Les images numériques : du Noir & Blanc aux niveaux de gris

1. La révolution des images numériques

La photographie argentique

La photographie a été inventée par le français Nicéphore Niépce vers 1824. D'abord sur des plaques d'argent recouvertes de bitumes, et qui nécessitaient plusieurs jours d'exposition aux rayons du soleil. le procédé fut amélioré par Louis Daguerre, qui recouvrit la plaque d'argent par une fine couche d'iodure d'argent, cette plaque étant ensuite exposée dans une chambre obscure puis soumise à des vapeurs de mercure qui provoquaient l'apparition de l'image. Le procédé s'améliora au fur et à mesure, par l'utilisation de papier argentiques réalisant un négatif de l'image souhaitée, la photographie se révélant après un bain dans différentes substances chimiques au sein d'une chambre noire.

Les procédés de photographie couleur ne sont apparus qu'au milieu des année 1930.

Au temps de la photographie argentique, obtenir une photographie nécessitait l'exposition puis le développement d'une pellicule sur du papier traité spécialement. Il fallait donc du temps, et chaque photo prise devait être développée - et donc coutait une certaine somme - quel que soit le résultat de la photo.

Pour mieux comprendre, rien de tel qu'une série de vidéo de *C'est pas sorcier* :

- Comment fonctionne la photographie argentique ?
- A quoi sert l'obturateur d'un appareil photo ?
- Comment développe-t'on une photo ?

Question

Énoncé

Quels sont les avantages et les défauts de la photographie argentique par rapport à la photographie numérique ?

Réponses

La **photographie numérique** - et par extension la **vidéo numérique** - a considérablement changé notre rapport à l'image. Tout possesseur de smartphone peut prendre des centaines de photographies, consulter immédiatement le résultat de la prise de vue, supprimer, éditer ou améliorer la photographie, et l'envoyer à l'autre bout du monde via internet, tout cela pour un coût extrêmement bas par photographie.

Bien entendu, comme tout progrès technologique, celui-ci n'est pas sans poser de nombreux problèmes. Par exemple, il est très facile de copier une photographie numérique (c'est ce qu'on appelle la non-rivalité de l'information numérique). D'où un certain nombre de problèmes :

- droits d'auteurs ;
- respect de la vie privée ;
- droit à l'effacement compromis ;
- deep-fake et manipulation de l'information
- etc...

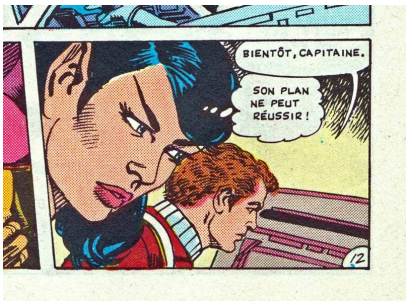
Dans ce thème, nous étudierons à la fois le côté numérique et technologique - comment est constituée une image numérique, sous quelle forme elle est stockée dans l'ordinateur - mais aussi sur les aspects sociétaux liés à l'utilisation des images numériques.

2. Quelques considérations optiques

2.1. Les trames Benday

Lors des années 1950 -1970, les comics américains ainsi qu'un certain nombre d'éditeurs de bandes dessinées européens utilisaient pour leurs couleurs une impression quadrichromique basée sur la méthode des **Benday dots**. Cette méthode consiste à appliquer de la couleur par une juxtaposition de points de couleurs primaires (par couches cyan, magenta, jaune et noire - ça ne vous rappelle rien ?) comme dans les images suivantes :

BD et Benday Dots



Cette méthode permettait à moindre coût d'obtenir des nuances de couleurs suffisantes pour une impression de bandes dessinées. L'artiste américain de pop art Roy Lichtenstein. s'est par ailleurs emparé de cette méthode pour réaliser de nombreuses oeuvres.

2.2. Le pouvoir de résolution de l'oeil

Si la méthode des **Benday Dots** fonctionne et nous donne l'impression de nuances de couleurs, c'est parce qu'à une distance suffisante, nous sommes dans l'impossibilité de distinguer ces points. Ceci est dû au pouvoir de résolution de l'oeil

Pour la vision humaine, selon wikipédia, «le pouvoir de résolution de l'œil est d'environ une minute d'arc ($1' = 1/60^\circ = 0,017^\circ$), soit environ 100 km sur la surface de la Lune vue de la Terre, ou plus proche de nous, un détail d'environ 1 mm pour un objet ou une image situé à 3 m de distance.»

