Partie 1. Représentation numérique de l'information

1.5 Codage numérique des images ***

On trouve les images numériques dans de nombreux domaines : loisir, médecine, transports, satellite, jeux vidéos, vidéo surveillance, détection de formes

Analyse d'image : Ce domaine de l'activité informatique a pour but de traiter, analyser, transformer l'image numérique

Pixel = **Pict**ure **el**ement

Les grecs, romains, arabes ont utilisé les techniques de la **mosaïque** pour représenter des images

L'industrie textile a souvent considéré les dessins comme des **matrices de points**, avec la programmation par des cartes perforées sur les métiers Jacquard

Avec le **pointillisme** (mouvement artistique du XIX° siècle) on utilise la juxtaposition de petites touches de couleurs primaires pour traduire l'espace, le mouvement et la lumière **Ces mouvements sont précurseurs de la numérisation des images**

Images matricielles

- **Le pixel** est l'unité la plus petite adressable par le contrôleur vidéo
- C'est **en nombre de pixels** qu'est donnée la résolution des écrans (VGA, SVGA...)
- Une **image matricielle** est un tableau de pixels de couleurs différentes que l'on peut projeter sur un écran d'ordinateur
- La **valeur du pixel** traduit la couleur, son intensité, sa luminosité, est donnée par un code couleur (hexadécimal, RVB...)
  Voir : http://fr.wikipedia.org/wiki/Liste_de_couleurs

- Place en mémoire :
  un pixel d'une image en noir et blanc est codé sur un bit : 0 pour le noir, 1 pour le blanc
  un pixel d'une image en niveaux de gris est codé sur 8 bits : 0 pour le noir, 255 pour le blanc, tous les niveaux de gris entre ces deux valeurs
  un pixel d'une image couleur est codé sur 16, 24 voire 32 bits.

Les formats

- Les images sont stockées en mémoire sous forme de fichiers images, aux divers formats
- Le format .bmp (BitMaP) :
  Il s'agit d'un stockage sous forme de code ASCII, dans lequel on trouve les caractères P1, suivis d'un blanc, puis la taille de l'image (deux nombre décimaux séparés par un blanc), puis la liste des valeurs des pixels parcourus de haut en bas et de gauche à droite

- Le format .pgm (Portable Grey Map) :
  Il s'agit d'un stockage d'images en niveaux de gris, sous forme de code ASCII dans lequel on trouve les caractères P2, suivis d'un blanc, puis la taille de l'image, puis une liste de valeurs de pixels compris entre 0 et 255

- La compression d'images :
  format JPEG : compression avec perte d'information (un pixel est remplacé par la moyenne de ses voisins, supporte 16 millions de couleurs format JPEG2000 : compression avec perte d'information, supporte 4milliards de couleurs, animations possibles format GIF : compression sans perte d'information, supporte une palette de 256 couleurs, animations possibles format PNG : compression sans perte d'information, supporte une palette de 256 couleurs ou 16 millions de couleurs

Le traitement de l'image

Exemple : la transformation Photomaton
Visible ici : http://interstices.info/jcms/int_67843/mona-lisa-au-photomaton

- **Des logiciels de traitement de l'image** permettent de retoucher une image
- **Diverses transformations et leurs algorithmes :**
  L'image est balayée pixel par pixel : le **balayage** le plus classique est celui qui parcourt l'image ligne après ligne, de gauche à droite, mais d'autres balayages sont possibles dans le but d'accélérer les programmes, ou de repérer certaines zones de l'image à traiter plus que d'autres
  Certains algorithmes **modifient l'image sur place dans la même matrice de pixels** (comme les transformations arithmétiques : addition, soustraction...), d'autres nécessitent la **reconstitution pas à pas de l'image dans une autre matrice de pixels** (comme les transformations morphologiques : dilatation, érosion...)
  Exemple : on peut **rendre une image plus floue** en remplaçant un pixel par la moyenne de ses voisins,
  Exemple : on peut **transformer une image niveaux de gris en image noire et blanc** en effectuant un **seuillage** (au-dessus d'un certain seuil, tous les pixels deviennent

blanc, en-dessous, ils deviennent noirs)
Exemple : on peut encore ajouter des images très rapprochées pour créer une animation, soustraire des images très rapprochées pour obtenir un contour, lisser les petits défauts d'une images, ou travailler sur les échancrures, isthmes et trous de l'image pour les renforcer (érosion : élimination des détails insignifiants) ou les combler (dilatation : accentuation des traits significatifs)

