est satisfaisant pour reproduire l'image avec une apparence satisfaisante. Des impressions de qualité photographique visualisées sous une bonne lumière ainsi que des écrans d'ordinateurs dans une pièce claire sont des cas typiques "d'environnements clairs".

Parfois, des images sont destinées à être vues en environnement "sombre", dans lesquels seule l'image est lumineuse. On trouve les films ou présentations (sur transparents) par projection dans ce cas de figure. Dans certaines circonstances, une reproduction exacte de l'original produit une sensation subjective de "platitude" et de manque de contraste. Il apparaît que l'image projetée nécessite un gamma d'environ 1,5 par apport à la source originale pour être perçue comme naturelle. Ainsi, les films pour transparents sont conçus pour présenter un gamma de 1,5, et non 1.

Il existe enfin une situation intermédiaire d'environnement "atténué", dans laquelle l'environnement reste toujours perceptible à l'oeil, mais plus sombre que l'image visualisée. Ceci est le cas lorsque l'on regarde la télévision, en début de soirée, ou dans des salles d'ordinateurs insuffisamment éclairées. Dasn ces conditions, le système devra présenter un gamma de 1,25 pour une perception naturelle. La nécessité d'augmenter le contraste (gamma) dans des environnements sombres découle de la façon dont la perception visuelle humaine fonctionne, et s'applique de ce fait aussi aux moniteurs informatiques. Ainsi, un client PNG désireux d'obtenir le plus de réalisme possible pour l'affichage des images doit connaître dans quelles conditions de luminosité ces images sont visualisées, et devra ajuster le gamme des images affichées en conséquence.

Si poser la question à l'utilisateur quant aux conditions d'éclairage dont il dispose se révèle inapproprié ou impossible, le client PNG devra choisir un gamma par défaut (gamma d'affichage comme défini ci-après) de 1 ou 1,25. C'est ce que la plupart des systèmes qui implémentent la correction de gamma font.

Qu'est ce que le gamma d'un écran ?

Presque tous les écrans vidéo informatiques ont une fonction de transfert en puissance affichant un gamma de 2,5. Cette valeur est la conséquence de la technologie permettant le contrôle du faisceau d'électrons dans le canon à électrons, et ne dépend pas des photophores.

Font exception à cette règle les moniteurs "de référence", disposant d'une électronique qui permet d'altérer leur fonction de transfert. Si vous disposez d'un tel moniteur, il est probable que le constructeur vous spécifiera son gamma. Dans tous les autres cas, on pourra estimer avec une certitude suffisante que le gamma vaut 2,5.

Il existe plusieurs mires de calibration qui permettent de mesurer le gamma, en général, en comparant l'intensité d'une zone contenant une séquence de noir et de blanc avec celle d'une zone contenant une suite de niveaux de gris de luminosité croissante. Ces techniques ne sont pas d'une très grande précision. Des images constituées d'un "damier" fin de blanc et de noir sont les moins efficaces, car la reproduction d'un pixel blanc paraîtra beaucoup plus sombre qu'une grande zone de blanc. Une image proposant une séquence de lignes horizontales blanches et noires (comme l'image "gamma.png" à l'adresse ftp://ftp.uu.net/graphics/png/images/suite/gamma.png) donne de bien meilleurs résultats, sauf à des très hautes définitions sur certains écrans.

Si vous disposez d'un bon photomètre, vous pouvez mesurer la lumière émise par une séquence de gris sur le moniteur en mesurant la tension d'entrée et calculer la fonction puissance qui correspond aux points de mesure. Cependant, cet 1000 te méthode reste très sensible au réglage du "noir" du moniteur, moyennement sensible à la définition, et encore l"gèrement sensible à la lumière ambiante. De plus, le verre de l'écran diffuse de la lumière depuis les zones éclairées vers les zones sombres limitrophes ; un pixel unique blanc sur un fond noir peut ainsi apparaître comma ayant un "halo". Votre technique de mesure devra faire en sorte de limiter ces effets parasites.

Comme il est effectivement assez difficile d'obtenir une mesure précise du gamma réel, que ce soit avec une image test ou par mesure photométrique, il est plus pratique d'accepter de prendre la valeur 2,5 comme estimation "suffisante".

Qu'entend-t-on par correction gamma ?

Un moniteur a un gamma de 2,5, et nous ne pouvons rien y changer. Pour atteindre un gamma global de 1 (ou du moins proche de 1) dans un système graphique, il sera nécessaire qu'au moins un autre composant de la "chaîne graphique" soit non linéaire. Si, dans les faits, un seul étage de la chaîne, autre que le moniteur, est non linéaire, alors il est de coutume de dire que le moniteur a un certain gamma, et que l'autre composant non linéaire effectue la "correction gamma" pour compenser la non linéarité du moniteur. Cependant, l'endroit réel où la "correction" est effectuée dépend des circonstances.

