Lycée Felix Le Dantec - Lannion

LE NUMÉRIQUE CODAGE D'UNE IMAGE MATRICIELLE

1STI2D
TD11 - 2h

Conditions :	• Travail seul ou en binôme; durée 2h
Matériel :	• PC sous Ubuntu avec les logiciels Geany, Bless, Pinta, Gimp et Libre Office Writer ou sous Windows avec les logiciels Notepad++, HxD, Paint.net et Libre Office Writer
Documents :	• le sujet du TD • le cours sur le codage des images matricielles

1. Édition d'une image noir et blanc au format PBM

Le format **PBM** (*Portable BitMap*) est l'un des plus simples pour coder une image matricielle en noir & blanc. Un fichier au format PBM est un fichier en ASCII qui se compose comme suit :

- les caractères P1, suivis d'un retour à la ligne ou d'un espace ;
- la largeur de l'image en nombre de pixels, en décimal, suivie d'un retour à la ligne ou d'un espace ;
- la hauteur de l'image en nombre de pixels, en décimal, suivie d'un retour à la ligne ou d'un espace ;
- la liste des pixels, ligne par ligne, de haut en bas et de gauche à droite (les retours à la ligne et les espaces sont ignorés dans cette partie).

Aucune ligne ne doit dépasser 70 caractères et toutes les lignes commençant par le caractère # sont des commentaires ignorés.

```
P1
# Mon premier fichier PBM : cercle
10 10
0000000000
0011111100
0110000110
0100000010
0100000010
0100000010
0100000010
0110000110
0011111100
0000000000
```

Exemple d'un fichier au format PBM

- Lancer l'éditeur Geany ou Notepad++ et taper le texte de l'exemple ci-dessus. Enregistrer le fichier avec l'extension pbm.
- Ouvrir le fichier créé avec un visualiseur d'image et vérifier le résultat obtenu.
- Créer avec Geany ou Notepad++ un fichier PBM réalisant l'image ci-contre de taille 20 x 20 pixels. Vérifier votre image avec un visualiseur d'image et faire valider par le professeur.

Validation Prof :

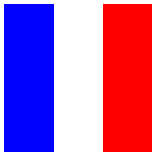
2. Édition d'une image couleur au format PPM

Le format PPM (*Portable PixMap*) ressemble beaucoup à un fichier PBM ; c'est un fichier ASCII qui se compose :

- des caractères P3, suivis d'un retour à la ligne ou d'un espace ;
- la largeur de l'image en nombre de pixels, en décimal, suivie d'un retour à la ligne ou d'un espace ;
- la hauteur de l'image en nombre de pixels, en décimal, suivie d'un retour à la ligne ou d'un espace ;
- la valeur maximale utilisée pour exprimer l'intensité des couleurs, par exemple 255 ;
- la liste des valeurs des couleurs, trois par pixel, dans l'ordre rouge, vert, bleu, ligne par ligne, de haut en bas et de gauche à droite, séparées par des retours à la ligne ou des espaces.

Comme en PBM, aucune ligne ne doit dépasser 70 caractères et toutes les lignes commençant par le caractère # sont des commentaires ignorés.

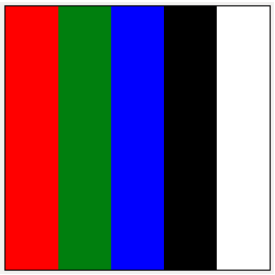
- Créer avec l'éditeur Geany ou Notepad++ un fichier PPM réalisant l'image ci-contre (drapeau bleu blanc rouge) de taille 12 x 12 pixels. Vérifier votre image avec un visualiseur d'image et faire valider par le professeur.



Validation Prof :

3. Analyse d'une image couleur au format BMP

- Lancer le logiciel éditeur d'image Pinta ou Paint.net. Créer une nouvelle image de taille 10 x 10 pixels.
- Zoomer sur votre image et créer la mire ci-dessous (rouge, vert, bleu, noir, blanc). Enregistrer votre fichier au format BMP.



