SVG : Scalable Vector Graphics

Christophe Cérin

28 octoble 2003

1 Introduction

SVG est un standard[1] encore en évolution poussé par le W3 consortium qui vise à décrire des objets graphiques par des vecteurs (par opposition à une description par des points comme dans les formats GIF, JPEG) et que l'on peut visualiser avec un navigateur. Il s'agit également d'une norme permettant de gérer des objets retaillables (pour pouvoir facilement les placer dans une mise en page, les zoomer). Les objets graphiques peuvent etre groupés, transformés (déformations axiales), composés dans d'autres objets et recevoir des attributs de style (au sens de CCS). Les objets SVG sont interactifs (création de formulaires, de boutons) et dynamiques (un crcle peut se déplacer dans la zone graphique par exemple). L'animation peut être définie soit à l'intérieur des fichiers SVG soit dans un langage de script externe. De plus, SVG posséde une interface avec DOM (Document Object Model) pour que l'animation puisse accéder à tous les objets et à leurs attributs.

SVG peut être vu comme une extension du langage HTML, comme une évolution du langage de description de pages et de programmation PostScript (introduit par la société Adobe en 1985). Tout comme HTML il s'agit d'un langage à balises. SVG est décrit par une grammaire XML : un document HTML peut donc contenir des graphiques SVG. Il ressemble également à PostScript à qui il emprunte beaucoup de concepts (chemin courant, matrice de transformation, déplacement relatif et absolu, mise à l'échelle...). Contrairement à PostScript, SVG n'est pas un langage de programmation puisqu'il a perdu toutes les instructions d'itération de PostScript par exemple.

Pour des raisons de lisibilité des textes en SVG, nous adoptons une ecriture indenté comme celle qui suit qui vise à aligner verticalement les balises :

```
1: <svg width="200" height="100">
2:      <circle cx="10" cy="86" r="15"
3:                  fill="red"
4:                  stroke="black" />
5: </svg>
```

Les lignes 1 et 5 marquent le début et la fin du code SVG (le cercle est imprimé dans une fenêtre de taille 200 par 100 points. L'objet cercle est décrit entre les balises `<cercle />`. Les attributs du cercle sont cx, cy, fill, stroke : nous les détaillerons plus tard. Un document SVG se compose d'un ou plusieurs fragments

1. `http://www.w3.org/Graphics/SVG/`

délimités par la balise `<svg>` Il peut y avoir plusieurs structures `<svg>` emboitées dans le même document ou dans des documents composites.

Le site du web3 consortium (W3C) est le point d'entrée incontournable pour trouver de l'information (normes, outils, projets...) sur SVG. SVG décrit les objets graphiques en utilisant une grammaire XML. SVG est un standard en évolution, non propriétaire, qui va être amené à évoluer, par exemple vers la présentation des mathématiques et plus précisemment vers l'obtension de réelles qualités typographiques : formatage de textes importants (sur plusieurs pages). Il devrait également offir un support pour la mobilité.

D'autres technologies liées au WEB comme CCS, XSLT, EcmaScript peuvent être appliquées à du code SVG ce qui permet de créer des documents web interactifs, dynamiques et conviviaux.

Comme les objets sont décrits par des vecteurs, l'espace de stockage et donc le traffic Internet entre un serveur et un client est fortement réduit par rapport à une description par des points (image bitmap). C'est le coté client qui interprête le code SVG envoyé depuis un serveur. Pour l'interprétation, des plugins sont disponibles (de chez Adobe ou chez Corel - Mozilla interprête directement le SVG (en fait un sous ensemble de la norme en vigeur)).

De nombreux outils de dessin permettent soit d'exporter en SVG soit d'importer du SVG. Notons Batik et Sodipodi dont vous trouverez les liens sur la page des maitrises. Cependant ces outils gratuits ne gèrent pas les animations.

2 Format des fichiers SVG

Un fichier SVG se présente de la facon suivante : il est composé d'une déclaration XML, de la déclaration de la DOCTYPE puis du fragment de code SVG.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
"http://www.w3.org/TR/SVG/DTD/svg10.dtd">
<svg width="350" height="300" xmlns="http://www.w3.org/2000/svg">
<title>SVG - Introduction</title>
<desc> This graphic demonstrates many exciting features of SVG.</desc>
<!-- commentaires : on met les objets à partir d'ici  -->
</svg>
```

La première ligne est l'instruction standard XML conforme aux spécifications XML 1.0. Elle précise la table de codage des caractères, ici UTF-8 (pas d'accents) et indique que le document dépend d'une définition du type de document (DTD : DOCTYPE) externe au document accessible à l'adresse `http://` mentionnée. La DTD sera appliquée à un élément du document nommé "svg" et indiquera la grammaire et les règles pour celui-ci.

La balise `title` permet de donner un titre et `desc` permet d'associer une description. Ces informations peuvent être utilisées par le navigateur Internet qui charge la page SVG : on peut les considérées comme des méta-données.

3 Objets graphiques de base

Dans ce paragraphe nous allons présenter à travers d'exemples, comment on peut dessiner des formes élémentaires (cercles, rectangles, traits, ellipse, polygone) et nous détaillerons les attributs que vous pouvez spécifier. Tous les attributs ne seront pas passés en revue : à cette fin, veuillez consulter la norme SVG en ligne.

À la fin de ce pragraphe, nous montrons comment nous pouvons définir dans un preambule des objets graphiques et comment les "appeler" (les référencer) plusieurs fois dans le code.

3.1 Les rectangles

C'est la balise `rect` qui permet de les dessiner. On spécifie les dimensions des côtés du triangle et la couleur de remplissage. Il y a un certain nombre de couleurs prédéfinies : voir la documentation en ligne.