Dans tous les systèmes de télédiffusion et de vidéo, la correction gamma est effectuée par la caméra. Ce choix à été fait à l'époque où l'électronique vidéo était entièrement analogique, et un bon circuit de correction gamma était très coûteux. Le standard vidéo NTSC original imposait aux caméras un gamma de 1/2,2 (à peu près 0,45). Récemment, une fonction de transfert plus complexe à deux zones a été adoptée [SMPTE-170M], mais son comportement peut être assimilé approximativement à une fonction de puissance de gamma 0,5. Lorsque l'image résultante est affichée à l'écran (disposant d'un gamma de 2,5), un gamma d'environ 1,25 par rapport à la scène originale est obtenu, qui correspond à un visionnage en environnement "atténué".

Aujourd'hui, le signal vidéo est de plus en plus souvent numérisé, et affiché dans des tampons graphiques numériques. Ceci fonctionne très bien, mais souvenez-vous que du fait que la correction gamma est "intégrée" au signal vidéo, les images numériques résultantes auront un gamma d'environ 0,5 par rapport à la scène réelle. Les logiciels de rendu d'image de synthèse produisent souvent des échantillons selon une échelle linéaire. Pour les afficher correctement, il est nécessaire que la luminosité de l'écran soit directement proportionnelle aux données enregistrées dans le tampon graphique. Ceci peut être obtenu en intercalant entre le tampon et l'électronique de l'écran une table de conversion. Cette table de conversion (souvent appelée LUT) est chargée avec une fonction de puissance de gamma 0,4, "corrigeant" ainsi la non linéarité du moniteur.

Il arrive que la correction gamma soit tantôt faite avant le tampon graphique, tantôt après. Tant que les images créées dans un environnement particulier sont affichées dans cet environnement, tout va bien. Mais dès que les images sont échangées avec d'autres personnes, les différences de conventions de correction gamma font que les images semblent trop claires ou fades (sous exposées), ou trop sombres et trop contrastées (surexposées).

Alors, faut-il mieux corriger avant ou après le tampon graphique ?

L'idéal serait que les échantillons soient enregistrés en virgule flottante. La précision serait très fine, et cette question deviendrait inutile. Mais dans la réalité, nous essayons toujours d'enregistrer une image avec le minimum de bits possible.

Si nous décidions d'utiliser un modèle d'échantillons linéaires, et effectuer la correction gamma au niveau de la table de conversion LUT du tampon graphique, alors la profondeur nécessaire pour chaque canal

primaire rouge, vert et bleu se révèle être de 12 bits pour obtenir une précision suffisante d'intensités. A moins que cela, on pourra v 1000 oir apparaître des "bandes de contour" dans les parties sombres de l'image, lorsque deux échantillons adjacents ont une valeur d'intensité encore trop différente, mais pas assez pour être visible.

Cependant, par une coïncidence heureuse, la perception visuelle humaine de la luminosité répond à une loi physique ressemblant très fortement à la fonction de puissance utilisée pour la correction gamma. Si nous appliquons la correction gamma sur les échantillons mesurés (ou calculés par un logiciel d'imagerie) avant leur quantification en entiers, nous aurons besoin de moins de bits pour enregistrer l'image. En réalité, 8 bits par couleur primaire sera en général suffisant pour éviter les artefacts de contour. Ceci est dû au fait que, comme la correction gamma est très proche de la perception visuelle, nous codons dans ce cas une variation linéaire de la perception d'intensité lumineuse sous forme d'une variation non linéaire d'excitation électrique du tube.

Comparé à un codage linéaire, nous associons moins d'étendue aux teintes claires de l'image et plus d'étendue aux teintes sombres.

Ainsi, tout en obtenant une image de qualité visuelle équivalente, des images enregistrées avec une correction gamma préalable utilisent un tiers de moins de place que des images codées en linéaire.

Généralisation de la gestion de gamma

Lorsque plus de deux éléments d'une chaîne graphique ont un comportement non linéaire, le terme "correction gamma" devient trop vague. Si nous considérons un système qui capture (ou calcule) une image, la convertit et l'enregistre dans un fichier, lit et décode le fichier, et enfin affiche l'image sur un écran, on trouve 5 endroits dans la chaîne qui peuvent présenter une fonction de transfert non linéaire. Nous allons donner un nom spécifique pour chacune de leur caractéristique gamma:

camera_gamma
Les caractéristiques du capteur d'acquisition

encoding_gamma
Le gamma de toute transformation effectuée par le logiciel lors de l'enregistrement du fichier

decoding_gamma
Le gamma de toute transformation effectuée par le logiciel lors de la lecture du fichier

LUT_gamma
Le gamma du tampon d'image LUT, s'il existe

CRT_gamma
Le gamma du moniteur, en général 2,5

Nous y ajouterons quelques définitions supplémentaires:

file_gamma
Le gamma résultant de l'image dans le fichier, par rapport à la scène originale. Il vaut

```
file_gamma = camera_gamma * encoding_gamma
```

display_gamma
Le gamma du "système d'affichage" complet. Il s'agit de

```
display_gamma = LUT_gamma * CRT_gamma
```

viewing_gamma
Le gamma global que nous souhaitons obtenir pour un affichage agréable des images, soit 1,0 ou 1,5.

