haute école
neuchâtel berne jura
arc ingénierie
saint-imier le locle delémont

Séance 9

XML et Multimédia
SVG

Prof. Yassin Aziz REKIK
Yassin.rekik@he-arc.ch

Hes·so
Haute Ecole Spécialisée
de Suisse occidentale

Introduction

- XML : modélisation des documents
- XML : Applications
  - **Publication et diffusion de documents**
  - **Format d'échange entre applications**
  - **Format de représentation de données spécifiques**
- Représentation de données multimédia
  - **SVG**
  - **SMIL**

Exemples SVG

- Avant de parler de SVG, voyons ce que ça peut faire !!!
  - **Cartes interactives**
  - **magasins virtuels,**
  - **jeux,**
  - **simulateurs, etc…)**

Définition

- SVG : Scalable Vector Graphics
- Une application XML qui permet de décrire du graphique vectoriel
  - **Incluant texte et images**
  - **Avec des couleurs dégradées, des filtres, une richesse de styles**
  - **Et des animations et des objets d'interaction**

Images Vectorielles vs Bitmaps

- Images Bitmaps :
  - **Principaux formats : GIF, JPEG, BMP**
  - **Représentées sous forme de fichiers binaires**
  - **Sur Internet, la création de l'image s'effectue coté serveur avant que l'image soit transférée au client (cf. librairies génération gif ACME pour Java)**

- Images Vectorielles :
  - **Utilisent des mécanismes de transformation très puissants, un système de coordonnées, des unités de mesures et des formes géométriques basées sur des vecteurs**
  - **Le rendu s'effectue sur le poste client (utilisation des ressources machines de l'internaute)**
  - **Peuvent être représentées sous forme de fichier texte ou binaire**
  - **Peuvent inclure des images Bitmaps**

Images Vectorielles vs Bitmaps

- La représentation d'une image vectorielle coute moins qu'une image bitmap (exemple pour une image de 640x480 : 3000 bytes en bitmap, 20 bytes en vectoriel)
- Pas de perte de qualité lors du redimensionnement d'une image vectorielle
- La description d'une image vectorielle repose sur du texte qui peut être sélectionné ou être soumis à des recherches
- Des liens peuvent être créés sur n'importe quelle partie d'une image vectorielle
- Les images vectorielles sont plus facilement manipulables
- Les animations sont plus simples à effectuer en utilisant des vecteurs

Historique

- Groupe de travail constitué par le W3C en 1998 (avec des représentants de Microsoft, Autodesk, Adobe, IBM, Sun, Netscape, Xerox, Apple, Corel, HP, …)
- Recommandation SVG 1.0 publiée en septembre 2001
- Recommandation SVG 1.1 publiée en janvier 2003
  - **comprend aussi une version SVG Mobile pour téléphones portables et PDAs**
- SVG 1.2 draft

Votre premier SVG

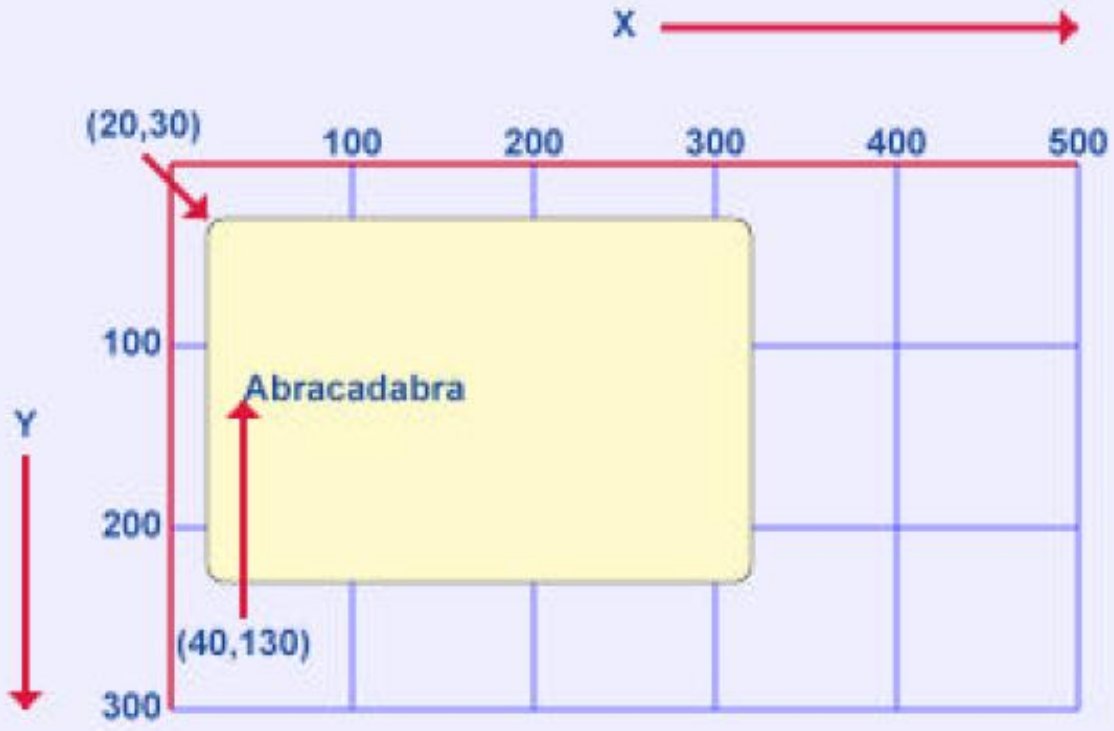
```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC " -//W3C//DTD SVG 1.0//EN"
   "http://www.w3.org/TR/2001/REC-SVG-
20010904/DTD/svg10.dtd">
<svg width="320" height="220">
<rect width="320" height="220" fill="white" stroke="black" />
<g transform="translate(10 10)">
<g stroke="none" fill="lime">
<path d="M 0 112
   L 20 124 L 40 129 L 60 126 L 80 120 L 100 111 L 120 104
   L 140 101 L 164 106 L 170 103 L 173 80 L 178 60 L 185 39
   L 200 30 L 220 30 L 240 40 L 260 61 L 280 69 L 290 68
   L 288 77 L 272 85 L 250 85 L 230 85 L 215 88 L 211 95
   L 215 110 L 228 120 L 241 130 L 251 149 L 252 164 L 242 181
   L 221 189 L 200 191 L 180 193 L 160 192 L 140 190 L 120 190
   L 100 188 L 80 182 L 61 179 L 42 171 L 30 159 L 13 140Z"/>
</g>
</g>
</svg>
```

Les coordonnées

- Voyons un autre exemple simple :

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC " -//W3C//DTD SVG 1.0//EN"
   "http://www.w3.org/TR/2001/REC-SVG-
20010904/DTD/svg10.dtd">
<svg width="320" height="220">
<rect x="20" y="30" width="300" height="200" rx="10" ry="10"
   style="fill:yellow;stroke:black" />
<text x="40" y="130" style="fill:black;stroke:none">Abracadabra</text>
</svg>
```

Les coordonnées




```
<rect x="20" y="30" width="300" height="200" rx="10" ry="10" />
<text x="40" y="130" >Abracadabra</text>
```

Les coordonnées

- Le système de coordonnée utilisateur est fixé par des attributs de l'élément racine <svg …>

- width, height , fixent la dimension de la fenêtre
  - en pixels px (par défaut)
  - en millimètres (mm), centimètres (cm), en pouces (in)
  - en grandeurs typographiques (em, ex, pt, pc)

Les coordonnées

- viewBox = "x y z h" : fixe la portion visible :

