SVG + Droste effect = VINS !

1 Introduction

SVG (*Scalable Vector Graphic*) est un langage de balisage (*markup language*) au format XML permettant de décrire des *images vectorielles*.

Contrairement aux images matricielles (*raster* ou *bitmap*, formats BMP, GIF, JPEG, PNG, *etc*), qui ont une taille bien définie et sont décrites par la donnée de la couleur de chacun des pixels, les images vectorielles (formats PDF, PS, SVG) sont décrites par la donnée des formes géométriques élémentaires (lignes, rectangles, cercles, *etc*) qui les composent et n'ont pas de taille fixe.

L'intérêt des images vectorielles est le passage à l'échelle : tandis qu'il n'est pas possible d'agrandir une image matricielle sans faire apparaître tôt ou tard les pixels qui la composent, une image vectorielle peut être affinée à l'infini, à la taille et la résolution que l'on souhaite. Il est par exemple possible de zoomer à l'infini sur les caractères d'un document PDF ou PostScript : à chaque zoom, la portion affichée de la page sera recalculée par le lecteur avec une résolution de plus en plus grande pour que l'image apparaisse nette.

Les formats vectoriels sont inadaptés à la description d'images photographiques (qui se décomposent très mal en un assemblage de formes géométriques élémentaires) mais très performants pour des assemblages géométriques complexes. Le jeu vidéo *Another World* (1991), utilisant un moteur intégralement vectoriel, ce qui était inédit à l'époque, est une belle illustration de cette performance (le format vectoriel permettant de décrire des scènes graphiquement très riches de façon très compacte en mémoire).

Un écran d'ordinateur n'étant qu'une matrice de pixels, l'affichage d'une image vectorielle passe par sa conversion en image matricielle d'une taille donnée, arbitrairement grande. Ce processus est appelé *rasterisation*.

L'objectif de ce projet est d'implémenter la rasterisation d'une variante originale, appelée VINS, du format SVG.

2 Le format SVG

Voici un exemple simple de document SVG et l'image correspondante.

```
<svg width="200" height="200">
   <circle cx="100" cy="100" r="75" fill="red" />
   <rect x="50" y="85" width="100" height="30" fill="white" />
</svg>
```

Une spécification complète, normalisée par le W3C, de SVG peut être trouvée ici : `http://www.w3.org/TR/SVG/`. Dans la suite, nous ne travaillerons qu'avec quelques balises simples, on pourra donc se contenter de parcourir la section *Shapes* du tutoriel du W3C : `http://www.w3schools.com/svg/`.

3 Le format VINS

Le format VINS (*VINS Is Not SVG*) reprendra un sous ensemble minimaliste des balises du langage SVG :

- `<line x1="`x_1`" y1="`y_1`" x2="`x_2`" y2="`y_2`" />`
- `<rect x="`x`" y="`y`" width="`w`" height="`h`" />`
- `<polygon points="`x_1`,`y_1 x_2`,`y_2 ... x_n`,`y_n`" />`
- `<circle cx="`x`" cy="`y`" r="`r`" />`

où toutes les dimensions (x, y, w, *etc*) sont des nombres flottants positifs.

Un document SVG n'étant qu'une liste finie de telles primitives, une limitation importante est qu'il ne permet pas de décrire des objets réellement infinis... Afin de palier ce défaut, le format VINS introduit une unique balise supplémentaire :

`<call target="`*cible*`" x="`x`" y="`y`" width="`w`" height="`h`" />`

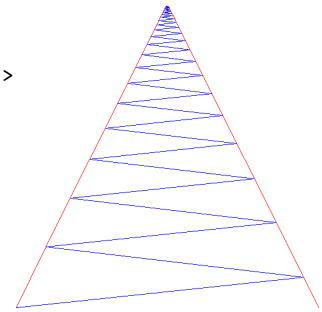
où *cible* vaut soit le nom d'un fichier VINS externe, soit `self` pour désigner le fichier courant, et la balise `call` a pour effet de faire référence à l'image décrite par le fichier cible dans le rectangle défini par les attributs `x`, `y`, `width` et `height`.