- Ouvrir le fichier BMP avec l'éditeur Bless ou HxD.
- A l'aide de l'annexe sur le format BMP, détailler le contenu de l'en-tête du fichier et de l'image : vous ferez une copie d'écran de la fenêtre de l'éditeur hexadécimal sur LibreOffice Writer et rédigerez votre réponse à la suite. Vérifier la cohérence de vos résultats.
- Examiner maintenant le corps de l'image et expliquer comment est codée votre mire : nombre de pixels par bande de couleur, codage de chaque couleur.
- Modifier le fichier hexadécimal avec Bless ou HxD afin de colorier en rouge la ligne du bas de l'image. Vérifier votre image avec un visualiseur d'image et faire valider par le professeur.

Validation Prof :

4. Compression d'une image

- Télécharger l'image ***phare.bmp*** disponible à l'adresse 192.168.11.1.
- Ouvrir l'image avec **Gimp** ou Paint.net. Retrouver les informations ci-dessous (menu Image / Propriétés de l'image).

Fichier	taille du fichier (kio)	dimensions en pixels	taille en cm	résolution
phare.bmp				

- Calculer la résolution en ppp de l'image à partir des dimensions en pixels et de la taille en cm de l'image. Comparer à celle donnée par Gimp.

- Calculer la taille du fichier bitMap en RVB, comparer à la taille du fichier.

- Modifier le nombre de couleurs (menu Image / Mode / Couleurs indexées) de l'image comme dans le tableau ci-dessous . Enregistrer chaque fichier (menu Fichier / Exporter). Compléter alors le tableau.

Fichier	Nombre de pixels	Nombre maxi de couleurs	Nombre de bits par pixel	Taille du fichier en octets	Qualité perçue de l'image	Rapport taille fichier / taille minimum*
phare.bmp		16 millions				
phrare1.bmp		256				
phare2.bmp		16				
phare3.bmp		2				

** ne remplissez cette colonne qu'à la fin, après avoir repéré le plus petit des fichiers, et arrondissez le résultat.*

- Que pouvez -vous dire du lien entre les valeurs obtenues pour la dernière colonne et le nombre de bits par pixel ?

Nous allons comparer la **compression** de l'image 256 couleurs, fichier ***phare1.bmp***, en utilisant plusieurs méthodes de compression.

- Ouvrir le fichier phare1.bmp. Enregistrer ce fichier (menu Fichier / Exporter) en cochant la case Encodé en Run-Length. Remplissez les deux premières lignes du tableau ci-dessous.
- Enregistrez maintenant l'image avec le format GIF puis JPEG (choisir un % de compression de 90). Complétez la suite du tableau.

Fichier	Taille en octets	taux de compression	Qualité perçue de l'image
phare1.bmp non compressé			
phare1_rle.bmp compression rle			
phare1_lzw.gif compression lzw			
phare1_jpeg.jpg compression jpeg			

- Quel est le meilleur algorithme de compression ?

- Percevez-vous une différence de qualité entre l'image JPEG et l'original au format BMP ?

ANNEXE : le format BMP

Bitmap, connu aussi sous son abréviation *BMP*, est un format d'image matricielle ouvert développé par Microsoft et IBM. C'est un des formats d'images les plus simples à développer et à utiliser pour programmer. Il est lisible par quasiment tous les visualiseurs et éditeurs d'images.
La structure d'un fichier bitmap est la suivante :

- En-tête du fichier (en anglais *file header*)
- En-tête du bitmap (en anglais *bitmap information header*, appelé aussi *information Header*)
- Palette (optionnellement)
- Corps de l'image

Le codage Little endian

Le format Bitmap utilise le codage **Little endian** pour l'enregistrement des octets dans le fichier, c'est-à-dire qu'il écrit l'octet de poids le plus faible en premier. Ce type de codage s'appelle encore **petit-boutistes** ou **mot de poids faible en tête**.
Par exemple la séquence dans le fichier BMP « 0xA0B70708 » donne dans l'ordre : 08 07 B7 A0.

