Tutoriel SVG

Table des matières

1 Introduction

Index

Le format SVG est basé sur le XML (langage de spécification de données). A cet égard, le fichier d'une image SVG (format *.svg*) est un fichier textuel (éditable avec l'importe quel éditeur de texte : WordPad, NEdit, ...). Il s'agit d'une suite de **balises** (ouvrante/fermante ou simple) qui décrivent des **objets**. Un plug-in permettra aux navigateurs internet d'interpréter ces fichiers comme des images proprement dit. Les visualiseurs d'**Adobe**, **Corel** ou Batik d'Apache peuvent également être utilisés. Le format SVG est une alternative **libre de droit** au format Flash. Il permet la

construction d'images vectorielles complexes (Formes de bases, utilisation d'effets, transformation géométriques, ...), d'interaction à l'aide de script et d'animations dynamiques.

Dans ce tutorial, on explique comment construire 'à la main' un fichier SVG afin d'en comprendre les principes généraux. En aucun cas il ne se veut exhaustif, il faudrait pour cela se référer à la spécification du W3C (traduite en français ici). La version utilisée est 1.1, la version 1.2 étant en construction lors de la construction de ce tutorial. Certain des exemples présentés sont extraits de ces documents.

AVERTISSEMENT Ce tutorial est loin d'être complet et peut faire apparaître même des erreurs liées au fait qu'il a été rédigé en utilisant un savant mélange entre la version 1.0 et la vrsion 1.1. Il a uniquement pour objectif de simplifier une utilisation rapide et de survole du format SVG dans le cadre d'une formation centrée sur la compréhension de ce qu'est une image vectorielle et sans aucun but de maîtrise de la spécification du SVG.

1.1 Première image

Index

L'exemple ci-dessous est de profile de base d'une image SVG.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN"
  "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">

<svg width="5cm" height="4cm" viewbox="0,0,500,400">

  <!-- Décrire ici les éléments de l'image -->
  <rect x="0.5cm" y="0.5cm" width="2cm" height="1cm"/>

</svg>
```

La première ligne (<?xml version="1.0" standalone="no"?>) permet de dire que le document est un document XML. La balise suivante va charger les spécifications (DTD) du format SVG afin de savoir interpréter correctement les mots clefs qui vont être utilisés dans la suite. L'image commence ensuite : elle est décrite par les balises qui se trouverons entre <svg ...> et </svg>. On a inséré ici un rectangle.

Les paramètres important de la balise <svg ...> sont les suivants :

xmlns[:*préfixe*] = "resource-name"	Attribut XML standard pour l'identification d'un espace de nommage. Se reporter à la recommandation « Les espaces de nommage dans XML ».
version = "1.1" ou "1.2"	Indique la version du langage SVG à laquelle ce fragment de document se conforme. Pour SVG 1.0, la valeur de cet attribut est fixée à "1.0". Cet attribut sera requis pour les fragments de document correspondants aux versions ultérieures de la spécification.
width = "<longueur>"	Pour les éléments 'svg' les plus externes, la largeur intrinsèque du fragment de document SVG. Pour les éléments 'svg' incorporés, la largeur de la région rectangulaire dans laquelle

	l'élément 'svg' se place. Si l'attribut n'est pas spécifié, ceci aboutit au même effet que si on avait spécifié une valeur de "100%".
height = "<longueur>"	Pour les éléments 'svg' les plus externes, la hauteur intrinsèque du fragment de document SVG. Pour les éléments 'svg' incorporés, la hauteur de la région rectangulaire dans laquelle l'élément 'svg' se place. Si l'attribut n'est pas spécifié, ceci aboutit au même effet que si on avait spécifié une valeur de "100%".
(Opt) viewbox = "<longueur>, <longueur>, <longueur>, <longueur>"	Permet de spécifier la zone visible par défaut.

On note au passage comment introduire des commentaires dans le code. L'utilisation des balises (<descr>...</decr>) permet également d'ajouter des descriptions (non visible sur l'image) des objects construits.

1.2 Visualisation d'une image dans un navigateur

Index

Un fichier image peut être lu tel quel par un navigateur disposant d'un plug-in approprié.

Pour insérer une image dans un document HTML, il faut utilisé une balise HTML <EMBED>:

```
<EMBED WIDTH="400" HEIGHT="267" SRC="mon_image.svg"
PLUGINSPAGE="http://www.adobe.com/svg/viewer/install/">
```

Le code précédent permet au navigateur de pouvoir charger le plug-in nécessaire.

2 Formes de base

Index

Dans cette section, on présente les formes de bases définies par le DTD du SVG qui permettent de construire un dessin vectoriel.

2.1 Rectangles

Index

Utilisation des balises rectangle

- pas de balise fermante
- transformable
- Exemple de balise

```
<rect x="1" y="1" width="1198" height="398"  fill="none" stroke="blue" stroke-width="2"/>
```

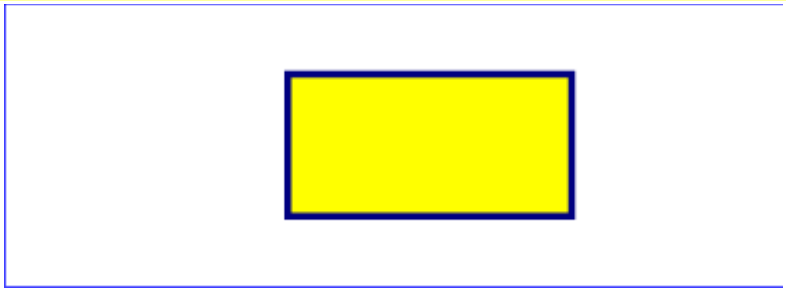
Les paramètres important de la balise <rect ...> sont les suivants :

x = "<position>"	Les attributs x et y définissent la position

y = "<position>"	du coin supérieur gauche du rectangle
width = "<longueur>" height = "<longueur>"	Les attributs high et width
stroke = "<color>" stroke-width = "<longueur>" stroke-opacity, stroke-dasharray, ...	L'attributs relatifs à 'stroke' font référence à la bordure de la forme. L'attribut stroke simple définie la couleur de la bordure. stroke-width définit la largeur de la bordure.
rx="<longueur>" ry="<longueur>"	Lorsqu'on veut définir un rectangle arrondi aux coins, rx et ry définissent les rayons sur x et y de cet arrondi.
fill="<color>"	L'attribut fill détermine la couleur de remplissage du rectangle. On peut utiliser les couleurs prédéfinies ou la commande rgb(<uint8>,<uint8>,<uint8>).

Exemple :

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN"
  "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg width="12cm" height="4cm" viewBox="0 0 1200 400"
     xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
  <desc>Exemple rect01 - rectangle avec des coins droits</desc>

  <!-- Montre le contour du canevas en utilisant l'élément 'rect' -->
  <rect x="1" y="1" width="1198" height="398"
        fill="none" stroke="blue" stroke-width="2"/>
  <rect x="400" y="100" width="400" height="200"
        fill="yellow" stroke="navy" stroke-width="10"  />
</svg>
```