La valeur *file_gamma*, telle que définie ci-dessus, est celle qui est notée dans le bloc gAMA d'un fichier PNG. Si *file_gamma* est différent de 1, nous pourrons détecter qu'une correction gamma a été faite avant d'enregistrer le fichier, et nous dirons que nous travaillons avec des "échantillons corrigés". Cependant, comme la chaîne graphique peut associer plusieurs gamma distincts, certains d'entre eux étant inconnus lorsque le fichier est écrit, la "correction" que nous détectons ne correspond pas forcément à une condition d'affichage identifiée. Comme nous utilisons une fonction de puissance lorsque nous convertissons les échantillons en entiers, qui correspond en tout point à une fonction gamma, il est plus correct de dire que les échantillons du fichier PNG sont "encodés gamma" plutôt que "corrigés". Lors de l'affichage d'une image d'un fichier, le programme qui décode l'image porte la responsabilité de restituer un gamma final égal au *viewing_gamma*, en choisissant une valeur appropriée pour le *decoding_gamma*. L b56 orsque nous décoderons un fichier PNG, la bloc gAMA nous donnera une valeur pour le *file_gamma*. Le *display_gamma* doit pouvoir être connu pour la machine concernée, ou peut être retrouvé dans les paramètres ou informations système (en dernier recours, il sera toujours possible de demander à l'utilisateur de rentrer une valeur de gamma). Le *viewing_gamma* à obtenir dépendra aussi des conditions d'éclairage environnant, information que nous ne pourrons généralement obtenir que de l'utilisateur.

En fin de chaîne, la formule finale donne

```
file_gamma * decoding_gamma * display_gamma = viewing_gamma
```

Quelques exemples spécifiques

Dans les systèmes vidéo numériques, le *camera_gamma* vaut à peu près 0,5 selon ce qui est spécifié dans la majorité des standards déclarés. Le *CRT_gamma* vaut 2,5 comme toujours, tandis que les *encoding_gamma*, *decoding_gamma*, et *LUT_gamma* valent tous 1. Le *viewing_gamma* résultant vaut alors 1,25. Sur des moniteurs graphiques de référence utilisant une table de correction au niveau du tampon graphique, et de ce fait calibrés pour effectuer un affichage linéaire, le *display_gamma* vaut 1. De nombreuses station de travail, les terminaux X et les écrans PC ne disposent pas de tables de correction gamma. Le *LUT_gamma* est donc toujours de 1, et *display_gamma* vaut 2,5. Les écrans Macintosh disposent d'une LUT. Par défaut, elle est chargée avec une table donnant une fonction gamma de 0,72, et donc un *display_gamma* (LUT et moniteur combinés) d'environ 1,8. Certains Macs ont un "tableau de bord Gamma" qui permet de redéfinir le *display_gamma* à 1,0, 1,2, 1,4, 1,8, ou 2,2. Ces réglages se font en chargeant des tables précalculées. Sur ces machines, la valeur de gamma mentionnée dans le tableau de bord "Gamma" peut être utilisée en lieu et place du *display_gamma* dans les calculs de la chaîne. Certains systèmes SGI récents (Silicon Graphics) disposent d'une correction gamma matérielle par une table dont le contenu est contrôle par le programme (privileged) "gamma". Le gamma mentionné par ce programme est l'inverse du *LUT_gamma*. Pour obtenir le *display_gamma*, vous devrez relever la valeur du *SGI_system_gamma* (soit en lisant dans le bon fichier, ou en le demandant à l'utilisateur) puis calculer

```
display_gamma = 2.5 / SGI_system_gamma
```

Vous trouverez des systèmes SGI de *system_gamma* 1.0 et 2.2 (ou supérieur), la valeur par défaut en sortie d'usine étant 1,7.

AP2. Appendice : Introduction aux couleurs

(Cet appendice ne fait pas partie de la spécification PNG).

A propos de chromatisme (ou chromie)

Le bloc cHRM est utilisé en combinaison avec le bloc gAMA pour donner une information précise des couleurs à utiliser dans le rendu d'une image PNG, que ce soit) l'écran ou à l'impression. Les chapitres précédents ont traité de la façon dont ces informations sont codées dans une image PNG. Ce chapitre résume brièvement la théorie colorimétrique sous-jacente pour ceux qui n'en auraient aucune notion.

Notez qu'une erreur de gamma produit un décalage chromatique nettement supérieur qu'une erreur d'interprétation des données de chromie. C'est pourquoi nous vous demandons de bien vérifier votre chaîne gamma avant de commencer à parler de chromie.

