Représenter les images

4/12/2012

1 Travail

- La séance de travail du 4/12 consiste à travailler l'ensemble de ce document.
- Les résolutions des exercices 3 et 11 sont à déposer dans les casiers numériques de vos enseignants ISN pour la fin de semaine (9/12).
- Les corrigés des exercices qui ne sont pas à rendre seront déposés dans vos casiers après le cours. Ces corrigés sont à travailler, ils vous permettent de contrôler votre travail. Les corrigés des exercices à rendre seront déposés dans vos casiers numériques en début de semaine suivante.

2 Le principe de base du codage d'une image matricielle

Si vous n'êtes pas doué en dessin, une façon efficace de reproduire une image est de la quadriller et de colorer chaque case d'un quadrillage d'une feuille blanche de la même couleur (ou de la couleur dominante) que la case de la grille de l'image. Plus le quadrillage est fin, meilleure est la copie.

On tient là l'idée essentielle de la notion d'image matricielle.
Pour coder une image, on dessine un quadrillage sur l'image.
On attribue à chaque cellule du quadrillage un code numérique correspondant à sa couleur (ou à sa couleur dominante ou à une "moyenne" de ses couleurs).
Les couleurs étant codées numériquement (comme toute information pour un ordinateur), on obtient ainsi une grille de nombres représentant l'image.
Un peu de lecture :
- principe de base pour le codage d'une image : http://blog.kleinproject.org/?p=719&lang=fr#more-719
- plus général : http://interstices.info/jcms/c_16351/vision-artificielle-et-traitement-dimages

3 Création d'un fichier image

Exercice 1

1. Exécuter le programme suivant enonces/A/cree.py

Jean-Manuel Mény- Ludovic Fasquelle – Irem de Lyon.

Python

```python
# -*- coding: utf-8 -*-

largeur=70
hauteur=70
n=largeur*hauteur

#ouverture d'un fichier texte en mode écriture :
f=open("coucou.txt","w")

#on écrit P1 sur la ligne 1 du fichier :
f.write("P1")
# on passe à la ligne :
f.write("\n")
# on écrit la valeur de largeur, espace, la valeur de hauteur :
f.write(str(largeur)+" "+str(hauteur))


# pour chaque ligne :
for j in range(hauteur):
        # passage à la ligne :
    f.write("\n")
    # pour chaque colonne :
    for i in range(largeur):
        # si j est pair
        # on inscrit 0
        if j%2==0:f.write("0" )
        # sinon on inscrit 1
        else: f.write("1")

# on ferme le fichier:
f.close()
```

2. Observer le contenu du fichier texte ainsi créé (avec un éditeur de texte). Faire le lien avec le code python.
3. Consulter les propriétés du fichier (clic droit). Quel est le type de ce fichier ?
4. Effacer l'extension txt du fichier coucou.txt. Consulter les propriétés du fichier. Quel est maintenant son type ?
5. Ouvrir le fichier avec un visionneur d'images.
6. Chercher sur le web la définition du format pbm (portable bit map) et expliquer pourquoi, sans l'extension txt, l'OS a identifié une image. On pourra consulter la page http://fr.wikipedia.org/wiki/Portable_pixmap.
7. Quelle est la largeur en pixels de l'image ? Quelle est sa hauteur en pixels ? Combien faut-il de valeurs numériques pour coder un pixel dans ce format d'image ?
8. Modifier le code du programme python pour que l'image produite ait une largeur de 10 pixels et une hauteur de 30 pixels. Observer l'image obtenue et son code.
9. Sur le fichier définissant l'image de largeur 10 et hauteur 30, changer quelques lignes de 0 par une ligne de 1 ("à la main"). Observer l'effet de cette modification.

Exercice 2

On reprend le programme précédent enonces/A/cree.py.
Modifier le programme python afin que l'image obtenue ne soit pas une alternance de lignes noires et lignes blanches mais une alternance de colonnes noires et colonnes blanches.