```
<svg viewBox="0 0 500 300">
<rect x="20" y="30" width="300" height="200" rx="10" ry="10"
   style="fill:yellow;stroke:black" />
<text x="40" y="130" style="fill:black;stroke:none">Abracadabra</text>
</svg>
```

Le système d'affichage

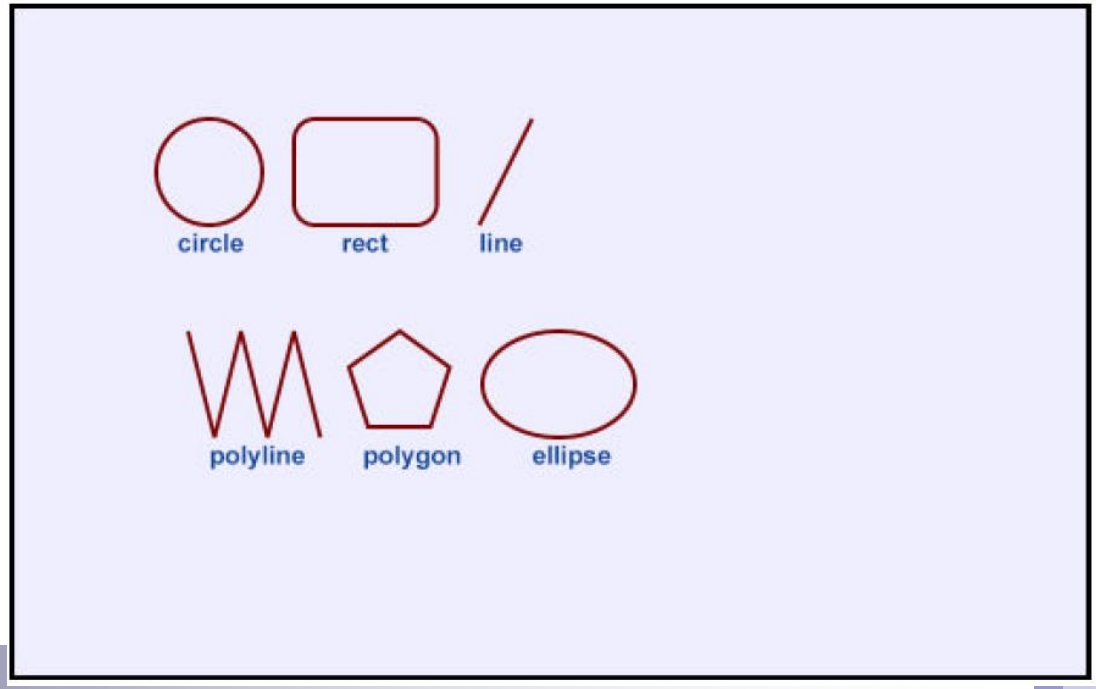
- Rendu graphique
  - **Un document SVG est représenté sur un plan graphique infini en 2D**
  - **Le rendu s'effectue sur une zone finie rectangulaire du plan appelée VIEWPORT**
  - **Ordre de rendu : premier élément rencontré, premier traité (FIFO)**
  - **Eléments supportés : formes géométriques, texte et images**

Exemple pour la suite

```
<svg viewbox= "0 0 512 320" >
<title>Slide Title</title>
<rect x="2" y="2" width="508" height="318"/>
<!-- ********** Coloured Screen Area 512 by 320
   ********** -->
<!-- ********** Examples added here ********** -->
</svg>
```

Formes de base

```
<circle cx="70" cy="100" r="50" />
<rect x="150" y="50" rx="20" ry="20" width="135" height="100" />
<line x1="325" y1="150" x2="375" y2="50" />
<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" />
<ellipse cx="400" cy="300" rx="72" ry="50" />
```

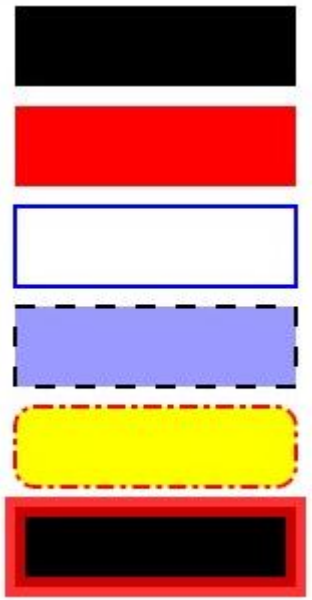


Rectangles

- Un rectangle (parallèle aux axes) est représenté par
  - **<rect x=... y=... width=... height=.../>**
- Sa présentation est caractérisée par les attributs
  - **fill="color" avec une spécification de couleur**
  - **fill-opacity="val" avec "val" entre 0 et 1**
  - **rx="val" et ry="val" pour définir des angles arrondis**
  - **stroke... pour la présentation des contours**

Exemple