Le problème

La couleur d'un objet dépend non seulement du spectre lumineux émis ou réfléchi par ce dernier, mais aussi de l'observateur --- son espèce, ce qu'il peut voir au même moment, et même ce qu'il vient de voir juste avant! De plus, deux spectres totalement différents peuvent produire la même impression de couleur. La perception de couleur n'est pas une propriété objective d'objets du monde réel; elle est sensation biologique subjective. Cependant, en formulant quelques hypothèses simplificatrices (comme par exemple: nous ne considérons que la vision humaine; qui plus est non daltonienne !), il est possible de définir un modèle mathématique de la couleur d'une précision suffisante pour notre propos.

Couleur dépendante du matériel

Affichez la même combinaison RVB sur trois moniteurs différents placés côte à côte, et vous vous rendrez tout de suite compte d'une différence de teinte entre ces trois écrans. Ceci est dû au fait que chaque moniteur émet une teinte et une intensité légèrement distincte de rouge, vert, et bleu. RVB est un exemple de modèle colorimétrique dépendant du matériel --- la couleur finale dépend en effet du moniteur utilisé. Ceci veut donc dire qu'une couleur --- représentée par, disons, une combinaison RVB 87, 146, 116 sur un certain écran --- devra être codée 98, 123, 104 sur un autre écran pour produire la même couleur.

Couleur indépendante du matériel

Une description physique d'une couleur devrait se baser sur une définition exacte de la composition spectrale de la source. Fort heureusement, notre oeil et notre cerveau ne sont pas suffisamment sensibles aux variations légères de spectre. Un modèle mathématique, indépendant du matériel utilisé existe, et décrit avec suffisamment de précision la perception de couleur telle que vu par nos yeux d'humains. Un des plus importants modèles colorimétriques, duquel tous les autres seront plus ou moins dérivés, a été développé par le International Lighting Committee (CIE, en Français) et est appelé XYZ.

Dans le modèle XYZ, X est l'intégrale d'une fonction de distribution de puissance lumineuse pondérée sur l'ensemble du spectre visible. De même pour Y et Z, mais avec d'autre poids de pondération. De ce fait, toute distribution arbitraire de puissance lumineuse est réduite à seulement trois réels. Les pondérations ont été établies sur la base de nombreuses expériences réalisées sur des sujets humains dans les années 1920. Le modèle CIE XYZ est devenu un standard international depuis 1931, et dispose d'un 1000 grand nombre de propriétés intéressantes:

- Deux couleurs de mêmes coordonnées XYZ apparaissent comme identiques.
- Deux couleurs de coordonnées XYZ différentes apparaissent différentes
- La valeur Y représente à elle seule la luminosité (luminance)
- La couleur XYZ de tout objet peut être mesurée objectivement

Le modèle de couleur XYZ a été utilisé depuis de nombreuses années par tous les professionnels qui

avaient besoin d'un contrôle très précis de la couleur - directeurs dela photo de films et TV, fabriquants de peintures et de filtres colorés, etc. Il est donc éprouvé par une longue utilisation industrielle. Les notions de précision, d'indépendance vis à vis du matériel ont commencé à se répandre dans des milieux moins "spécialisés" depuis les années 1980 et 1990, et le format PNG a pris en compte cette tendance.

Couleurs dépendantes du matériel et calibrage

Traditionnellement, les formats d'image numérique se sont appuyés sur des modèles de couleurs non calibrés et dépendant du matériel utilisé. Si l'on dispose de suffisamment d'information sur le système de restitution original, il devient alors possible de convertir cet espace colorimétrique dépendant du matériel dans un autre totalement indépendant du contexte. En faisant les simplifications suffisantes, par exemple en ne prenant que le cas des moniteurs (bien plus faciles à traiter que celui des imprimantes), tout ce que nous avons besoin de connaître est les différentes coordonnées XYZ de chaque couleur primaire de l'écran, ainsi que le *CRT_gamma*.

Alors pourquoi le format PNG n'enregistre pas les images directement au format XYZ ? Il y a deux raisons à cela. Tout d'abord, l'enregistrement en coordonnées XYZ nécessite plus de bits pour la même précision, ce qui aurait tendance à "enfler" les fichiers. Deuxièmement, tous les programmes seraient alors obligés de convertir les images avant de pouvoir les visualiser sur un écran. Calibrées ou non, toutes les variantes du modèle RVB sont suffisamment proches l'une par rapport à l'autre pour que des programmes de visualisation bas de gamme puissent se permettre de ne pas faire de correction chromatique avant l'affichage. En enregistrant du RVB calibré, le format PNG reste compatible avec tous les programmes existants acceptant des données RVB, et transporte en plus suffisamment d'informations pour effectuer une conversion XYZ à destination d'applications qui nécessitent un modèle colorimétrique plus précis. Nous gardons donc les avantages de chaque système.

Qu'est-ce que le chrominance et la luminance ?