```
<svg width="350" height="300">
     <rect x="50" y="30" width="200" height="100" fill="lightgray" />
</svg>
```

Tous les objets graphiques SVG ont un intérieur (nommé fill) et un pourtour nommé stroke. C'est la même chose en PostScript d'ailleurs.

L'intérieur peut être plus ou moins opaque : pour cela, nous disposons de l'attribut `fill-opacity` qui prend ses valeurs sur 0..1 (1 : opaque, 0: transparent).

La propriété `fill-rule` détermine les parties intérieures et extérieures d'un objet qui seront dessinés. L'attribut prend ses valeurs parmi `nonzero | evenodd | inherit`. Ces valeurs correspondent à des méthodes de remplissage d'objets. Nous les détaillerons plus tard.

La propriété `stroke-width` permet de spécifier l'epaisseur d'un trait alors que la propriété `stroke` permet de choisir une couleur pour le trait du contour de la forme graphique.

Les propriétés suivantes permettent également de spécifier des formes pour les traits de contour : `stroke-opacity stroke-linecap stroke-linejoin stroke-dasharray stroke-dashoffset`. ces attributs permettent dans l'ordre de choisir une opacité, une forme pour les "bouts" des traits (arrondis, biseautés, droits), une méthode pour la jointure de deux traits. Les deux derniers attributs permettent de spécifier la forme des pointillés si l'on choisi un tracé en pointillé pour le contour.

Examinons des exemples pour la forme `line` qui permet de tracer des droites.

3.2 Les lignes

```
<svg>
   <line x1="25" y1="30" x2="125" y2="30"
           stroke="darkslategray" stroke-width="4"/>
  <line x1="40" y1="20" x2="200" y2="20"
          stroke-dasharray="1 10" fill="none"
```

```
          stroke="darkslategray" stroke-width="2" stroke-linecap="round" />
  <line x1="40" y1="80" x2="200" y2="80"
         stroke-dasharray="1 10 10 10 1 10" fill="none"
         stroke="darkslategray" stroke-width="2" stroke-linecap="round" />
</svg>
```

On remarque ici que l'on demande à dessiner des droites entre les points de coordonnées (x_1, y_1) et (x_2, y_2).

3.3 Les cercles

L'élément 'circle' possède trois attributs géométriques: `cx, cy, r`. Les attributs cx et cy donnent la position du centre du cercle tandis que r indique le rayon du cercle. Si vous voulez créer un cercle avec un diamètre de 80 unités, vous devez donner à l'attribut r la valeur de 40 comme sur cet exemple :

```
<svg width="350" height="300">
        <circle cx="100" cy="50" r="40"
                  stroke="darkslategrey" stroke-width="2"
                  fill="grey" fill-opacity="0.4"/>
</svg>
```

3.4 Les rectangles

Pour les spécifier on dit à partir de quel point on veut les dessiner et on précise les deux largeurs formant les côtés. On peut spécifier également des rayons permettant de dessiner des coins arrondis.

```
<svg width="500" height="300">
    <!-- Elliptical Rectangles -->
   <rect x="30" y="45" width="80" height="50" rx="50" ry="30"
           fill="lightgrey" stroke="black" stroke-width="1"/>
   <rect x="0" y="40" width="90" height="60" rx="40" ry="30"
          fill="white" stroke="none"/>
    <!-- Horizontal Rectangle -->
     <rect x="50" y="45" width="100" height="50" rx="10"
            fill="none" stroke="black" stroke-width="2"/>
    <!-- Verticle Rectangle -->
     <rect x="20" y="20" width="50" height="100" rx="10" ry="10"
           fill="peachpuff" stroke="forestgreen" stroke-width="2"/>
</svg>
```

3.5 Les ellipses

Pour les spécifier, on précise les coordonnées du centre ainsi que les demi axes de l'ellipse :

```
<svg width="350" height="300">
```

```
<ellipse cx="110" cy="55" rx="70" ry="35"
           stroke="darkslategray" stroke-width="2"
           fill="lightgray" fill-opacity="0.4" />
</svg>
```

Rappel : les attributs de type `stroke, fill` peuvent s'appliquer à une ellipse également ainsi qu'aux éléments graphiques qui suivent.

3.6 Les polygones

C'est l'attribut `point` qui permet de spécifier les points deffinissant le polygone.

```
<svg width="350" height="300">
        <polygon fill="silver" stroke="black" stroke-width="2"
                 points="50,100 50,80 120,50 150,20 200,80 200,100" />
</svg>
```

3.7 Les lignes brisées

Comme pour `polygon`, on utilise l'attribut `point` :

```
<svg width="350" height="250">
   <polyline points="120,20 140,120 180,120 200,20"
        fill="cyan" fill-opacity="0.4" stroke="black"
        stroke-width="6" stroke-linecap="round" stroke-opacity="0.5" />
</svg>
```

Remarquez qu'en executant le code, vous n'avez pas de ligne rejoignant les points de départ et d'arrivée. De plus remarquez qu'il y a une ligne virtuelle entre ces deux points extrème qui nous servent de délimiteur pour le remplissage en cyan.

3.8 Regroupement d'objets

La balise `<g>` permet de regrouper et de nommer des éléments qui se partagent des attributs communs comme la couleur, style. Exemple :

```
<g style="fill:red" id="Grands rectangles rouges">
      <rect x="100" y="100" width="200" height="200 />
      <rect x="300" y="400" width="100" height="100 />
</g>
<g style="fill:blue" id="Petits rectangles bleus">
       <rect x="10" y="10" width="20" height="20 />
       <rect x="30" y="40" width="10" height="10 />
</g>
```

3.9 Définir des objets avant de les utiliser

SVG permet de structurer son code en définissant des objets, des motifs de remplissage etc ayant des attributs déterminés puis de créer des objets sur lesquels on pourra appliquer des propriété définies.

Dans l'exemple qui suit, nous avons prédéfini entre les balises `<defs> </defs>` un motif référencé par "Pat01" une méthode de remplissage référencée par "Grad01" ainsi qu'un cercle dont le remplissage se fait avec la méthode Grad01.

Puis nous avons précisé que le graphique (définis entre les balises `<g> </g>`) comporte un rectangle dont le remplissage se fait avec le motif Pat01 ainsi qu'un autre objet graphique qui est en l'occurence du texte (remarquer au passage les attributs spécifiés pour un objet de type text).

La balise `<g> </g>` dont de spécifier des attributs géneraux qui seront appliqués à tous les objets contenus entre les balises.

