TP – les images vectorielles

Éléments de correction en fin de chaque partie étudiée. **Mon rôle en TP : vérifiez avec vous que vous avez bien compris.**

Objectifs :

- découvrir le principe de codage des images vectorielles fondamentalement différent du principe de codage des images bitmap

- découvrir des avantages des deux types d'images : vectorielles et bitmap (=matricielles)

Préalable : revoir le principe général de codage des images bitmap et le principe général de codage des images vectorielles vus en cours.

Démarche

Vous allez observer le codage d'images vectorielles intégrées dans des **fichiers** avec l'extension **.svg** .

SVG est un langage XML utilisé pour décrire des images vectorielles en 2 dimensions, c'est un langage bénéficiant d'une recommandation du W3C.

Il n'est pas question d'apprendre à utiliser ce langage mais de comprendre, plus précisément, en observant les lignes de description, comment sont décrites les images vectorielles. Et donc comment elles sont codées.

Vous ouvrirez les fichiers contenant les images vectorielles avec l'éditeur de texte Bloc-notes (Notepad) et avec le logiciel Adobe Illustrator.

Dans Bloc-notes vous observerez les lignes de description des images et dans Illustrator vous observerez les images affichées sur l'écran.

Vous utiliserez **Hex Editor Neo** HEN pour observer les contenus en octets (donc afficher en base 16) des fichiers.

Vous utiliserez **Photoshop** pour les images bitmap.

Exercices :

Fichier : **cercleBleu.svg**

A) DESCRIPTION ET CODAGE DE L'IMAGE

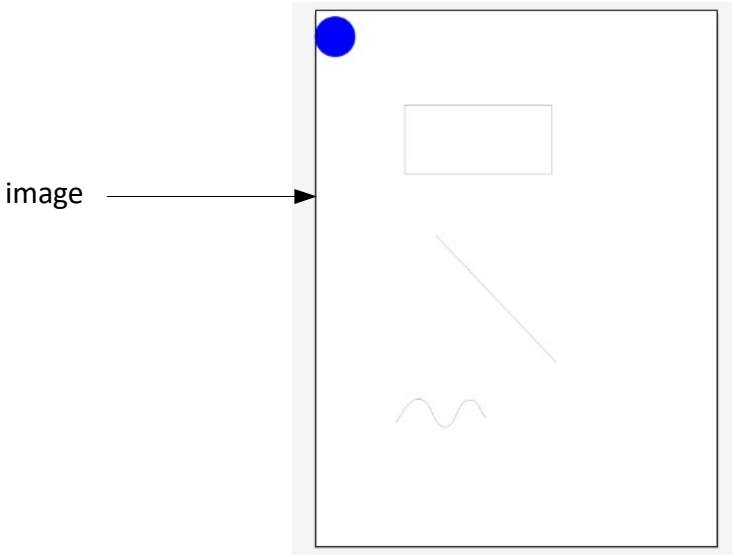
1) Vérifier que la fonction qui permet de décrire le cercle est la fonction **ellipse** caractérisée par des **paramètres** définissant la **couleur de remplissage**, la **couleur du tour du cercle**, les **coordonnées du centre du cercle** et le **rayon du cercle**.

Pour cela changer les caractéristiques géométriques du cercle dans l'éditeur de texte Bloc-notes et vérifier l'effet dans Illustrator.

Relever ces **paramètres**, expliquer leur rôle et commenter leur « valeurs ».

2) Observer avec HEN la ligne de description de la forme géométrique « cercle » et tous les autres caractères présents dans le fichier. Dites avec quels codes ces informations sont inscrites dans le fichier. Justifier.

3) Avec Illustrator ajouter quelques formes géométriques au cercle (dont un rectangle, un segment de droite et un tracé au crayon) et observer leur description dans Bloc-notes. Citer la fonction de description de chaque forme.



Eléments de correction
ASCII
Line pour le segment de droite
Path pour le train crayon
Rect pour le rectangle

B) POIDS DU FICHIER IMAGE

1) Comment pourrait-on trouver le poids du fichier en regardant la colonne centrale de HEN? Vérifier la valeur du poids en utilisant le système d'exploitation de l'ordinateur.

