Comment expliquez-vous les différentes valeurs ?

Quand on clique sur différents pixels les valeurs RGB (Rouge, Vert, Bleu) changent parce que chaque pixel de l'image peut avoir une intensité différente pour chaque canal de couleur.

L'image est composée de points (pixels), et chacun possède une combinaison spécifique de R, G et B, ce qui donne la couleur finale.

Pourquoi les valeurs H et S sont toujours nulles ?

H = teinte (Hue)
S = saturation
V = valeur (luminosité)

Si H = 0 et S = 0, ça veut dire que la couleur est un gris.
En effet, dans une image en niveaux de gris, il n'y a pas de teinte, ni de saturation, donc seule la luminosité varie.

Si vous deviez représenter cette image par programmation, quelle structure de donnée utiliseriez-vous ?

Tableau 2D: image[x][y] = (R, G, B) et image[x][y] = (H, S, V)

Selon vous, quelle serait le type de donnée le plus adapté pour cette structure ?

Pour chaque canal (R, G, B), les valeurs vont de 0 à 255. Donc le type adapté = entier non signé sur 8 bits (byte en Java par exemple).

H (Hue) : réel (float/double), exprimé en degrés 0–360.
S (Saturation) : réel (float/double), entre 0.0 et 1.0.
V (Value) : réel (float/double), entre 0.0 et 1.0.

Donc le type adapté pour HSV = float sur 32 bits ou double sur 64 bits

Image en couleur

Comment expliquez-vous les différentes valeurs ?

Chaque pixel a sa propre couleur donc chaque combinaison R, G et B peut être différente.
Image couleur contrairement à une image en niveaux de gris :
H (teinte) ≠ 0, S (saturation) ≠ 0, V (luminosité) varie selon la couleur du pixel.

À quoi correspondent ces 3 calques ?

Chaque calque correspond à un canal de couleur :

R (Rouge) = intensité du rouge pour chaque pixel
G (Vert) = intensité du vert
B (Bleu) = intensité du bleu

Si vous deviez représenter cette image couleur par programmation, quelle structure de donnée utiliseriez-vous ?

Tableau 2D: pixel[x][y] = (R, G, B)

Selon vous, quelle serait le type de donnée le plus adapté pour cette structure ?

Chaque canal va de 0 à 255 donc entier non signé 8 bits (uint8 ou unsigned char).

Au lieu d'utiliser 4 Byte/Unsigned char pour représenter un pixel, quel autre type pourrait-on utiliser ?

uint32_t (ou unsigned int 32 bits)

Chaque pixel = 1 entier de 32 bits :
0xRRGGBBAA ou 0xAARRGGBB selon l'ordre choisi

Exemple :

```
uint32_t pixel = 0xFF1122CC; // R=0x11,G=0x22,B=0xCC,A=0xFF
uint8_t red   = (pixel >> 16) & 0xFF;
uint8_t green = (pixel>> 8) & 0xFF;
uint8_t blue  = pixel & 0xFF;
uint8_t alpha = (pixel >> 24) & 0xFF;
```

Tracez les histogrammes des fichiers suivants : lena.pgm, anel.pgm, elna.pgm, ovmz.pgm et lena riendesuspect.pgm. Que constatez-vous ?

J’ai utilisé GIMP pour comparer les histogrammes, car ceux que je lis depuis mon code sont erronés : il y a trop de pixels à la valeur 0. Malheureusement, je n’ai pas réussi à résoudre ce problème pour le moment.

Pour les fichiers lena.pgm, anel.pgm, elna.pgm et lena riendesuspect.pgm, les histogrammes sont presque identiques, ce qui montre que la répartition des niveaux de gris reste la même malgré de possibles permutations des pixels.

Pour ovmz.pgm, l’histogramme apparaît inversé ou miroir, indiquant que les valeurs de pixels ont été modifiées ou inversées.

Dans tous les cas, on constate que la majorité des pixels se situe dans des nuances de gris, allant du noir au blanc, c’est-à-dire des tons de gris, blanc et noir prédominants.

Bit de poids faible.
Que constatez-vous ? Observez-vous une différence sur l’histogramme ?

L’image ressemble presque exactement à l’originale. Les modifications sur le bit de poids faible sont très subtiles, donc l’œil humain ne les voit presque pas.

Histogramme : Les pixels impairs ont été convertis en pairs donc les barres correspondant aux valeurs impaires diminuent ou disparaissent et les barres correspondant aux valeurs paires augmentent légèrement.