```
<svg width="100%" height="100%">
  <title>Scalable Vector Graphics - Introduction</title>
  <desc>This graphic demonstrates many exciting features of SVG.</desc>
  <defs>
        <circle id="Ball01" cx="100" cy="100" r="40" fill="url(#Grad01)"/>
        <pattern id="Pat01" width="10" height="10" patternUnits="userSpaceOnUse">
            <rect width="10" height="10" fill="#FFFFFF" stroke="#000000"
                      stroke-width="0.1"/>
       </pattern>
       <radialGradient id="Grad01" gradientUnits="userSpaceOnUse"
                 cx="120" cy="60"
                 fx="130" fy="50" r="100">
                <stop offset="0%" stop-color="white"/>
                <stop offset="20%" stop-color="#cccccc"/>
                <stop offset="100%" stop-color="rgb(100,20,150)"/>
       </radialGradient>
</defs>
<g letter-spacing="0.02em" enable-background="new">
     <!--FOND-->
     <rect id="Background" fill="url(#Pat01)" stroke-width="0.5"
             stroke="#000000" x="0" y="0" width="100%" height="100%">
            <desc>Background Pattern</desc>
     </rect>
    <g id="watermark" fill="#000000" font-weight="bold" font-size="180"
              font-family="sans-serif,Arial,Verdana" opacity="0.2">
             <text x="50%" y="70%" text-anchor="middle" stroke="none"
                   writing-mode="lr">S V G</text>
    </g>
</g>
</svg>
```

Cette méthode de structuration permet par exemple de prédéfinir des embouts de traits pour dessiner des flêches qui sont composées alors d'un trait et d'un embout :

```
<defs>
```

```
<marker id="Marker01" viewBox="0 0 10 10" refX="0" refY="5"
      markerUnits="strokeWidth" markerWidth="10" markerHeight="6"
      orient="auto">
      <path d="M 0 0 L 15 5 L 0 10 z" />
</marker>
</defs>
<!--Utilisation de la définition-->
<line x1="165" y1="16" x2="165" y2="132"
        stroke="black" stroke-width="1.5"
        marker-end="url(#Marker01)"/>
```

Dans l'exemple précédent, remarquez également la balise `path` qui permet de définir un contour c'est à dire ici le bout de flêche. Nous reviendrons ultérieurement sur cette balise au moment de présenter la description des courbes.

4 Les courbes

En fait, la balise `path` permet de décrire tous les objets graphiques vus précédemment. Il faut donc voir toutes les balises précédentes comme des raccourcis de balises `path`. Avec cette dernière nous allons aussi pouvoir décrire des courbes (de Bézier).

Cette notion de chemin (path) que l'on décrit, que l'on ferme par une méthode stroke est une notion héritée de PostScript.

Le principal attribut de `path` est `d` qui permet de préciser le chemin, la forme de l'objet graphique. En fait, il s'agit de préciser des points qui seront reliés par des courbes ou plus exactement, il s'agit de spécifier un déplacement entre des points au moyen des commandes suivantes :

```
Syntaxe: moveto
( se déplacer sans tracer et ouvrir un nouveau tracé)
M | m x, y
      x = x du nouveau point
      y = y du nouveau point
Syntaxe: lineto
( se déplacer avec un tracé rectiligne depuis le point courant )
L | l x, y
      x = x du nouveau point
      y = y du nouveau point
Syntaxe: closepath
( rejoindre le point de départ du tracé courant avec un tracé rectiligne )
Z | z
Syntaxe: horizontal lineto
( déplacement horizontal avec tracé rectiligne )
H | h x = x du nouveau point
Syntaxe: vertical lineto
( déplacement vertical avec tracé rectiligne )
```

```
V | v y = y du nouveau point
```

Voici un exemple :