```
<svg ...>
  <rect x="10" y="10" width="140" height="40"/>
  <rect x="10" y="60" ... fill="red"/>
  <rect x="10" y="110" ... fill="none"
   stroke="blue" stroke-width="2"/>
  <rect x="10" y="160" ... fill="#9999FF"
   stroke="black" stroke-width="2"
   stroke-dasharray="10 10"/>
  <rect x="10" y="210" ... fill="yellow"
   stroke="red" stroke-width="2"
   stroke-dasharray="9 3 3 3" rx="10" ry="10"/>
  <rect x="10" y="260" ... stroke="red"
   stroke-opacity="0.8" stroke-width="10"/>
</svg>
```



Cercles et ellipses

- Un cercle est représenté par
  - **<circle cx=... cy=... r=.../>**
- Une ellipse est représenté par
  - **<ellipse cx=... cy=... rx=... ry=.../>**
- Leur présentation est régi par les mêmes attributs que les rectangles

Polygones et lignes polygonales

- Un polygone (courbe fermé) est représenté par
  - **<polygon points="x1,y1 x2,y2 ..."/>**
- Une ligne polygonale (ouverte) est représentée par
  - **<polyline points="x1,y1 x2,y2 ..."/>**
  - **normalement, on utilise polyline avec l'attribut fill="none«**
- Les attributs suivants contrôlent la présentation
  - **stroke-linecap avec les valeurs butt, round, square définit la forme des extrémités**
  - **stroke-linejoin avec les valeurs miter, round, bevel définit la forme des angles**

Les formes quelconques

- SVG se base principalement sur deux éléments graphiques :
  - **Les lignes**
  - **Les textes**

- Une forme est un ensemble de segments, ouverts ou fermés.
- Chaque segment est représenté par un point de départ (position courante), un point d'arrivée (à préciser) et un type de déplacement.

- <path d="M 0 0 L 100 100">
  - **M : Move to the point 0,0**
  - **0,0 est la position de départ**
  - **L : Line to 100,100**
  - **100,100 est la position d'arrivée (départ pour la prochaine commande)**

Les formes quelconques

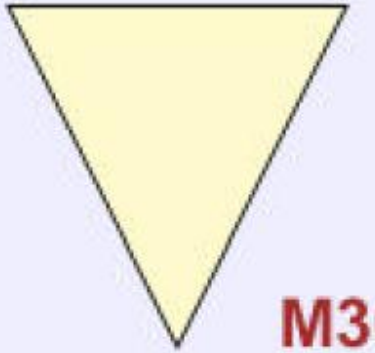
- Représentation d'un triangle
  - **<path d="M 0 0 L 100 0 L50 100 Z">**
  - **M : move**
  - **L : line**
  - **Z : fermer le chemin**
- Exemple de deux chemins
  - **<path d="M 0 0 L 100 0 L50 100 Z M300,300 L400,300 L350,400 Z">**

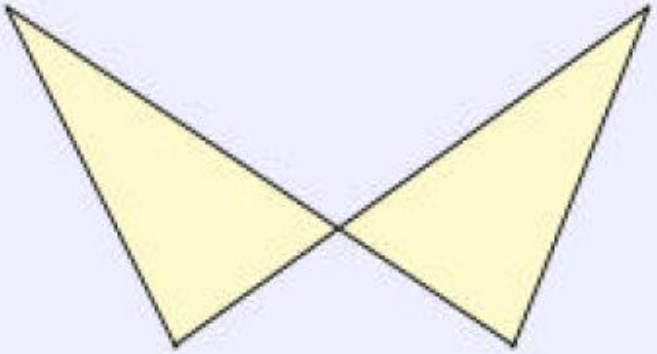
Les formes quelconques

Points and Lines

M150,50 L200,100 H250 V50 h50 v-40 l50,50

Areas





M300,140 L450,240 L490,140 L350,240 Z

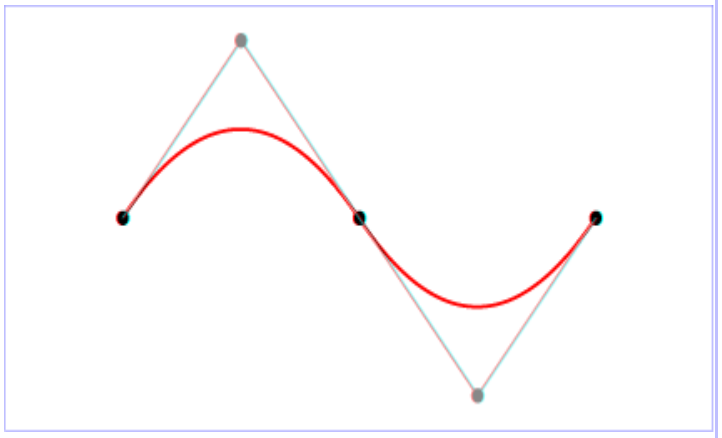
M50,170 L150,170 L100,270 Z

Les formes quelconques

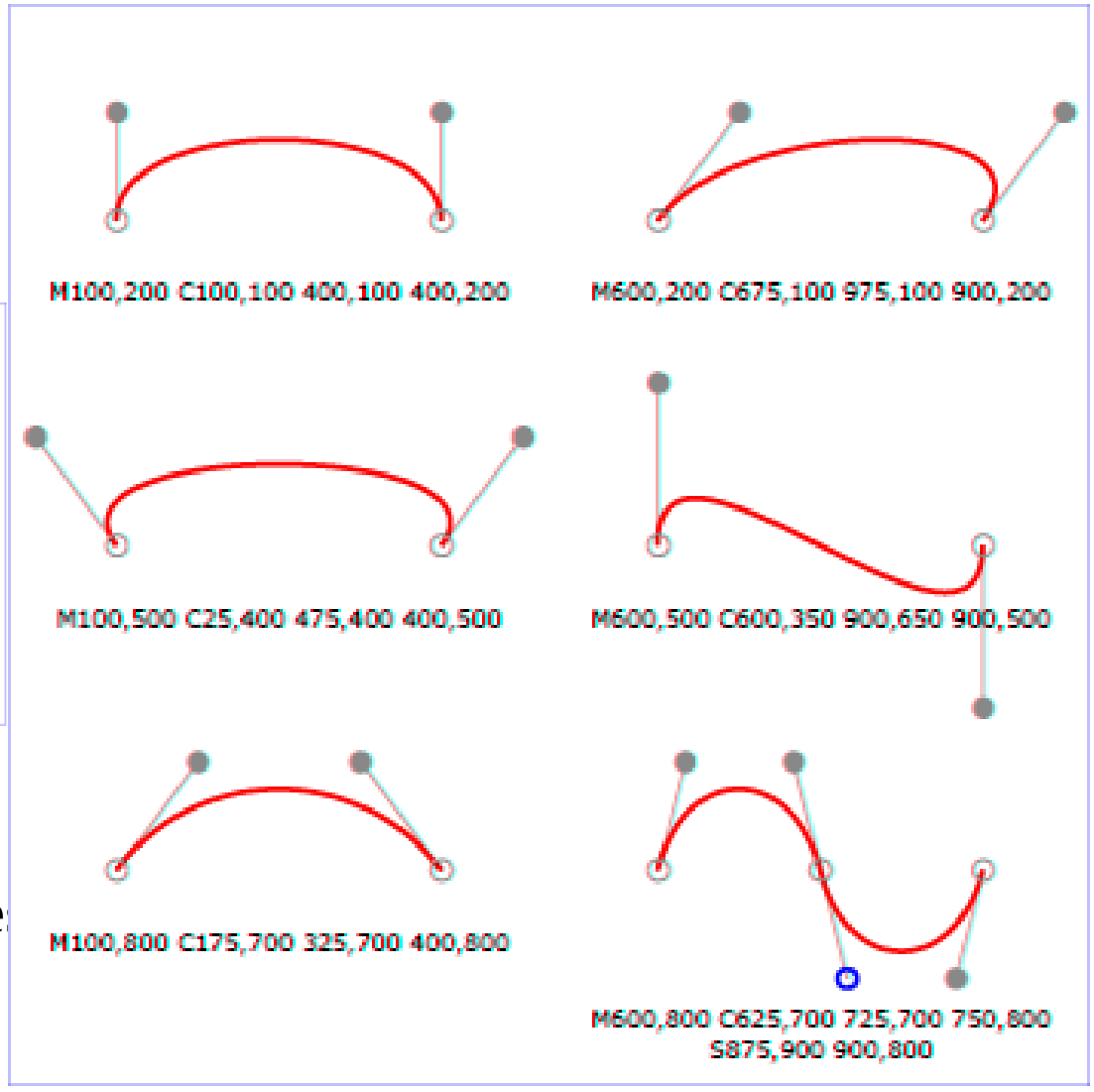
- SVG permet de définir des formes quelconques
  - **<path d="data"/> où data contient les codes**
    - M x y : aller à (x,y)
    - L x y : dessiner ligne vers (x,y)
    - H x : dessiner une ligne horizontale jusqu'à x
    - V y : dessiner une ligne verticale jusqu'à y
    - Z : fermer le chemin
    - A … : dessiner un arc d'ellipse
    - Q … , T … . dessiner une courbe de Bézier quadratique
    - C … , S … : dessiner une courbe de Bézier cubique
  - **les codes minuscules (m,l,h,v,...) sont utilisés avec des coordonnées relatives**

Les formes quelconques

Exemple de courbes
de Bézier quadratiques

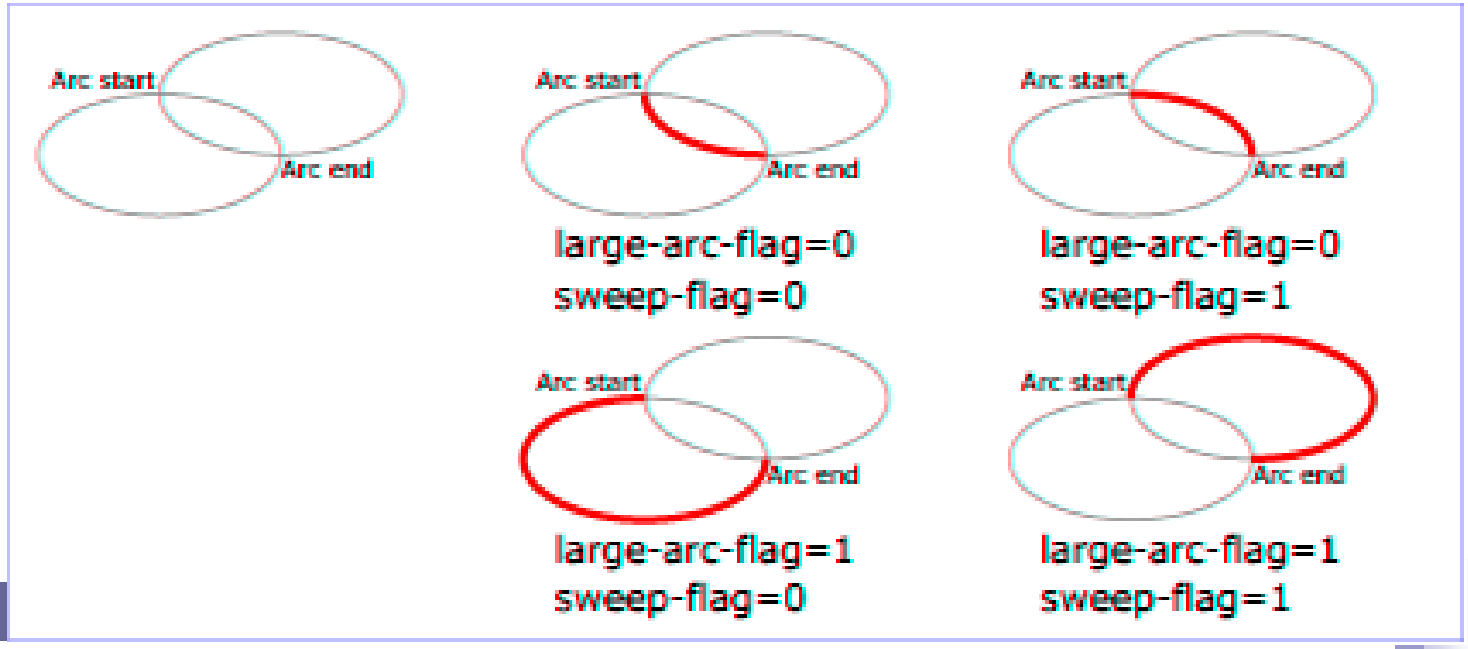


... et cubique

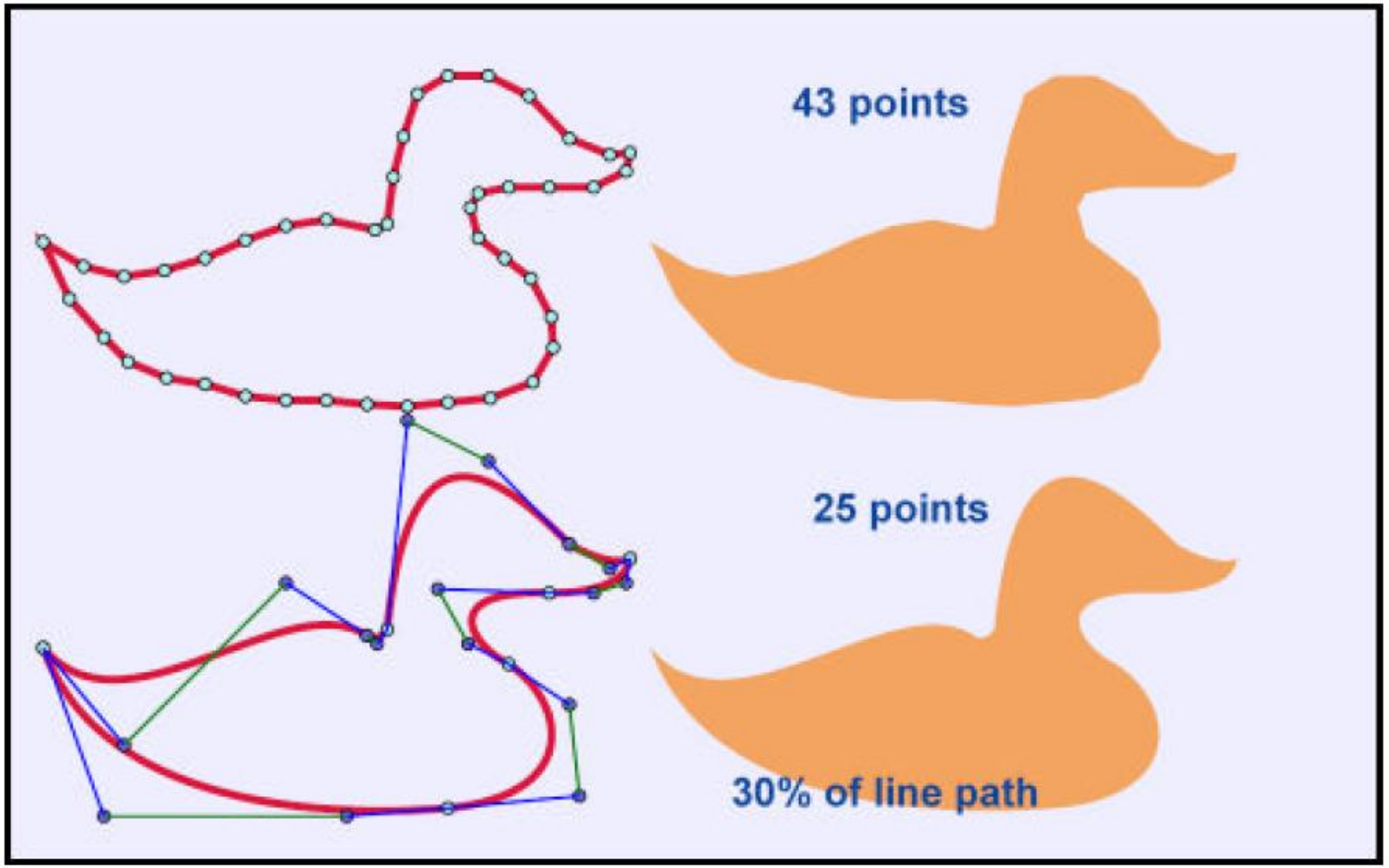


Arc elliptiques

- Dans un chemin un arc elliptique est représenté par
  - **A rx ry x-rot large-arc sweep x y où**
    - rx ry représentent les rayons en x et y
    - x-rot représente l'angle de rotation par rapport à l'axe des x
    - large-arc et sweep définissent le segment
    - x y représentent le point final de l'arc



Les formes quelconques



Les textes

- Une chaîne de caractères (texte) est représentée par
  - **<text x=... y=... >content</text>**
- Sa présentation est caractérisée par les attributs
  - **font-family, font-size, font-weight et font-style, text-decoration, letter-spacing, ... (comme dans CSS)**
  - **text-anchor avec les valeurs start, middle et end pour l'alignement**
  - **fill="color" pour la couleur**
- L'élément <tspan …> permet de redéfinir les attributs d'une sous-chaîne
- SVG supporte les caractère Unicode et il est possible d'écrire
  - **verticalement (caractères droits ou inclinés)**
  - **selon une orientation quelconque**
  - **en suivant une courbe**

Les textes : exemple