Exercice avec corrigé 1

On reprend le programme enonces/A/cree.py

Python

```python
# -*- coding: utf-8 -*-

largeur=70
hauteur=70
n=largeur*hauteur

#ouverture d'un fichier texte en mode écriture :
f=open("coucou.txt","w")

#on écrit P1 sur la ligne 1 du fichier :
f.write("P1")
# on passe à la ligne :
f.write("\n")
# on écrit la valeur de largeur, espace, la valeur de hauteur :
f.write(str(largeur)+" "+str(hauteur))


# pour chaque ligne :
for j in range(hauteur):
        # passage à la ligne :
    f.write("\n")
    # pour chaque colonne :
    for i in range(largeur):
        # si j est pair
        # on inscrit 0
        if j%2==0:f.write("0" )
        # sinon on inscrit 1
        else: f.write("1")

# on ferme le fichier:
f.close()
```

Modifier le programme python afin d'obtenir l'image d'un cercle.

Une résolution

On cherche à placer en noir les points de coordonnées (x, y) tels que $(x - x_\Omega)^2 + (y - y_\Omega)^2 = r^2$ où l'on a pris pour Ω un point centré dans l'image (de coordonnées (largeur//2 , hauteur//2)) et pour rayon r.
Si l'on "allume" les pixels par if (x−largeur//2)**2+(y−hauteur//2)**2==r2 :f.write("1"), on voit clairement la différence entre le pixel (plus petit élément d'une image) et le point mathématique puisque l'on obtient l'image suivante :

(voir l'image dans le dossier enonces/A).
On propose ci-dessous une solution "naïve" basée sur l'idée de ne pas allumer uniquement les pixels lorsqu'il y a égalité avec r^2 mais pour tout un intervalle de valeurs. Dans le plan de la géométrie usuelle, on afficherait ainsi un "cercle épais" (c'est à dire en fait une couronne).
Un code possible enonces/A/cercle2.py :

Python

```python
# -*- coding: utf-8 -*-

largeur=70
hauteur=70
n=largeur*hauteur

#ouverture d'un fichier texte en mode écriture :
f=open("coucou.pbm","w")

#on écrit P1 sur la ligne 1 du fichier :
f.write("P1")
# on passe à la ligne :
f.write("\n")
# on écrit la valeur de l, espace, la valeur de h :
f.write(str(largeur)+" "+str(hauteur))


# carre du rayon :
r=10**2

for j in range(hauteur):
    f.write("\n")
    for i in range(largeur):
        if (i-largeur//2)**2+(j-hauteur//2)**2 in range(r-10,r+11):f.write("1" )
        else: f.write("0")

# on ferme le fichier:
f.close()
```

Exercice avec corrigé 2

On reprend le programme enonces/A/cree.py

Python

```python
# -*- coding: utf-8 -*-

largeur=70
hauteur=70
n=largeur*hauteur

#ouverture d'un fichier texte en mode écriture :
f=open("coucou.txt","w")

#on écrit P1 sur la ligne 1 du fichier :
f.write("P1")
# on passe à la ligne :
f.write("\n")
# on écrit la valeur de largeur, espace, la valeur de hauteur :
f.write(str(largeur)+" "+str(hauteur))


# pour chaque ligne :
for j in range(hauteur):
        # passage à la ligne :
    f.write("\n")
    # pour chaque colonne :
    for i in range(largeur):
        # si j est pair
        # on inscrit 0
        if j%2==0:f.write("0" )
        # sinon on inscrit 1
        else: f.write("1")

# on ferme le fichier:
f.close()
```

Modifier le programme python afin d'obtenir l'image d'un carré.

Une résolution

Un code possible (carré) `enonces/A/carre.py` :

Python

```
# -*- coding: utf-8 -*-

largeur=70
hauteur=70
n=largeur*hauteur

#ouverture d'un fichier texte en mode écriture :
f=open("coucou.pbm","w")

#on écrit P1 sur la ligne 1 du fichier :
f.write("P1")
# on passe à la ligne :
f.write("\n")
# on écrit la valeur de l, espace, la valeur de h :
f.write(str(largeur)+" "+str(hauteur))


# côté /2
r=20//2

for j in range(hauteur):
    f.write("\n")
    for i in range(largeur):
        if max(abs(i-largeur//2),abs(j-hauteur//2))==r:f.write("1" )
        else: f.write("0")

# on ferme le fichier:
f.close()
```

Exercice avec corrigé 3

1. Dans le dossier enonces/B, vous trouverez une photo de l'entrée du lycée de la Plaine de l'Ain.

Avec le logiciel GIMP, enregistrer cette photo au format pbm ascii (utiliser "enregistrer sous" et choisir le format adapté). Ouvrir ensuite le fichier avec un éditeur de texte (modifier éventuellement l'extension en txt).