2.2 Cercles et ellipses

Index

Utilisation des balises circle/ellipse :

- pas de balise fermante
- transformable
- Exemple de balise

```
<circle cx="600" cy="200" r="100"  fill="red" stroke="blue" stroke-width="10"  />
<ellipse cx="600" cy="200" rx="250" ry="100" fill="none" stroke="blue" stroke-width="20"  />
```

Les paramètres de la balise <circle ...> sont les suivants :

cx = "<position>" cy = "<position>"	Les attributs cx et cy définissent la position du centre du cercle.
r = "<longueur>"	Rayon du cercle.
stroke = "<color>" stroke-width = "<longueur>" stroke-opacity, stroke-dasharray, ...	L'attributs relatifs à 'stroke' font référence à la bordure de la forme. L'attribut stroke simple définie la couleur de la bordure. stroke-width définit la largeur de la bordure.
fill="<color>"	L'attribut fill détermine la couleur de remplissage du rectangle. On peut utiliser les couleurs prédéfinies ou la commande rgb(<uint8>,<uint8>,<uint8>).

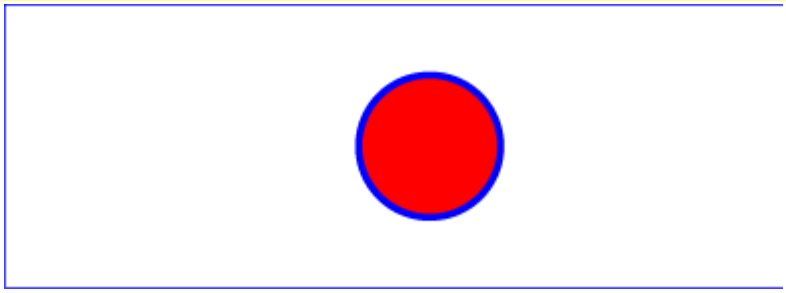
Les paramètres de la balise <ellipse ...> sont les suivants :

cx = "<position>" cy = "<position>"	Les attributs cx et cy définissent la position du centre de l'ellipse
rx = "<longueur>" ry = "<longueur>"	Rayons de l'ellipse suivant les axes x et y.
stroke = "<color>" stroke-width = "<longueur>"	L'attributs relatifs à 'stroke' font référence à la bordure de la forme. L'attribut stroke simple définie la couleur de la bordure. stroke-width définit la largeur de la bordure.

stroke-opacity, stroke-dasharray, ...	
fill="<color>"	L'attribut fill détermine la couleur de remplissage du rectangle. On peut utiliser les couleurs prédéfinies ou la commande rgb(<uint8>,<uint8>,<uint8>).

Exemple :

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN"
  "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg width="12cm" height="4cm" viewBox="0 0 1200 400"
     xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
  <desc>Exemple circle01 - cercle rempli de rouge avec un liseré bleu</desc>

  <!-- Montre le contour du canevas avec un élément 'rect' -->
  <rect x="1" y="1" width="1198" height="398"
        fill="none" stroke="blue" stroke-width="2"/>

  <circle cx="600" cy="200" r="100"
        fill="red" stroke="blue" stroke-width="10"  />
</svg>
```



2.3 Lignes et Lignes multiples

Index

Utilisation des balises line :

- pas de balise fermante
- transformable
- Exemple de balise

```
<line x1="600" y1="200" x2="200" y2="30" stroke-width="5" stroke="blue"/>
```

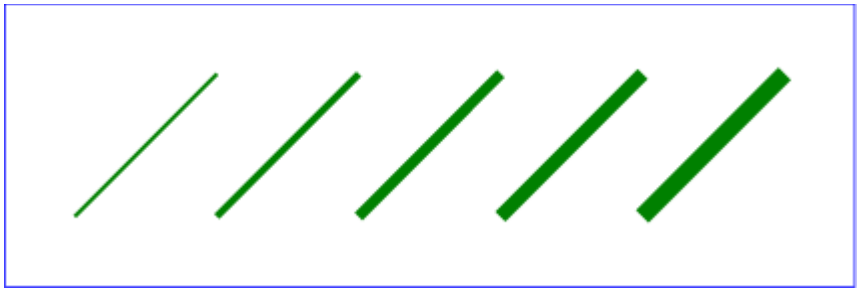
Les paramètres de la balise <line ...> sont les suivants :

x1 = "<position>" y1 = "<position>"	x1, y1 est la position de l'origine de la ligne et x2, y2 la position de fin.
stroke = "<color>"	Ici, l'attributs relatifs à 'stroke' font référence à la ligne (pas de bordure). L'attribut

stroke-width = "<longueur>" stroke-opacity, stroke-dasharray, ...	stroke simple définie la couleur de la ligne. Et stroke-width définit sa largeur.

Exemple :

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN"
"http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg width="12cm" height="4cm" viewBox="0 0 1200 400"
xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
<desc>Exemple line01 - droites exprimées en coordonnées utilisateur</desc>

<!-- Montre le contour du canevas avec un élément 'rect' -->
<rect x="1" y="1" width="1198" height="398" fill="none" stroke="blue" stroke-width="2" />


<line x1="100" y1="300" x2="300" y2="100" stroke-width="5" stroke="green" />
<line x1="300" y1="300" x2="500" y2="100" stroke-width="10" stroke="green" />
<line x1="500" y1="300" x2="700" y2="100" stroke-width="15" stroke="green" />
<line x1="700" y1="300" x2="900" y2="100" stroke-width="20" stroke="green" />
<line x1="900" y1="300" x2="1100" y2="100" stroke-width="25" stroke="green"  />

</svg>
```



Les lignes multiples (polyline) permettent de construire une ligne comportant plusieurs segments sans discontinuités à l'aide d'une commande unique (et non pas une suite de commande de ligne).

Utilisation des balises polyline :

- pas de balise fermante
- transformable

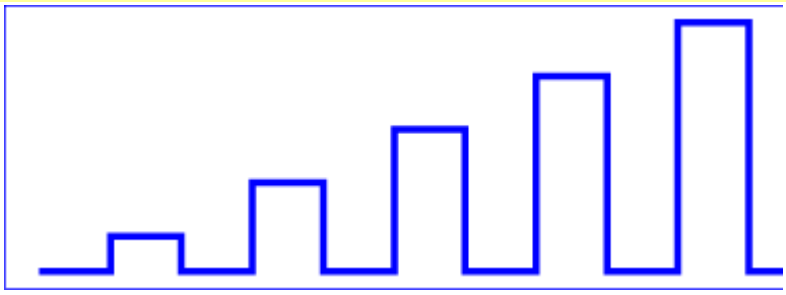
Les paramètres de la balise <polyline ...> sont les suivants :

points = "<position>,<position> <position>,<position> ..."	points représente la suite les points (définies par des valeurs x et y séparée une virgule). Chaque point est séparer par un espace.
stroke = "<color>" stroke-width = "<longueur>" stroke-opacity, stroke-dasharray, ...	Ici, l'attributs relatifs à 'stroke' font référence à la ligne (pas de bordure). L'attribut stroke simple définie la couleur de la ligne. Et stroke-width définit sa largeur.

