Structure d'un fichier SVG

- **Type MIME : image/svg**
- **Déclaration XML**
  <?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">
- **L'élément <svg>**
  - Un document SVG se compose d'un (ou plusieurs) fragment(s) délimités par la balise **<svg>**
  - Il peut y avoir plusieurs structures <svg> emboîtées dans le même document ou dans des documents composites résultants de plusieurs espaces de noms
  - La balise <svg> redéfinit l'espace utilisateur
  - *Attributs principaux de la balise <svg>*
    - x = "x0" ; position en x du coin supérieur gauche (pour les structures internes)
    - y = "y0" ; position en y du coin supérieur gauche (pour les structures internes)
    - width = "w0" ; largeur en pixels de l'espace (pour les structures externes)
    - height = "h0" ; hauteur en pixels de l'espace (pour les structures externes)
    - viewbox
    - style

Définir un style

- Peut s'appliquer :
  - soit par CSS2 (Cascadind Style Sheet Level 2)
  - soit par XSL (XML Style Language)
  - soit par l'attribut *style* commun à beaucoup de balises SVG
    - Exemple d'emploi avec une balise *text*
      - <text style="font-size: 12pt> Texte en 12 pt</text>
- Il est recommandé de ne pas mélanger les styles CSS et XSL dans le même document SVG

Grouper des éléments

- **La balise <g>**
- Regroupe et nomme des éléments qui partagent des attributs communs : 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>
```

- **On peut imbriquer autant de structures <g> que l'on veut :**
  - **Les objets isolés (ne figurant pas dans une structure <g>) sont considérés comme figurant dans leur propre groupe (leurs attributs ne sont pas propagés)**

Référencement

- On peut utiliser plusieurs types de référencement
- **Référencement relatif**
  - La référence est locale
  - *Exemple*

```
<linearGradient id="myGradient"> .... </linearGradient>
  ...
<rect style="fill:url(#myGradient) />
```

- **Référencement absolu**
  - La référence est une URI générale
  - La référence ne se trouve pas dans le fichier courant

La balise <defs>

- Elle autorise la définition d'objets référencés **plus tard** dans le même fichier

- *Exemple* :

```
<defs>
  <linearGradient id="Gradient01"> .... </linearGradient >
</defs>
...
<rect style="fill:url(#Gradient01) ..../>
```

- Il est requis que toutes les définitions d'objets devant être référencés plus tard soient faites dans la même structure <defs>
- Il n'y a donc **qu'une structure <defs> par fichier SVG**

La balise <symbol>

- Elle permet de définir des objets graphiques réutilisables dans les cas suivants
  - Objet à instancier de multiples fois
  - Objet classique référencé par de nombreux éléments
  - Définition d'un élément de police textuel
  - ...

La balise <use>

- Référencement d'éléments définis dans une structure <defs > (forme d'inclusion)

- *Exemple*

```
<defs>
  <symbol id="s1" >
  ......
  </symbol>
</defs>
<g >
 <use xlink:href="#s1" />
</g>
```

- L'élément use peut référencer :
  - soit un élément du même fichier dont l'ancêtre est un élément <defs>
  - soit un élément d'un autre fichier dont l'ancêtre est un élément <defs>

Image

- **La balise <image>** autorise le référencement d'images entières dans une zone rectangulaire définie dans les coordonnées utilisateur
- Les formats d'images acceptés doivent être :
  - PNG
  - JPEG
  - SVG
- *Exemple :*

```
<image x="200" y="300" width="100px"
height="100px" xlink:href="monimage.png" />
```

Structures des métadonnées

- Ces balises sont importantes car utilisées par les moteurs de recherche

- **La structure <desc>** autorise l'insertion de commentaires non rendus
- **La structure <title>** autorise un titre pouvant être rendu par les viewers, dans le bandeau par exemple
- *Exemple* :

```
<g>
 <title> Mon image </title>
 <desc> Cette image ne contient qu'un rectangle </desc>
 <rect ..... />
</g>
```

Traitement conditionnel

- Il est assuré comme dans SMIL par la balise switch (même principe que dans SMIL)
- Attributs :
  - xml:lang
  - system-required = liste de fonctions
    - Définit les fonctionnalités minimales devant être implémentées par exemple
      - SVGLang : toutes les fonctionnalités de la spécification
        SVGStatic : un sous-ensemble
        SVGDOMStatic : + les interfaces DOM de SVGStatic
        SVGDOMDynamic , ...
  - system-language = liste de langues
    - idem SMIL

Le système de coordonnées

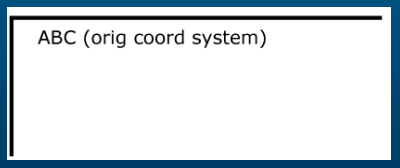
- A l'origine l'élément <svg> le plus externe établit un espace utilisateur et un point de vue confondus comme suit :

```
<?xml version="1.0" standalone="no"?>
      <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG December
      1999//EN"
        "http://www.w3.org/Graphics/SVG/SVG-19991203.dtd">
      <svg width="300px" height="100px">
        <desc>Example InitialCoords - SVG's initial coordinate
      system</desc>    <g style="fill:none; stroke:black; stroke-width:3">
          <line x1="0" y1="1.5" x2="300" y2="1.5" />
          <line x1="1.5" y1="0" x2="1.5" y2="100" />
        </g>
        <g style="fill:red; stroke:none">
          <rect x="0" y="0" width="3" height="3" />
          <rect x="297" y="0" width="3" height="3" />
          <rect x="0" y="97" width="3" height="3" />
        </g>
        <g style="font-size:14 font-family:Verdana">
          <text x="10" y="20">(0,0)</text>
          <text x="240" y="20">(300,0)</text>
          <text x="10" y="90">(0,100)</text>
        </g>
      </svg>
```

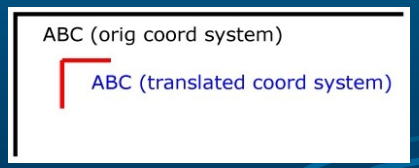
- Chaque élément <svg> interne redéfinit un nouvel espace utilisateur et un nouveau point de vue associé

Transformation et coordonnées

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG December 1999//EN"
 "http://www.w3.org/Graphics/SVG/SVG-19991203.dtd">
<svg width="400px" height="150px">
 <desc>Example xform1Orig - Simple transformations: original picture</desc>
 <g style="fill:none; stroke:black; stroke-width:3">
  <!-- Draw the axes of the original coordinate system -->
  <line x1="0" y1="1.5" x2="400" y2="1.5" />
  <line x1="1.5" y1="0" x2="1.5" y2="150" />
 </g>
 <g>
  <text x="30" y="30" style="font-size:20 font-family:Verdana">
   ABC (orig coord system)
  </text>
 </g>
</svg>
```