Exemple

A titre d'exemple, on peut considérer cette image sous trois résolutions différentes :

Si de près l'image de gauche est beaucoup plus nette que l'image de droite, en vous reculant par rapport à l'image, vous trouverez une distance pour laquelle vous ne verrez plus de différence entre les images. Vous aurez ainsi atteint la limite de résolution de votre oeil.

3. Du noir et blanc aux niveaux de gris

3.1. Du caractère à l'image

Du caractère à l'image

Enoncé

1. Créez un dossier `ImagesNumeriques` dans votre dossier `SNT`.
2. Téléchargez et enregistrez dans le dossier précédent le fichier reallyNumbers.txt
3. Ouvrez avec `Notepad++` le fichier `reallyNumbers.txt`. Que contient-t-il ?
4. A l'aide de la combinaison `CTRL+Molette`, dézoomer le fichier. Quel constat pouvez-vous faire ?
5. A l'aide de l'outil de remplacement (`Recherche > Remplacer`), remplacer toutes les valeurs « `255` » par **rien** (c'est-à-dire PAS PAR UN ESPACE), puis remplacez toutes les valeurs « `0` » (c'est-à-dire un zéro suivi d'un espace) par « `0` » (c'est-à-dire zéro non suivi d'un espace). Dézoomez de nouveau. Que constate-t-on ?
6. Combien de lignes possède ce fichier ?
7. Combien de caractères, espaces compris, y-a-t-il par ligne ?

Solution

3.2. Une image en noir et blanc

Images en noir et blanc

Enoncé

1. Téléchargez et enregistrez dans le répertoire `ImagesNumeriques` le fichier champiNB.bmp
2. Ouvrez le fichier `champiNB.bmp` dans un visionneur d'images.
3. Quelle est la largeur en pixel de l'image ?
4. Quelle est la hauteur en pixel de l'image ?
5. Combien y-a-t-il de pixels dans l'image ?
6. Quelle est la dimension en octet de cette image ?

Solutions

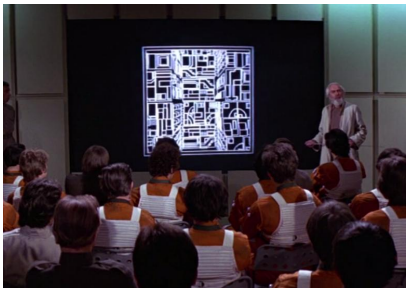
Représenter une image en noir et blanc

En simplifiant beaucoup, l'image obtenue dans la partie précédente est une image numérique en noir et blanc. Le fichier `reallyNumbers.txt` est une version lisible du fichier `champiNB.bmp`. Chaque **pixel** de l'image est représenté par deux valeurs : 255 pour du blanc et 0 pour du noir - soit un octet plein (`1111 1111`) pour du blanc et un octet vide (`0000 0000`) pour du noir.

Le nombre d'octets dans l'image BMP correpond donc à peu près au nombre de pixels de l'image. La différence provient d'un petit nombre d'informations - les métadonnées - situées dans l'entête du fichier pour préciser le réel format de l'image - ici le format `BMP` (*BitmaP*, ou *windows BitMap*).

Ecrans monochromes

Au début de l'informatique - des années 1960 aux années 1980, il était très fréquent d'utiliser des écrans **monochromes**. Ces écrans étaient constitués de luminophores - une substance qui émet de la lumière si elle subit une excitation. Ces luminophores donnaient à l'écran, en fonction de leurs caractéristiques, soit une couleur blanche, soit une couleur orange, mais le plus souvent une couleur verte ! D'où les images bien connues du film `Matrix`, ou les couleurs des affichages des ordinateurs de bord dans la première trilogie de `Star Wars`.



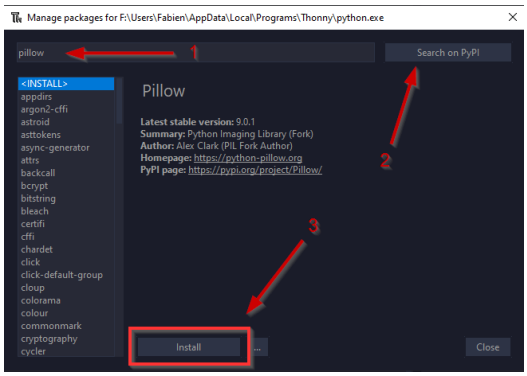
Certains écrans monochromes sont toujours utilisés en raison de leur très grande lisibilité.