```
<text x="220" y="20">
<tspan x="220" dy="30">This is multi-line</tspan>
<tspan x="220" dy="30">text or text</tspan>
<tspan x="220" dy="30" style="fill:white;stroke:green">with different
   properties</tspan>
<tspan x="20" dy="30" rotate="10 20 30 40 0 50 60 70 0 80 90 0 100 110
   120 140 150 160 170 180">that can be produced</tspan>
<tspan x="220" dy="30">using the tspan element</tspan>
</text>
```

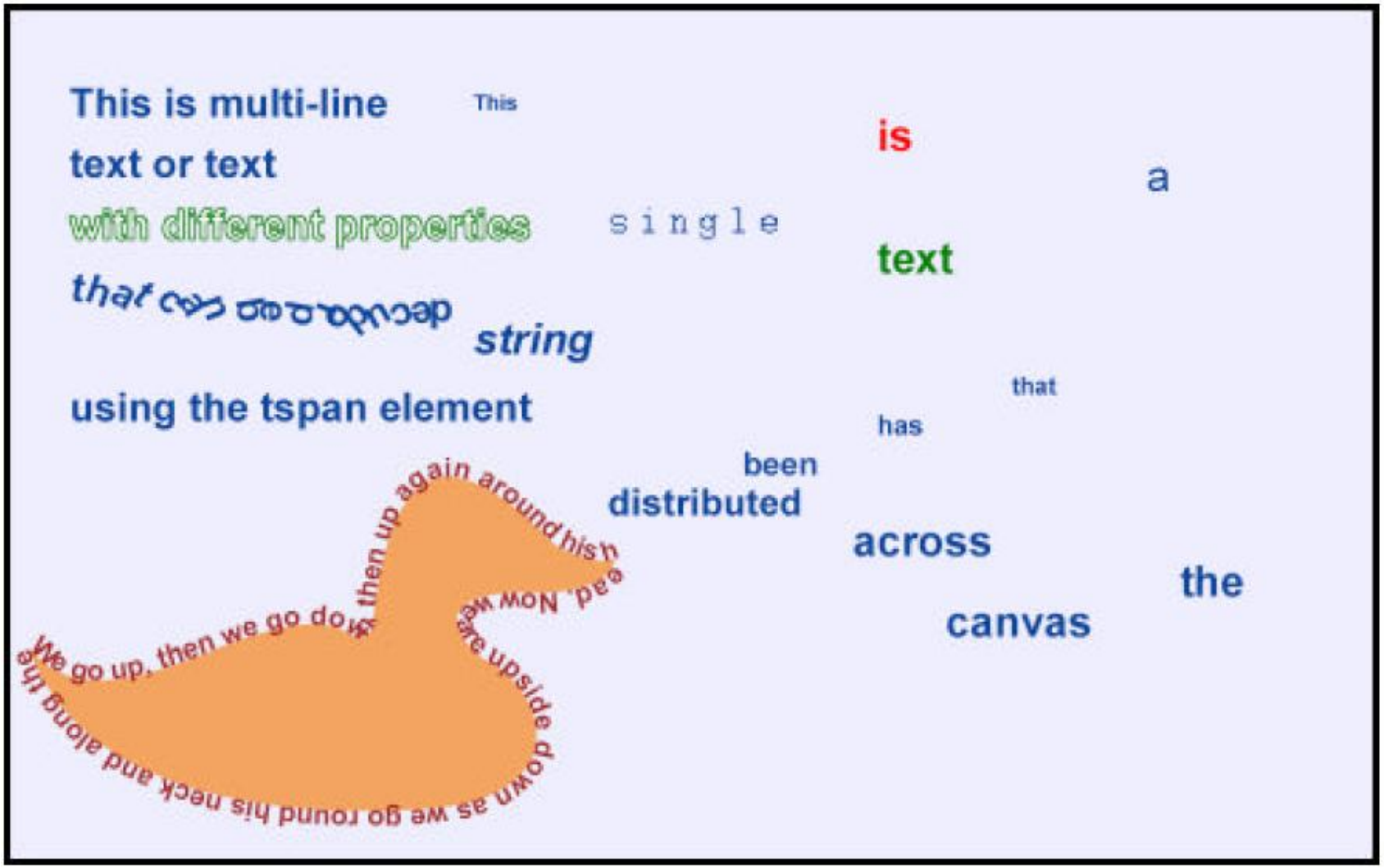
Les textes : exemple - suite