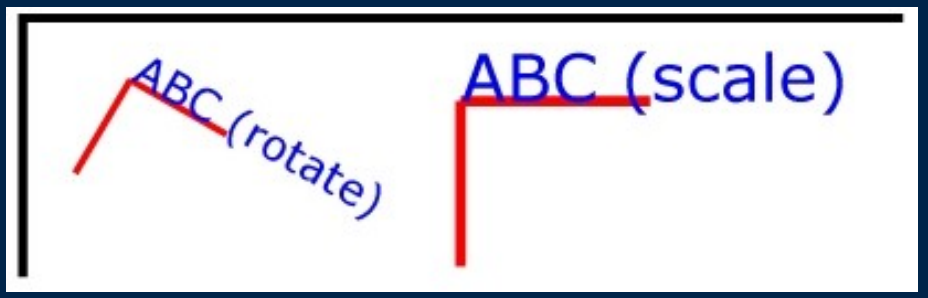



```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG December 1999//EN"
"http://www.w3.org/Graphics/SVG/SVG 19991203.dtd">
<svg width="400px" height="150px">
 <desc>Example xform1NewCoordSys - New user coordinate system</desc>
 <g style="fill:none; stroke:black; stroke-width:3">
  <!-- Draw the axes of the original coordinate system -->
  <line x1="0" y1="1.5" x2="400" y2="1.5" />
  <line x1="1.5" y1="0" x2="1.5" y2="150" />
 </g>
 <g>
  <text x="30" y="30" style="font-size:20 font-family:Verdana">
   ABC (orig coord system)
  </text>
 </g> <!-- Establish a new coordinate system, which is  shifted (i.e., translated) from the initial coordinate system by 50 user units along each axis. -->
 <g transform="translate(50,50)">
  <g style="fill:none; stroke:red; stroke-width:3"> <!-- Draw lines of length 50 user units along  the axes of the new coordinate system -->
   <line x1="0" y1="0" x2="50" y2="0" style="stroke:red"/>
   <line x1="0" y1="0" x2="0" y2="50" />
  </g>
  <text x="30" y="30" style="font-size:20 font-family:Verdana">
   ABC (translated coord system)
  </text>
 </g>
</svg>
```



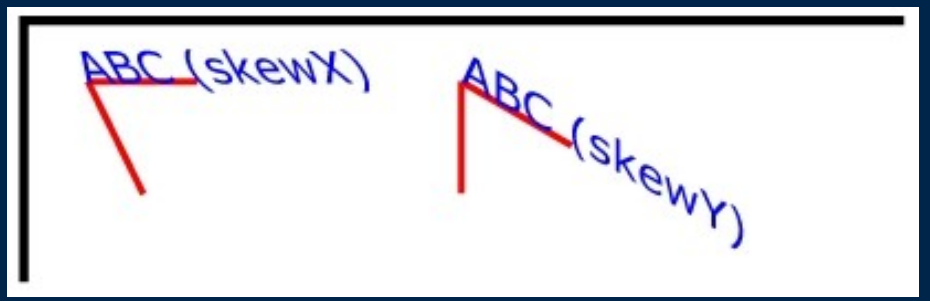
Rotation et mise à l'échelle

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG December 1999//EN"
 "http://www.w3.org/Graphics/SVG/SVG-19991203.dtd">
<svg width="400px" height="120px">
 <desc>Example RotateScale - Rotate and scale transforms</desc>
 <g style="fill:none; stroke:black; stroke-width:3">
  <!-- Draw the axes of the original coordinate system -->
  <line x1="0" y1="1.5" x2="400" y2="1.5" />
  <line x1="1.5" y1="0" x2="1.5" y2="120" />
 </g>
 <!-- Establish a new coordinate system whose origin is at (50,30)
      in the initial coord. system and which is rotated by 30 degrees. -->
 <g transform="translate(50,30)">
  <g transform="rotate(30)">
   <g style="fill:none; stroke:red; stroke-width:3">
    <line x1="0" y1="0" x2="50" y2="0" />
    <line x1="0" y1="0" x2="0" y2="50" />
   </g>
   <text x="0" y="0" style="font-size:20; font-family:Verdana; fill:blue">
    ABC (rotate)
   </text>
  </g>
 </g>
 <!-- Establish a new coordinate system whose origin is at (200,40)
      in the initial coord. system and which is scaled by 1.5. -->
 <g transform="translate(200,40)">
  <g transform="scale(1.5)">
   <g style="fill:none; stroke:red; stroke-width:3">
    <line x1="0" y1="0" x2="50" y2="0" />
    <line x1="0" y1="0" x2="0" y2="50" />
   </g>
   <text x="0" y="0" style="font-size:20; font-family:Verdana; fill:blue">
    ABC (scale)
   </text>
  </g>
 </g>
</svg>
```



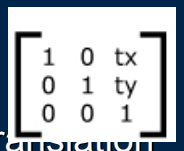
Inclinaison

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG December 1999//EN"
  "http://www.w3.org/Graphics/SVG/SVG-19991203.dtd">
<svg width="400px" height="120px">
  <desc>Example Skew - Show effects of skewX and skewY</desc>
  <g style="fill:none; stroke:black; stroke-width:3">
    <!-- Draw the axes of the original coordinate system -->
    <line x1="0" y1="1.5" x2="400" y2="1.5" />
    <line x1="1.5" y1="0" x2="1.5" y2="120" />
  </g>
  <!-- Establish a new coordinate system whose origin is at (30,30)
       in the initial coord. system and which is skewed in X by 30 degrees. -->
  <g transform="translate(30,30)">
    <g transform="skewX(30)">
      <g style="fill:none; stroke:red; stroke-width:3">
        <line x1="0" y1="0" x2="50" y2="0" />
        <line x1="0" y1="0" x2="0" y2="50" />
      </g>
      <text x="0" y="0" style="font-size:20; font-family:Verdana; fill:blue">
        ABC (skewX)
      </text>
    </g>
  </g>
  <!-- Establish a new coordinate system whose origin is at (200,30)
       in the initial coord. system and which is skewed in Y by 30 degrees. -->
  <g transform="translate(200,30)">
    <g transform="skewY(30)">
      <g style="fill:none; stroke:red; stroke-width:3">
        <line x1="0" y1="0" x2="50" y2="0" />
        <line x1="0" y1="0" x2="0" y2="50" />
      </g>
      <text x="0" y="0" style="font-size:20; font-family:Verdana; fill:blue">
        ABC (skewY)
      </text>
    </g>
  </g>
</svg>
```