2) Vérifier qu'en ajoutant des espaces sur une ligne vide, entre <g id="rond"> et <ellipse fill= … , le poids du fichier change. Pourquoi ?

3) Même question si vous supprimez un saut de ligne.

Eléments de correction
654 octest
un espace est codé avec l'octet 20 (en base 16)
un saut de ligne est codé sur 2 octets

C) COMPARAISON POIDS IMAGE VECTORIELLE/ IMAGE BITMAP

La description de l'image dans Bloc-note indique la définition 1800 x 2400 pixels.

1) Créer une image bitmap RVB de même définition avec Photoshop et l'enregistrer dans le fichier **cercleBITMAP.bmp.** Relever la valeur du poids avec le système d'exploitation de l'ordinateur.

2) Quel serait le poids d'une image bitmap (et non vectorielle) **indexée** , de même définition, avec, pour rappel, 8 bit/pixel ?

3) Comparer les trois poids. Conclusion ?

Eléments de correction
Environ 12657 Kio /654 Kio
Environ 4219 Kio /654 Kio
les images bitmap contiennent bcq de pixels donc leurs poids sont très élevés par rapport à l'image vectorielle, en effet chaque pixel est codé sur 24 bits ou 8 bits (pas de compression après pour ces deux images bitmap)

Fichier : **petit-cercleBleu.svg** et **petit-cercleBleu.bmp**

la définition de **petit-cercleBleu.bmp** est la définition indiquée dans la description textuelle de **petit-cercleBleu.svg** : 180 x 240 pixels

D) AGRANDISSEMENT D'IMAGE VECTORIELLE ET D'IMAGE BITMAP

1) Changer le code de description de l'image vectorielle **petit-cercleBleu.svg** pour obtenir une image affichée 10 fois plus grande en largeur et en hauteur.

- donner le nom **petit-cercleBLEU-plus-grand.svg**

- l'affichage est il de qualité ?

- relever le poids du fichier image avec le système d'exploitation de l'ordinateur et vérifier que cette valeur est bien en accord avec les modifications faites dans la description de l'image

2) **Modifier le nombre de pixels** , avec le même objectif, de **petit-cercleBleu.bmp** et enregistrer l'image dans le fichier **petit-cercleBLEU-plus-grand.bmp** (avec 24 bit/pixel). Pour cela, avec Photoshop, en partant de l'image dans **petit-cercleBleu.bmp**:

. suivre le chemin : Image/Taille de l'image

. choisir l'algorithme d'agrandissement* *Bicubique plus lisse (adapté à l'agrandissement)*

. cliquer sur Ré échantillonnage (= changer le nombre de pixels)

. **modifier le nombre de pixels**

- l'affichage est il de qualité ? Que penser des algorithmes d'agrandissement* des images bitmap ? Conclusion concernant les images de vos futurs projets ?

- relever le poids du fichier image avec le système d'exploitation de l'ordinateur et justifier l'ordre de grandeur par calculs

3) Conclusion de 1) et 2)?

Eléments de correction affichage de qualité 10 octets de plus mauvaise qualité d'affichage ne jamais sur échantillonner poids élevé 12657 Kio/127 Kio →	→ l'image vectorielle du cercle agrandi n'est pas très lourde par rapport à l'image initiale. l'affichage est de qualité car les pixels ne sont pas codés mais l'image est décrite l'image bitmap agrandie est bcq plus lourde car le nb de pixels a beaucoup augmenté et le poids est proportionnel à la définition de l'image. La qualité d'affichage est médiocre car PS doit inventer des couleurs de pixels qui n'existent pas initialement

E) ZOOM

1) Dans Illustrator, zoomer l'image contenue dans le fichier **petit-cercleBleu.svg** jusqu'à obtenir sur votre écran la même image que l'image contenue dans **petit-cercleBLEU-plus-grand.svg .** En déduire l'opération réalisée sur l'image contenue dans le fichier **petit-cercleBleu.svg** , par Illustrator, pour obtenir l'image zoomée ?

