ISN S12: Les images

Objectif :
Découvrir l'utilisation des images sous processing.

1. Formats d'image:

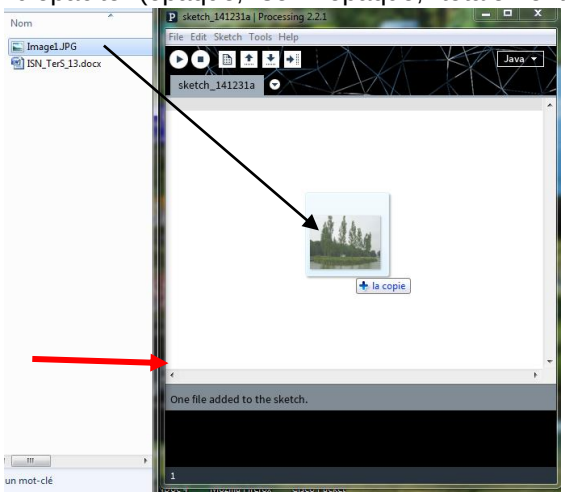
Trois formats de fichier d'image sont acceptés dans Processing : **PNG, JPEG, ou GIF.**

- JPEG est souvent utilisé pour compresser des images de type photographique. Ce format ne permet pas d'avoir des zones transparentes.
- GIF est historiquement utilisé pour l'illustration de boutons et autres éléments graphiques de l'espace de travail d'un programme. Sur les sites internet, ce format est utilisé pour les logos et de manière générale pour les dessins réalisés par ordinateur (notamment ceux qui comportent des aplats de couleurs). Ce format peut contenir des zones transparentes binaires (soit opaques soit transparentes, sans possibilité d'opacité intermédiaire). Il existe des images gif animées, mais Processing ignorera cet aspect.
- PNG est de plus en plus utilisé pour les deux usages (photos + dessins) et peut contenir des zones transparentes non binaires, offrant des niveaux d'opacité (opaque, semi-opaque, totalement transparent).

2. Importer une image:

a) Déposer/glisser :

Localisez maintenant votre fichier d'image, et glissez-le directement sur la fenêtre Processing :
lors de l'appel de l'image dans le programme, elle doit être dans le répertoir du sketch dans data



b) Importer une image:

Il y a trois étapes pour afficher une image dans Processing :

1. Créer une variable qui contiendra les données (en pixels) de notre image.
2. Importer les pixels d'un fichier dans notre variable.
3. Dessiner les pixels de cette variable dans notre espace de dessin.

Syntaxe : PImage a;
a=loadImage("Nom du fichier");
image (a, X, Y, "Largeur", "hauteur");

I) Ecrire le programme ci-dessous et sauvegarder sous exemple1.pde

```
size(500,400);

PImage ile;
ile = loadImage("Image1.JPG");

image(ile,50,10,400,380);
```

Qu'obtenez-vous?

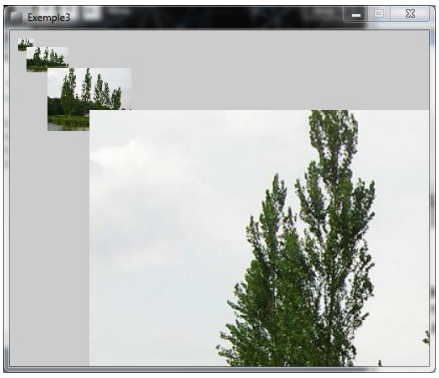
c) Image importé du web

II) Ecrire le programme ci-dessous et sauvegarder sous exemple2.pde

```
size(400,400);

PImage webcam;
webcam = loadImage("http://www.gutenberg.org/files/3913/3913-h/images/rousseau.jpg");
image(webcam,0,0,width,height);
```

Qu'obtenez-vous?



3. Changer la taille d'une image:

Pour changer la taille de l'image, il suffit de rajouter deux paramètres à votre image, {*largeur*, *hauteur*},

III) Ecrire le programme qui affiche l'image ci-contre et sauvegarder sous exemple3.pde
Taille de la fenêtre 500*400 pixels²
Taille image 1 : 20*15 pixels²
Taille image 2 : 50*30 pixels²
Taille image 3 : 100*75 pixels²
Taille image 4 : 1000*750 pixels²

4. Rendre transparent l'arrière-plan d'une image:

Souvent nous avons besoin d'importer des images qui ne sont ni carrées, ni rectangulaires, comme dans le cas d'un petit jeu vidéo utilisant des silhouettes de personnages en deux dimensions : nous ne voulons pas voir en permanence un carré blanc autour du profil de notre héros. Malheureusement, à ce jour, les images à base de pixels doivent être importées dans un format carré.