```
<path stroke="blue" stroke-width="3" fill="none"
          d="M 40,40
              L 40,80
              L 100,80
              L 100,40
              M 40,30
              L 70,10
              L 100,30
              Z" />
```

Note : chaque élément path doit commencer par un moveto.

Le chemin (path) peut être également composé d'arcs de cercle. Pour cela nous avons à notre disposition la commande A ou a (en minuscule) dont la description complète peut se trouver sur le WEB. Prenez quelques minutes pour comprendre les options de cette commande qui est technique.

Le chemin (path) peut également comporter des courbes de Bezier qui sont obtenues (par exemple) à partir d'équations paramétriques en consiérant 3 points : le point courant, un point dit de contrôle et un point d'arrivée. Les équations qui permettent le rendu de la courbe entre le point courant et le point d'arrivée ne sont pas explicitées ici (voir le polycopié PostScript).

En ce qui concerne SVG, c'est la commande Q qui permet de spécifier une courbe de Bézier. Par ailleurs, la commande T permet de s'affranchir de spécifier le point de contrôle. Cette commande demande simplement les coordonnées du nouveau point, fin de tracé de la courbe de Bézier. Le point de contrôle pour ce tracé est créé automatiquement, il est le symétrique du précédent point de contrôle par rapport au point courant. Ceci suppose qu'une commande Q soit utilisée avant les commandes T afin de donner au moins un point de contrôle.

Enfin, la commande C permet de créer des courbes de Bézier quadratiques qui introduisent deux points de contrôle. Ce format permet de créer des courbes avec des rendus encore plus variés que précédemment.

Attention, pour toutes ces commandes, nous avons utilisé les lettres majuscules, mais nous pouvons aussi utiliser les lettres minuscules. La différence est importante, avec les majuscules les paramètres sont définis dans le système de coordonnées de l'élément `<path>` alors que pour les minuscules les paramètres sont donnés dans un système de coordonnées relatif au point courant.

4.1 Remplissage d'un chemin

Une fois qu'un chemin a été défini, son "intérieur" peut être rempli. Intuitivement, les algorithmes de remplissage se basent sur un rayon qui traverse le contour (le path) et selon le mode de remplissage choisi, on détermine ce que l'on doit faire (remplir ou pas) la portion du rayon entre deux points appartenant au contour.

En fait, on lance un rayon (une droite virtuelle) et on décide, pour tous les points sur cette droite virtuelle s'ils sont à l'intérieur ou à l'extérieur du contour. S'ils sont à l'intérieur, le point est peint, sinon on ne fait

rien. L'orsque le point considéré du rayon intersecte un point du contour on applique une règle (spécifiée par les atributs evenodd, nonzero ou inherit) pour décider si on passe de l'état "être à l'intérieur" à l'état "être à l'extérieur" du contour (ou l'inverse).

De plus, la règle fait intervenir le "sens" du dessin c'est à dire si le trait est déssiné de la gauche vers la droite ou de ladroite vers la gauche c'est à dire que l'on considère que le chemin est orienté.

Reportez vous maintenant à la norme pour examiner tous les détails techniques.

4.2 Marqueurs sur une courbe

Nous ne pouvons placer ces marqueurs qu'à des points précis - les sommets (points de départ et d'arrivée) de notre tracé. L'attribut `marker-start` place un marqueur au point de départ de notre tracé, `marker-end` le place à la fin du tracé et enfin `marker-mid` place un marqueur à chaque sommet intermédiaire. Si nous voulons les trois à la fois nous pouvons utiliser simplement le raccourci `marker`. Les spécifications ne traitent pas explicitement de l'ambiguité d'avoir à la fois 'marker-start' et 'marker-end' pour des courbes. Aussi le résultat peut dépendre du visualiseur utilisé.

L'idée des marqueurs est de pouvoir placé automatiquement des objets le long d'une courbe. Dans la version actuelle de la norme SVG, cette notion n'est pas complètement aboutie. Par exemple, on ne peut pas spécifier (sans passer par unscript) que l'on place un objet à "intervales réguliers le long d'un chemin".

5 Afficher du texte

SVG nous permet avec l'objet 'text' d'afficher des titres, des légendes, des boutons et même des paragraphes complets de texte. Les éléments `text, tspan` sont à la base de l'affichage de texte en SVG. L'élément 'text' nécessite la donnée des attributs "x" et "y" qui définissent le point de départ de l'affichage de notre texte. Voici l'exemple classique "Hello world" en SVG.

```
<svg width="350" height="300">
     <text x="30" y="20"
              fill="black" font-size="10">
              Hello world.
     </text>
     <text x="30" y="50"
            fill="black" font-size="10">
            I&apos;m a Scalable Vector Graphic!
     </text>
</svg>
```

La syntaxe de text est la suivante:

```
<text id="name"
   x="coordinate+"
   y="coordinate+"
   dx="lengths+"
```

```
   dy="lengths+"
   rotate="numbers+"
   textLength="length"
   lengthAdjust="spacing|spacingAndGlyphs"
   transform="transform properties"
   style-attribute="style-attribute">
   <!-- texte ö afficher -->
</text>
```

Les attributs `dx, dy` permettent de spécifier un décalage par rapport au point (x, y). L'attribut `rotate` permet de faire une orientation du texte. La définition des autres attributs peut se trouver sur le WEB.

La balise `<tspan> </tspan>` ressemble à la balise précédente sauf qu'elle permet d'appliquer des positionnements automatiques de chaque caractère du texte concerné. On peut donc ajuster la position du texte, la valeur du texte ou la police du texte grace à l'élément `<tspan>`.

Il est souvent préférable de rassembler dans une formle "text" plusieurs lignes de texte délimitées par les balise `<tspan></tspan>` ceci afin de créer des blocs de texte sur lesquels ont pourra agir globalement. Voici un exemple qui n'utilise cependant pas d'identificateur pour référencer les 4 lignes de texte :

```
<?xml version="1.0"?>
<svg width="350" height="300">
   <text x="10" y="30" fill="black" font-size="10">
      <tspan>They might not need me - yet they might -</tspan>
      <tspan x="10" dy="20">I'll let me Heart be just in sight -</tspan>
      <tspan x="10" dy="20">A smile so small as mine might be </tspan>
      <tspan x="10" dy="20">Precisely their necessity -</tspan>
  </text>
</svg>
```

Les attributs `fill` et `stroke` permettent de nombreux rendus pour les textes : voir les discussions précédentes pour les usages.

SVG permet de spécifier de nombreux paramètres pour les polices de caractères. Les attributs principaux sont 'font-weight', 'font-style' et 'text-decoration'. Le premier permet de spécifier une épaisseur de trait pour la police, le deuxième permet de typer la police (italique, oblique, normal...), le troisième permet d'ajouter un sous-lignement, un clignotement par exemple.

La propriété 'text-rendering' peut affecter la qualité du rendu de votre texte selon la méthode deremplissage choisie, selon que votre police soit vectorielle ou pas. Certaines propriété permettent de justifier votre texte comme dans l'exemple qui suit :

```
<svg width="350" height="300">
   <text x="70" y="30" font-size="15">
          start
   </text>
  <text x="70" y="50" text-anchor="middle" font-size="15">
          middle
  </text>
```