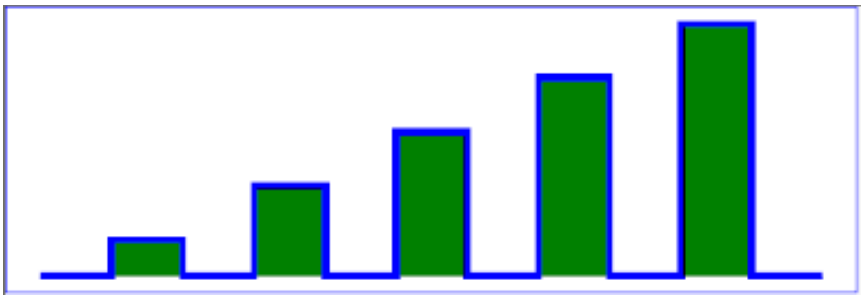
fill = "<color>"	Si fill n'est pas none, une partie 'sous la courbe' sont remplies.

Exemple :

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN"
  "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg width="12cm" height="4cm" viewBox="0 0 1200 400"
     xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
  <desc>Exemple polyline01 - des barres de plus en plus grandes</desc>

  <!-- Montre le contour du canevas avec un élément 'rect' -->
  <rect x="1" y="1" width="1198" height="398"
        fill="none" stroke="blue" stroke-width="2" />

  <polyline fill="none" stroke="blue" stroke-width="10"
            points="50,375
                    150,375 150,325 250,325 250,375
                    350,375 350,250 450,250 450,375
                    550,375 550,175 650,175 650,375
                    750,375 750,100 850,100 850,375
                    950,375 950,25 1050,25 1050,375
                    1150,375" />
</svg>
```



Même image avec fill="green"



La commande <polygone ...> reprend le même principe et les mêmes attributs. Il ferme automatiquement à la fin de la liste de point la courbe pour obtenir un polygone fermé. Ces objets ne dispose pas de commande de position. On verra qu'avec les transformations on pourra les positionner.

2.4 Textes

Index

Utilisation des balises text :

- balises ouvrante/fermante
- transformable
- Exemple de balise

```
<text x="250" y="150" font-family="Verdana" font-size="55" fill="blue" >
Mettre le texte ICI
```

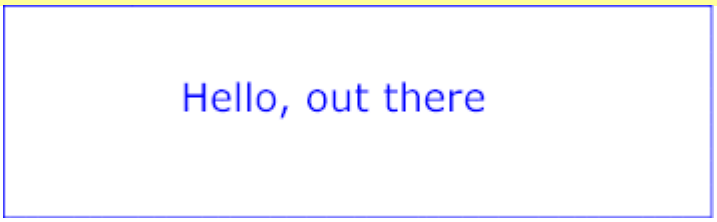
```
</text>
```

Les paramètres de la balise <text ...> sont les suivants :

x = "<position>" y = "<position>"	x, y est la position du texte dans l'image.
font-family = "<type de font>" font-size = "<nombre>" font-weight = "normal | bold | bolder | lighter ", font-style="normal | italic | oblique | inherit"	Le choix de la police est fait par un ensemble d'attributs : font-family pour la typographie, font-size détermine la taille de la police. font-weight permet de spécifier l'épaisseur du trait et font-style l'italique.
fill = "<color>"	Donne la couleur du texte.

Exemple :

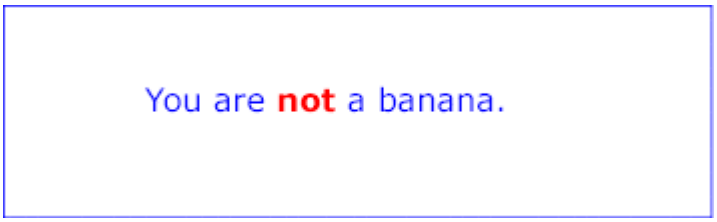
```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="10cm" height="3cm" viewBox="0 0 1000 300" xmlns="http://www.w3.org/2000/svg"
version="1.1">
<desc>Exemple texte01 - 'Hello, out there' en bleu</desc>
<text x="250" y="150" font-family="Verdana" font-size="55" fill="blue" >
Hello, out there
</text>

<rect x="1" y="1" width="998" height="298" fill="none" stroke="blue" stroke-width="2" />
</svg>
```



Pour modifier au milieu d'un texte les caractéristiques d'une partie de celui-ci (Mettre en gras un mot ou changer une couleur), la commande <tspan> peut être utilisé à l'intérieur de la description du texte.

Exemple:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="10cm" height="3cm" viewBox="0 0 1000 300" xmlns="http://www.w3.org/2000/svg"
version="1.1">
<desc>Exemple tspan01 - utilisation des tspan</desc>

<text x="200" y="150" fill="blue" font-family="Verdana" font-size="45">
You are
<tspan font-weight="bold" fill="red" >not</tspan>
a banana.
</text>

<rect x="1" y="1" width="998" height="298" fill="none" stroke="blue" stroke-width="2" />
</svg>
```



2.5 Tracés

Index

Utilisation des balises path :

- pas de balise fermante
- transformable
- Exemple de balise

```
<path d="M 100 100 L 300 100 L 200 300 z" fill="red" stroke="blue" stroke-width="3" />
```

La commande path est la commande la plus générale pour la construction de formes. Toutes les formes précédentes sont en fait définies à partir de cette commande. Elle permet de définir une courbe, une suite de courbes, fermée ou non par la description d'un chemin.

Les paramètres de la balise <path ...> sont les suivants :

d	Description du chemin.
stroke = "<color>" stroke-width = "<longueur>" stroke-opacity, stroke-dasharray, ...	Ici, l'attributs relatifs à 'stroke' font référence à la ligne (pas de bordure). L'attribut stroke simple définie la couleur de la ligne. Et stroke-width définit sa largeur.
pathLength = "<nombre>"	Défini la longueur totale à tracer effectivement. Cet attribut sert principalement pour les animations.
fill = "<color>"	Donne la couleur de remplissage 'sous le tracé' ou à l'intérieur si la forme est fermée.

Description d'un chemin

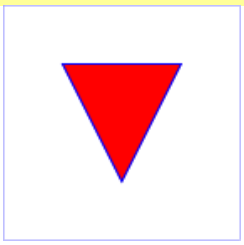
La description de chemin fait appel à la notion de point courant.