Un document VINS complet ne sera qu'une séquence finie de balises parmi les cinq précédentes, précédée par une balise de début `<vins width="`w`" height="`h`">` et terminée par une balise de fin `</vins>`. Les attributs `width` et `height` de la balise `vins` ne définissent **pas** la taille (en pixels) de l'image vectorielle (par définition,

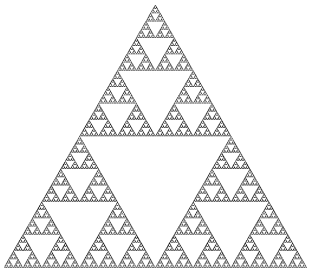
celle-ci n'en a pas) mais les dimensions de la zone de dessin (c'est-à-dire les valeurs maximales des coordonnées flottantes des points manipulés).

On trouvera ci-après quelques exemples simples.

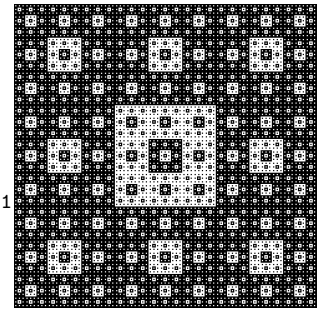
```
<vins width="200" height="200">
  <line x1="20" y1="160" x2="0" y2="200" stroke="rgb(255,0,0)" />
  <line x1="180" y1="160" x2="200" y2="200" stroke="rgb(255,0,0)" />
  <line x1="0" y1="200" x2="190" y2="180" stroke="rgb(0,0,255)" />
  <line x1="190" y1="180" x2="20" y2="160" stroke="rgb(0,0,255)" />
  <call target="self" x="20" y="0" width="160" height="160" />
</vins>
```



```
<vins width="100" height="100">
  <polygon points="50,0␣100,100␣0,100" fill="rgb(0,0,0)" />
  <polygon points="25,50␣75,50␣50,100" fill="rgb(255,255,255)" />
  <call target="self" x="0" y="50" width="50" height="50" />
  <call target="self" x="50" y="50" width="50" height="50" />
  <call target="self" x="25" y="0" width="50" height="50" />
</vins>
```



```
<!-- fichier bisierp0.vins -->
<vins width="300" height="300">
  <rect x="0" y="0" width="300" height="300" fill="rgb(0,0,0)" />
  <call target="self" x="0" y="0" width="100" height="100" />
  <call target="self" x="100" y="0" width="100" height="100" />
  <call target="self" x="200" y="0" width="100" height="100" />
  <call target="self" x="0" y="100" width="100" height="100" />
  <call target="bisierp1.vins" x="100" y="100" width="100" height="1
  <call target="self" x="200" y="100" width="100" height="100" />
  <call target="self" x="0" y="200" width="100" height="100" />
  <call target="self" x="100" y="200" width="100" height="100" />
  <call target="self" x="200" y="200" width="100" height="100" />
</vins>
```



4 À vous de jouer...

Vous devez écrire en Java un programme `vins2png` qui produit une image PNG obtenue par rasterisation à une résolution arbitrairement grande d'un fichier VINS. Ainsi, l'exécution de `./vins2png in.vins 1000 1000 out.png` devra produire une image PNG `out.png` de taille 1000×1000 pixels à partir de l'image VINS `in.vins`.

Afin de parser le XML d'un fichier VINS, vous pourrez utiliser les bibliothèques `javax.xml.parsers` et `org.w3c.dom` afin de récupérer une représentation facile à manipuler et conforme au modèle objet DOM (`http://www.w3schools.com/dom/`) du W3C.

Évidemment, on ne vous demande ni de manipuler directement le format PNG, ni d'implémenter par vous même les primitives de dessin. Vous utiliserez pour cela les bibliothèques `java.awt` (pour les primitives permettant de dessiner des lignes, des cercles, *etc*) et `javax.imageio` (pour enregistrer au format PNG).

5 Compléments possibles

Prendre en charge plus de balises de dessin de SVG ; gérer une partie des styles et couleurs de SVG ; gérer les rotations via un attribut `angle` ajouté à la balise `call` ; gérer au mieux les imprécisions du calcul ; optimiser la gestion des appels récursifs de paramètres identiques (mêmes cible et dimensions) redondants ; proposer une interface graphique avec outil de zoom pour jouer avec l'image en temps réel...