2. En consultant les propriétés du fichier image (ou du fichier txt), lire la taille du fichier (en octets). Expliquer la taille de ce fichier à l'aide des dimensions (en pixels) de l'image.

Remarque. Une ligne du fichier pbm ne doit pas dépasser 70 caractères alors qu'une ligne de pixels dépasse en général 70 pixels : la lecture se fait donc par paquets de "largeur pixels" plutôt que ligne par ligne.

Une résolution

1.

2. Mon image fait environ 88 ko. La largeur de l'image, lue dans le fichier txt, est de 340 pour une hauteur de 255. Ce qui fait $340 \times 255 = 86700$ octets (puisque chaque pixel est codé par un caractère ascii, ce qui occupe un octet). Si on ajoute les octets de l'entête et ceux des changements de ligne, on trouve les 88 ko annoncés.

Exercice 3
On reprend la photo du lycée du dossier enonces/B.

1. Avec le logiciel GIMP, la transformer en pbm binaire.
2. Comparer la taille du fichier pbm ascii avec celle du fichier pbm binaire. Expliquer.
3. Pour visualiser le code du fichier pbm codé en binaire, utiliser un éditeur hexadécimal (par exemple le logiciel GHex). Vous lisez par exemple 3F dans la partie correspondant au codage des pixels. Que signifie ce 3F ?

4 Les formats d'images

Il existe de nombreux formats d'images : pbm, pgm, ppm, png, jpeg, gif, ps, pict, tiff ... Vous pourrez vous faire une idée des définitions de ces formats en cherchant sur le web.
Un point de départ possible (qui ne concerne pas que les images) :
http://fr.wikipedia.org/wiki/Liste_d%27extensions_de_fichiers
Des caractéristiques distinguant les différents formats d'images :
– Noir et blanc, en niveaux de gris, en couleurs.
– Vectorielle ou bitmap.
– Image compressée ou non.
– Formats publics ou non. Certains concepteurs gardent leur format secret afin de contraindre les utilisateurs à utiliser les logiciels du concepteur pour traiter ces images. D'autres formats ont au contraire une définition publique.
– Formats propriétaires ou libres. Pour certains formats, des droits doivent être payés aux concepteurs, pour d'autres non.
Le format pbm est par exemple libre, public, non compressé, bitmap, noir et blanc.

Exercice 4
Donner (recherche web) les caractéristiques des formats gif et png.
gif : Graphics Interchange Formats.
png : Portable Network Graphics.

Exercice 5
Après une recherche sur le web, expliquer la différence entre « définition » et « résolution » d'une image matricielle. Donner une définition précise de ces deux termes et des exemples explicites à l'aide d'images matricielles.

5 Images en niveaux de gris

On a vu précédemment le format pbm qui est un format codant les images en noir et blanc.
Certains formats, comme le format pgm (portable greymap), utilisent des nuances de gris.

Rappelons la structure d'un fichier PBM :

– P1 est inscrit sur la ligne 1 (code ascii) ou P4 (code binaire).
– En ligne 2 : la largeur de l'image suivie d'un caractère d'espacement, suivie de la hauteur de l'image (en pixels).
– Sur les lignes suivantes : la liste des couleurs des pixels (ligne par ligne, de haut en bas et de droite à gauche), chaque ligne compte au plus 70 caractères (on peut donc inscrire entre 1 et 70 caractères par ligne de codage ascii, une ligne ascii ne correspond pas à une ligne de pixels de l'image, une ligne de pixels correspond à "largeur" caractères pris à la suite).
– On peut ajouter des commentaires (ligne commençant par le symbole #).