- **M**, MoveTo : La commande MoveTo définie un nouveau point courant (départ de courbe). 2 paramètres de position absolue.
- **L** ou **l**, LineTo : déplace le point courant en traçant une ligne droite. L utilise un point absolue et l utilise une position du nouveau point en coordonnées relative.
- **H** ,**h** : déplacement horizontal. 1 paramètre uniquement qui définie la nouvelle abscisse en absolue (H) ou relatif (h).
- **V** ,**v** : déplacement vertical. 1 paramètre uniquement qui définie la nouvelle ordonnée en absolue (V) ou relatif (v).
- **Z**, Close Path : commande de fermeture de la courbe (Optionnel)

D'autres commandes (**C, S, Q, A,** ...) permettent de définir de courbes pour faire des arrondis.

Exemple :

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN" "http://www.w3.org/TR/2001/REC-SVG-
20010904/DTD/svg10.dtd">
<svg width="4cm" height="4cm" viewBox="0 0 400 400" xmlns="http://www.w3.org/2000/svg">
<title>Exemple triangle01- exemple simple d'un 'tracé'</title>
<desc>Un tracé dessinant un triangle</desc>
<rect x="1" y="1" width="398" height="398"fill="none" stroke="blue" />
<path d="M 100 100 L 300 100 L 200 300 z" fill="red" stroke="blue" stroke-width="3" />
</svg>
```

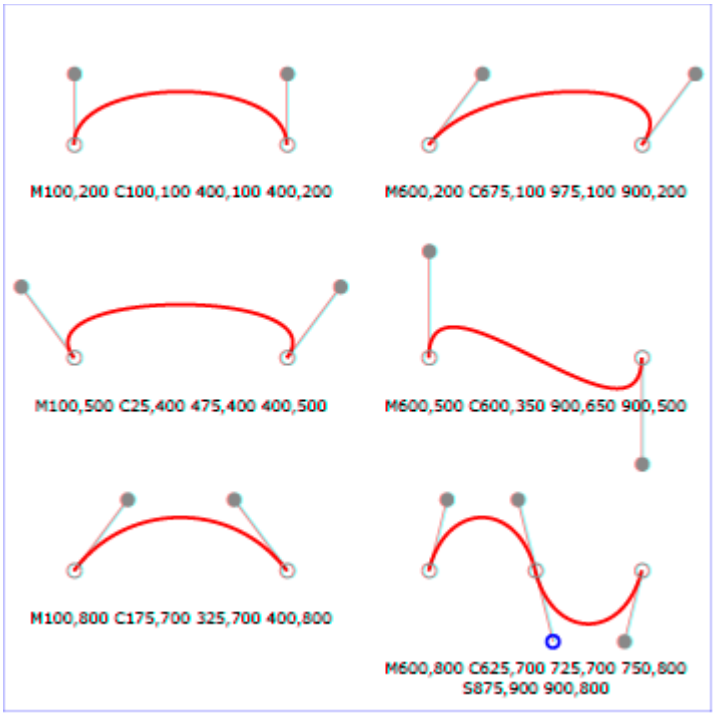


2.6 Courbes

Index

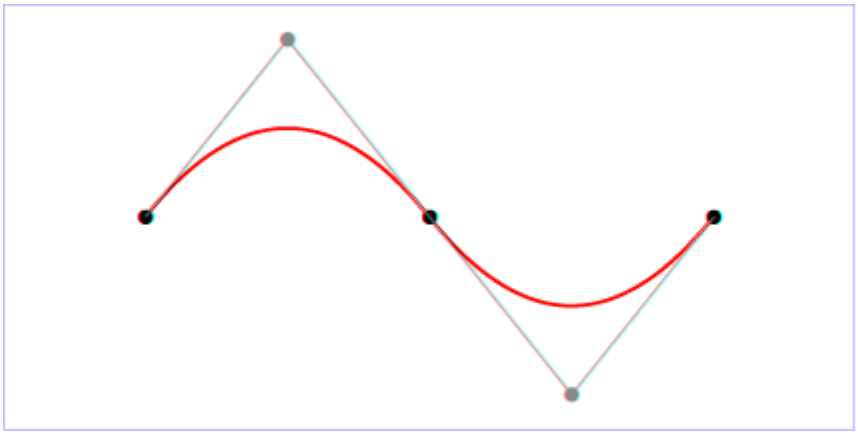
Exemple de chemin pour le path:

Courbes de Béziers :



Courbes Quadratic :

```
<path d="M200,300 Q400,50 600,300 T1000,300" fill="none" stroke="red" stroke-width="5"/>
```



Pour plus de détail, voir les références du W3C.

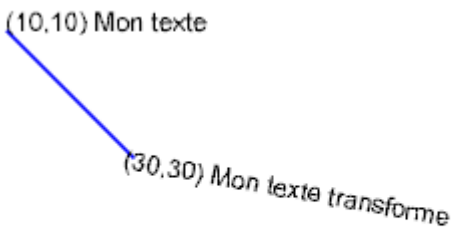
3 Les transformations

Index

Les transformations peuvent être appliqué à la plupart des objets (y compris les regroupements) à l'aide de l'attribut `transform=""` . On décrit dans un premier temps comment faire les opérations élémentaires de transformations géométriques. Dans la dernière sous section, on utilise des matrices de transformation pour combiner celles-ci.

Exemple :

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg xml:space="preserve" width="10cm" height="10cm" viewBox="0 0 100 100" version="1.1">
<desc>Transformations - T</desc>

<line x1="10" y1="10" x2="30" y2="30" stroke="blue" stroke-width="0.5" />
<text x="10" y="10" fill="black" font-size="4">(10,10) Mon texte</text>
<text transform="translate(20,20) rotate(10)" x="10" y="10" fill="black" font-size="4">(30,30) Mon
texte transforme</text>

</svg>
```



Toutes les transformations peuvent exprimer par la combinaison des transformations élémentaires du plan : rotation, translation, échelle ou par une matrice de transformation (qui effectue d'un coup plusieurs opérations élémentaires). Comme dans l'exemple ci-dessus, il peut y avoir plusieurs transformations du même objet. Les opérations sont effectuées les unes après les autres dans l'ordre de lecture de l'attribut de transformation.

L'attribut transform="" est disponible pour toutes les formes de base.

3.1 Translations

Index

Les translations d'un objet sont définis dans l'attribut transform par la commande

```
translate (x,y)
```

Les paramètres sont:

x = "<longueur>" y = "<longueur>"	x et y définissent la translation de l'objet à faire.

cf. Exemple précédent

3.2 Rotation

Index

Les rotations d'un objet sont définis dans l'attribut transform par la commande

```
rotate (a)
```

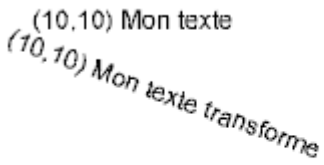
Les paramètres sont:

a = "<angle>"	a défini l'angle de rotation à effectuer

La rotation est effectué autour du coin supérieur gauche de l'objet. ATTENTION : L'utilisation d'une translation est donc différente du positionnement par les attributs x et y d'un élément de l'objet.