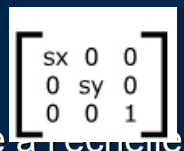


Les attributs de transformations

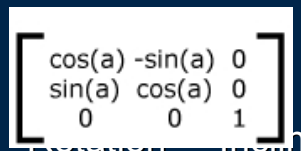
- L'attribut transform a comme valeur possible une liste de transformations séparées par des blancs et/ou une virgule. Ces transformations individuelles sont appliquées l'une après l'autre dans l'ordre où elles apparaissent. Ce peut être :
  - matrix (a,b,c,d,e,f) ; les valeurs e et f peuvent recevoir des unités CSS



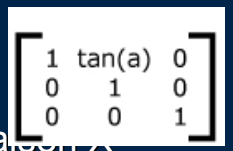
Translation



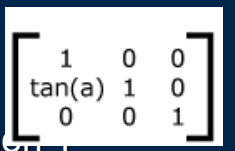
Mise à l'échelle



Rotation



Inclinaison X



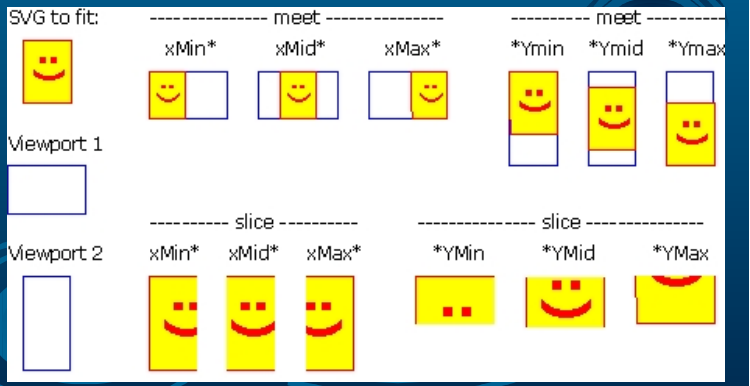
Inclinaison Y

  - translate (tx, [ty]) ; tx et ty peuvent recevoir des unités CSS
  - scale (sx, [sy]) ; si sy n'est pas fourni, on suppose sy = sx
  - rotate (alpha) spécifie une rotation d'angle alpha autour de l'origine de l'espace de coordonnées utilisateur courantes
  - skewX (beta) spécifie une inclinaison d'angle beta le long de l'axe des x
  - skewY (beta) spécifie une inclinaison d'angle beta le long de l'axe des y
- Toutes les valeurs numériques sont réelles ; les valeurs d'angle sont exprimées dans les unités précisées pour les angles

- L'attribut **transform** est appliqué à l'élément AVANT tout calcul de coordonnées ou de longueurs pour cet élément

Les points de vues

- Pour contraindre un ensemble d'objets graphiques à rester confiné dans un espace donné, on peut établir un nouveau point de vue
- Tous les éléments qui peuvent établir un nouveau point de vue ont l'attribut viewBox
  - viewBox (minx , miny, largeur, hauteur)
- On peut contrôler la mise à l'échelle lors du changement de point de vue par l'attribut preserveAspectRatio = *align* [meet ou slice]
    - rien (défaut) ; la mise à l'échelle s'arrange pour que les valeurs extrêmes en x et en y touchent les bords du rectangle de point de vue
    - xMinYmin ; Mise à l'échelle uniforme ; min des x = x min du point de vue min des y = y min du point de vue
    - XMidYMin ; Mise à l'échelle uniforme ; val moyenne des x = x moyen du point de vue min des y = y min du point de vue
  - <svg preserveAspectRatio="xMaxYMax slice"



Formes basiques,path,traitement de l'image,texte,liens

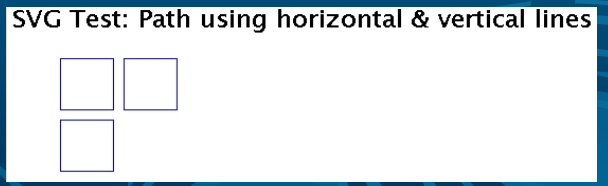
La balise <path>

- L'objet graphique de base est un chemin (path)
  - Contour d'une forme dont on peut spécifier l'épaisseur, la couleur, le remplissage

- **Syntaxe** : <path d="path data" nominalLenght="number" />
  - *path data* est un ensemble de commandes élémentaires (détaillées plus loin)
  - nominalLenght est la longueur totale du chemin en coordonnées utilisateur

- **Commandes de base**
  - M ou m : moveto : x,y démarre un nouveau sous-chemin
  - Z ou z : closepath : ferme un sous-chemin en traçant une ligne droite entre le point courant et le dernier moveto/lineto
  - L ou l : lineto : x , y trace une ligne droite entre le point courant et le point (x,y)
  - H ou h : horizontal lineto : x trace une ligne horizontale entre le point courant et le point (x,y0)
  - V ou v : vertical lineto : y trace une ligne verticale entre le point courant et le point (x0,y)

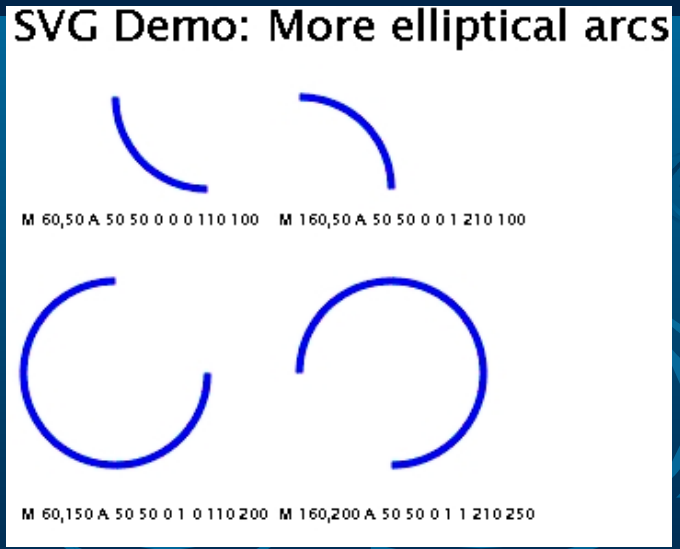
-
```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg SYSTEM "svg-19991203.dtd" > <svg width="600px" height="200px">
  <text x="5" y="20" style="font-size:24">SVG Test: Path using horizontal &amp;amp; vertical lines</text>
  <g style="fill:none; stroke:blue">
    <path d="M 50,50  h 50 v 50 h -50 z"/>
    <path d="M 110,50 h 50 v 50 h -50 v -50"/>
    <path d="m 50,110 h 50 v 50 h -50 z" />
  </g>
</svg>
```