Le format PGM (codé en ascii) est construit sur le même modèle :
– P2 est inscrit sur la ligne 1 (P5 pour le pgm binaire).
– En ligne 2 :la largeur de l'image suivie d'un espace, suivie de la hauteur de l'image (en pixels).
– En ligne 3, la valeur maximale utilisée pour estimer les niveaux de gris, par exemple 255. 255 correspond alors au blanc, 0 au noir et les valeurs entre 1 et 254 à différents gris du plus foncé au plus clair.
– Sur les lignes suivantes : la liste des couleurs des pixels (c'est à dire des entiers entre 0 et la valeur indiquée en ligne 3) séparés par des espaces ou des retours à la ligne, ligne par ligne, de haut en bas et de droite à gauche, chaque ligne compte au plus 70 caractères.
– On peut ajouter des commentaires (ligne commençant par le symbole #).

Exercice 6

1. Reprendre la photo du lycée précédemment transformée. Avec le logiciel GIMP, l'enregistrer sous le format pgm ascii. Ouvrir le fichier avec un éditeur de textes.
2. Quelle est la taille (en ko) de cette image ? Justifier (approximativement) cette taille en tenant compte de la dimension en pixels.

Exercice 7

Écrire un programme python :
– input : un fichier pgm ascii.
– output : un fichier pgm ascii aux teintes de gris inversées. Si la valeur maximale utilisée est 255, chaque valeur x sera donc remplacée par $255 - x$.
On donne ci-dessous quelques éléments techniques. Compléter.

Python

```python
# ouverture du fichier source en mode lecture (dans le même dossier que le .py) :
fsource = open("NomDuFichierSource.pgm","r")
# ouverture d'un nouveau fichier (en mode écriture)
# qui contiendra au final le code de l'image aux gris inversés :
fbut=open("NomDuFichierBut.pgm","w")
# on crée une liste des lignes du fichier initial :
L=fsource.readlines()
# on recopie dans le fichier but les lignes de l'entête (cas où l'entête occupe 4 lignes :
# P2, commentaire, dimensions en pixels, valeur max correspondant au blanc)
for i in range(4):
    fbut.write(L[i])

 # inversion des codes correspondant aux teintes de gris :
 ...

# fermeture des fichiers :
fsource.close()
fbut.close()
```

L'effet attendu pourra être visualisé avec les fichiers du dossier enonces/C.

6 Un format avec couleurs

6.1 Le principe de la formation des couleurs sur un écran.

Notre oeil contient des cellules (les cônes) sensibles à la couleur (longueur d'onde) de la lumière qu'ils reçoivent. Ces cônes sont de trois sortes, de maximum de sensibilité respectivement dans le rouge, le vert et le bleu. Quelle que soit la lumière percue, l'oeil ne communique au cerveau que l'intensité de la réaction des cônes. Deux lumières distinctes mais stimulant les trois types de cônes de la même façon nous seront donc indiscernables (par exemple, le mélange de lumière rouge et verte peut donner à l'oeil la même sensation qu'une lumière jaune).
Sur l'écran d'un ordinateur, chaque pixel est composé de trois sources de lumière (rouge, verte, bleue) et en faisant varier l'intensité de chacune d'elles, on peut simuler un grand nombre de couleurs.
Si, comme certains animaux, nous avions quatre type de cônes, nos écrans devraient être conçus avec quatre sources de lumière.

6.2 Le format ppm

Le format ppm (portable pixel map) est un format (parmi beacoup d'autres) concernant les images couleurs.
Dans ce format, on associe à chaque pixel trois nombres entre 0 et 255 correspondant à l'intensité des couleurs rouge, vert, bleu (RGB).
Consulter la page `http://fr.wikipedia.org/wiki/Portable_pixmap` pour la définition du format ppm.

Exercice avec corrigé 4
Dans plusieurs formats d'image couleur, les couleurs sont définies par le code RVB (rouge, vert, bleu ou encore RGB avec les initiales en anglais). La composante "rouge", la composante "vert" et la composante "bleu" sont chacune codée sur un octet, donc par un nombre binaire compris entre 00000000 et 11111111.
Combien de couleurs différentes peut-on ainsi coder ?

Une résolution
Sur un octet, on code les entiers entre 0 et $2^0 + 2^1 + \cdots + 2^7 = 2^8 - 1 = 255$. Chaque composante prend donc 256 valeurs différentes.
On peut donc coder $256 \times 256 \times 256 = 16\,777\,216$ couleurs différentes. □

Exercice 8
Lire, exécuter et comprendre le fichier `enonces/F/couleurs.py` dont le code est rappelé ci-dessous.