Exemple 1: les attributs x et y sont non nul : la position du coin supérieur gauche change!

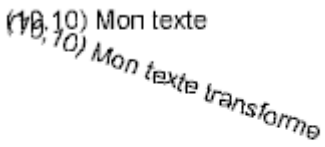
```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg xml:space="preserve" width="10cm" height="10cm" viewBox="0 0 100 100" version="1.1">
<desc>Transformations - T</desc>

<text x="10" y="10" fill="black" font-size="4">(10,10) Mon texte</text>
<text transform="rotate(20)" x="10" y="10" fill="black" font-size="4">(10,10) Mon texte
transforme</text>
</svg>
```



Exemple 1: les attributs x et y sont nul : la position du coin supérieur gauche ne change plus! En utilisant une translation, on a replacer l'objet en (10,10).

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg xml:space="preserve" width="10cm" height="10cm" viewBox="0 0 100 100" version="1.1">
```

```
<desc>Transformations - T</desc>

<text x="10" y="10" fill="black" font-size="4">(10,10) Mon texte</text>
<text transform="translate(10,10) rotate(20)" x="0" y="0" fill="black" font-size="4">(10,10) Mon
texte transforme</text>

</svg>
```



3.3 Echelle

Index

Les changements de taille d'un objet sont définis dans l'attribut transform par la commande

```
scale (x[,y])
```

Les paramètres sont:

x = "<longueur>" y = "<longueur>"	x et y définissent les coefficients d'échelle suivant x et y. Si il n'y a qu'un paramètre, le coefficient est attribué pour les deux axes.

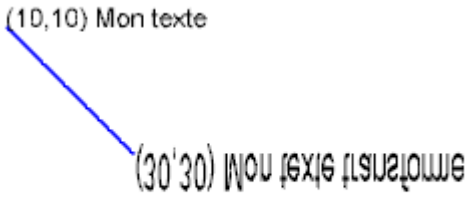
Lorsque les coefficients sont inférieurs à 1 (en valeur absolue), on obtient une réduction. Sinon on a un agrandissement. Les valeurs négatives permettent de faire une symétrie axiale suivant l'axe correspondant. Si les deux coefficients sont négatifs, on obtient une symétrie centrale.

Le changement d'échelle est fait dans le système de coordonnée de l'objet. La conséquence est donc différente si on effectue une translation de l'objet et un changement d'échelle d'un objet changé d'échelle positionné par les attribut x et y. Dans le second cas, il y a modification.

Exemple

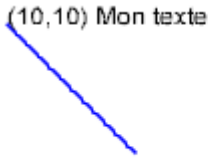
Dans ce premier exemple, on inverse le sens suivant les y en augmentant la taille suivant ce même axe. On note que les attributs x et y sont nuls.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg xml:space="preserve" width="10cm" height="10cm" viewBox="0 0 100 100" version="1.1">
<desc>Transformations - T</desc>

<line x1="10" y1="10" x2="30" y2="30" stroke="blue" stroke-width="0.5" />
<text x="10" y="10" fill="black" font-size="4">(10,10) Mon texte</text>
<text transform="translate(30 30) scale(1 -1)" x="0" y="0" fill="black" font-size="4">(30,30) Mon
texte transforme</text>

</svg>
```



On place maintenant l'objet à l'aide de x et y en (30,30) et on fait le même changement d'échelle : Le texte se trouve alors en (30,-60) on ne le voit plus sur l'image!

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg xml:space="preserve" width="10cm" height="10cm" viewBox="0 0 100 100" version="1.1">
<desc>Transformations - T</desc>

<line x1="10" y1="10" x2="30" y2="30" stroke="blue" stroke-width="0.5" />
<text x="10" y="10" fill="black" font-size="4">(10,10) Mon texte</text>
<text transform="scale(1 -1)" x="30" y="30" fill="black" font-size="4">(30,30) Mon texte
transforme</text>
</svg>
```



3.4 Bending

Index

Le bending/skew permet d'incliner un objet. On peut incliner suivant les x ou les y, il y a donc deux commandes correspondantes :

```
skewX (a)
skewY (a)
```

Les paramètres sont:

a = "<angle>"	a est l'angle d'inclinaison

Les mêmes précautions que pour les autres transformations sont à prendre concernant les différences entre translation et attribut x et y.

3.5 Transformations générales

Index

Mathématiquement, on peut représenter toutes les transformations comme des matrices de transformation 3x3 de la forme suivante (cf. cours de calcul matriciel et transformations géométriques du plan):

$$\begin{bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{bmatrix}$$

Comme on n'utilise que six valeurs dans la matrice 3x3 ci-dessus, une matrice de transformation s'exprime également comme un vecteur : **[a b c d e f]**.

La transformation peut être définie par une matrice en définissant l'attribut transform ainsi :