La chrominance est une mesure objective de la couleur d'un objet, hormis toute considération de luminosité. La chrominance s'appuie sur deux paramètres x et y, et qui peuvent être calculés à partir des coordonnées XYZ:

```
x = X / (X + Y + Z)
y = Y / (X + Y + Z)
```

Des couleurs XYZ de même chrominance apparaîtront comme ayant la même "teinte" mais peuvent varier en luminosité. Notez que les coordonnées x et y sont sans dimension, et garderont la même valeur quelles que soient les dimensions de X,Y et Z.

La coordonnée Y du modèle XYZ est directement proportionnelle à la luminosité absolue et est appelée luminance de la couleur. Une couleur peut donc être décrite sous forme de coordonnées XYZ ou par une chrominance x,y associée à une luminance Y. Le modèle XYZ présente l'avantage d'être lié à l'espace RVB par une transformée linéaire.

Comment sont décrites les couleurs d'un moniteur ?

Le "point blanc" d'un moniteur est la chrominance x,y du blanc nominal du moniteur (rougr = vert = bleu = maximum).

Il est d'usage de spécifier un moniteur en donnant la chrominance de chacun de ses photophores R, V, et B, plus celui du point blanc. Le point blanc permet de déduire la luminosité relative des trois photophores, qui n'est pas déterminée par leur seule chrominance.

Notez que la luminosité absolue d'un 1000 moniteur n'est pas spécifiée. En infographie, il est peu courant de se référer à des valeurs absolues de luminosité. Au lieu de manipuler des coordonnées XYZ absolues (par lesquelles X, Y et Z seraient exprimées en unités physiques de puissance émise, par exemple des candelas par mètre carré), il sera plus pratique de considérer des "XYZ" relatifs, calibrées sur la luminance nominale du point blanc du moniteur (Y = 1,0). Avec cette simplification, il devient simple de calculer les coordonnées XYZ de chaque couleur primaire rouge, verte et bleue du moniteur à partir des valeurs de chrominance.

Pourquoi le bloc cHRM utilise le couple x,y plutôt que les coordonnées XYZ ?

Tout simplement parce que ce sont les valeurs qui sont données par les constructeurs dans leurs spécifications ! Ainsi, la première opération que fera un programme sera de convertir les informations de chrominance du bloc cHRM pour obtenir un espace XYZ relatif.

Que vais-je en faire ?

Si un fichier PNG dispose des blocs gAMA et cHRM, les valeurs RVB_source peuvent être converties dans l'espace XYZ. Ceci permet :

- une conversion précise en niveaux de gris (on utilise que la coordonnée Y)
- une conversion RVB pour votre propre moniteur (pour voir les couleurs originales)
- une impression de l'image en PostScript niveau 2 avec une bien meilleure fidélité que celle obtenue par simple conversion RVB vers CMJN
- le calcul d'une palette optimale
- le transfert des données d'image à un système de gestion de couleur
- etc.

Comment convertir du RVB_source en XYZ ?

Faisons tout d'abord quelques suppositions. Supposons d'abord que le moniteur est parfaitement noir en absence d'excitation. Supposons ensuite que les canons à électrons n'interfèrent pas. En connaissant les valeurs CIE XYZ pour chacune des trois couleurs primaires rouge, vert et bleu du moniteur, placez les dans une matrice m:

$$m = \begin{matrix} Xr & Xg & Xb \\ Yr & Yg & Yb \\ Zr & Zg & Zb \end{matrix}$$

Nous supposerons de plus que nous travaillons avec des échantillons RVB à virgule flottante codés par rapport à une échelle normalisée sur un intervalle 0..1. Si le gamma est différent de 1,0, on convertira les échantillons. On peut à ce moment obtenir les coordonnées XYZ par multiplication de matrice:

$$\begin{matrix} X \\ Y \\ Z \end{matrix} = m \begin{matrix} R \\ G \\ B \end{matrix}$$

En d'autres termes, `X = Xr*R + Xg*G + Xb*B`, de même pour Y et Z. L'opération inverse est bien sûr valide :

$$\begin{matrix} R \\ G \\ B \end{matrix} = m^{-1} \begin{matrix} X \\ Y \\ Z \end{matrix}$$

Qu'est-ce qu'un gamut ?

Le gamut d'un matériel est le sous ensemble de couleurs visibles que l'appareil peut reproduire. (aucun rapport avec le gamma). Le gamut d'un appareil RVB peut être visualisé sous forme d'un polyèdre dans l'espace XYZ; Les arrêtes correspondent aux niveaux noir, bleu, rouge, vert, magenta, cyans, jaune et blanc du moniteur.

Des appareils différents ont des gamuts différents, en d'autres termes, tel appareil sera capable de reproduire tel ensemble de couleurs (en général plus saturées) alors qu'un tel ne le pourra pas. Le gamut d'un appareil RVB particulier peut être déterminé à partir de ses caractéristiques chromatiques R, V, et B ainsi que du point blanc (les valeurs données dans le bloc cHRM). Le gamut d'une imprimante couleur est plus compliqué et ne peut être souvent que mesuré. Ce que l'on peut dire, c'est que les gamut d'imprimantes sont toujours plus réduits que celui d'un écran, et ce qui explique que de nombreuses couleurs affichables à l'écran ne sont pas imprimables.

