Pixel Art

Quelle belle image !

Une image numérisée est composée de pixels, c'est-à-dire en petites zones carrées. Chaque pixel possède une couleur.
Le nombre de lignes et de colonnes donnent les dimensions de l'image, et le produit de ces deux nombres est égal au nombre de pixels contenus dans l'image.
Avec une image de 150 pixels de large sur 100 pixels de hauteur, on a donc ..... pixels.
Les logiciels de traitement d'images numériques comme Photoshop ou Gimp permettent par un fort agrandissement de mettre en évidence les pixels d'une image.

Rappel :
Une couleur est composée d'une composante verte, une composante rouge et une composante bleue. (R,V,B). Chaque composante est un entier compris enter 0 et 255.

Q1/ Pourquoi 255 ?

Comment accéder aux composantes de couleur de chaque pixel ?

Pour manipuler les images dans l'environnement WEB, nous allons utiliser un canvas. Il permet de manipuler chaque pixel d'une image. Ci-dessous, les balises HTML pour créer un élément Canvas.

```
<canvas id="monCanvas" width="600px" height="600px"></canvas>
```

Voici le code javascript qui permet de travailler sur les pixels d'une image :

```
img = new Image();      /* Ce code permet de créer un objet image et de lui associé son URL */
img.src="un-chat.jpg";
```

```
var canvas = document.getElementById("monCanvas");

var ctx = canvas.getContext("2d");

     /* On definit la longueur et la largeur du canvas*/
           canvas.width=image.width;
           canvas.height=image.height;

     /* On dessine l'image dans le canvas */
           ctx.drawImage(image,0,0);

     /* On remplit un tableau contenant les composantes et l'opacité de chaque pixel */

        pixels = ctx.getImageData(0, 0,image.width,image.height);
```

L'instruction `ctx.getImageData` permet de récupérer un objet qui contient les informations d'image d'une zone du `canvas`. Il contient les dimensions de la zone et la propriété `data` qui est un tableau à une dimension contenant les différents octets qui codent les pixels de l'image.

1ère valeur du tableau : composante rouge du 1er pixel (nombre de 0 à 255)
2ème valeur du tableau : composante verte du 1er pixel (nombre de 0 à 255)
3ème valeur du tableau : composante bleue du 1er pixel (nombre de 0 à 255)
4ème valeur du tableau : composante d'opacité du 1er pixel (nombre de 0 à 255)
5ème valeur du tableau : composante rouge du 2ème pixel (nombre de 0 à 255)
6ème valeur du tableau : composante verte du 2ème pixel (nombre de 0 à 255)
7ème valeur du tableau : composante bleue du 2ème pixel (nombre de 0 à 255)
8ème valeur du tableau : composante d'opacité du 2ème pixel (nombre de 0 à 255)
...

Traiter les images

L'image affichée dans le `canvas` est constitué de pixels, chaque pixel est codé sur 4 octets :

- un octet pour la composante rouge,
- un octet pour la composante verte,
- un octet pour la composante bleue,
- un octet pour la composante alpha qui indique la transparence (de 0 pour un pixel complètement transparent à 255 pour un pixel complètement opaque).

Exemples :
pixels[0] permet d'obtenir la composante rouge du du 1er pixel.
pixels[1] permet d'obtenir la composante verte du du 1er pixel.
pixels[5] permet d'obtenir la composante verte du 2ème pixel.

A RETENIR :
Pour agir sur les pixels d'une image on va UNIQUEMENT modifier les valeurs du tableau : PixelsImage

manip'1
Lisez attentivement le code source du fichier dans ArtPixel.js de la fonction annuler_R(). Commenter le code.

1/ Que fait cette fonction ?
2/ Compléter la fonction **annuler_V** qui annule la composante verte de l'image.
3/ Compléter la fonction **annuler_B** qui annule la composante bleue de l'image.
4/ Faire les tests

Les nuances de gris ...

Pour avoir une image en noir et blanc, il faut faire en sorte que pour chaque pixel : composante rouge = composante verte = composante bleue = Gris

Voici quelques exemples pour le choix de la variable Gris :

- méthode 1 : Gris=(R+V+B)/3 (moyenne)
- méthode 2 : Gris=min(R,V,B)+max(R,V,B)/2 (moyenne entre le min et max des 3 composantes)
- méthode 3 : Gris=0,2125×R+0,7154×V+0,0721×B (Couleurs « vraies »)
- méthode 4 : Gris=0,299×R+0,587×V+0,114×B (Signaux vidéos)

manip'2
Ajouter deux fonctions en utilisant 2 méthodes pour convertir l'image en niveaux de gris.