```
transform="matrix(10 10 20 0 0 10)"
```

Dans l'ordre, les arguments définissent la valeur de a, b, c, d, e et f. Les effets sur l'image sont combiner. Pour définir une matrice, se référer à la décomposition en transformation élémentaire ci-dessous.

Les transformations font correspondre les coordonnées et les longueurs d'un système de coordonnées précédent avec celles d'un nouveau système de coordonnées :

$$\begin{bmatrix} x_{\text{préc.Sys.Coord.}} \\ y_{\text{préc.Sys.Coord.}} \\ 1 \end{bmatrix} = \begin{bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_{\text{nouv.Sys.Coord.}} \\ y_{\text{nouv.Sys.Coord.}} \\ 1 \end{bmatrix}$$

Les transformations simples sont représentées sous une forme matricielle comme suit :

- Une translation équivaut à la matrice

$$\begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix}$$

ou encore **[1 0 0 1 tx ty]**, où; les valeurs *tx* et *ty* sont les distances de déplacement sur *X* et *Y*, respectivement.

- Un changement d'échelle équivaut à la matrice

$$\begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

ou encore **[sx 0 0 sy 0 0]**. Une unité sur les directions *X* et *Y* dans le nouveau système de coordonnées égale *sx* et *sy* unités dans le système de coordonnées précédent, respectivement.

- Une rotation sur l'origine équivaut à la matrice

$$\begin{bmatrix} \cos(a) & -\sin(a) & 0 \\ \sin(a) & \cos(a) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

ou encore **[cos(a) sin(a) -sin(a) cos(a) 0 0]**, dont l'effet est une rotation d'un angle *a* des axes du système de coordonnées.

- Une inclinaison sur l'axe-x équivaut à la matrice

$$\begin{bmatrix} 1 & \tan(a) & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

ou encore **[1 0 tan(a) 1 0 0]**, dont l'effet est d'incliner les coordonnées X d'un angle *a.*

- Une inclinaison sur l'axe-y équivaut à la matrice

$$\begin{bmatrix} 1 & 0 & 0 \\ \tan(a) & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

ou encore **[1 tan(a) 0 1 0 0]**, dont l'effet est d'incliner les coordonnées Y d'un angle *a*.

4 Structuration

Index

La structuration du document permet d'utiliser les concepts objets que permet la formalisme de XML pour simplifier la construction d'image complexes (ici, utilisant de nombreux éléments). Certains 'objets' graphiques sont à décrire à l'aide de plusieurs formes de bases : par exemple une grille constituée de moult lignes.

Le regroupement d'objets va permettre de faire simplement des transformations d'une forme complexes. La rotation d'une grille serait sinon un véritable casse-tête, s'il fallait placer et tourner chaque élément.

Combiner avec des commandes de définition de formes (`<defs>` et `id`) dans notre image, on ne définira qu'une seule fois les objets qui se répètent. On utilisera ensuite en la positionnant à notre guise à l'aide de transformation.

Une structuration de niveau supérieur (dans le sens où elle donne des informations générales sur la façon d'afficher le document XML) peut être utilisé à l'aide de propriétés de styles partagées avec le CSS et le XSL.

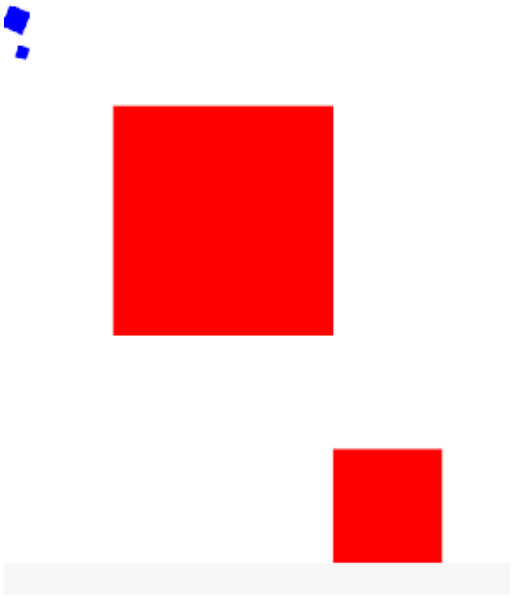
4.1 Regroupement de formes, commande <g>

Index

la balise `<g>` permet de regrouper des formes élémentaires afin de 1) construire des objets plus complexes qui pourront être facilement utilisés (déplacés, transformés) et réutilisé. Et 2), les

balises <g> permettent de regrouper des attributs à valeurs identiques pour les objets contenus.

```
<g style="fill:red">
<rect x="100" y="100" width="200" height="200" />
<rect x="300" y="400" width="100" height="100" />
</g>

<g style="fill:blue" transform="rotate(20)">
<rect x="10" y="10" width="20" height="20" />
<rect x="30" y="40" width="10" height="10" />
</g>
```



La commande <g> peut être utilisée de façon à imbriquer des objets :

```
<g>
        <g>
                        <g>
                                ...
                        </g>
        </g>
        <g>
                        ...
        </g>
</g>
```

Les attributs x et y qui spécifient les positions des objets (<rect x="???" y="???" ...>) définissent en fait la position d'un objet dans le conteneur juste au dessus.

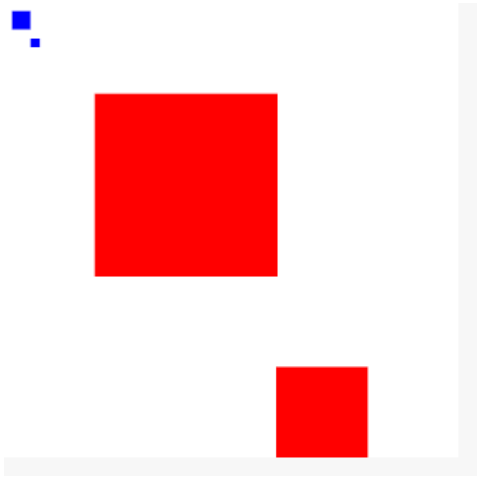
La commande g dispose des attributs génériques des autres objets :

x = "<position>" y = "<position>"	x, y est la position de l'image dans le conteneur supérieur.
fill="<color>"	Couleur de remplissage des objets contenus
strokes="<color>"	Tout ce qui concernent les bordures des objets

...	
...	
transform="..."	Transformation de l'ensemble des objets contenus (cf. ci dessous)

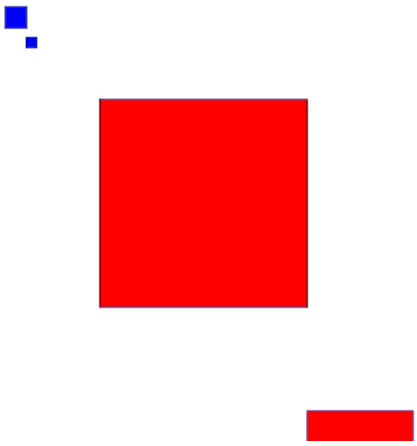
Pour les transformations, comme précédemment, elles se font en prennant comme point de référence le coin supérieur gauche de l'objet : c'est toute la zone qui est transformée. Si des sous-objets sont placés à l'aide des attributs x et y, leur position changera, mais l'allure générale de la forme est conservé.

Dans l'exemple ci-dessus, les carrés bleus ont été tournés par rapport à l'image ci dessous. On voit clairement que la position du plus petit carré à subit également la transformation (rotation).



Exemple: On a imbriqué ici les objets avec un niveau supérieur pour les translater et ajouter des bordures.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg xml:space="preserve" width="10cm" height="10cm" viewBox="0 0 500 500" version="1.1">
<desc>utilisation de g</desc>

<g transform="translate(70,70)" stroke="navy" stroke-width="1">
   <g style="fill:red">
   <rect x="100" y="100" width="200" height="200" />
   <rect x="300" y="400" width="100" height="100" />
   </g>

   <g style="fill:blue">
   <rect x="10" y="10" width="20" height="20" />
   <rect x="30" y="40" width="10" height="10" />
   </g>
</g>
</svg>
```



4.2 Référencement de formes, commandes de <defs> Index

Les styles peuvent être définis entre les balises <defs> et </defs> . On ajout dans les attributs des objets un attribut `id` (identifiant) que permettra d'y faire référence (ie les utiliser) ensuite. La balise <defs> doit être unique dans le svg, mais elle peut contenir autant d'objets référencé que souhaité. Elle se place au début du svg.

Pour utiliser un style ou une forme, le référencement est fait à l'aide de la commande

```
url(#id)
```

on remplace *id* par l'identifiant voulu (en laissant #).