La restitution d'une image créée sur un matériel, sur un autre matériel se heurte souvent à une différence de gamuts - des couleurs ne pouvant être correctement représentées sur le matériel destinataire. Le procédé qui permet de faire 29e coïncider les couleurs, et qui va d'un simple écrêtage jusqu'à des transformées non linéaires élaborées, est appelé "gamut mapping" ou conversion de gamut. L'objectif est de pouvoir offrir une restitution raisonnablement proche de l'image originale.

Autres lectures

Les références bibliographiques [COLOR-1] jusqu'à [COLOR-5] donnent des détails supplémentaires sur la théorie des couleurs.

AP3. Appendice : Code source pour le CRC

Le code source qui suit est une implémentation pratique du calcul de CRC (Cyclic Redundancy Check) utilisé dans les blocs PNG. (Voir aussi ISO 3309 [ISO-3309] ou ITU-T V.42 [ITU-V42] pour une spécification formelle).

Le source est écrit en C ANSI. Les notations suivantes sont explicitées à destination des personnes peu familières avec le C:

&
Opérateur ET binaire (bit à bit).

^
Opérateur OU EXCLUSIF binaire. (Attention: dans les autres parties de ce document, ^ représente la mise à la puissance.)

>>
Décalage bit à droite. Lorsqu'utilisé avec des valeurs non signées, des 0 sont insérés par la gauche.

!
NON Logique.

++
"n++" incrémente la variable n d'une unité.

0xNNN
0x introduit la notation d'une constante hexadécimale (base 16). Le suffixe L indique une notation en tant

que "long" (au moins 32 bits).

```
/* Table of CRCs of all 8-bit messages. */
unsigned long crc_table[256];

/* Flag: has the table been computed? Initially false. */
int crc_table_computed = 0;
 /* Make the table for a fast CRC. */
void make_crc_table(void)
{
  unsigned long c;
  int n, k;

  for (n = 0; n < 256; n++) {
    c = (unsigned long) n;
    for (k = 0; k < 8; k++) {
      if (c & 1)
        c = 0xedb88320L ^ (c >> 1);
      else
        c = c >> 1;
    }
    crc_table[n] = c;
  }
  crc_table_computed = 1;
}

/* Update a running CRC with the bytes buf[0..len-1]--the CRC
```

AP4. Bibliographie

[COLOR-1]
Hall, Roy, Illumination and Color in Computer Generated Imagery. Springer-Verlag, New York, 1989. ISBN 0-387-96774-5.

[COLOR-2]
Kasson, J., and W. Plouffe, "An Analysis of Selected Computer Interchange Color Spaces", ACM Transactions on Graphics, vol 11 no 4 (1992), pp 373-405.

[COLOR-3]
Lilley, C., F. Lin, W.T. Hewitt, and T.L.J. Howard, Colour in Computer Graphics. CVCP, Sheffield, 1993. ISBN 1-85889-022-5. Also available from

[COLOR-4]
Stone, M.C., W.B. Cowan, and J.C. Beatty, "Color gamut mapping and the printing of digital images", ACM Transactions on Graphics, vol 7 no 3 (1988), pp 249-292.

[COLOR-5]
Travis, David, Effective Color Displays --- Theory and Practice. Academic Press, London, 1991. ISBN 0-12-697690-2.

[GAMMA-FAQ]
Poynton, C., "Gamma FAQ".

[ISO-3309]
International Organization for Standardization, "Information Processing Systems --- Data Communication High-Level Data Link Control Procedure --- Frame Structure", IS 3309, October 1984, 3rd Edition.

[ISO-8859]
International Organization for Standardization, "Information Processing --- 8-bit Single-Byte Coded

Graphic Character Sets --- Part 1: Latin Alphabet No. 1", IS 8859-1, 1987. Also see sample files at ftp://ftp.uu.net/graphics/png/documents/iso_8859-1.*

[ITU-BT709]
International Telecommunications Union, "Basic Parameter Values for the HDTV Standard for the Studio and for International Programme Exchange", ITU-R Recommendation BT.709 (formerly CCIR Rec. 709), 1990.

[ITU-V42]
International Telecommunications Union, "Error-correcting Procedures for DCEs Using Asynchronous-to-Synchronous Conversion", ITU-T Recommendation V.42, 1994, Rev. 1.

[PAETH]
Paeth, A.W., "Image File Compression Made Easy", in Graphics Gems II, James Arvo, editor. Academic Press, San Diego, 1991. ISBN 0-12-064480-0.

[POSTSCRIPT]
Adobe Systems Incorporated, PostScript Language Reference Manual, 2nd edition. Addison-Wesley, Reading, 1990. ISBN 0-201-18127-4.

