Images numériques

Cette série d'activités a pour but de démontrer aux élèves l'affirmation selon laquelle les images informatiques sont numériques, c'est-à-dire constituées de nombres. L'idée générale consiste à ouvrir un même fichier, d'une part en utilisant un éditeur de texte (comme WordPad sous Windows, ou emacs sous Linux), et d'autre part avec un logiciel d'imagerie (comme Gimp). Dans le premier cas, les élèves pourront observer des nombres, et dans le second cas, une image constituée de pixels. Il est possible de modifier des nombres dans le fichier texte et d'observer les modifications obtenues sur l'image. Certains des exercices sont utilisables pour des élèves du cycle 3 (CM1-CM2-6e), d'autres sont plutôt destinés à des élèves de la fin du cycle 4 (classe de 3e), car ils comportent une part d'algorithmique et de programmation en python. Les fichiers et les programmes utilisés se trouvent dans l'archive `FichiersEtProgrammes.zip`, et les corrigés dans l'archive `Corriges.zip`.

1 Pour l'enseignant

Pour les activités proposées, des images aux formats simples : `PPM`, `PGM` et `PBM` sont utilisées. Par contre, les fichiers sont assez volumineux puisque les pixels sont représentés par des caractères ASCII qui occupent 1 octet chacun. Le format PBM est utilisé pour des images en deux couleurs, le format PGM pour les images avec des niveaux de gris et le format PPM pour les images en couleurs. Ces fichiers sont construits sur la même base :

- Le nombre « *magique* » du format (P1 pour les images en noir et blanc, P2 pour celles en niveaux de gris et P3 pour les images en couleurs).
- Un caractère d'espacement qui peut être un espace, une tabulation ou une nouvelle ligne.
- La largeur de l'image (codée en caractères ASCII).
- Un caractère d'espacement qui peut être un espace, une tabulation ou une nouvelle ligne.
- La hauteur de l'image (codée en caractères ASCII).
- Pour les format PGM et PPM, un caractère d'espacement qui peut être un espace, une tabulation ou une nouvelle ligne, suivi d'un nombre représentant le nombre de couleurs utilisées.
- Un caractère d'espacement qui peut être un espace, une tabulation ou une nouvelle ligne.
- Les données binaires de l'image :
  - ⋆ L'image est codée ligne par ligne en partant du haut.
  - ⋆ Chaque ligne est codée de gauche à droite.

De plus, toutes les lignes commençant par # sont ignorées.
La représentation des pixels dans les trois formats utilisés est la suivante :

PBM : ces images utilisent le caractère 1 pour les pixels noirs et 0 pour les pixels blancs, comme l'indique l'image de la figure 1.

```
P1
#  Image de la lettre "J"
6 10
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
1 0 0 0 1 0
0 1 1 1 0 0
0 0 0 0 0 0
0 0 0 0 0 0
```

(a) Résultat (b) Code de l'image

FIGURE 1 – Image PBM représentant la lettre « J » (a) et son code (b)

PGM : ces images sont constituées de caractères qui représentent des niveaux de gris dont la valeur maximale est fixée dans l'image et est inférieure à 65536 (codée en caractères ASCII). Sur la figure 2, il y a 15 niveaux de gris possibles. Le noir est codé par 0 et le blanc par la valeur maximale.

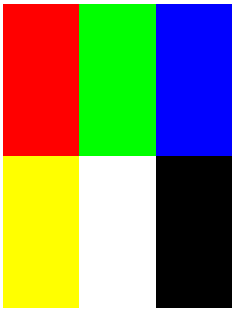
(a) Résultat

```
P2
# Affiche le mot "LOOP"
24 7
15
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  3  0  0  0  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15 15 15 15  0
0  3  0  0  0  0  0  7  0  0  7  0  0 11  0  0 11  0  0 15  0  0 15  0
0  3  0  0  0  0  0  7  0  0  7  0  0 11  0  0 11  0  0 15 15 15 15  0
0  3  0  0  0  0  0  7  0  0  7  0  0 11  0  0 11  0  0 15  0  0  0  0
0  3  3  3  3  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

(b) Code de l'image

FIGURE 2 – Image PGM représentant de l'image « LOOP » (a) et son code (b)

PPM : Cette fois les pixels sont colorés. Il faut indiquer pour chaque pixel 3 valeurs qui indiquent la proportion de rouge, de vert et de bleu. La valeur maximale des couleurs est fixée dans l'image et est inférieure à 65536 (codée en caractères ASCII). Sur l'exemple de la figure 3 cette valeur vaut 255.



(a) Résultat