Dessiner des courbes

- Il existe de plus trois groupes de commandes qui dessinent **des courbes** :
  - C ou c, S ou s : Courbes de Bézier cubiques . On spécifie un point de départ, un point d'arrivée et 2 points de contrôle
  - Q ou q, T ou t : Courbes de Bézier quadratiques . On spécifie un point de départ, un point d'arrivée et 1 point de contrôle
  - A ou a : Arc elliptique . On spécifie un morceau d'ellipse par 2 rayons et un sens de parcours

```
<?xml version="1.0" standalone="no" ?>
<!DOCTYPE svg SYSTEM "svg-19991203.dtd" >
<svg width="500" height="700" >
  <g style="text-rendering:optimizeLegibility;shape-rendering:default">
    <text x="5" y="20" style="font-size:24">SVG Demo: More elliptical arcs</text>
    <g style="stroke-width:4 ; fill:none; stroke:blue" >
      <path d="M 60,50   A 50 50 0 0 0 110 100" />
      <path d="M 160,50  A 50 50 0 0 1 210 100" />
      <path d="M 60,150  A 50 50 0 1 0 110 200" />
      <path d="M 160,200 A 50 50 0 1 1 210 250" />
    </g>
    <g style="font-size:8" >
      <text x="10"  y="120">M 60,50  A 50 50 0 0 0 110 100</text>
      <text x="150" y="120">M 160,50 A 50 50 0 0 1 210 100</text>
      <text x="10"  y="280">M 60,150 A 50 50 0 1 0 110 200</text>
      <text x="150" y="280">M 160,200 A 50 50 0 1 1 210 250</text>
    </g>
  </g>
</svg>
```



Formes élémentaires (1)
Rectangles, cercles, ellipses

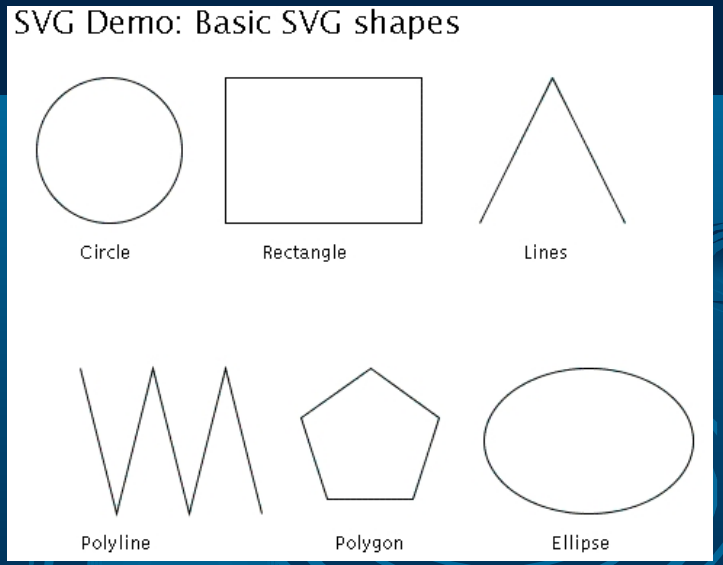
- Il existe un certain nombre de formes prédéfinies (chemins particuliers) qui permettent de s'affranchir de la description complète d'un chemin
- **les rectangles**
  - <rect x="coord" y="coord" width="longueur" height="longueur" rx="longueur" ry="longueur" />
    - x : coordonnée x du coté du rectangle de plus bas x
    - y : coordonnée y du coté du rectangle de plus bas y
    - width : largeur du rectangle
    - height : hauteur du rectangle
    - rx : pour des rectangles à coins arrondis rayon en x de l'ellipse assurant le raccord
    - ry : pour des rectangles à coins arrondis rayon en y de l'ellipse assurant le raccord
- **les cercles**
  - <circle cx="coord" cy="coord" r="longueur" />
    - cx : coordonnée x du centre du cercle, 0 par défaut
    - cy : coordonnée y du centre du cercle, 0 par défaut
    - r : rayon du cercle
- **les ellipses**
  - <ellipse cx="coord" cy="coord" rx="longueur" ry="longueur" />
    - cx : coordonnée x du centre de l'ellipse, 0 par défaut
    - cy : coordonnée y du centre de l'elipse, 0 par défaut
    - rx : rayon de l'ellipse suivant l'axe des x
    - ry : rayon de l'ellipse suivant l'axe des y

Formes élémentaires (2) lignes, polylignes et polygones

- **les lignes**
  - <line x1="coord" x2="coord" y1="coord" y2="coord" />
    - x1 : coordonnée x du point de départ
    - y1 : coordonnée y du point de départ
    - x2 : coordonnée x du point d'arrivée
    - y2 : coordonnée y du point d'arrivée
- **les polylignes**
  - Une polyligne est un ensemble de lignes droites connectées entre elles. Elle définit une forme ouverte.
  - <polyline points="liste de points" />
    - *liste de points* est une liste de coordonnées x , y séparées par des virgules. Ces coordonnées sont exprimées dans l'espace utilisateur
- **Les polygones**
  - Un polygone est un ensemble de lignes droites connectées entre elles définissant une forme fermée
  - <polygon points="liste de points" />
    - *liste de points* est une liste de coordonnées x , y séparées par des virgules. Ces coordonnées sont exprimées dans l'espace utilisateur

