

Le format BMP

Une image numérique matricielle est constituée de **pixels** (picture elements). Le pixel est la plus petite surface homogène d'une image, il est en quelque sorte l'équivalent d'un point en imprimerie.

Une image numérique est définie par :

- sa **résolution** : le nombre de pixels qui la composent en largeur et en hauteur.

Plus on a de pixels par unité de distance meilleure est la qualité. Cette information est généralement mesurée en nombre de pixels par pouce ou ppp. En anglais on nomme cette mesure dot per inch ou dpi. Un pouce équivaut à peu près à 2,54 centimètres.

- sa **profondeur** : le nombre de couleurs que peut prendre un pixel.

Pour définir la couleur d'un pixel, il est courant d'utiliser le codage RGB, qui définit la quantité de **rouge**, de **vert** et de **bleu** nécessaire à la composition de la couleur du pixel. Généralement on utilise un octet par composante. Dans un tel cas la profondeur est donc de $2^{24} = 16\,777\,216$ couleurs différentes¹. On réduit souvent cette profondeur au nombre de bits utilisé pour coder un pixel, donc à 24 dans notre exemple.

Avec ce codage le noir est représenté par `0x000000` et le blanc par `0xffffffff`.

- sa **matrice** : les lignes de pixels qui composent effectivement l'image.

Une telle image numérique est généralement stockée dans un fichier dont la structure est un codage particulier de la matrice de pixels.

Le format BMP a été créé par IBM et Microsoft pour structurer des fichiers contenant des images numériques matricielles. C'est un format binaire simple qui est organisé sous la forme suivante :

1. une entête de fichier sur **14** octets
2. une entête d'image sur **40** octets
3. une palette d'image (facultatif et non utilisé dans ce TP) sur **4** octets
4. le codage de la matrice de pixels sur une taille variable

Tous les nombres sont stockés en little-endian.

Entête de fichier

La partie entête de fichier permet de stocker des informations sur le fichier lui-même. Elle fournit notamment les informations sur la taille du fichier et la position exacte du début des données de l'image (la matrice elle-même) dans le fichier. En détail elle se compose de quatre champs :

1. la signature (magic number) sur **2** octets
Les lettres `BM`, soit `0x42 0x4d`, correspond, par exemple, au format des images bitmap de Windows
2. la taille totale du fichier sur **4** octets
Un entier
3. un champ réservé sur **4** octets
4. l'offset sur **4** octets
Il s'agit du décalage (l'adresse relative) du début des informations par rapport au début du fichier

Entête d'image

La partie entête de l'image fournit des informations sur l'image, notamment ses dimensions et ses couleurs. Elle est composée de onze champs :

1. la taille de l'entête de l'image sur **4** octets
Pour le format BMP windows la taille est de `0x28`.

1. Il est généralement accepté que l'œil humain ne peut pas distinguer plus de couleurs.

2. la largeur de l'image sur **4** octets
C'est le nombre de pixels sur l'axe horizontal.
3. la hauteur de l'image sur **4** octets
C'est le nombre de pixels sur l'axe vertical.
4. le nombre de plans sur **2** octets
Ce champ vaut toujours 1.
5. la profondeur de l'image sur **2** octets
Les profondeurs possibles sont 1, 4, 8, 16, 24 ou 32.
6. la méthode de compression sur **4** octets
Les images non compressées ont pour valeur de ce champ 0.
7. la taille totale de l'image sur **4** octets
La taille est exprimée en octets.
8. la résolution horizontale sur **4** octets
La résolution est spécifiée en pixels par mètre.
9. la résolution verticale sur **4** octets
La résolution est spécifiée en pixels par mètre.
10. le nombre de couleurs de la palette sur **4** octets
11. le nombre de couleurs importantes de la palette sur **4** octets
Si toutes les couleurs sont importantes ce champ vaut 0.

Palette d'image

Cette partie est optionnelle. Quand elle est présente, elle contient des octets représentant la composante bleue, verte et rouge puis un octet réservé.

Code de la matrice

Le codage de l'image se fait simplement en écrivant successivement les bits correspondant à chaque pixel, ligne par ligne en commençant par le pixel en bas à gauche.