```
P3
# par 3 colonnes et 4 lignes,
 3 4
# ayant 255 pour valeur maximum
255
255   0   0     0 255   0     0   0 255
255   0   0     0 255   0     0   0 255
255 255   0   255 255 255     0   0   0
255 255   0   255 255 255     0   0   0
```

(b) Code de l'image

FIGURE 3 – Image PPM représentant 12 pixels colorés (a) et son code (b)

Dans l'archive `FichiersEtProgrammes.zip`, se trouvent :
- Le fichier `1Maison.txt` (représentant une maison) destiné à un premier contact.
- Trois fichiers `3Coco400NB.txt`, `5Coco400Gris.txt` et `7Coco400Couleurs.txt` (représentant un perroquet) permettant un travail plus approfondi.
- Trois programmes en python pouvant servir de base à des exercices d'algorithmique destinés à effectuer des traitements d'images. Ces programmes se contentent de copier l'image initiale dans un fichier de sortie.

Dans l'archive `Corriges.zip`, se trouvent :
- Trois programmes en python contenant les corrections des exercices de programmation. La réponse à chaque question se trouve sous forme de commentaire (lignes commençant par ♯) : il suffit de décommenter les lignes correspondantes (c'est-à-dire enlever les signes ♯ en début de ligne) et d'exécuter le programme.

2 Pour les élèves

- Une image numérique est composée de petit carrés appelés des **pixels**, comme une mosaïque.
- Chaque pixel possède une couleur.
- Cette couleur est noir ou blanc dans le cas d'une image en deux couleurs, gris plus ou moins foncé dans le cas d'une image en niveaux de gris et une vraie couleur dans le cas d'une image en couleurs.
- La couleur d'un pixel est représentée par un nombre (ou par trois nombres dans le cas d'une image en couleurs).
- La position de chaque pixel dans une image est repérée par un numéro de ligne i et un numéro de colonne j, appelés coordonnées du pixel.
- Pour une image de hauteur H et de largeur L, les numéros de lignes vont de 0 à $H-1$ et les numéros de colonnes vont de 0 à $L-1$.
- Attention, la ligne numéro 0 est celle d'en haut !
- La colonne numéro 0 est celle de gauche.

Images en deux couleurs

Exercice 1.

1. Ouvrir le fichier `Maison.txt` avec un éditeur de texte. Voyez-vous la maison ?
2. Ouvrir le logiciel `GIMP`, et ouvrir le fichier `Maison.txt` avec ce logiciel.
3. Quel est le nombre représentant la couleur noire pour un pixel ? Et pour la couleur blanche ?
4. Pouvez-vous ajouter une fenêtre au rez-de-chaussée à droite de la porte ?

- Pour effectuer des traitement plus compliqués sur des images plus grandes, on ne change pas les pixels un par un à la main, mais on utilise des logiciels de traitement d'images.
- Le programme `TraitementNoirEtBlanc.py` est écrit en langage Python. Il peut-être ouvert, modifié et exécuté à l'aide du logiciel `IDLE-Python`
- Il s'agit d'un programme de traitement d'images très simple.
- À partir d'un fichier image d'entrée, il permet de parcourir chaque ligne de l'image, colonne par colonne, et de modifier la valeur de chaque pixel pour effectuer le traitement souhaité.
- L'image obtenue est enregistrée dans un fichier de sortie appelé `monimage.txt`, qui peut être ouvert à l'aide d'un éditeur de texte ou d'un visionneur d'images comme `GIMP`.
- Certaines informations sur l'image sont également affichées par le programme.
- Au départ, aucun traitement n'est réalisé, et l'image de sortie est simplement une copie de l'image d'entrée.

Exercice 2.