1/ Ajouter deux boutons au menu
2/ Ecrire la fonction niveauGrisVers1().
3/ Ecrire la fonction niveauGrisVers2().
4/ tester les fonctions

Négatif...

Pour obtenir le négatif d'une image, il faut remplacer la valeur des composantes de chaque pixel par son complément.

NouvelleValeur = 255 - AncienneValeur

manip'3
Ajouter une fonction permettant d'obtenir le négatif d'une image.

Sépia...

Pour obtenir une image en sépia, il faut :

1/ calculer pour chaque composante les formules suivantes

tr = 0.393R + 0.769G + 0.189B
tg = 0.349R + 0.686G + 0.168B
tb = 0.272R + 0.534G + 0.131B

2/ Ne prendre que la valeur entière

3/ Les nouvelles valeurs RGB sont obtenues de la manière suivante

Si (tr>255)
  Alors R = 255
  Sinon R = tr
Fin si

Faire la même chose pour les trois composantes

https://www.dyclassroom.com/image-processing-project/how-to-convert-a-color-image-into-sepia-image

manip'4
Ajouter une fonction permettant d'obtenir une image en sépia

La tête en bas...

Le pixel en position (x,y) passe en position (x,hauteur-y). L'utilisation de la fonction `conversion` (voir ci-dessous) est indispensable pour gérer facilement ce "jeu" sur les coordonnées.

manip'5
1/ Ajouter une fonction qui assure le retournement horizontal de l'image
2/ Ajouter une fonction qui assure le retournement vertical de l'image

Aide

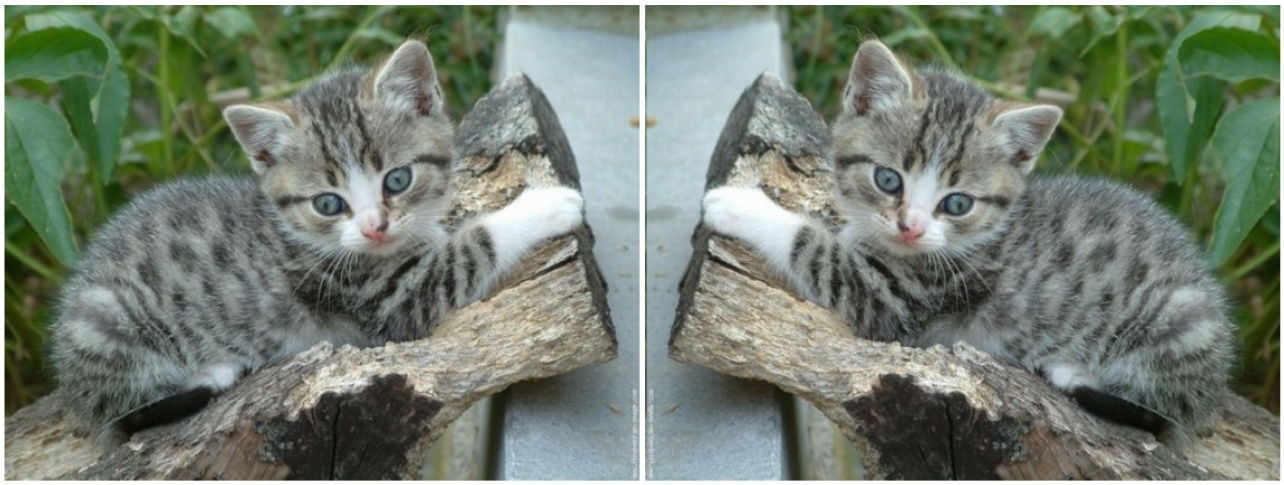
Position physique et position dans `data`

Habituellement lorsque l'on travaille avec des images les pixels se repèrent physiquement en plaçant l'origine en haut à gauche et du coup les ordonnées croissent vers le bas.

Par contre dans le tableau `data` les données des pixels sont rangées les unes à la suite des autres, ce qui rend toute manipulation qui fait intervenir les coordonnées impossible à gérer directement.
Pour contourner le problème voici la fonction `conversion` qui à partir des coordonnées physiques du pixel retourne la position dans `data` du premier octet qui défini ses caratéristiques.

```
function conversion(x,y){
     return (y*image.width+x)*4;
}
```

Retournement horizontal



Retournement vertical