- les images en 2 couleurs utilisent 1 bit par pixel, ce qui signifie qu'un octet permet de coder 8 pixels
- les images en 16 couleurs utilisent 4 bits par pixel, ce qui signifie qu'un octet permet de coder 2 pixels
- les images en 256 couleurs utilisent 8 bits par pixel, ce qui signifie qu'un octet code chaque pixel
- les images en couleurs réelles utilisent 24 bits par pixel, ce qui signifie qu'il faut 3 octets pour coder chaque pixel, en prenant soin de respecter l'ordre de l'alternance Rouge, Vert puis Bleu (RGB).

Chaque ligne de l'image doit comporter un nombre total d'octets qui soit un multiple de 4. Si ce n'est pas le cas, la ligne doit être complétée par des 0 de manière à respecter ce critère.

Exercices

Dans ce TP vous :

- devrez travailler dans le dossier nommé R1.03/TP02 dans votre répertoire principal ;
- allez visualiser et manipuler des images représentées en binaire. Ces informations, stockées sur le disque dur de votre machine, sont accessibles dans des fichiers.

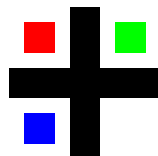
Exercice 1 : Compréhension du format BMP

Q1. Sur une feuille, dessinez 54 octets (en écrivant des lignes de 16 octets) puis, en utilisant le préambule de ce TP, remplissez ce dessin pour représenter le positionnement des différents éléments qui composent les entêtes d'une image bitmap.

Q2. À quoi correspondent les octets d'une image bitmap à partir du 55e octet ?

Exercice 2 : Manipuler le format BMP

- Q1.** Déplacez-vous dans votre répertoire `~/R1.03/TP02` et copiez-y le fichier `/home/public/baste/R1.03/TP02/image.bmp`.
- Q2.** Utilisez la commande `stat` pour obtenir la taille du fichier `image.bmp`.
- Q3.** Convertissez cette taille en hexadécimal.
- Q4.** Quelle commande `od` permet de n'afficher que l'entête de fichier ?
- Q5.** En utilisant cette commande, déterminez :
— la signature du fichier
— la taille du fichier
— l'offset de l'image
- Q6.** Vérifiez que la taille que vous avez obtenu avec `stat` et la taille contenue dans le fichier soient bien identiques.
- Q7.** Quelle commande `od` permet de n'afficher que l'entête de l'image ?
- Q8.** En précisant la commande `od` utilisée à chaque fois, déterminez :
— la taille de l'entête de l'image
— la largeur de l'image
— la hauteur de l'image
— la profondeur de chaque pixel
- Q9.** Quel est le nombre d'octets utilisé pour le bourrage avec des zéros de chaque ligne ?
- Q10.** Remplissez dans un tableau les couleurs des pixels dans l'image. Utilisez le symbole X pour remplir les cases de bourrage.
- Q11.** À quelle couleur correspondent les valeurs dans le tableau ?
- Q12.** Dessinez l'image.
- Q13.** Faites vérifier votre dessin par votre enseignant.
- Q14.** En utilisant l'utilitaire `hexedit`, modifiez les octets du fichier `image.bmp` et observez après chaque changement le résultat sur l'image
— Passer le 58e octet à la valeur `0x00`.
— Passer le 59e octet à la valeur `0x00`.
— Passer le 60e octet à la valeur `0x00`.
— Passer le 61e octet à la valeur `0x00`.
Pour visualiser votre image vous pouvez, par exemple, utiliser la commande `eom -f image.bmp` qui vous l'affiche en plein écran. Il vous faudra également sans doute désactiver le lissage dans les préférences de façon à la voir nettement. Pour cela, allez dans `Edit -> Preferences` et désactivez les options `smooth images when zoomed-out` et `smooth images when zoomed-in`.
- Q15.** En jouant avec les octets 58, 59 et 60, déduisez-en comment sont stockées les couleurs.
- Q16.** En utilisant l'utilitaire `hexedit`, modifiez le fichier `image.bmp` pour que l'image en question devienne :



Exercice 3 : Chasse au trésor

- Q1.** Copiez le fichier `/home/public/baste/R1.03/TP02/secret.bmp` dans votre répertoire de travail.
Une phrase de 8 caractères ASCII a été cachée dans le fichier `secret.bmp`. Un caractère est caché dans chaque ligne, sauf pour les lignes complètement en noir.
Chaque pixel blanc contient un bit caché. Sachant qu'il y a 8 pixels blancs dans chaque ligne, l'octet composant le caractère caché peut alors être constitué en assemblant ces bits.
Le bit caché se trouve dans le bit de poids faible dans l'octet qui représente le composant rouge du pixel.
- Q2.** Quelle est la phrase cachée ?