3.3. Générer ses propres images

Fabriquer en PixelArt

Enoncé

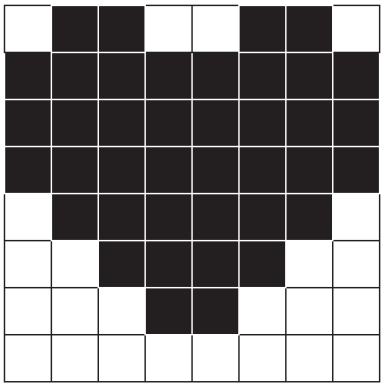
1. Ouvrir le logiciel `Thonny`.
2. Ouvrir le menu `Tools > Manage packages` et dans la barre de recherche, chercher le module `pillow`, et l'installer si il n'est pas déjà installé.



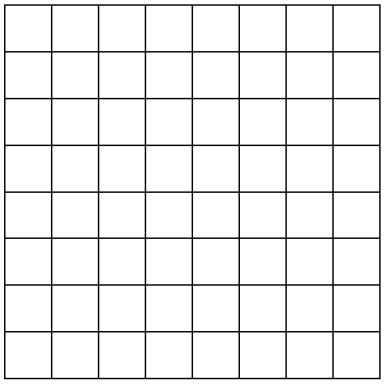
3. Téléchargez et sauvegardez le fichier GenereBMP.py, puis ouvrez le dans `Thonny`.
4. Exécutez le programme, en sauvegardant le fichier sous le nom `Stripes.bmp`. Ouvrez l'image obtenue. Qu'obtient-on ?
5. Dans le code, quelle structure représente l'image ?

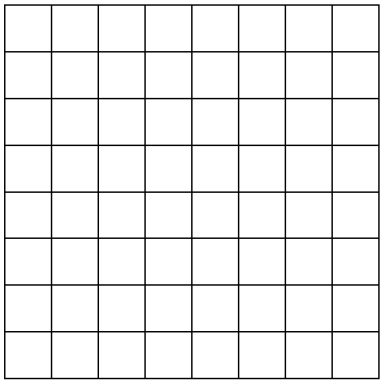
> **Matrice des pixels**
>
> En Python, une telle structure est **un tableau** (plus précisément, un tableau est une liste de listes). Un tableau de nombre est appelé une **matrice**. Dans notre cas chaque nombre de la matrice `pix` représente un pixel soit de couleur noire (0), soit de couleur blanche(255).

6. Combien y-a-t-il de pixels dans cette image ?
7. Modifiez la matrice afin d'obtenir l'image suivante :



8. A l'aide des grilles suivantes, créez deux nouvelles images :





9. Modifiez la matrice pour créer vos deux images.

10. Combien de valeurs sont nécessaires pour une image de 16×16 pixels ? de 32×32 pixels ? de 64×64 pixels ?

3.4. Les niveaux de gris

Téléchargez et sauvegardez les fichiers champisGris.txt et champisGris.bmp.

Le fichier `champiGris.txt` contient l'extraction des pixels du fichier `champiGris.bmp`. Quelle est la principale différence par rapport au fichier `champiNB.txt` ?

Nuances de gris

Contrairement aux écrans monochromes, qui ne pouvaient pour chaque luminophore prendre que deux valeurs (allumés et éteints), les écrans de télévision à tubes cathodiques dits « en noir et blanc » étaient en fait capable de prendre toute une nuance de couleur entre le noir et le blanc, ce qu'on appelle des **niveaux de gris**.

De la même manière qu'en noir et blanc, ont peut donc créer et manipuler des images en niveau de gris, chaque pixel pouvant alors prendre une valeur entière comprise entre 0 et 255, soit 256 nuances différentes.