```
<path id="duck" d="M 0 312 C 40 360 120 280 160 306 C 160 306 165
    310 170 303
C 180 200 220 220 260 261 C 260 261 280 273 290 268 C 288 280 272
    285 250 285
C 195 283 210 310 230 320 C 260 340 265 385 200 391 C 150 395 30
    395 0 312 Z"/>
<text style="font-size:10">
<textPath xlink:href="#duck">
We go up, then we go down, then up again around his head. Now we are
    upside down as we
go round his neck and along the bottom to the tail.
</textPath>
</text>
```

Les textes : exemple – suite et fin

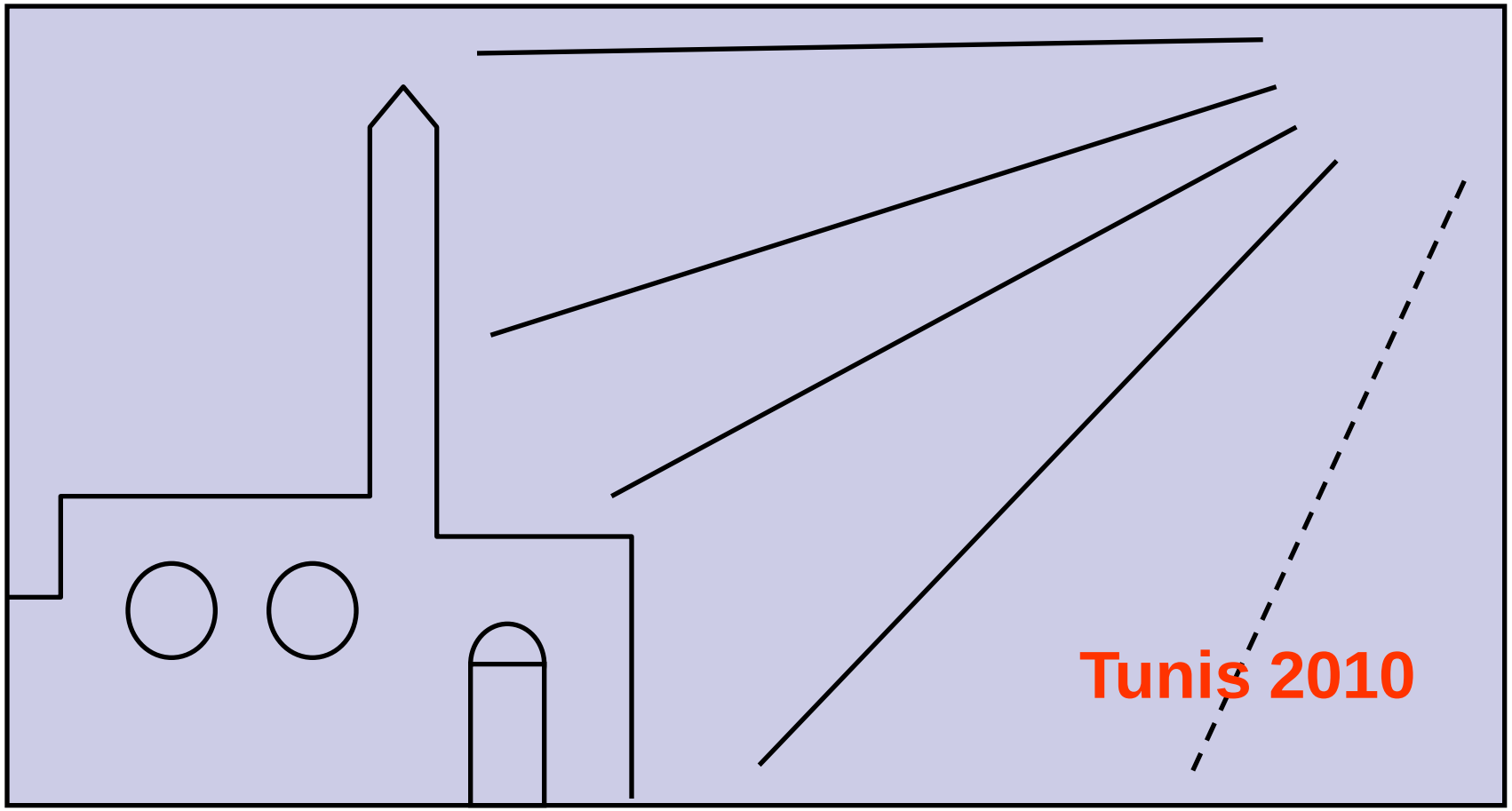
```
<text>
<tspan x="30" dy="30" font-size="16">This </tspan>
<tspan x="330" dy="30" fill="red">is </tspan>
<tspan x="530" dy="30" font-weight="normal">a </tspan>
<tspan x="130" dy="30" font-family="Courier" font-size="28">single
   </tspan>
<tspan x="330" dy="30" fill="green">text </tspan>
<tspan x="30" dy="60" font-style="italic">string </tspan>
<tspan x="430" dy="30" font-size="18">that </tspan>
<tspan x="330" dy="30" font-size="20">has </tspan>
<tspan x="230" dy="30" font-size="24">been </tspan>
<tspan x="130" dy="30" font-size="28">distributed <</tspan>
<tspan dx="30" dy="30">across </tspan>
<tspan dx="130" dy="30">the </tspan>
<tspan dx="-230" dy="30">canvas</tspan>
</text>
```

Les textes : exemple – suite et fin



Exercices récapitulatif

- Essayer de reproduire le graphique suivant :



Groupes d'éléments

- Grouper des éléments permet de :
  - **Partager des attributs**
  - **Utiliser des coordonnées relatives**
  - **Déplacer et modifier le groupe**