L'exemple suivant définit un mode de remplissage (dégradé linéaire du bleu ou majenta)

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN" "http://www.w3.org/TR/2001/REC-SVG-
20010904/DTD/svg10.dtd">
<svg width="8cm" height="3cm"
     xmlns="http://www.w3.org/2000/svg">
  <desc>Des références d'URI locales dans l'ancêtre de l'élément 'defs'.</desc>
  <defs>
    <degradeLineaire id="Degrade01">
      <stop offset="20%" stop-color="#39F" />
      <stop offset="90%" stop-color="#F3F" />
    </degradeLineaire>
  </defs>
  <rect x="1cm" y="1cm" width="6cm" height="1cm" fill="url(#Degrade01)"  />

  <!-- Montre le contour du canevas en utilisant un élément 'rect' -->
  <rect x=".01cm" y=".01cm" width="7.98cm" height="2.98cm"
        fill="none" stroke="blue" stroke-width=".02cm" />

</svg>
```

4.3 Utilisation multiples d'objets

Index

Pour construire des objets qui peuvent être utilisé plusieurs fois dans le svg, il faut utiliser la commande `<symbol id="...">`. De même que pour les styles prédéfinis (cf. section précédente), la déclaration des objets se fait entre les balises `<defs>...</defs>`.

Pour les insérer effectivement dans l'image, la commande `<use ...>` permet de faire référence à un objet prédéfini. Les attributs de l'objet instancié peuvent être modifier dans les attributs de la balise `<use ...>`.

Exemple :

```
<defs>
<symbol id="s1" >
......
</symbol>
</defs>

<g >
<use xlink:href="#s1" />
</g>
```

5 Les animations

Index

On ne s'intéresse ici qu'aux animations SMIL et pas aux autres possibilités qui peuvent être envisager avec l'utilisation de script (JavaScript et ECMAScript). Les animations sont représentées par des balises <animate> et ses dérivées <animateTransform>, <animateColor>, et enfin <animateMotion>. On présente les trois premiers.

5.1 Element <animate>

Index

L'élément 'animate' s'utilise pour animer un seul attribut, ou une seule propriété, au cours du temps.

attributeName = <attributeName>	Spécifie l'attribut dont la valeur est à faire variée (une seule possible)
attributeType = "CSS | XML | auto"	Spécifie si l'attribut est un attribut CSS ou un attribut XML. 'auto' par défaut, et ca marche bien comme ca pour les animations simples !!
from, to, by = "..."	Les attributs from, by et to définissent les limites de l'attribut qui va changer linéairement .
values=" ..."	L'attribut values consiste en une liste de valeurs que prendra l'attribut à faire varier, séparées par des points-virgules.
keySplines = "<list>" KeyTimes = "<list>"	Lorsque ces attributs sont utilisés, l'attribut variant suit les valeurs définies par les points de controles données dans keySplines et en suivant le timing définie dans keyTimes.

`calcMode` = "discrete | linear | paced | spline"	Différents modes de calculs des valeurs entre les points de contrôles des valeurs de l'attribut animé.

Exemple:

Animation simple d'un attribut :

```
<rect width="20px" ...>
<animate attributeName="width" from="0px" to="10px" dur="10s"/>
</rect>
```

Pour donner un effet de fondu en fermeture sur un rectangle, qui se répète toutes les 5 secondes, on peut spécifier

```
<rect>
  <animate attributeType="CSS" attributeName="opacity"
         from="1" to="0" dur="5s" repeatCount="indefinite" />
</rect>
```

Exemples avec Value :

```
<use xlink:href="#cube" x="150" y="150">
       <animate attributeName="y" dur="2s" values="150; 140; 130; 120; 110; 100; 110; 120; 130;
140; 150"/>
</use>
```

Exemple d'animations avec keySpline:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE svg SYSTEM "svg10.dtd" [
    <!ENTITY dur "2s">
    <!ENTITY values "0; -45; 0; 16; 0; -7; 0; 3; 0; -2; 0; 1; 0">
    <!ENTITY keyTimes "0; 0.2564; 0.5128; 0.6154; 0.6923; 0.7436; 0.7949; 0.8462; 0.8974; 0.9231;
0.9487; 0.9744; 1">
    <!ENTITY keySplines "0 .75 .5 1; .5 0 1 .25; 0 .25 .25 1; .5 0 1 .5; 0 0 1 1; 0 0 1 1; 0 0 1 1;
0 0 1 1;  0 0 1 1;  0 0 1 1;  0 0 1 1;  0 0 1 1">
    <!ENTITY calcMode "spline">
    <!ENTITY begin "mouseover">
]>
<svg width="665px" height="250px" viewBox="0 0 665 250" xml:space="preserve">
    <defs>
        <symbol id="cube" stroke="#000000" stroke-width="0.5" stroke-linejoin="bevel">
            <path d="M 0.112,26.271 l 25.032,12.485 V 25.164 L 0.112,12.68 V 26.271 z" fill="#333"/>
            <path d="M 25.144,38.756 l25.033 -12.485 H 50.12 V 12.708 L 25.144,25.164 V 38.756 z"
fill="#666"/>
            <path d="M 50.12,12.708 l 0.057 -0.028 L 25.144,0.224 L 0.112,12.68 l 25.032,12.484 L
50.12,12.708 z" fill="#ccc"/>
        </symbol>
    </defs>
    <rect width="100%" height="100%" fill="#999" stroke="none"/>
    <g id="cubes" transform="translate(300.25,143.45)">
        <use xlink:href="#cube" transform="translate(0,-72)">
            <animate attributeName="y" dur="&dur;" values="&values;" keyTimes="&keyTimes;"
keySplines="&keySplines;" begin="&begin;" restart="whenNotActive" calcMode="&calcMode;"/>
        </use>
        <use xlink:href="#cube" transform="translate(20,-72)">
            <animate attributeName="y" dur="&dur;" values="&values;" keyTimes="&keyTimes;"
keySplines="&keySplines;" begin="&begin;" restart="whenNotActive" calcMode="&calcMode;"/>
        </use>
        <use xlink:href="#cube" transform="translate(40,-72)">
            <animate attributeName="y" dur="&dur;" values="&values;" keyTimes="&keyTimes;"
keySplines="&keySplines;" begin="&begin;" restart="whenNotActive" calcMode="&calcMode;"/>
        </use>
        <use xlink:href="#cube" transform="translate(60,-72)">
            <animate attributeName="y" dur="&dur;" values="&values;" keyTimes="&keyTimes;"
keySplines="&keySplines;" begin="&begin;" restart="whenNotActive" calcMode="&calcMode;"/>
        </use>
    </g>
```

```
</svg>
```

5.2 Element <animateTransform> Index

Pour animer des objets introduit par `<use>` ou bien des groupements entiers `<g>`, les attributs de positionnement ne sont plus disponibles être modifiés avec une animation `<animate>`. La solution est prévue est de pouvoir « animer » la transformation, ou plus précisément faire varier les paramètres de la transformation.

Toutes les transformations (translation, rotation, ...) peuvent être animées, mais également l'animation d'éléments CSS (e.g. ppacité)!