```
  <text x="70" y="70" text-anchor="end" font-size="15">
       end
  </text>
</svg>
```

SVG permet également de faire de l'interlettrage c'est à dire d'ajouter ou de retirer du blanc entre les caractères ou entre les mots. Les attributs associés sont 'letter-spacing', 'word-spacing' et 'kerning'. Voici un exemple d'usage de ces attributs :

```
<svg width="350" height="300">
   <text x="10" y="20" fill="black" font-family=""
          word-spacing="normal">Word spacing is normal.</text>
   <text x="10" y="40" fill="black" font-family=""
          word-spacing="5">Word spacing is 5.</text>
   <text x="10" y="60" fill="black" font-family="Lucida Handwriting"
         letter-spacing="normal">Letter spacing is normal.</text>
   <text x="10" y="80" fill="black" font-family="Lucida Handwriting"
        letter-spacing="3">Letter spacing is 3.</text>
   <text x="10" y="100" fill="black" font-family="Georgia"
        kerning="auto">Kerning is auto.</text>
   <text x="10" y="120" fill="black" font-family="Georgia"
        kerning="0">Kerning is 0.</text>
   <text x="10" y="140" fill="black" font-family="Georgia"
       kerning="3">Kerning is 3.</text>
</svg>
```

Vous pouvez, avec ces propriétés 'textLength' et 'lengthAdjust' forcer le texte à occuper une largeur précise. La propriété "rotate" permet de faire tourner des lettres ou des lignes complètes de texte. Voici un exemple :

```
<svg width="350" height="300">
       <text x="40" y="30" fill="black"
               font-size="20" rotate="-30">Rotate word</text>
       <text x="40" y="60" fill="black"
              font-size="20" rotate="-40 -35 -30 -20 -10">
              Rotate letters</text>
       <text x="40" y="90" fill="black" font-size="20"
              rotate="0 0 0 0 0 0 0 -20 -15 0 15 20">
             <tspan>Rotate tspan</tspan>
       </text>
</svg>
```

La propriété 'writing-mode' permet par exemple de positionner verticalement le texte. La balise `<textpath> </textpath>` permet de positionner un texte le long d'une courbe de bézier : voir la documentation en ligne.

On peut imposer au texte de suivre un chemin prédéfini par la balise `<textPath starOffset="longueur | pourcentage" xlink:href="uri" />`. starOffset est le décalage par rapport au début du texte,

longueur représente une distance le long du chemin mesurée selon la métrique de l'espace utilisateur et pourcentage représente un pourcentage par rapport au chemin entier toujours selon la métrique de l'espace utilisateur

5.1 Incorporation de polices

Avec SVG, vous pouvez définir vos propres polices de caractère et les utiliser comme police par défaut. Voici un exemple remodelé à partir des exemples fournis avec Batik (`http://xml.apache.org/batik/`). Nous remarquouns que chaque caractère est décrit par une courbe de Bézier, que la police porte le nom de 'SciFiFont et qu'elle est utilisée de façon habituelle via l'attribut 'font-family'. Les différents attributs de l'élément 'font' ne sont pas commentés ici : voir la documentation en ligne.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20000303 Stylable//EN"
  "http://www.w3.org/TR/2000/03/WD-SVG-20000303/DTD/svg-20000303-stylable.dtd">
<svg viewBox="0 0 861.75 279" xml:space="preserve">
<desc>examples de fontes</desc>

  <font horiz-adv-x="858" ><font-face
               font-family="SciFiFont"
               units-per-em="1000"
               panose-1="0 0 4 0 0 0 0 0 0 0"
               ascent="825"
               descent="-200"
               alphabetic="0" />
               <!-- Embeded font: "ZeroHour" font from Ray Larabie.
                    See http://www.larabiefonts.com -->
               <missing-glyph horiz-adv-x="500" d="M63
                    0V800H438V0H63ZM125 63H375V738H125V63Z" />
               <glyph unicode="A" glyph-name="A" horiz-adv-x="984"
                   d="M948 0H828V320H148Q148 327 148 0H28V382Q28 560
                   126 660T406 760H948V0ZM828 640Q757 640 650 641T505
                   643Q305 643 241 609Q169 571 154 440H828V640Z" />
               <glyph unicode="B" glyph-name="B" horiz-adv-x="985"
                   d="M942 760V378Q942 200 844 100T564 0H22V382Q22
                   560 121 660T400 760H942ZM822 640H404Q276 640 216
                   595T145 440H822V640ZM819 320H142V120H560Q688 120
                   748 165T819 320Z" />
               <glyph unicode="I" glyph-name="I" horiz-adv-x="245"
                   d="M194 0H74V760H194V0Z" />
               <glyph unicode="K" glyph-name="K" horiz-adv-x="966"
                   d="M956 69L881 -24L328 418L131 339V0H11V760H131V469L905
                   778L949 667L457 470L956 69Z" />
               <glyph unicode="T" glyph-name="T" horiz-adv-x="886"
                   d="M924 640H524V0H404V640Q281 640 220 606Q164 574
                   143 499Q127 446 120 317L1 324Q13 546 86 643Q173
                   760 382 760H924V640Z" />
           </font>

       <g font-size="80" text-anchor="middle" font-family="SciFiFont"
          transform="translate(225,200)" >
```

```
            <text x="-1" y="-1" fill="black" >BATIK</text>
            <text x="-1" y="-110" fill="red" >BATIK</text>
        </g>
</svg>
```

6 Systèmes de coordonnées et transformations

Les spécifications SVG définissent une région rectangulaire finie dans laquelle seront rendus les objets graphiques. Cette fenêtre est nommée 'Viewport', nous traduirons ce terme par *fenêtre de visualisation*. Le point $(0, 0)$ est situé en haut à gauche.