Python

```python
# -*- coding: utf-8 -*-


largeur=100
hauteur=600
n=largeur*hauteur

#ouverture d'un fichier en mode écriture :
f=open("couleurs.ppm","w")

#on écrit P3 sur la ligne 1 du fichier :
f.write("P3")
# on passe à la ligne :
f.write("\n")
# on écrit la valeur de largeur, espace, la valeur de hauteur :
f.write(str(largeur)+" "+str(hauteur))
f.write("\n")
# valeur max pour l'intensité des couleurs :
f.write("255")
f.write("\n")


for j in range(hauteur//6):
        for i in range(largeur):
                f.write("255 0  0")
                f.write("\n")

for j in range(hauteur//6):
        for i in range(largeur):
                f.write("0 255  0")
                f.write("\n")

for j in range(hauteur//6):
        for i in range(largeur):
                f.write("0 0 255")
                f.write("\n")

for j in range(hauteur//6):
        for i in range(largeur):
                f.write("0 255 255")
                f.write("\n")

for j in range(hauteur//6):
        for i in range(largeur):
                f.write("255 0 255")
                f.write("\n")

for j in range(hauteur//6):
        for i in range(largeur):
                f.write("255 255 0")
                f.write("\n")

# on ferme le fichier:
f.close()
```

Exercice 9

Quelle sera l'image créée par le programme python ci-dessous ?

Python

```python
# -*- coding: utf-8 -*-

largeur=50
hauteur=256
n=largeur*hauteur

#ouverture d'un fichier en mode écriture :
f=open("coucou.ppm","w")

#on écrit P3 sur la ligne 1 du fichier :
f.write("P3")
# on passe à la ligne :
f.write("\n")
# on écrit la valeur de largeur, espace, la valeur de hauteur :
f.write(str(largeur)+" "+str(hauteur))
f.write("\n")
# valeur max pour l'intensité des couleurs :
f.write("255")
f.write("\n")

for j in range(hauteur):
    for i in range(largeur):
        f.write(str(j)+" 0"+" 0")
        f.write("\n")


# on ferme le fichier:
f.close()
```

Fichier enonces/D/D.py

Exercice 10

On veut créer l'image d'un carré noir plein de largeur 50 pixels et hauteur 50 pixels.

1. Donner la taille (en octets) du fichier obtenu si l'on définit l'image au format pbm ascii.
2. Donner la taille (en octets) du fichier obtenu si l'on définit l'image au format ppm ascii.

Exercice 11

Dans le dossier enonces/E, vous trouverez une image de fleur au format ppm ascii.
La première ligne du fichier texte contient le code P3 correspondant à ce format.
La ligne 2 de ce fichier texte contient les dimensions largeur et hauteur en pixels de l'image : 800 688
La ligne 3 de ce fichier texte contient 255, valeur maximale utilisée pour coder les couleurs.
Chacune des lignes qui suit contient un nombre entre 0 et 255.

1. Quelle est la signification des nombres des lignes 4, 5 et 6 du fichier texte ?
2. Caractériser les numéros de ligne, dans le fichier texte, dont le numéro correspondant à la composante "vert" d'un pixel.
3. Expliquer comment calculer le nombre de lignes de ce fichier texte à partir des indications précédentes.
4. Ci-dessous un extrait de programme dont la boucle principale a été effacée. A vous de la réécrire.

Python

```
# -*- coding: utf-8 -*-

# ouverture du fichier source en lecture:
fsource = open("rose.ppm","r")
# ouverture d'un nouveau fichier
# qui contiendra au final le code de l'image aux rouges et verts intervertis :
fbut=open("verte.ppm","w")
# on crée une liste des lignes du fichier initial :
L=fsource.readlines()

# on recopie dans le fichier but les lignes de l'entête :
for i in range(3):
    fbut.write(L[i])

# boucle à écrire intervertissant les composantes : les valeurs correspondants au rouge
    deviendront les valeurs pour le vert
# les valeurs du vert deviennent les valeurs du bleu et les valeurs du bleu deviennent
    celles du rouge.


# fermeture des fichiers :
fsource.close()
fbut.close()
```

L'objectif est de passer de l'image de gauche à l'image de droite :