Rappel de quelques instructions utilisées dans la gestion des couleurs de pixel

1) Les fonctions de conversion pour passer d'une position en X et en Y à un index dans un tableau de pixels, et inversement :

```
int positionXyVersIndex(float posX, float posY, int largeur) {
  int monIndex = int((posY*largeur)+posX);
  return monIndex;
}

int indexVersPositionX (int index) {
  int maPositionEnX;
  maPositionEnX = index%img.width;
  return maPositionEnX;
}

int indexVersPositionY (int index) {
  int maPositionEnY;
  maPositionEnY = index/img.width;
  return maPositionEnY;
}
```

2) La formule permettant de décomposer les valeurs A, R, G, B d'une couleur depuis un pixels, à l'aide de la méthode Bit Shifting :

```
// On récupère la donnée couleur d'un pixel en particulier
int donneeCouleur = pixels[index];

// Pour le rouge :
int r = (donneeCouleur >> 16) & 0xFF;
// Pour le vert :
int v = (donneeCouleur >> 8) & 0xFF;
// Pour le bleu :
int b = donneeCouleur & 0xFF;
// Pour l'alpha :
int a = (donneeCouleur >> 24) & 0xFF;

println(r + ", " + v + ", " + b + ", " + a);
color c1 = color(r, v, b, a);
```

Ou dans une seule ligne :

```
color c2 = color((donneeCouleur >> 16) & 0xFF,(donneeCouleur >> 8) & 0xFF,
donneeCouleur & 0xFF, (donneeCouleur >> 24) & 0xFF);
```

3) La formule permettant d'attribuer d'un seul coup une valeur de couleur à un pixel en utilisant la technique du Bit Shifting :

```
// La situation de traitement pourrait être l'inversion des valeurs trouvées ci-
dessus:
int donneeRouge = abs(r-255);
int donneeVert = abs(v-255);
int donneeBleu = abs(b-255);

pixels[i] = 0xFF000000 | (donneeRouge << 16) | (donneeVert << 8) | donneeBleu;

// Notons qu'avec l'instruction 0xFF000000, l'alpha demandé sera opaque à 100%
(équivaut à la valeur 255)
```