SVG permet de définir plusieurs fenêtres de visualisation comme descendants de l'élément racine. Voici un exemple :

```
<svg width="200" height="100">
      <circle cx="10" cy="86" r="15" fill="red" stroke="black"/>
      <svg x="125" y="15" width="50" height="70">
             <circle cx="10" cy="60" r="15" fill="blue" stroke="black"/>
     </svg>
</svg>
```

Dans cet exemple le second repère est défini à l'intérieur du repère principal, son origine est localisée en $(125, 15)$, il a 50 unités de large et 70 unités de haut, tout ceci exprimé dans le repère parent, ici le repère de l'élément racine.

6.1 Transformations élémentaires

Les spécifications SVG fournissent l'attribut 'transform' pour redimmentionner par exemple un objet. Cet attribut peut être appliqué à la grande majorité des éléments graphiques. Il peut être également appliqué à l'élément `<g>` qui permet de définir (regouper) des objets. Voici sa syntaxe :

```
transform = "translate(tx [ ty])
                    rotate(angle [cx cy])
                    scale(sx [sy])
                    skewX(angle)
                    skewY(angle)"
```

L'attribut 'translate' permet de déplacer le système de coordonnées de l'objet, l'attribut 'rotate' permet de faire tourner l'objet autour d'un centre spécifié par cx, cy, 'scale' permet de le (re)-dimentionner et les attributs skewX et skewY correspondent à des déplacement angulaires des axes de x et des y (les valeurs doivent être comprises entre -90 et 90). Ceci permet par exemple d'avoir un rendu en perspective.

L'attribut 'transform' supporte un dernier type de transformation, le plus général, une transformation représentée par une matrice. c'est encore un héritagede PostScript. Nous ne détaillons pas cette possibilités ici et nous renvoyons le lecteur à la documentation en ligne.

7 Les gradients

Avec SVG, on peut remplir un objet ou en souligner le contour par une couleur, un gradient (de couleurs), un motif. Un gradient de couleurs consiste en une transition douce entre deux couleurs selon un vecteur. Les gradients de couleur peuvent être linéaires (de la gauche de l'objet vers la droite par exemple) ou radiaux (du centre de l'objet vers les bords par exemple). Voici un exemple de gradient linéaire :

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg SYSTEM "svg-19991203.dtd" >
<svg width="500px" height="600px" >
  <g style="text-rendering:optimizeLegibility;shape-rendering:default">
    <defs>
      <linearGradient id="grad1" x1="0" y1="0" x2="400" y2="400">
        <stop offset="0%"   style="stop-color:#FF0000"/>
        <stop offset="25%"  style="stop-color:#0000FF"/>
        <stop offset="50%"  style="stop-color:#00FF00"/>
        <stop offset="75%"  style="stop-color:#0000FF"/>
        <stop offset="100%" style="stop-color:#FF0000"/>
      </linearGradient>
    </defs>
    <text x="5" y="20" style="font-size:22">SVG Demo: Linear gradients</text>
    <path style="fill:url(#grad1)" d="M0 50 h200 v200 h-200 z"/>
    <circle  style="fill:url(#grad1)" cx="320" cy="150" r="100" />
    <polygon style="fill:url(#grad1)" points="0,400,200,400,100,280" />
    <ellipse style="fill:url(#grad1)" cx="300" cy="350" rx="100" ry="75"/>
    <rect style="fill:url(#grad1)" x="0" y="450" width="400" height="100" />
  </g>
</svg>
```

Voici un exemple de gradient radial :

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg SYSTEM "svg-19991203.dtd" >
<svg width="500px" height="500px" >
  <g style="text-rendering:optimizeLegibility;shape-rendering:default">
    <defs>
      <radialGradient id="grad1" cx="250" cy="250" r="300" fx="250" fy="250">
        <stop offset="0%"   style="stop-color:#FF0000"/>
        <stop offset="25%"  style="stop-color:#0000FF"/>
        <stop offset="50%"  style="stop-color:#00FF00"/>
        <stop offset="75%"  style="stop-color:#0000FF"/>
        <stop offset="100%" style="stop-color:#FF0000"/>
      </radialGradient>
    </defs>
    <text x="5" y="20" style="font-size:22">SVG Demo: Radial gradient</text>
    <rect style="fill:url(#grad1)" x="50" y="50" width="400" height="400" />
  </g>
</svg>
```

Un motif est utilisé pour remplir ou dessiner le contour d'un objet en utilisant un objet graphique pré-défini qui peut être dupliqué à intervalles fixes en x et en y pour couvrir l'aire à peindre. voici un exemple de motif

qui servent de contours :

```
<?xml version="1.0"?>
<!DOCTYPE svg SYSTEM "svg-19991203.dtd" >
<svg width="600px" height="500px" >
  <defs>
     <pattern id="mypattern" x="0" y="0" width="30" height="30">
       <rect width="30" height="30" style="fill:yellow" />
       <path style="fill:none;stroke:blue" d="M0,0 L 30,30 M0,30 L 30,0" />
     </pattern>
  </defs>
  <g style="text-rendering:optimizeLegibility" >
    <text x="5" y="20" style="font-size:22">
       SVG Demo: Filling and stroking with a custom pattern
    </text>
  </g>
  <g style="stroke:none; shape-rendering:default" >
    <ellipse cx="150" cy="150" rx="100" ry="100"
           style="fill:url(#mypattern)" />
    <rect x="300" y="50" width="150" height="150"
           style="fill:url(#mypattern)" />
    <ellipse cx="250" cy="350" rx="200" ry="75"
           style="stroke:url(#mypattern);fill:none;stroke-width:20" />
  </g>
</svg>
```