Pour lever cette contrainte, certains formats de fichiers font appel à une **couche alpha** pour gérer la transparence de certaines zones de l'image et ainsi laisser apparaître les éléments qui sont situés derrière elles (un fond coloré ou illustré par exemple). Cette couche se superpose aux couches *rouge, vert, bleu* utilisées pour composer l'image et indique l'intensité de la transparence pour chaque pixel, avec même la possibilité de pixels semi-transparents dans le cas du format *PNG*.

IV) Ecrire le programme qui affiche l'image ci-contre et sauvegarder sous exemple4.pde
Taille de la fenêtre 400*300 pixels²
Taille image 1 : 400*300 pixels²
Taille de Rousseau : celle d'origine.

5. Colorier les images:

On peut également colorier les images. Autrement dit, on peut changer la couleur des pixels, soit dans leur ensemble, soit individuellement. La plus simple de ces méthodes concerne le coloriage de l'ensemble des pixels. Cette méthode de coloriage ressemble à peu près au coloriage de rectangles, mais nécessite une nouvelle instruction, tint(), pour le distinguer du coloriage direct des pixels à l'intérieur du rectangle. Dans le cas de tint(), Processing va modifier la couleur de chaque pixel au moment où celui-ci est dessiné dans l'espace de dessin.

Syntaxe :
```
PImage a;
a=loadImage("Nom du fichier");
tint(R,G,B,Alpha);
image (a, X, Y, "Largeur", "hauteur");
```

V) Ecrire le programme qui affiche l'image ci-contre et sauvegarder sous exemple5.pde
size(500,150);
Taille de la fenêtre 500*150 pixels²
Taille des images : 110*110 pixels²

6. Création d'une image à partir de la valeur de chaque pixel:

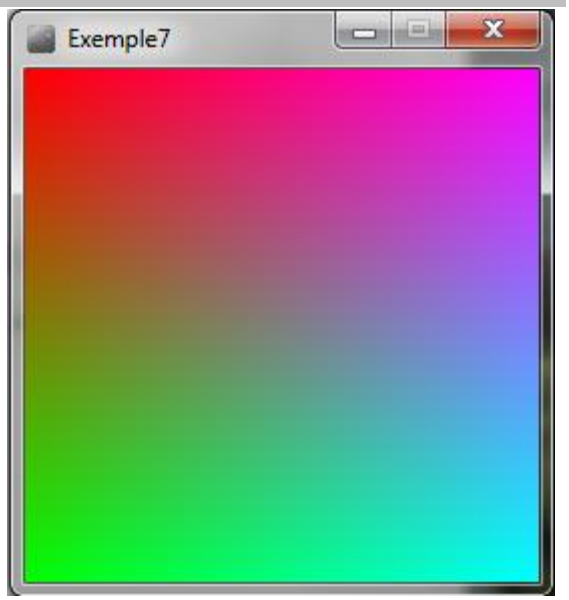
VI) Ecrire le programme ci-dessous et sauvegarder sous exemple6.pde

```
size(200, 200);
// Conversion en pixels
loadPixels();
// Définit chaque pixel
for (int i = 0; i < pixels.length; i++) {
  float R = random(255);
  float G = random(255);
  float B = random(255);
```

```
  // Création de la couleur
  color c = color(R,G,B);
  //On met cette valeur dans le tableau de pixels
  pixels[i] = c;
}
// Affiche le résultat
updatePixels();
```

Qu'obtenez-vous?

VII) Ecrire le programme qui affiche l'image ci-contre et sauvegarder sous exemple7.pde



Taille de la fenêtre 256*256 pixels²
Pour trouver les valeurs RGB, il faut deux boucles FOR:

```
for (int j = 0; j < 256; j++)
{
   for (int k = 0; k < 256; k++)
  {
    float R = 255-j;
    float G = j;
    float B = k;
  }
}
```