```
<g style="fill:red;stroke:black">
    <circle cx="70" cy="100" r="50" />
    <rect x="150" y="50" rx="20" ry="20" width="135" height="100" />
    <line x1="325" y1="150" x2="375" y2="50" />
    <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" />
    <ellipse cx="400" cy="300" rx="72" ry="50" />
</g>
```

Les transformations

- Une transformation « transform » appliquée à un groupe g permet de déplacer les coordonnées de l'ensemble des éléments de ce groupe.

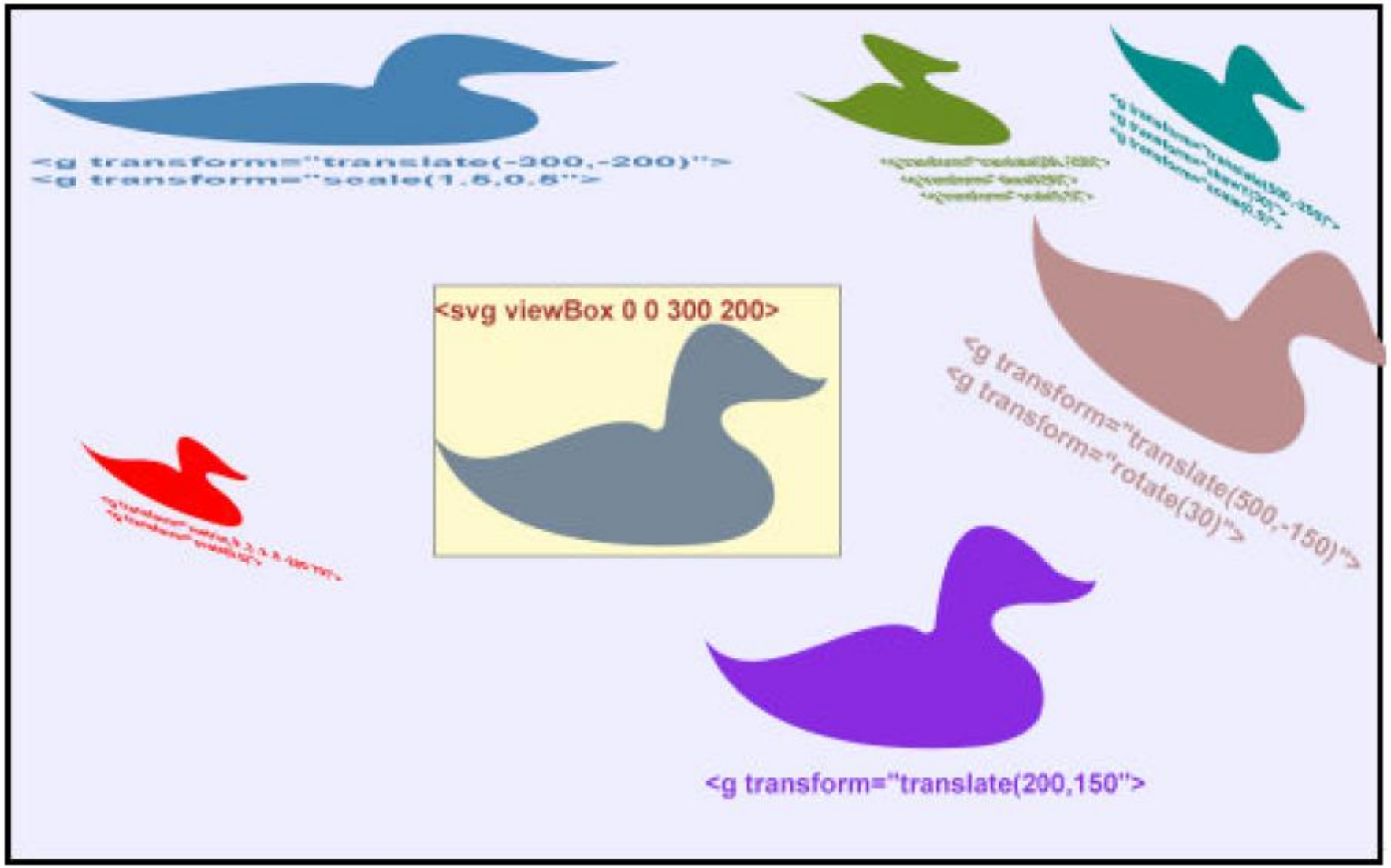
- Exemple :