[PNG-EXTENSIONS]
PNG Group, "PNG Special-Purpose Public Chunks". Available in several formats from ftp://ftp.uu.net/graphics/png/documents/pngextensions.*

[RFC-1123]
Braden, R., Editor, "Requirements for Internet Hosts --- Application and Support", STD 3, RFC 1123, USC/Information Sciences Institute, October 1989.

[RFC-2045]
Freed, N., and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, Innosoft, First Virtual, November 1996.

[RFC-2048]
Freed, N., Klensin, J., and J. Postel, "Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures", RFC 2048, Innosoft, MCI, USC/Information Sciences Institute, November 1996.

[RFC-1950]
Deutsch, P. and J-L. Gailly, "ZLIB Compressed Data Format Spec 2a5 ification version 3.3", RFC 1950, Aladdin Enterprises, May 1996.

[RFC-1951]
Deutsch, P., "DEFLATE Compressed Data Format Specification version 1.3", RFC 1951, Aladdin Enterprises, May 1996.

[SMPTE-170M]
Society of Motion Picture and Television Engineers, "Television --- Composite Analog Video Signal --- NTSC for Studio Applications", SMPTE-170M, 1994.

AP5. Glossaire

a^b
Exponentiation; a élevé à la puissance b. Les programmeurs C devront éviter la confusion avec l'opérateur ou exclusif du C. Notez que dans les calculs de gamma, zéro élevé à n'importe quelle puissance est valide et doit donner zéro.

Alpha
Une valeur codant le degré de transparence d'un pixel. Plus un pixel est transparent, moins il cache le fond sur lequel est présentée l'image. Dans le format PNG, l'alpha est en fait le degré d'opacité : zéro alpha représente un pixel complètement transparent, une valeur alpha maximale représente un picel complètement opaque. Cependant, le canal alpha est dans le langage courant associé à la transparence, et non à l'opacité, et nous suivront cette habitude ici.

Bloc auxiliaire
Un bloc renfermant une information additionnelle, mais non nécessaire pour l'affichage correct de l'image.

Profondeur bit
Le nombre de bits par canal de couleur primaire destiné à coder la couleur d'une entrée de palette (PNG en couleurs indexées) ou d'un échantillon (PNG en couleurs vraies). Cette valeur apparaît dans l'IHDR.

Bloc
Une section d'un fichier PNG. Chaque bloc est identifié par un type déduit de son nom. La plupart des types prédéfinis incluent une partie données. Le format et la signification des données sont déterminés par le nom du type de bloc associé au bloc.

Bloc critique
Un bloc dont les données sont essentielles à la restitution correcte de l'image contenu dans le fichier PNG.

Canal
L'ensemble de tous les octets codant une grandeur similaire dans une image; par exemple, tous les échantillons codant l'intensité de la couleur primaire bleue (Le terme "composante" est aussi utilisé, mais pas dans cette spécification). Un échantillon est l'intersection entre un canal et un pixel.

Chrominance
Une paire x,y de coordonées qui définit précisément la teinte d'un pixel dans un espace chromatique donné, mais ne donne en rien la luminosité de ce pixel, ni une couleur "perçue".

Compression non destructive
Toute technique de compression qui permet de reconstituer les données originales exactement, bit par bit.

Compression destructive
Toute méthode de compression qui ne permet la reconstitution de l'original qu'approximativement.

Couleurs indexées
Une représentation graphique qui attribue à chaque pixel une couleur prise dans une table. La valeur du pixel est alors l'index d'entrée dans la table, encore appelée "palette de couleurs", ou simplement "palette".

CRC
Cyclic Redundancy Check. Un CRC est une valeur calculée sur les données dont le but est de détecter le plus d'erreurs de transmission ou de lecture. Un décodeur calcule le CRC des données reçues et les compares à la valeur du CRC que l'encodeur a transmise, à la fin des données. Une différence de valeur entre les CRC indique la présence d'erreurs de transmission.

CRT
Cathode Ray Tube: Ecran cathodique.

Déflation
Le nom de l'algorithme de compression utilisé en standard dans les fichiers PNG, comme dans les fichiers zip, gzip, pkzip, et autres produits de programmes de compression. La déflation est une méthode de la famille d'algorithm 1000 es de compression LZ77.

Echantillon
Un entier unique codant un paramètre d'un pixel; par exemple, la luminosité de la couleur primaire rouge pour un pixel. Un pixel est défini par un ou plusieurs échantillons. Lorsque nous avons présenté l'organisation physique des données (en particulier, dans la section 2.3), nous avons utilisé le terme "échantillon" pour désigner un nombre enregistré dans le tableau de valeurs composant l'image. Il eût été plus exact, mais moins compréhensible de dire "échantillon, ou index de palette" dans ce contexte. En tout autre endroit de la spécification, "échantillon" désigne bien une valeur de couleur primaire ou de transparence (alpha). Dans le cas du modèle en couleur indexées, l'échantillon est une composante de la couleur enregistrée dans la palette, et non l'index de palette inscrit pour le pixel.