En-tête du fichier

L'entête du fichier, composée de 4 champs, fournit des informations sur le type de fichier (Bitmap), sa taille et indique où commencent les informations concernant l'image à proprement parler.

Offset#	Taille	Valeur
0x0000	2 octets	le nombre magique correspondant à l'utilisation du fichier BMP : • **BM** - Windows 3.1x, 95, NT, • **BA** - OS/2 *Bitmap Array* • **CI** - OS/2 Icône Couleur (*Color Icon*) • **CP** - OS/2 Pointeur Couleur (*Color Pointer*) • **IC** - OS/2 Icône (*Icon*) • **PT** - OS/2 Pointeur (*Pointer*)
0x0002	4 octets	la taille du fichier BMP en octets du poids faible vers le poids fort Ex : 28 4A 01 00 = $00014A28 octets = 84 520 octets
0x0006	4 octets	réservé pour l'identifiant de l'application qui a créé le fichier
0x000A	4 octets	l'*offset* (l'adresse de départ) du contenu du BMP

En-tête du bitmap

L'entête de l'image, composée de 11 champs, fournit des informations sur l'image, notamment ses dimensions et ses couleurs :

Offset#	Taille	Valeur
0x000E	4 octets	La taille de l'entête de l'image en octets. Les valeurs hexadécimales suivantes sont possibles suivant le type de format BMP : – 28 pour Windows 3.1x, 95, NT, … – 0C pour OS/2 1.x – F0 pour OS/2 2.x
0x0012	4 octets	La largeur de l'image, c'est-à-dire le nombre de pixels horizontalement (en anglais *width*)
0x0016	4 octets	La hauteur de l'image, c'est-à-dire le nombre de pixels verticalement (en anglais *height*)
0x001A	2 octets	Le nombre de plans. Cette valeur vaut toujours 1
0x001C	2 octets	La profondeur de codage de la couleur, c'est-à-dire le nombre de bits utilisés pour coder la couleur. Cette valeur peut-être égale à 1, 4, 8, 16, 24 ou 32
0x001E	4 octets	La méthode de compression (sur 4 octets). Cette valeur vaut 0 lorsque l'image n'est pas compressée, ou bien 1, 2 ou 3 suivant le type de compression utilisé : 1 pour un codage RLE de 8 bits par pixel ; 2 pour un codage RLE de 4 bits par pixel ; 3 pour un codage bitfields, signifiant que la couleur est codée par un triple masque représenté par la palette
0x0022	4 octets	La taille totale de l'image en octets
0x0026	4 octets	La résolution horizontale, c'est-à-dire le nombre de pixels par mètre horizontalement
0x002A	4 octets	La résolution verticale, c'est-à-dire le nombre de pixels par mètre verticalement
0x002E	4 octets	Le nombre de couleurs de la palette
0x0032	4 octets	Le nombre de couleurs importantes de la palette. Ce champ peut être égal à 0 lorsque chaque couleur a son importance.

La palette de couleurs

Les logiciels de lecture utilisent trois octets pour coder la couleur (système rouge vert bleu, RVB) :

- en BMP 24 bits (1 octet bleu, 1 octet vert, 1 octet rouge par pixel), la palette n'est pas nécessaire (on peut représenter toutes les couleurs). Dans ce cas, la palette n'existe pas dans le fichier BMP ;
- en BMP 8 bits, on ne peut représenter que 256 couleurs, il faut définir une correspondance entre la couleur du pixel et les trois composantes. Il a donc été ajouté une table juste après l'en-tête (octet 56), qui donne pour chaque valeur (de 0 à 255) les trois composantes RVB qui y correspondent).

Le corps de l'image

Enfin voilà le contenu de l'image ! Elle suit normalement directement la zone d'en-tête du bitmap (ou la palette, s'il y en a une), mais il est préférable d'y accéder via l'offset définit dans le *header.*
Le codage de l'image se fait 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 bleu, vert et rouge.

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 telle manière à respecter ce critère.