```
<g transform="translate(100,0)">
<circle cx="70" cy="100" r="50" />
<rect x="150" y="50" rx="20" ry="20" width="135" height="100" />
</g>
```

Les transformations

- Avant d'être affiché, un objet graphique peut subir une transformation définie par transform="…"
  - **Translation des coordonnées : translate(tx,ty)**
  - **Rotation angulaire : rotate(a) ou rotate(a,cx,cy)**
  - **Mise en échelle : scale(f) ou scale(fx,fy)**
  - **Inclinaison : skewX(a), skewY(a)**
    - où les angle sont exprimés en degrés
- Les transformations peuvent être combinées

Les transformations



La découpe (clipping)

- Tout groupe d'élément peut être découpé selon un clipPath
- Le clipPath peut être une forme, un chemin ou un texte

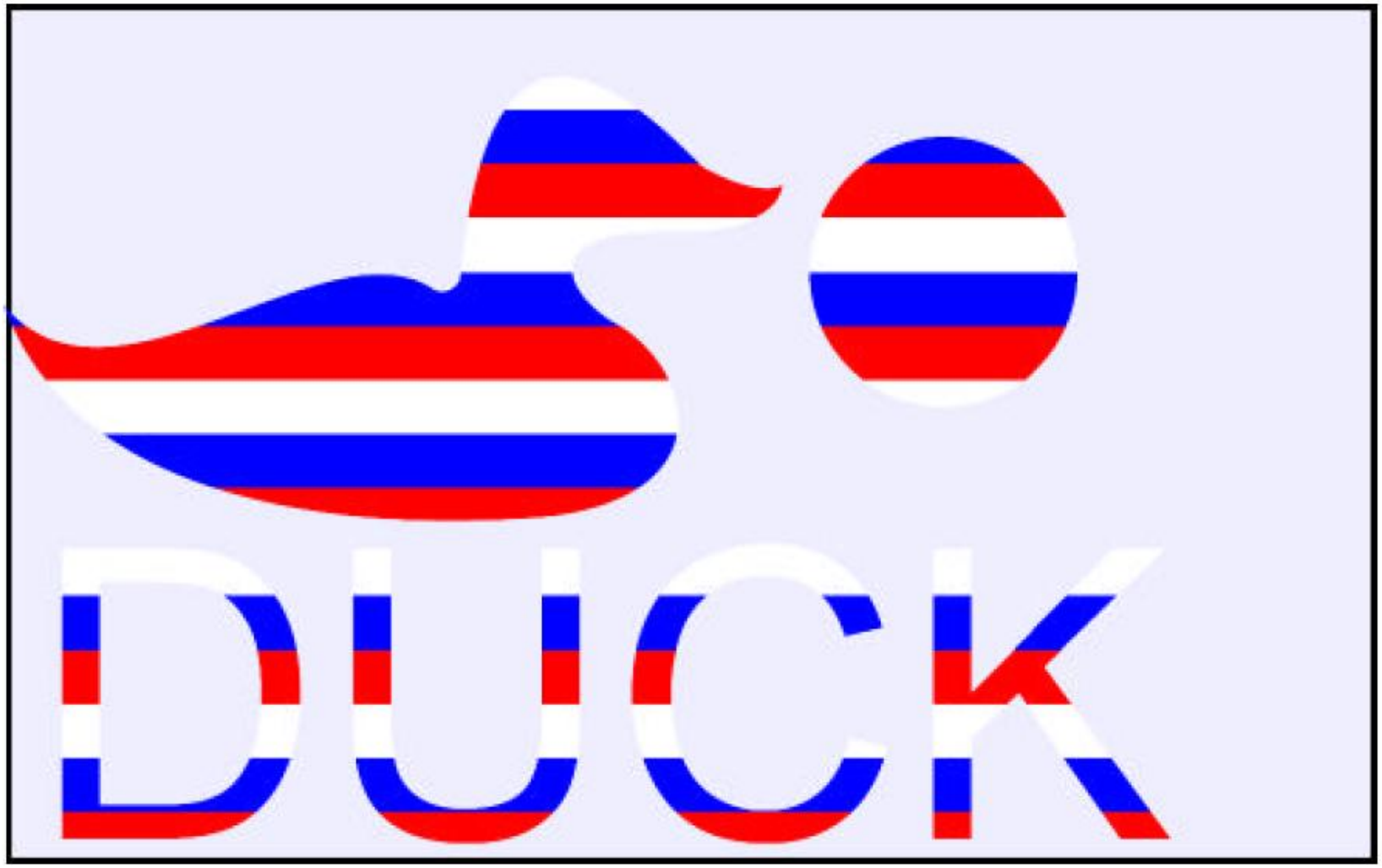
```
<clipPath id="myClip">
<circle cx="350" cy="100" r="50"/>
</clipPath>
<g style="stroke:none;clip-path:url(#myClip)">
<rect style="fill:red" x="0" y="0" width="500" height="20" />
<rect style="fill:white" x="0" y="20" width="500" height="20" />
<rect style="fill:blue" x="0" y="40" width="500" height="20" />
<rect style="fill:red" x="0" y="60" width="500" height="20" />
<rect style="fill:white" x="0" y="80" width="500" height="20" />
<rect style="fill:blue" x="0" y="100" width="500" height="20" />
<rect style="fill:white" x="0" y="120" width="500" height="20" />
<rect style="fill:blue" x="0" y="160" width="500" height="20" />
<rect style="fill:red" x="0" y="180" width="500" height="20" />
<rect style="fill:white" x="0" y="200" width="500" height="20" />
<rect style="fill:blue" x="0" y="220" width="500" height="20" />
<rect style="fill:red" x="0" y="240" width="500" height="20" />
<rect style="fill:white" x="0" y="260" width="500" height="20" />
<rect style="fill:blue" x="0" y="280" width="500" height="20" />
<rect style="fill:red" x="0" y="300" width="500" height="20" />
<rect style="fill:white" x="0" y="320" width="500" height="20" />
</g>
```

La découpe (clipping) – suite exemple