Editeur PNG
Un programme qui modifie un fichier PNG et préserve les informations auxiliaires, y compris les blocs qu'il ne peut pas reconnaître. Un tel programme doit respecter les règles d'ordonnancement des blocs (Chapitre 7).

Filtre
Une transformation opérée sur une image dans le but d'optimiser la compression. PNG n'exploite que des filtres bijectifs (réversibles).

Flux
Une séquence d'octets. Ce terme est préféré au mot fichier car plus général. Un flux est une séquence qui peut constituer un fichier, être prélevée sur un câble, en un point d'un réseau, etc...

Gamma
Mesure la luminosité de la couleur médiane. Plus précisément, une valeur qui mesure la linéarité d'un élément de la chaîne graphique. La fonction de transfert de cet élément est alors :

```
output = input ^ gamma
```

dans laquelle l'entrée input et la sortie output sont calibrées entre 0 et 1.

Ligne
Une ligne horizontale de pixels dans une image.

LSB
Octet contenant les poids faibles d'une grandeur codée sur plusieurs octets.

Luminance
Luminosité perçue, ou niveau de gris, d'une couleur. La combinaison luminance et chromatisme définit une couleur perçue.

LUT
Look Up Table, ou table de transfert. En général, il s'agit d'une table permettant la conversion rapide de données. Dans une carte graphique, la LUT peut être utilisée pour transcoder une entrée d'index en une couleur vraie, ou pour effectuer une correction de gamma par table. D'un point de vue logiciel, une LUT est une méthode rapide pour implémenter une fonction mathématique non triviale.

MSB
Octet contenant les poids forts d'une grandeur codée sur plusieurs octets.

Niveaux de gris
Une représentation graphique dans laquelle chaque pixel est codé par une valeur unique qui indique le niveau global de luminosité du pixel (sur une échelle allant du noir au blanc). Le format PNG permet de plus l'association d'une valeur alpha (transparence) pour chaque valeur de luminosité.

Octet

Huit bits.

Palette
Le sous ensemble de couleurs disponibles a un moment donnée pour afficher une image. Dans le format PNG, la palette est constituée d'un ensemble de triolets de valeur codant les composantes rouge, verte et bleue d'une entrée de palette. (des valeurs alpha peuvent compléter cette définition, via le bloc tRNS).

Pixel
L'information nécessaire à l'affichage d'un point unique de l'image. L'image complète est un tableau rectangle de pixels.

Point blanc
Le chromatisme nominal pour l'affichage d'un point blanc à l'écran.

Profondeur bit
La précision, en bits, avec l

AP6. Crédits

Editeur

Thomas Boutell, boutell@boutell.com

Co-Editeur

Tom Lane, tgl@sss.pgh.pa.us

Auteurs

Noms par ordre alphabétique.

- *Mark Adler, madler@alumni.caltech.edu*
- *Thomas Boutell, boutell@boutell.com*
- *Christian Brunschen, cb@df.lth.se*
- *Adam M. Costello, amc@cs.berkeley.edu*
- *Lee Daniel Crocker, lee@piclab.com*
- *Andreas Dilger, adilger@enel.ucalgary.ca*
- *Oliver Fromme, fromme@rz.tu-clausthal.de*
- *Jean-loup Gailly, gzip@prep.ai.mit.edu*
- *Chris Herborth, chrish@qnx.com*
- *Alex Jakulin, Aleks.Jakulin@snet.fri.uni-lj.si*
- *Neal Kettler, kettler@cs.colostate.edu*
- *Tom Lane, tgl@sss.pgh.pa.us*
- *Alexander Lehmann, alex@hal.rhein-main.de*
- *Chris Lilley, chris@w3.org*
- *Dave Martindale, davem@cs.ubc.ca*
- *Owen Mortensen, 104707.650@compuserve.com*
- *Keith S. Pickens, ksp@swri.edu*
- *Robert P. Poole, lionboy@primenet.com*
- *Glenn Randers-Pehrson, glennrp@arl.mil or randeg@alumni.rpi.edu*
- *Greg Roelofs, newt@pobox.com*
- *Willem van Schaik, willem@gintic.gov.sg*

- *Guy Schalnat*
- *Paul Schmidt, pschmidt@photodex.com*
- *Tim Wegner, 71320.675@compuserve.com*
- *Jeremy Wohl, jeremyw@anders.com*

Traducteur

Votre serviteur Valéry G Frémaux, Valery.Fremaux@eisti.fr

"The authors wish to acknowledge the contributions of the Portable Network Graphics mailing list, the readers of comp.graphics, and the members of the World Wide Web Consortium (W3C). The Adam7 interlacing scheme is not patented and it is not the intention of the originator of that scheme to patent it. The scheme may be freely used by all PNG implementations. The name "Adam7" may be freely used to describe interlace method 1 of the PNG specification."