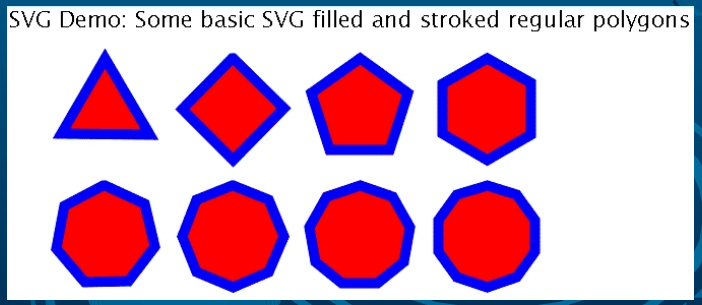
Formes élémentaires
Exemple (1)

```
<?xml version="1.0" standalone="no"?>
      <!DOCTYPE svg SYSTEM "svg-19991203.dtd" >
      <svg width="500" height="500" >
        <g style="text-rendering:optimizeLegibility;shape-rendering:default">
          <text x="5" y="20" style="font-size:22">SVG Demo: Basic SVG shapes</text>
          <g style="stroke:black; fill:none; shape-rendering:default" >
            <circle cx="70"  cy="100" r="50"  />
            <rect x="150" y="50" width="135" height="100" />
            <line x1="325" y1="150" x2="375" y2="50" />
            <line x1="375" y1="50"  x2="425" y2="150" />
            <polyline points="50,250,75,350,100,250,125,350,150,250,175,350" />
            <polygon points="250,250,297,284,279,340,220,340,202,284,250,250" />
            <ellipse cx="400" cy="300" rx="72" ry="50" />
          </g>
          <g style="text-rendering:optimizeSpeed">
            <text x="50"  y="175">Circle</text>
            <text x="175" y="175">Rectangle</text>
            <text x="355" y="175">Lines</text>
            <text x="50"  y="375">Polyline</text>
            <text x="225" y="375">Polygon</text>
            <text x="375" y="375">Ellipse</text>
          </g>
        </g>
      </svg>
```



Formes élémentaires
Exemple (2)

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg SYSTEM "svg-19991203.dtd" >
<svg width="680" height="500" >
        <g style="text-rendering:optimizeLegibility">
          <text x="5" y="20" style="font-size:22">SVG Demo: Some basic SVG filled and stroked regular
            polygons</text>
         <g style="stroke:blue; fill:red; shape-rendering:default; stroke-width:10" >
            <polygon points="99,50,143,125,56,124,99,50" />
            <polygon points="225,50,275,99,225,150,175,100,224,50" />
            <polygon points="350,50,397,84,379,140,320,140,302,84,350,50" />
            <polygon points="475,50,518,74,518,125,475,150,431,124,431,75,475,50" />
            <polygon points="99,175,138,193,148,235,122,269,79,270,51,237,59,195,97,175" />
            <polygon
               points="225,175,260,189,275,225,260,260,225,275,189,260,175,225,189,189,224,175" />
            <polygon
               points="350,175,382,186,399,216,393,250,367,271,332,271,306,250,300,216,317,186,350,
                        175" />
            <polygon
               points="475,175,504,184,522,209,522,240,504,265,475,275,445,265,427,240,427,209,445,
                        184,475,175" />
         </g>
       </g>
</svg>
```



La balise <text>

- Le texte suit les recommandations générales des caractères XML . Il n'effectue ni retour à la ligne ni césure automatique
- La balise <text> est traitée comme un objet graphique
- En tant que telle, elle subit l'influence :
  - des changements de coordonnées
  - du mode de rendu
  - du clipping

<text x="coord" y="coord" />

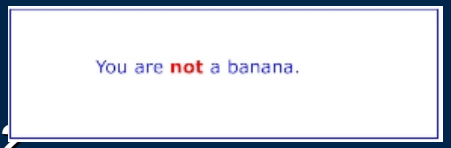
- x représente l'abscisse de départ du texte
  - s'il n'est pas suivi d'unité, la valeur est calculée dans l'espace utilisateur
  - s'il est suivi d'une unité CSS ou de %, la valeur est calculée par rapport au point de vue
- y représente l'ordonnée de départ du texte
  - si elle n'est pas suivie d'unité, la valeur est calculée dans l'espace utilisateur
  - si elle est suivie d'une unité CSS ou de %, la valeur est calculée par rapport au point de vue

- A l'intérieur d'un élément <text>, on peut ajuster la position du texte, la valeur du texte ou la police du texte grâce à l'élément <tspan>
  - <tspan x="coord+" y="coord+" dx="coord+" dy="coord+" rotate="auto|nombre" />

Exemple de textes

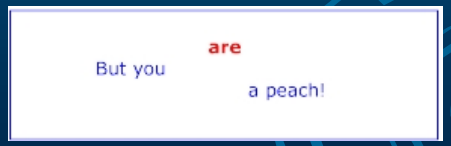
- **Exemple 1**

```
<?xml version="1.0" standalone="no"?>
    <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG December 1999//EN"
      "http://www.w3.org/Graphics/SVG/SVG-19991203.dtd">
    <svg width="10cm" height="3cm">
      <desc>Example tspan01 - using tspan to change visual attributes</desc>
        <g style="font-family:Verdana; font-size:12pt">
        <text x="2cm" y="1.5cm" style="fill:blue">
          You are <tspan style="font-weight:bold; fill:red">not</tspan> a banana.
        </text>
      </g>
    </svg>
```



- **Exemple 2**

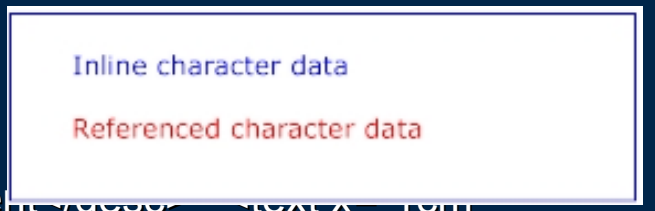
```
<?xml version="1.0" standalone="no"?>
    <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG December 1999//EN"
      "http://www.w3.org/Graphics/SVG/SVG-19991203.dtd">
    <svg width="10cm" height="3cm">
      <desc>Example tspan02 - using tspan's dx and dy attributes
            for incremental positioning adjustments</desc>
       <g style="font-family:Verdana; font-size:12pt">
        <text x="2cm" y="1.5cm" style="fill:blue">
          But you <tspan dx="2em" dy="-.5cm" style="font-weight:bold; fill:red"> are </tspan>
            <tspan dy="1cm"> a peach! </tspan>
        </text>
      </g>
    </svg>
```



Les textes

- A l'intérieur de la balise <text>, on peut :
  - soit spécifier le texte directement
  - soit référencer le texte d'un autre élément par la balise <tref>

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG December 1999//EN"
      "http://www.w3.org/Graphics/SVG/SVG-19991203.dtd">
    <svg width="10cm" height="3cm">
      <defs>
        <text id="ReferencedText">
          Referenced character data
        </text>
      </defs>
      <desc>Example tspan04 - inline vs reference text content</desc>
        y="1cm" style="font-size:12pt; fill:blue">
        Inline character data
      </text>
      <text x="1cm" y="2cm" style="font-size:12pt; fill:red">
        <tref xlink:href="#ReferencedText"/>
      </text>
    </svg>
```