2) Dans Photoshop, zoomer (x 1000%) l'image contenue dans le fichier **petit-cercleBleu.bmp .**

- Comparer l'affichage avec celui de **petit-cercleBLEU-plus-grand.bmp** ? L'affichage est-il de qualité ?

3) Conclusion ?

Eléments de correction
Illustrator a augmenté les valeurs des paramètres comme vus en D avant l'affichage

Photoshop : le nb de pixels a été augmenté pour que la taille de l'image à l'écran soit plus grande, on trouve un affichage de mauvais qualité comme précédemment (D) mais pas avec le même algorithme de sur échantillonnage

Zoomer une image vectorielle ne la détériore pas, son principe de codage permet un affichage de qualité, par contre le principe de codage de l'image bitmap ne permet pas un affichage de qualité car les couleurs de nouveaux pixels doivent être inventés à partir de pixels préexistants

F) VECTORISATION D'UNE IMAGE BITMAP

1) Ouvrir le fichier **mures.bmp** avec Illustrator et vectoriser (.svg) l'image *Fenêtre - Vectorisation de l'image* en choisissant des paramètres adaptés.

- Observer avec attention l'image vectorielle affichée par Illustrator en zoomant. Décrire ce que vous observez. Expliquer.

2) Comparer les poids des deux fichiers. Expliquer.

Elements de correction

Illustrator a essayé de trouver des formes géométriques pour décrire au mieux l'image, mais cela n'est pas très réussi car l'image bitmap est une image photographique : difficile d'obtenir des formes géométriques suffisamment précises. Quand on zoom on voit les formes générées bien nettement. Cela donne l'impression d'un tableau peint.

Le poids de l'image .svg est très grand car la vectorisation a induit un nb lignes de codes de description important

G) AFFICHAGE D'UNE IMAGE VECTORIELLE SUR UN ECRAN

Cette partie ne nécessite aucun logiciel

1) Un observateur ou une observatrice ne peut pas voir une image vectorielle sur un écran sans qu'il y ait **conversion** de cette image en une image bitmap. Justifier cette affirmation.

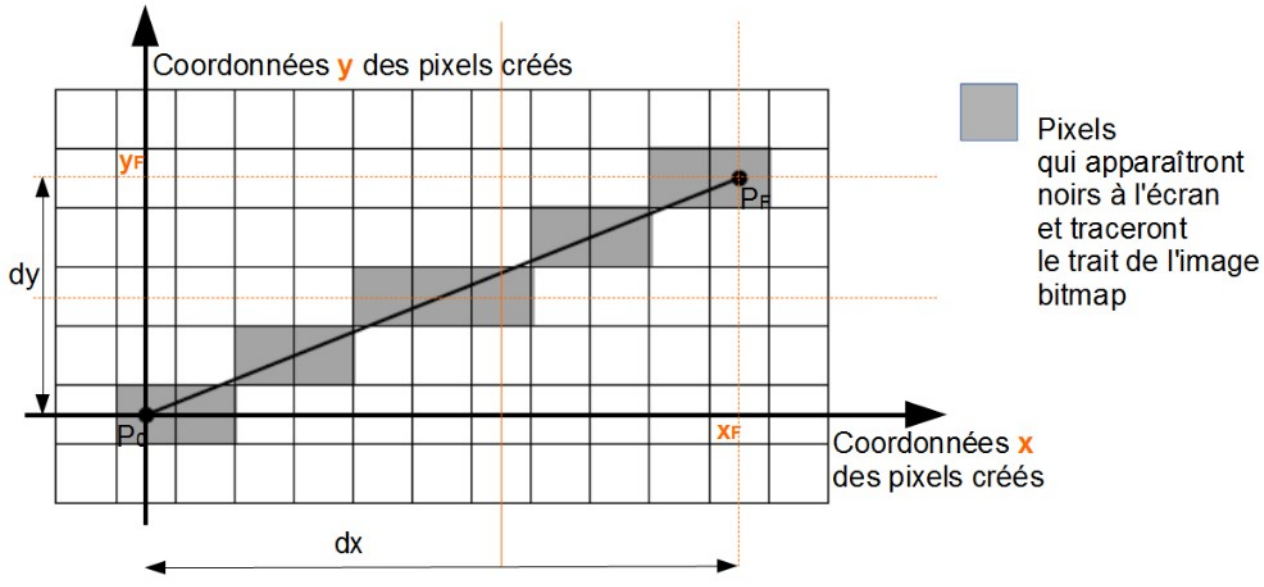
Un fichier d'image vectorielle contient la description d'un segment noir. Les coordonnées des deux points extrémités de ce segment sont données par Illustrator, en pixels : (0;0) et (117; 58).