Les attributs sont similaires à l'élément `<animate>`. On peut également faire varier plusieurs paramètres de la transformation en même temps comme le montre le second exemple.

attributeName= 'transform'	Donne l'attribut dont il faut changer la valeur. L'attribut doit prendre comme valeur une couleur, e.g. Les attributs 'fill' et 'strokes'
type = 'translate | scale | rotate | skewX | skewY'	Type donne le type de transformation dont il faut modifier les paramètres.

Exemples :

```
<rect transform="skewX(30)"...>
       <animateTransform attributeName="transform" attributeType="XML"
       type="rotate" from="0" to="90" dur="5s"
       additive="replace" fill="freeze"/>
       <animateTransform attributeName="transform" attributeType="XML"
       type="scale" from="1" to="2" dur="5s"
       additive="replace" fill="freeze"/>
</rect>
```

```
<use ...>
       <animateTransform attributeType="CSS" attributeName="opacity" from="1" to="0" dur="5s"
repeatCount="indefinite" />
</use>
```

```
<g transform="rotate(20 10 10)">
       <path .../>
       <path .../>
       <animateTransform attributeName="transform" type="rotate" dur="2s" from="20 10 10" to="200
100 100"/>
</g>
```

5.3 Element <animateColor> Index

L'élément 'animateColor' spécifie une transformation de couleur au cours du temps.

attributeName= '...'	Donne l'attribut dont il faut changer la valeur. L'attribut doit prendre comme valeur une couleur, e.g. Les attributs 'fill' et 'strokes'

from, to, by = "<color>"	Les attributs from, by et to admettent des valeurs de couleur, chaque couleur pouvant être exprimée par un codage hexadécimal.
values="<color>; <colors>; ..."	L'attribut values consiste en une liste de valeurs de couleur, séparées par des points-virgules.
color-interpolation	La propriété s'applique aux interpolations de couleur issues des animations avec 'animateColor'.

Exemples:

```
<circle ... fill="#5e5d06">
       <animateColor attributeName="fill" dur="2s" from="#5e5d06" to="#fffc00" fill="freeze"/>
</circle>
```

5.4 Contrôles de la succession des animations

Index

Il est souvent pratique de définir une animation comme un décalage, ou un delta, par rapport à la valeur d'un attribut, plutôt que comme des valeurs absolues. Une simple animation « agrandir » peut accroître la largeur d'un objet de 10 pixels :

```
<rect width="20px" ...>
  <animate attributeName="width" from="0px" to="10px" dur="10s"
           additive="sum"/>
</rect>
```

Il est souvent pratique, pour des animations répétées, de se construire sur des résultats précédents, qui s'accumulent avec chaque itération. L'exemple suivant fait que le rectangle continue à grandir au fur et à mesure de la répétition de l'animation :

```
<rect width="20px" ...>
  <animate attributeName="width" from="0px" to="10px" dur="10s"
           additive="sum" accumulate="sum" repeatCount="5"/>
</rect>
```

À la fin de la première répétition, le rectangle a une largeur de 30 pixels, à la fin de la deuxième une largeur de 40 pixels et à la fin de la cinquième une largeur de 70 pixels.

Les attributs suivants sont communs aux éléments 'animate', 'animateMotion', 'animateColor' et 'animateTransform' :

additive = 'replace | sum'	Contrôle si l'animation est additive, ou non sum : Spécifie que l'animation va s'ajouter à la valeur sous-jacente de l'attribut et les autres animations. Replace : Remplace l'animation en cours.(Certaines exceptions)
accumulate = 'none | sum'	Contrôle si l'animation est cumulative, ou non. Sum : Spécifie que chaque itération repeat suivant la première se bâtit sur la précédente.

	None : Spécifie que les itérations repeat ne sont pas cumulative.

5.5 Contrôles de la synchronisation de l'animation

Index

Les attributs suivants, communs à tous les éléments d'animation, contrôlent la synchronisation de l'animation, y compris ce qui entraîne le début ou la fin d'animation, la répétition, ou non, de l'animation et la conservation, ou non, de l'état final de l'animation une fois celle-ci achevée.

begin='Clock-value |...'	Définit quand l'élément devrait commencer (i.e. devenir actif). La valeur de l'attribut est une liste de valeurs, séparées par des points-virgules. En donnant une valeur de temps (en seconde) on peut synchroniser plusieurs transformations entre elles.
end='...'	Définit une valeur de fin pour l'animation, qui peut contraindre la durée active. La valeur de l'attribut est une liste de valeurs, séparées par des points-virgules.
dur = 'Clock-value |indefinite'	Spécifie la durée simple : clock-value. (e.g. en seconde « 1s ») "indefinite":Spécifie la durée simple comme indéfinie.
fill = "freeze" | "remove"	Spécifie comme se termine l'animation. Si on indique freeze, on reste dans l'état de fin, si on spécifie remove, on se replace dans l'état initial.
restart='always|whenNotActive|never'	
repeatCount : numeric value | "indefinite"	Spécifie le nombre d'itérations de la fonction d'animation. Si on donne un nombre, c'est le nombre de répétition, si on indique 'indefinite', la répétition sera infinie.
min / max	Spécifie la valeur minimum ou maximum de la durée active

Pour les évènements déclenchant qui peuvent être utilisés pour les attributs begin et end , il en existe plusieurs sortes :

Exemples d'évènements :

- focusin, focusout, activate : Déclenchement sur l'activation de la forme.

- **id.click, id.mouseup, id.mousedown, id.mouseover, id.mousemove, id.mouseout : Evènements de la souris lorsqu'elle est sur l'objet id. (Avec id qui est l'identifiant donné à une forme quelconque.)**
- SVGError, SVGZoom, SVGScroll : Evènement de gestion du SVG
- id.beginEvent, id.endEvent, id.repeatEvent : Déclenchés quand l'élément d'animation id commence, s'arrete ou se répète. Permet de synchroniser des animations entre elles. (Avec **id** qui est l'identifiant donné à un élément d'animation quelconque.)

Exemple de synhronisation explicite:

```
<... fill="rgb(0,0,255)" x="300" ...>
       <animate attributeName="x" attributeType="XML"
       begin="0s" dur="9s" fill="freeze" from="300" to="0" />
       <animateColor attributeName="fill" attributeType="CSS"
       from="rgb(0,0,255)" to="rgb(128,0,0)"
       begin="3s" dur="6s" fill="freeze" />
</...>
```

Exemple de déclenchement à l'aide d'un bouton:

```
<rect width="80" height="15" stroke="black" fill="#0ee" id="bouton"/>
<text x="10" y="12" stroke="black" stroke-width="1" id="textbouton">Lancer!</text>

<use xlink:href="#cube" x="150" y="150">
       <animate attributeName="y" dur="2s" values="150; 140; 130; 120; 110; 100; 110; 120; 130;
140; 150" begin="bouton.click" fill="freeze"/>
</use>
```