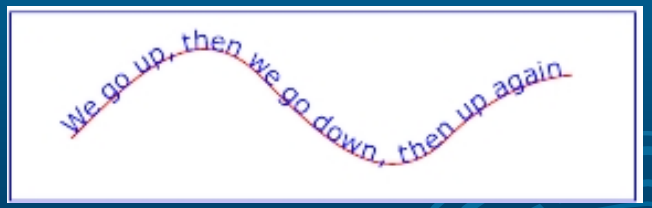
Lier le texte avec un chemin

- On peut imposer au texte de suivre un chemin prédéfini par la balise <path>

<textPath starOffset="longueur|pourcentage" xlink:href="uri" />

- starOffset est le décalage par rapport au début du texte
- une longueur représente une distance le long du chemin mesurée selon la métrique de l'espace utilisateur
- un pourcentage représente un pourcentage par rapport au chemin entier selon la métrique de l'espace utilisateur

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG December 1999//EN"
  "http://www.w3.org/Graphics/SVG/SVG-19991203.dtd">
<svg width="10cm" height="3cm" viewBox="0 0 1000 300">
  <defs>
    <path id="MyPath"
          d="M 100 200
             C 200 100 300   0 400 100
             C 500 200 600 300 700 200
             C 800 100 900 100 900 100" />
  </defs>
  <desc>Example toap01 - simple text on a path</desc>   <use
xlink:href="#MyPath" style="stroke:red" />
  <text style="font-family:Verdana; font-size:42.3333; fill:blue">
    <textPath xlink:href="#MyPath">
      We go up, then we go down, then up again
    </textPath>
  </text>
</svg>
```



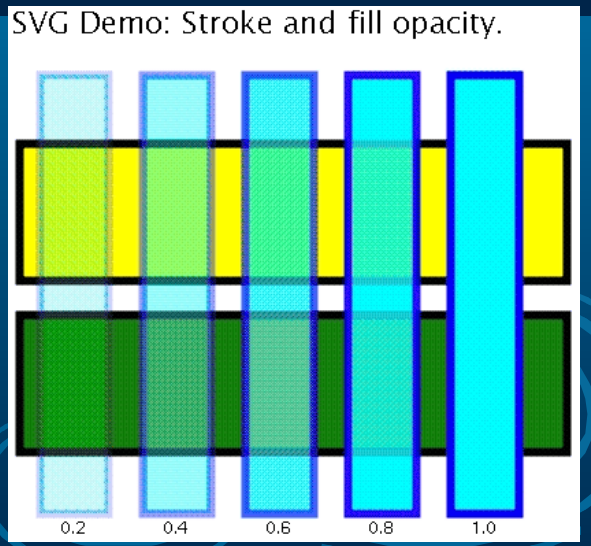
Traitement de l'image : Le rendu

- Les éléments <path>,<text> et les formes de base peuvent être remplis et coloriés (c'est à dire peints sur les bords). On appelera cette opération le rendu
- En SVG, on peut rendre avec :
  - une couleur simple
  - un gradient (linéaire ou radial)
  - un motif (vecteur ou image)
  - des motifs personnalisés disponibles par extension

- Deux propriétés fill et stroke se partagent les attributs suivants :
  - couleur
  - uri de la couleur ou du gradient
- Propriétés de fill (remplissage)
  - opacité
- Propriétés de stroke (dessin)
  - épaisseur
  - jonction de lignes
  - arrondi des angles
  - pointillés
- La propriété de couleur s'applique aux propriétés fill et stroke

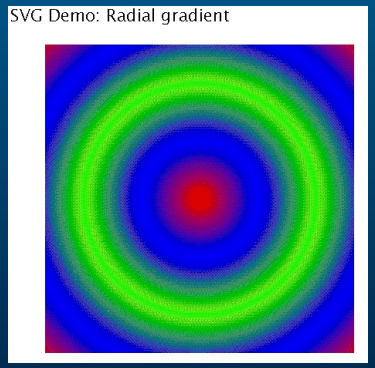
Exemple de remplissage

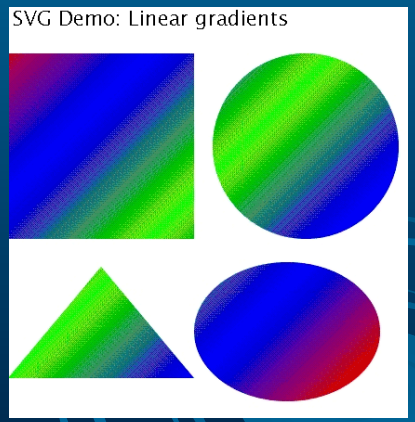
```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg SYSTEM "svg-19991203.dtd" >
<svg width="500" height="500" >
<g style="stroke:black; stroke-width:6; text-rendering:optimizeLegibility; shape-rendering:default" >
   <text x="5" y="20" style="font-size:22">SVG Demo: Stroke and fill opacity.</text>
   <rect x="10"  y="100" width="400" height="100" style="fill:yellow"  />
   <rect x="10"  y="225" width="400" height="100" style="fill:green"  />
   <g style="stroke:blue;fill:cyan">
         <rect x="25"  y="50"  width="50"  height="320" style="stroke-opacity:0.2;fill-opacity:0.2"  />
         <rect x="100" y="50"  width="50"  height="320" style="stroke-opacity:0.4;fill-opacity:0.4"  />
         <rect x="175" y="50"  width="50"  height="320" style="stroke-opacity:0.6;fill-opacity:0.6"  />
         <rect x="250" y="50"  width="50"  height="320" style="stroke-opacity:0.8;fill-opacity:0.8"  />
         <rect x="325" y="50"  width="50"  height="320"   />
    </g>
   <text x="40"  y="385">0.2</text>
   <text x="115" y="385">0.4</text>
   <text x="190" y="385">0.6</text>
   <text x="265" y="385">0.8</text>
   <text x="340" y="385">1.0</text>
 </g>
</svg>
```



Gradients et motifs (ne marche plus)

- En 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
      radiaux