Images vectorielles

- Ce sont des images créées de manière dynamique par des opérations géométriques, comme les tracés de courbe de la calculatrice, les dessins effectués avec la tortue Logo
- Elles sont stockées au format SVG (Scalable Vector Graphics), peuvent être agrandies à l'infini

Images en 3D

Le codage des images numériques en 3D utilise les règles de la perspective pour placer les point, dont on calcule les coordonnées en utilisant les règles de la géométrie et les équations des droites de fuite. La profondeur intervient comme un facteur de division des coordonnées. (Voir livre p. 243).

TP Codage numérique des images :

► **Exercice 1 : Pour commencer avec la proglet Codage de pixel de Java'scool**

- Le téléchargement d'une image s'obtient par load(" adresse de l'image "), ce qui affiche l'image en couleurs.
- Pour télécharger, utiliser et diffuser des images libres de droits, on pourra utiliser le site `http://encyclopedia.erpi.com/encyclopedia/accueil` autorisant la reproduction pour usage personnel ou scolaire sous condition de citer le propriétaire de l'image (ici © Dorling Kindersley)
- Pour afficher l'image en niveaux de gris
  En Java'scool :

```
int w = getWidth();
int h = getHeight();
//coordonnées du point
int i;
int j;
for (i = - w; i <= w; i = i + 1) {
  for (j = - h; j <= h; j = j + 1) {
    setPixel(i, j, getPixel(i, j));
    //getPixel est un entier définissant un niveau de gris
     //compris entre zéro (noir) et 255 (blanc)
     // setPixel(a,b,valeur) affiche un point monochrome
     //aux coordonnées (a,b)
  }
}
```

- Pour la transformer en matrice de pixels noirs et blancs : compléter l'algorithme
  En Java'scool :

```
int w = getWidth();
int h = getHeight();
int i;
int j;
for (i = - w; i <= w; i = i + 1) {
  for (j = - h; j <= h; j = j + 1) {
    .....
```

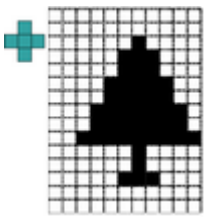
► **Exercice 2 : A l'aide d'un logiciel de traitement d'image libre**, ouvrir une de vos images préférées, puis :
Modifier luminosité et contraste
Inverser les couleurs
Transformer l'image en niveaux de gris, puis noir et blanc, en négatif, en sépia, ajouter des effets
Recadrer l'image
Obtenir ses dimensions, son nombre de couleurs, sa taille en mémoire
Compresser l'image, lui ajouter du texte

► **Exercice 3 :** Ecrire l'algorithme qui inverse une image (négatif), qui l'éclaircit, l'assombrit, augmente son contraste, le diminue

► **Exercice 4 :** Que fait cet algorithme ?
En Java'scool : **int valeurPixel = getPixel(i, j) + getPixel(i + 1, j) + getPixel(i - 1, j) + getPixel(i, j - 1) + getPixel(i, j + 1) ; setPixel(i, j, valeurPixel / 5) ;**

► **Exercice 5 :** La transformation Photomaton : expérimentation à la main
- Représenter sur papier une matrice de 16 lignes et 16 colonnes
- Représenter ensuite l'image obtenue par la transformation de Photomaton
- Réitérer le procédé : en combien de transformations retrouve-t-on l'image initiale ?
- Ecrire l'algorithme de la transformation de l'image

► **Exercice 6 :** Sur ce dessin est représentée une image comportant des cases blanches et



des cases noires

1. Choisir un codage pour ces cases
2. On définit pour chaque case la relation "a pour voisins" de la manière suivante : les voisins d'une case sont les 4 cases formant la croix, en couleur, autour de cette case. Écrire la fonction booléenne de chacune des transformations suivantes du dessin complet :
   - Algorithme 1 : Une case noire reste noire, une case blanche devient noire si l'un au moins de ses voisins est noir (le noir est "contagieux")
   - Algorithme 2 : Une case blanche reste blanche, une case noire devient blanche si l'un au moins de ses voisins est blanc (le blanc est "contagieux")

► **Exercice 7 :** Créer ses premiers fichiers d'images Exercices 9.3 et 9.4 p. 126 et 127