SVG permet aussi de masquer des zones ou de découper des formes pour rendre visible l'intérieur par exemple. Dans le texte qui suit, une ellipse est définie puis une image bitmap est affichée mias seulement la partie située à l'intérieur de l'ellipse sera visible. Cette notion de chemin de clipping est héritéee de PostScript.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg SYSTEM "svg-19991203.dtd" >
<svg width="500" height="500" >
  <defs>
    <clipPath>
      <ellipse cx="200" cy="220" rx="80" ry="120" id="myClip"  />
    </clipPath>
  </defs>
  <g style="text-rendering:optimizeLegibility;shape-rendering:default" >
    <text  x="5" y="25" style="font-size:24">
         SVG Demo: An image clipped by an ellipse.
    </text>
    <ellipse cx="200" cy="220" rx="90" ry="130" style="fill:#CE8A77" />
      <image x="40" y="100" width="320" height="240"
                 xlink:href="krl1.jpg" style="clip-path:URL(#myClip)"/>
  </g>
</svg>
```

8 Interactivité et animation

8.1 Eléments d'interactivité

Nous avons présenté jusqu'à maintenant les principaux éléments vous permettant de dessiner en SVG. Nous avons laissé de côté les notions de filtres et de masques qui permettent une gestion fine de motifs bitmap.

SVG permet en plus de réaliser les fonctions suivantes :

- répondre aux actions de l'utilisateur, comme presser un bouton du pointeur (par exemple, une souris) peuvent démarrer des animations ou déclencher l'exécution d'un script.
- L'utilisateur peut suivre des liens vers d'autres pages Web (voir les liens externes en SVG: l'élément 'a') par des actions telles qu'un click de souris quand le pointeur est sur des éléments graphiques particuliers.
- Dans de nombreux cas, suivant les valeurs de l'attribut 'zoomAndPan' de l'élément 'svg' et les caractéristiques du rendu, les utilisateurs peuvent zoomer ou faire un panoramique sur le contenu SVG.
- Les déplacements du pointeur peuvent provoquer des changements dans le dessin du curseur indiquant la position du pointeur.

Syntaxe de l'élément 'a':

```
<a id="name"
   xlink:href = '<uri>'
   xlink:type = 'simple'
   xlink:role = '<uri>'
   xlink:arcrole = '<uri>'
   xlink:title = '<string>'
   xlink:show = 'new | replace'
   xlink:actuate = 'onRequest'
   target = "<frame-target>" >
   <!-- objets svg pour activer le lien -->
</a>
```

Exemple de code:

```
<a xlink:href="MyFile.svg">
    <rect x="0" y="0" width="150" height="150" style="fill:green"/> </a>
```

Quand le pointeur est sur ce rectangle, il prend la forme d'une main et si l'utilisateur clique, le fichier "MyFile.svg" est ouvert dans la même fenêtre.

Nous pouvons choisir comme lien n'importe quel élément du document en utilisant son attribut 'id', mais comme le contenu SVG représente souvent un dessin, une image, il est intéressant de pouvoir accéder à une vue particulière du document. Ceci se réalise avec la balise `<view>`. Exemple :

```
<defs> <view id="MyView" viewBox="150 0 200 200"/> </defs>
```

```
\end{veerbatim}

Le lien est alors codé comme ceci :

\begin{verbatim}
<a xlink:href="#MyView"> <!-- élément activateur --> </a> ou
<a xlink:href="#xpointer(id('MyView'))"> <!-- élément activateur --> </a>
```

Avec cette méthode, on peut créer un zoom quand on clique sur une région.

8.2 Eléments d'animation

Les graphiques SVG peuvent être animés par les élements d'animation propre à SVG ou par l'interface DOM de SVG.

SVG offre différents éléments d'animation, 'animate', 'animateMotion', 'animateColor', 'animateTransform' et 'set'. Ils seront en général placés comme descendants de l'objet graphique à animer. Cela veut dire que l'on commence par définir toutes les propriété de l'objet autres que l'animation, puis on "applique" une animation sur cet objet.

Les éléments suivants ont pour rôle :

animate : permet aux attributs scalaires et aux propriétés de changer avec le temps ;

set : raccourci pour assigner des valeurs d'animation et des attributs et propriétés non numériques ;

animateMotion : déplace un élément le long d'un chemin ;

animateColor : modifie la couleur d'attributs ou de propriétés au cours du temps ;

animateTransform : modifie les attributs des transformations au cours du temps (attributs path, keyPoints, rotate).

Une animation consiste à modifier les valeurs d'un attribut de l'élément cible. Par exemple, quand l'utilisateur clique sur l'objet qui a l'identificateur 'red' (événement 'red.click'), le rectangle qui était blanc devient rouge.

SVG utilise 'attributeName' pour accéder au nom de l'attribut ou de la propriété et 'attributeType' pour accéder à son type (CCS, XML).

Vous pouvez contrôler la durée de l'animation, son départ par les attributs 'begin'. On peut itérer une fois, deux fois ou indéfiniment l'animation.

L'élément 'set' permet simplement d'attribuer une nouvelle valeur ö un attribut pour une durée spécifiée ou sur un évènement. Ceci permet de créer des menus déroulant par exemple ou les options changent de couleur quand on passe sur l'option.

Passons en revue les principaux éléments permettant d'animer un objet, une propriété. Le premier élément est `<animate>`. Voici un exemple et reportez vous à la documentation en ligne pour la signification précise des attributs de cet élément :