```
<?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>
```

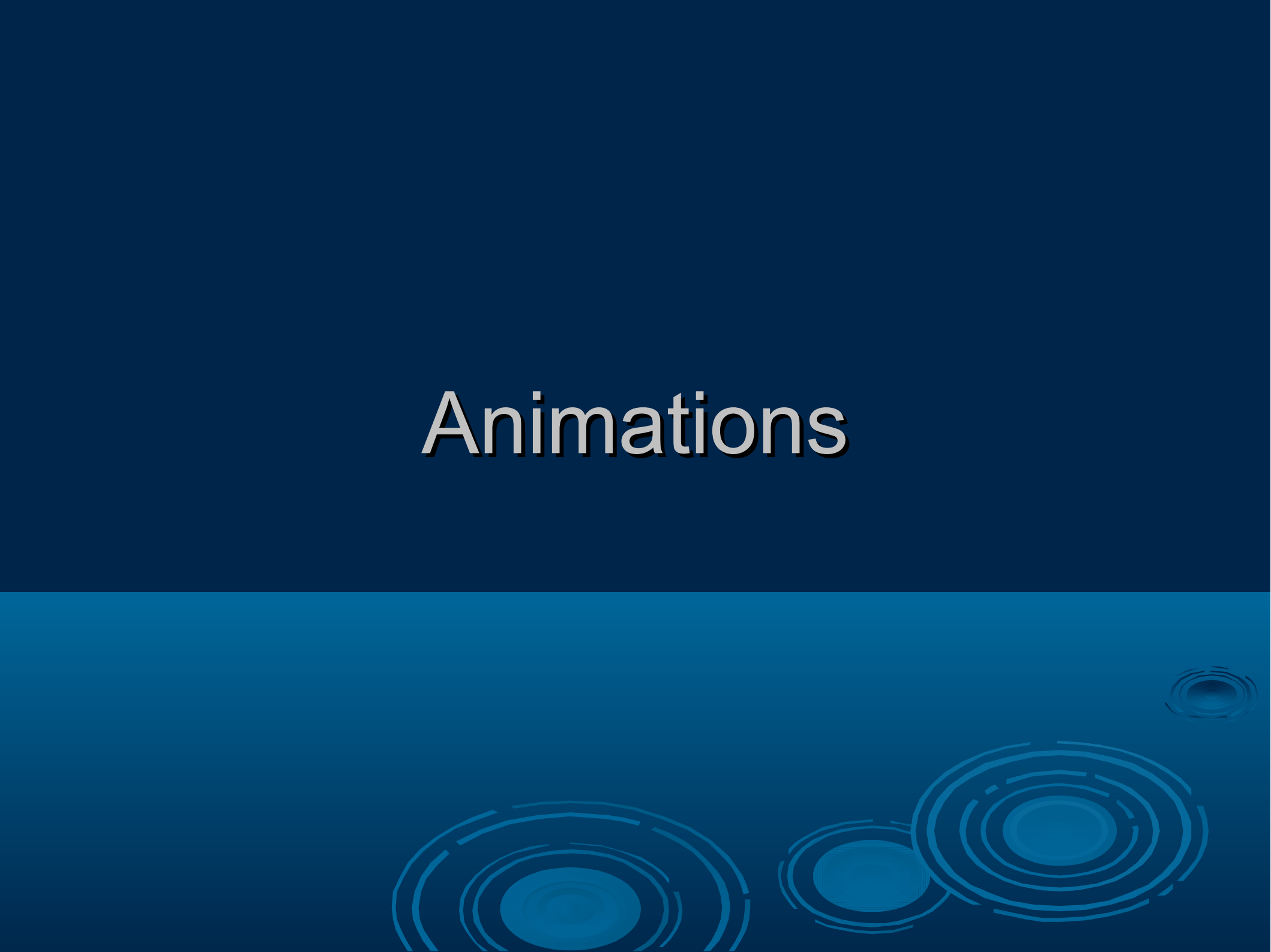
Les Liens

- On distingue les liens extra et intra document SVG
- **Liens extra document :**
  - Les liens en dehors du document courant sont pris en charge par l'élément <a>, analogue à l'élément correspondant de HTML ou SMIL
  - A noter que l'élément <a> utilise la syntaxe de XLink (en cours de spécification)
- *Exemple :*

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG December 1999//EN"
"http://www.w3.org/Graphics/SVG/SVG-19991203.dtd">
<svg width="4in" height="3in">
    <desc>This valid svg document draws a triangle which is a hyperlink
    </desc>
    <a xlink:href="http://www.w3.org">
        <path d="M 0 0 L 200 0 L 100 200 z"/>
    </a>
</svg>
```

- **Les Liens : liens internes**
  - Nécessite de spécifier un fragment SVG
  - Analogue à HTML : MyDrawing.svg#MyView
  - Référence compatible avec XPointer : MyDrawing.svg#xptr(id('MyView'))
  - Spécification d'une vue SVG : MyDrawing.svg#svgView(viewBox(0,200,1000,1000))

Animations



Introduction culture générale animation...

- Les éléments d'animation SVG ont en été développés en collaboration avec le Groupe de Travail Multimédias Synchronisés du W3C (SYMM), les développeurs de la spécification du Langage d'Intégration Multimédia Synchronisé (SMIL) version 1.0 [SMIL1].
  Un fichier SMIL (The Synchronous Multimedia Integration Language) peut contenir du texte, des images fixes et en mouvement, des animations, ainsi que du son. Le code décrit quels éléments multimédia sont présents dans le document, où ils sont situés, à quel moment ils apparaissent et combien de temps ils durent.
  (Les formats de fichiers acceptés dans SMIL sont les suivants : .swf, .rm, .wav, .aif, .mov, .mp3, .gif, .jpg, .rt, .rm, .avi, .mov, .asf, .viv, .mpeg.)
- Le Groupe de Travail SYMM, en collaboration avec le Groupe de Travail SVG, a édité la spécification SMIL Animation [SMILANIM] http://www.w3.org/TR/smil20/animation.html#animationNS-Introduction, qui représente un jeu de fonctions d'animation XML d'usage général.
  SVG incorpore les fonctions d'animation définies dans la spécification SMIL Animation et fournit des développements propres à SVG.