2) Rappeler le nom de la fonction qui décrit un segment de droite.

L'objectif est de découvrir le premier algorithme historique de **conversion** : l'**algorithme de Bresenham**, qui permet de transformer la description d'un segment vectoriel en une image de catégorie bitmap RVB. Il a été amélioré par la suite.

L'algorithme de conversion détermine les coordonnées entières de tous les pixels du segment qui seront noirs. Par défaut les autres seront blancs.

Le premier pixel noir est le pixel P_0 et ses coordonnées sont (0;0) , le dernier est P_F et ses coordonnées sont (117; 58).



Les coordonnées des pixels calculées sont entières

Le segment [P_0 P_F] est le segment théorique.

3) Suivre le pseudo code de l'algorithme donné ci-dessous et déterminer les coordonnées des 5 premiers pixels générés lors de la **conversion** vectoriel/bitmap.

```
Fonction TraceSegmentPointMilieu(x0, y0, xF, yF : entier,
couleur : caractère) :
    dx, dy, dp, deltaE, deltaNE, x, y : entier
    dx ← xF − x0
    dy ← yF − y0
    dp ← 2∗dy−dx (Valeur initiale de dp)
    deltaE ← 2 ∗ dy
    deltaNE ← 2 ∗ (dy − dx)
    x ← x0
    y ← y0
    DessinePixel (x, y, couleur)
    Tant que (x < xF) faire
        Si (dp ≤ 0) Alors
            (On choisit le point E)
            dp ← dp + deltaE
            x ← x + 1
        Sinon
            (On choisit le point NE)
            dp ← dp + deltaNE
            x ← x + 1
            y ← y + 1
        Fin Si
        DessinePixel (x, y, couleur)
    Fait
Fin
```

Remarques :

- lors de la détermination, par l'algorithme, d'un nouveau pixel à une abscisse N+1 par rapport à l'abscisse N du dernier pixel déterminé, l'algorithme a le choix entre deux pixels : l'un avec la même ordonnée que le pixel précédent et l'autre avec une ordonnée augmentée de 1. Ces deux pixels sont mentionnés dans le pseudo code respectivement par le point E(est) et le point NE (nord-est). Le choix est fait en fonction de la valeur de *dp* qui correspond à la distance entre le point sur le segment théorique mathématique et la moyenne entre les deux ordonnées des points E et NE.

- on doit cet algorithme à **Jack E. Bresenham** qui, en mai 1962, travaillait dans un laboratoire d'informatique d'IBM. Il mit au point son algorithme de tracé de segment alors qu'il cherchait à piloter un traceur attaché à une console texte. Cet algorithme a été publié en 1965 dans la revue IBM Systems Journal. Il a été amélioré ensuite.

Éléments de correction
L’écran doit disposer d’une image RVB pour pouvoir afficher
Line

Dp=-1 dx=117 dy=58 deltaE= 116 deltaNE=-118 **(0;0)**	dp=115 x=1 **(1;0) corrdonnées du second pixel**	Dp=-3 x=2 et y=1 **(2;1)**
Dp=113 x=3 et y=1 **(3;1)**	Dp=-5 x=4 y=2 (4;2)	