4. Transformer des images

Manipuler et transformer des images

Enoncé

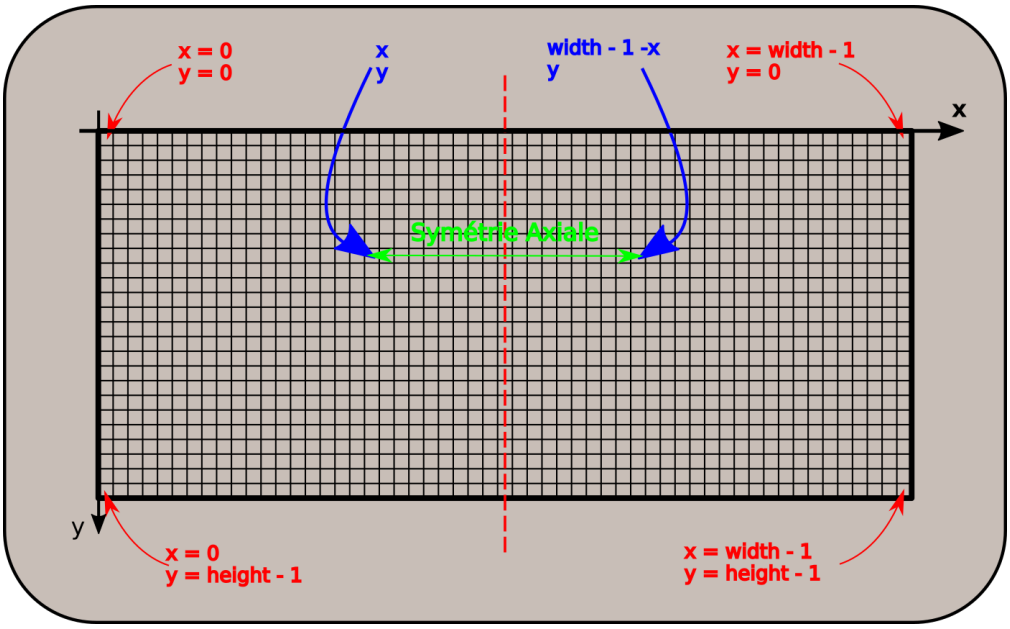
1. Téléchargez et enregistrez le fichier vador.jpeg dans votre dossier `SNT/ImagesNumériques`.
2. Téléchargez et enregistrez traitementGris.py puis ouvrez-le dans `Thonny`.
3. Exécuter le fichier `traitementGris.py`.
4. Comment s'appelle la transformation effectuée entre les deux images ?

La carte des pixels (*Pixel Map*)

Chaque pixel d'une image est repéré par ses *coordonnées*, sous la forme d'un couple $(x\ ;\ y)$ d'entiers, compris entre :

- `0` et `width-1`, où `width` est la **largeur** de l'image;
- `0` et `height-1`, où `height` est la **hauteur** de l'image.

L'axe des abscisses est *horizontal*, et dirigé vers la droite (comme en maths). L'axe des ordonnées est *vertical*, et dirigé **vers le bas** (au contraire des maths !)



Le pixel obtenu par symétrie axiale d'axe vertical médian en partant du pixel de coordonnées $(x\ ;\ y)$ est donc celui de coordonnées $(width - 1 - x\ ;\ y)$.

On voit cette transformation dans le code Python :

```
gs, alpha = file.getpixel((x,y))
# On place dans la nouvelle image le symétrique du pixel travaillé
pixels[width-1-x,y] = (gs, alpha)
```

Détaillons le code :

- `gs, alpha = file.getpixel((x,y))` : on crée deux variables `gs` et `alpha`, qui contiennent les valeurs `gs` et `alpha` du pixel de coordonnées $(x;y)$ de l'image d'origine (`file`). A titre d'information :
  - `gs` (pour *greyscale*) contient le niveau de gris ;
  - `alpha` contient le niveau de transparence (canal alpha) du pixel (inutile dans ce TP).
- `pixels[width-1-x,y] = (gs, alpha)` : on place dans la nouvelle image `pixels` aux coordonnées symétriques $(width - 1 - x; y)$ les valeurs `gs` et `alpha` récupérées.

5. Comment transformer le code pour obtenir une symétrie d'axe horizontal ?
6. Comment transformer le code pour obtenir une rotation à 180° de l'image originale ?
7. Pour obtenir un négatif de l'image originelle, il faut changer la valeur de chaque pixel de la manière suivante :

- un pixel de valeur 255 devient de valeur 0 et inversement ;
- un pixel de valeur 250 devient de valeur 5 et inversement :
- un pixel de valeur 100 devient de valeur 1555 et inversement :
- ...

Que changer dans la ligne suivante pour obtenir un tel négatif ?

```
pixels[width-1-x,y] = (gs, alpha)
```

Autres images

Vous pouvez aussi vous amuser avec d'autres images, il suffit de les mettre dans votre dossier `SNT/Images` et de changer le nom dans la ligne 6 du programme `traitementGris.py`.