```
<svg  viewBox="0 0 320 200">
```

```
  <animate attributeName="viewBox"
           values="0 0 720 800;
                   50 50 100 100;
                   150 50 200 100;
                   0 0 720 800"
           dur="8s"
           repeatDur="indefinite"/>
   <g id="dessin" >
   <rect x="10" y="20" width="100" height="200"
         style="fill:yellow; stroke:blue;stroke-width:4" />
   <circle style="fill:blue;stroke:yellow;stroke-width:4"
           cx="250" cy="160" r="90" />
</svg>
```

Dans l'exemple qui suit, différents attributs sont animés :

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20000303 Stylable//EN"
  "http://www.w3.org/TR/2000/03/WD-SVG-20000303/DTD/svg-20000303-stylable.dtd">
<svg xml:space="preserve" width="400" height="400">
<desc>examples of animations</desc>
<circle style="fill:red;" id="c1" cx="25" cy="25" r="20"/>
<!-- red disk gets very big -->
<animate attributeName="r" attributeType="XML"
    begin="1s" dur="10s" from="20" to="800"
    xlink:href="#c1" fill="freeze" />
<circle style="fill:blue;" id="c2" cx="350" cy="50" r="20"/>
<!-- blue disk goes left repeatedly -->
<animate attributeName="cx" attributeType="XML"
   begin="3s" dur="2s" repeatCount="10" by="-200"
   xlink:href="#c2" />
<circle style="fill:yellow;" id="c3" cx="50" cy="250" r="20"/>
<!-- yellow disk goes up and down -->
<animate attributeName="cy" attributeType="XML"
   begin="5s" dur="3s" repeatCount="10" values="250;50;250"
   xlink:href="#c3" />
<!-- yellow disk goes smaller and bigger -->
<animate attributeName="r" attributeType="XML"
   begin="7s" dur="6s" repeatCount="indefinite" values="20;1;20"
   xlink:href="#c3" />
<circle style="fill:green;" id="c4" cx="250" cy="250" r="50"/>
<!-- green disk goes transparent -->
<animate attributeName="opacity" attributeType="CSS"
   xlink:href="#c4" from="1" to="0" dur="5s"
   repeatCount="indefinite" />
</svg>
```

Dans l'exemple qui suit, le texte `Christophe` se déplace le long d'un chemin :

```
<svg width="400" height="300" viewBox="-20 -20 400 300">
```

```
 <path style="fill:none;stroke:green;stroke-width:2"
       d="M 200 100
          c 0 55 -44 100 -100 100
          s -100 -44 -100 -100
          C 0 45 45 1 100 1
          s 100 44 100 100 z "/>
  <text style="fill:black;font-size:24;">
  Christophe
  <animateMotion dur="15s"
         repeatCount="indefinite"
         rotate="auto"
         path="M 200 100
               c 0 55 -44 100 -100 100
               s -100 -44 -100 -100
               C 0 45 45 1 100 1
               s 100 44 100 100" />
  </text>
</svg>
```

Autre exemple : nous voulons faire évoluer les valeurs de l'attribut 'd' d'un élément 'path', la courbe se déformera. Nous pouvons même avoir un morphing - passage continu d'une forme à la suivante. Dans l'exemple qui suit, le carré devient progressivement une étoile et on change de couleur durant lestransformations :

```
<svg width="600" height="400" viewBox="-300 -200 600 400">
    <rect x="-300" y="-200" width="600" height="400" style="fill:#E0E0E0"/>
    <path d="M-100 -100l100 0 100 0 0 100 0 100 -100 0 -100 0 0 -100z"
      style="stroke:blue;fill-opacity:0.8; fill:blue">
      <animate begin="go.click" dur="10s" repeatCount="1" attributeName="d"
            values="M-100 -100l100 0 100 0 0 100 0 100 -100 0 -100 0 0 -100z;
            M-20 -20l20 -80 20 80 80 20 -80 20 -20 80 -20 -80 -80 -20z;
            M-100 -100l100 0 100 0 0 100 0 100 -100 0 -100 0 0 -100z"/>
      <animate begin="go.click" dur="10s" repeatCount="1" attributeName="fill"
            values="blue;green;blue"/>
    </path>
    <text x="250" y="180"
          style="font-size:25;fill:black;text-anchor:middle"> GO </text>
    <rect id="go" x="220" y="155" width="60" height="30"
         style="fill:black;fill-opacity:0.1"/>
</svg>
```

L'élément `animatecolor` permet d'animer uniquement une couleur. L'élément 'animateTransform' s'applique uniquement à l'attribut 'transform' et les valeurs qu'il peut utiliser sont 'translate(tx,ty)', 'rotate(r,cx,cy)', 'scale(sx,sy)', 'skewX(a)' et 'skewY(a)'. SVG offre donc le moyen d'animer un texte en le faisant tourner en le redimentionnant etc

L'élément 'animateMotion' provoque le déplacement d'un objet le long d'une courbe définie par 'path'. Les types d'objets qui peuvent être déplacés sont : 'g' , 'defs' , 'use' , 'image' , 'switch' , 'path' , 'rect' , 'circle' , 'ellipse', 'line', 'polyline', 'polygon' , 'text' , 'clipPath' , 'mask' , 'a' et 'foreignObject'. Reportez vous à la documentation en ligne pour les objets que nous n'avons pas présenté dans ce texte.

Pour définir la trajectoire sur laquelle l'objet se déplace, nous pouvons utiliser l'élément 'mpath' descendant de l'élément 'animateTransform' qui fera référence à un élément 'path' défini dans le document avec l'attribut 'xlink:href'. Au lieu de :

```
<animateMotion path="M100,250 C 100,50 400,50 400,250" ..... />
```

Nous pouvons écrire :

```
<defs> <path id="MyPath" d="M100,250 C 100,50 400,50 400,250" /> </defs>
<animateMotion ..................... >
         <mpath xlink:href="#MyPath" />
</animateMotion>
```

9 Conclusion

Cette introduction permet d'appréhender les notions importantes du langage de description SVG. Il reste à examiner l'interface de SVG avec d'autres langages, l'écriture de scripts et d'explorer la définition de motifs et de gradients de couleur. Enfin, le domaine des polices de caractères (de leurs définitions jusqu'à leurs utilisations) restent un domaine qui est encore du domaine des propositions dans le cadre du W3C. Pensez donc à suivre l'évolution du langage dans les prochaines années.