Une petite animation

- Un language d'animation doit permette de spécifier quant commence l'animation d'un élément, quand cette animation s'arrête, si cette animation doit se répéter et combien de fois. Et biensûr, Il doit permettre de décrire ce qui se passe durant l'animation.

```
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
    "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg width="250" height="100" viewBox="0 0 250 100">
<title>Simple Animation</title>
<rect x="10" y="10" width="200" height="20" style="stroke: black; fill: none;">
<animate
attributeName="width"
attributeType="XML"
from="200" to="20"
begin="0s" dur="5s"
repeatCount="indefinite"
fill="freeze" />
</rect>
</svg>
```

..\..\..\..\..\..\..\Bureau\TP SVG\rectangleDiminue.svg

Svg et animations

- 'animate'permet aux attributs et aux propriétés **scalaires** de recevoir différentes valeurs au cours du temps.
- 'set'un raccourci pratique pour l'élément 'animate', utile lors de l'assignation de valeurs d'animation à des attributs et propriétés **non-numériques**, telle que la propriété 'visibility'
- 'animateMotion' déplace un élément le long d'un tracé de mouvement
- 'animateColor'modifie la valeur de couleur des attributs et des propriétés particuliers au cours du temps

- En supplément, SVG comprend les extensions à SMIL Animation compatibles suivantes :
  - 'animateTransform' modifie l'un des attributs de transformation de SVG au cours du temps, tel que l'attribut transform
  - l'attribut path SVG autorise la spécification de toute fonction, issue de la syntaxe des données de tracé de SVG, dans l'attribut path d'un élément 'animateMotion' (SMIL Animation ne permet qu'un sous-ensemble de la syntaxe des données de tracé de SVG dans un attribut path)
  - 'mpath' elementSVG ajoute un attribut keyPoints à l'élément 'animateMotion' pour offrir un contrôle précis des animations d'un tracé de mouvement
  - l'attribut keyPoints SVG ajoute un attribut keyPoints à l'élément 'animateMotion' pour offrir un contrôle précis des animations d'un tracé de mouvement
  - l'attribut rotate SVG ajoute un attribut rotate à l'élément 'animateMotion' pour le contrôle automatique de la rotation d'un objet, de façon à ce que son axe-x se dirige dans la même direction (ou dans la direction opposée) que le vecteur tangent directionnel du tracé de mouvement

Une autre animation

```
<svg width="300" height="250" viewBox="0 0 300 250">
    <title>Multiple Animations on a Single Object</title>
    <rect x="10" y="10" width="20" height="20"
      style="stroke: black; fill: green; style: fill-opacity: 0.25;">
    <animate attributeName="width" attributeType="XML"
      from="20" to="250" begin="0s" dur="8s" fill="freeze"
      repeatCount="indefinite" />
    <animate attributeName="height" attributeType="XML"
      from="20" to="200" begin="0s" dur="8s" fill="freeze"
      repeatCount="indefinite"/>
    <animate attributeName="fill-opacity" attributeType="CSS"
      from="0.25" to="1" begin="0s" dur="3s" fill="freeze"
      repeatCount="indefinite"/>
    <animate attributeName="fill-opacity" attributeType="CSS"
      from="1" to="0.25" begin="3s" dur="3s" fill="freeze"
      repeatCount="indefinite"/>
    </rect>
</svg>
    ..\..\..\..\..\..\Bureau\TP SVG\multipleanim.svg
```

Animation le long d’un chemin

```
<svg width="300" height="200" viewBox="0 0 300 200">
    <title>Animation Along a Complex Path</title>
    <!-- show the path along which the triangle will move -->
    <path d="M50,125 C 100,25 150,225, 200, 125"
          style="fill: none; stroke: blue;"/>
    <!-- Triangle to be moved along the motion path.
          It is defined with an upright orientation with the base of the
    triangle centered horizontally just above the origin. -->
    <path d="M-10,-3 L10,-3 L0,-25z" style="fill: yellow; stroke: red;">
      <animateMotion
          path="M50,125 C 100,25 150,225, 200, 125"
          dur="6s" fill="freeze" repeatCount="indefinite"/>
    </path>
</svg>..\..\..\..\..\..\Bureau\TP SVG\animatemotion.svg
```

Animation le long d'un chemin

```
<svg width="300" height="200" viewBox="0 0 300 200">
   <title>Motion Along a Complex Path Using mpath</title>
   <path id="cubicCurve"
    d="M50,125 C 100,25 150,225, 200, 125"
    style="fill: none; stroke: blue;"/>
   <path d="M-10,-3 L10,-3 L0,-25z" style="fill: yellow;
    stroke: red;" >
    <animateMotion dur="6s" rotate="auto" fill="freeze"
     repeatCount="indefinite">
     <mpath xlink:href="#cubicCurve"/>
    </animateMotion>
   </path>
</svg>..\..\..\..\..\..\..\Bureau\TP SVG\animatemotion2.svg
```

Découpage et masquage

```
<?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>
```

Effets de filtre

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG December
1999//EN"
 "http://www.w3.org/Graphics/SVG/SVG-19991203.dtd">
<svg width="278px" height="118px">
 <defs>
<filter id="Shadow">
<feGaussianBlur in="SourceAlpha"
  stdDeviation="3"
  result="blurredAlpha" />
 <feOffset in="blurredAlpha"
   dx="2" dy="1"
   result="offsetBlurredAlpha" />
 <feDiffuseLighting in="blurredAlpha"
   diffuseConstant=".5"
    surfaceScale="5"
    resultScale="5"
    LightColor="white"
    result="bumpMapDiffuse" >
 <feDistantLight azimuth="135" elevation="60"/>
</feDiffuseLighting>
<feComposite in="bumpMapDiffuse" in2="SourceGraphic"
  operator="arithmetic" k1="1"
  result="litPaint" />
<feSpecularLighting in="blurredAlpha"
        surfaceScale="5"
        specularConstant=".5"
        specularExponent="10"
        lightColor="white"
        result="bumpMapSpecular" >
       <feDistantLight azimuth="135" elevation="60"/>
      </feSpecularLighting>
      <feComposite in="litPaint" in2="bumpMapSpecular"
        operator="arithmetic" k2="1" k3="1"
        result="litPaint" />
       <feComposite in="litPaint"
         in2="SourceAlpha"
         operator="in"
         result="litPaint" />
      <feMerge>
        <feMergeNode in="offsetBlurredAlpha" />
         <feMergeNode in="litPaint" />
       </feMerge>
      </filter>
      </defs>
       <desc>Example filters02 - text with shadowing
     effect</desc>
 <text style="font-size:36px; fill:red; filter:url(#Shadow)"
       x="10px" y="70px">Shadowed Text</text>
     </svg>
```

Insertion de scripts

- Il est possible d'insérer des « javascripts »
- <?xml version="1.0" standalone="no"?>

```
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG December 1999//EN"
      "http://www.w3.org/Graphics/SVG/SVG-19991203.dtd">
    <svg width="4in" height="3in">
      <defs>
        <script><![CDATA[
          /* Beep on mouseclick */
          MouseClickHandler() { beep(); }
         ]]>
        </script>
      </defs>
      <circle onclick="MouseClickHandler()" r="85"/>
    </svg>
```