1. Saurez-vous modifier la partie traitement du programme `TraitementNoirEtBlanc.py` pour que la couleur de chaque pixel de l'image de sortie soit l'inverse de la couleur du pixel correspondant de l'image d'entrée ? Vous pouvez vérifier le résultat en visualisant l'image obtenue à partir du fichier `Maison.txt` comme image d'entrée.
2. On souhaite maintenant obtenir une image représentant une inondation qui atteint le quartier de la maison. Autrement dit, on veut tracer une bande noire occupant tout le bas de l'image.
3. Ajouter un soleil (carré pour simplifier) dans la partie en haut à droite de l'image de la maison.
4. *Attention, plus difficile !* Un pixel du contour du dessin est un pixel noir dont au moins un des 8 pixels voisins est blanc. Dans cette question, sur l'image de sortie, on veut afficher les pixels du contour du dessin en noir et tous les autres pixels de l'image en blanc.
5. En changeant le nom du fichier contenant l'image d'entrée au début du programme, vous pouvez tester le résultat de certains de vos traitements sur l'image du perroquet `Coco400NB.txt`.

Images en niveaux de gris

- Dans une image en niveaux de gris, le gris plus ou moins clair d'un pixel est représenté par un nombre entre 0 et 255.
- Attention, un pixel noir a pour valeur 0 et un pixel blanc a pour valeur 255 !
- Le programme `TraitementGris.py` fonctionne sur le même principe que le programme précédent, mais pour des images en niveaux de gris.

Exercice 3.

1. Créer une image représentant l'image `Coco400Gris.txt` en négatif.
2. Créer une image représentant Coco avec la tête en bas.
3. On veut créer une image représentant le perroquet en applats noirs et blancs. Pour cela, on décide que tous les pixels clairs sur l'image d'entrée deviendront blancs sur l'image de sortie, et que tous les pixels sombres deviendront noirs.
4. On décide maintenant de fixer quatre paliers de couleurs : noir (0), gris foncé (85), gris clair (170) et blanc (255), et pas simplement deux.
5. *Attention, plus difficile !* On se fixe un écart (par exemple 90), et on décide qu'un pixel du contour du dessin est un pixel dont la couleur différe d'au moins cet écart de la couleur d'au moins un de 4 pixels voisins. Dans cette question, sur l'image de sortie, on veut afficher les pixels du contour du dessin en noir et tous les autres pixels de l'image en blanc.

Images en couleurs

- Dans une image en couleurs au format `RVB`, pour Rouge, Vert, Bleu (en Anglais on dit `RGB`, je vous laisse deviner pourquoi), la couleur de chaque pixel est codée par trois nombres compris entre 0 et 255.
- Le premier nombre correspond au rouge, le deuxième au vert et le troisième au bleu.
- Un pixel rouge vif a ainsi pour valeurs 255 0 0, un pixel vert vif a pour valeurs 0 255 0 et un pixel bleu vif a pour valeurs 0 0 255.
- Un pixel dont la valeur du premier nombre est plus grande que les deux autres a une couleur à dominante rouge.
- En donnant des valeurs variables aux trois nombres, on obtient des couleurs intermédiaires, mais la prévision du résultat demande une certaine habitude. Ainsi, un pixel de valeurs 255 255 0 sera jaune vif, un pixel de valeurs 255 00 255 sera plutôt violet vif.
- Le programme `TraitementCouleurs.py` fonctionne sur le même principe que les deux précédents.

Exercice 4.

1. Le perroquet de l'image `Coco400Couleurs.txt` est rouge. Que peut-on dire des valeurs des pixels ?
2. Le programme `TraitementCouleurs.py` permet de transformer Coco en perroquet vert ou en perroquet bleu. Pouvez-vous expliquer comment fonctionne le traitement ?