```
<clipPath id="myClip">
<path d="M 0 112 C 40 160 120 80 160 106 C 160 106 165 110 170 103 C
    180 0 220 20 260 61 C 260 61 280 73 290 68 C 288 80 272 85 250 85
    C 195 83 210 110 230 120 C 260 140 265 185 200 191 C 150 195 30
    195 0 112 Z"/>
</clipPath>
```

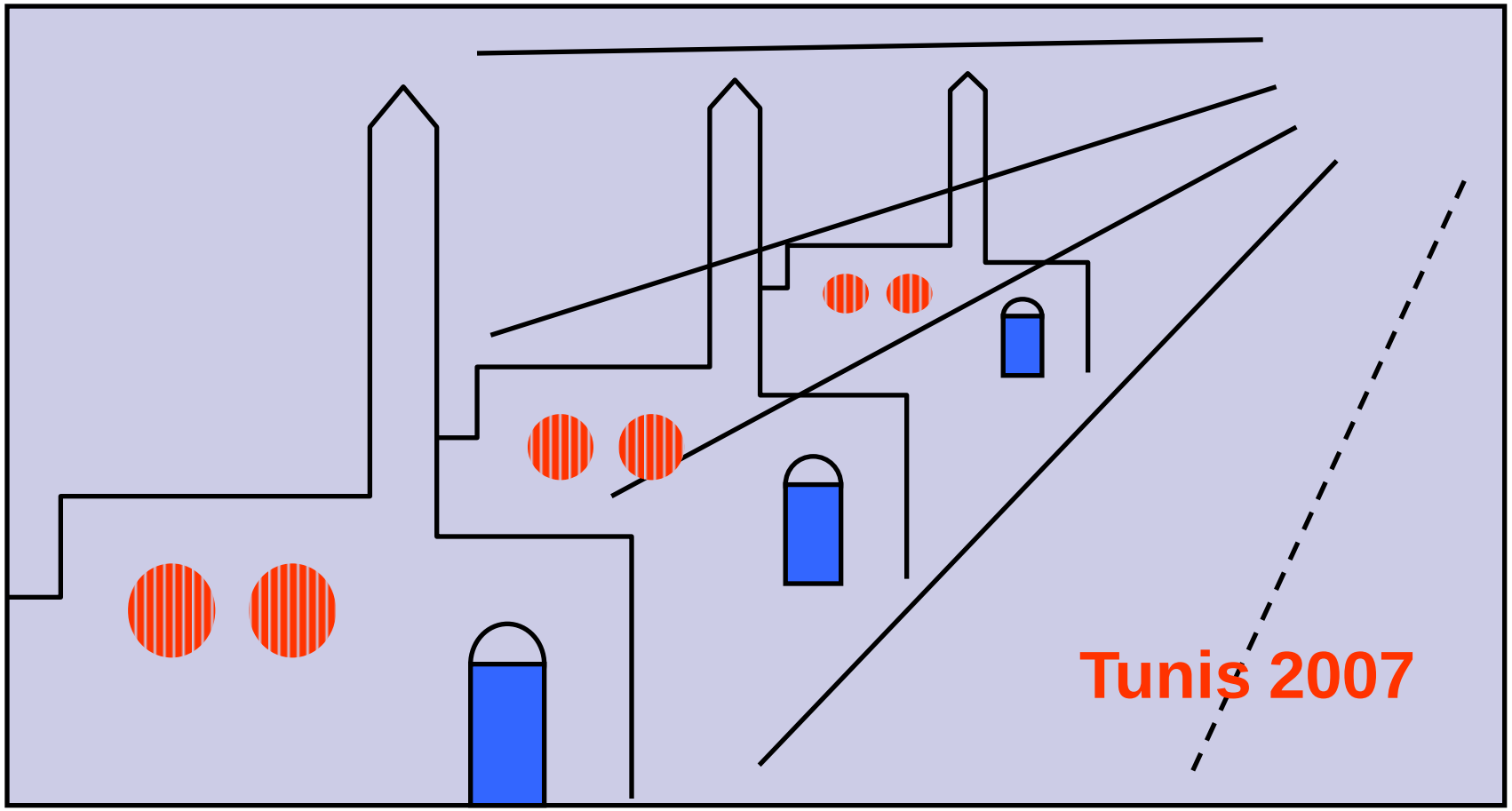
```
<clipPath id="myClip">
<text x="10" y="310" style="font-size:150">DUCK</text>
</clipPath>
```

La découpe (clipping) – suite exemple



Exercice récapitulatif

- Essayer de reproduire le graphique suivant :



Remplissage

- Les éléments et les groupes en SVG sont caractérisés par des attributs permettant de gérer :
  - **La couleur intérieure et la couleur des bordures en RGB ou en utilisant 149 couleurs prédéfinies**
  - **Opacité de l'intérieur et des bordures en allant de l'opaque au transparent**
  - **Le découpage en utilisant tout élément SVG fermé**
  - **Épaisseur et style des lignes**
  - **Règles de remplissage**
  - **Utilisation des patrons de couleurs ou des dégradés**

Remplissage

- Les attributs de base sont :
  - **Fill : la couleur de remplissage**
  - **Opacity : la transparence**
  - **Fill-rule : la région remplie dans les chemins complexes**

```
<path style="fill:red;opacity:0.5;fill-rule:evenodd" d="M10,20h100v50h-
    80v-70h-20v20z" />
```

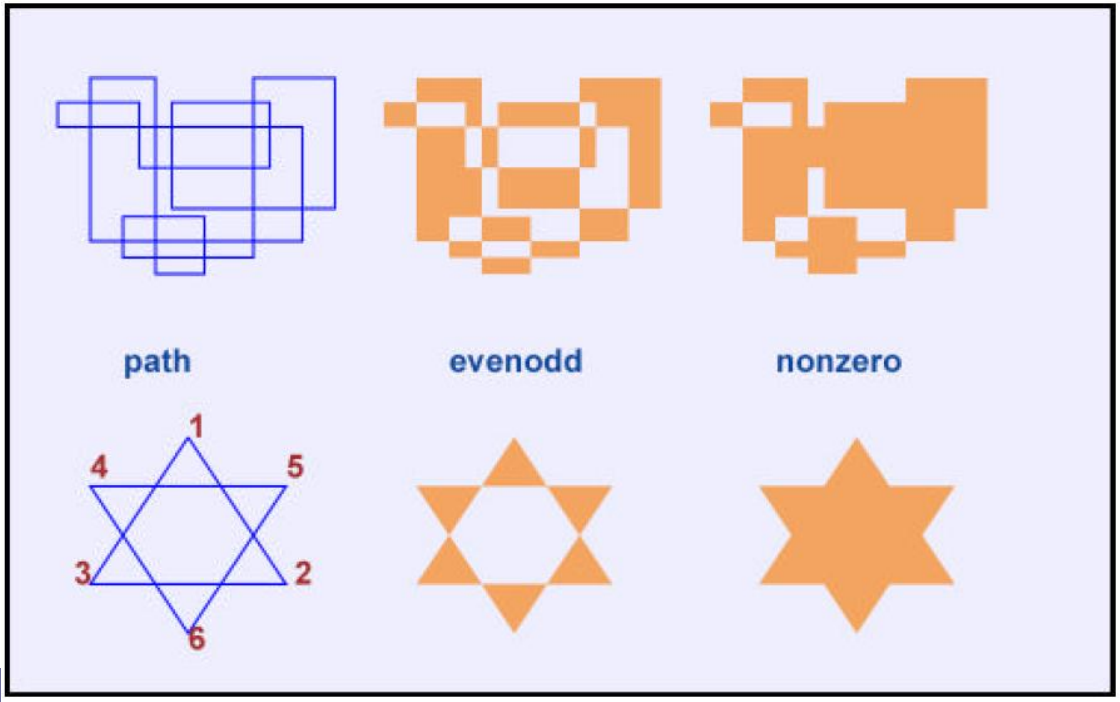
Couleurs

- Utilisation des noms prédéfinis
- Utilisation de valeurs RGB en décimal ou en Hexa

```
<rect width="10" height="10" style="fill:coral" />
<rect width="10" height="10" style="fill:rgb(255,127,80)" />
<rect width="10" height="10" style="fill:#f75" />
```

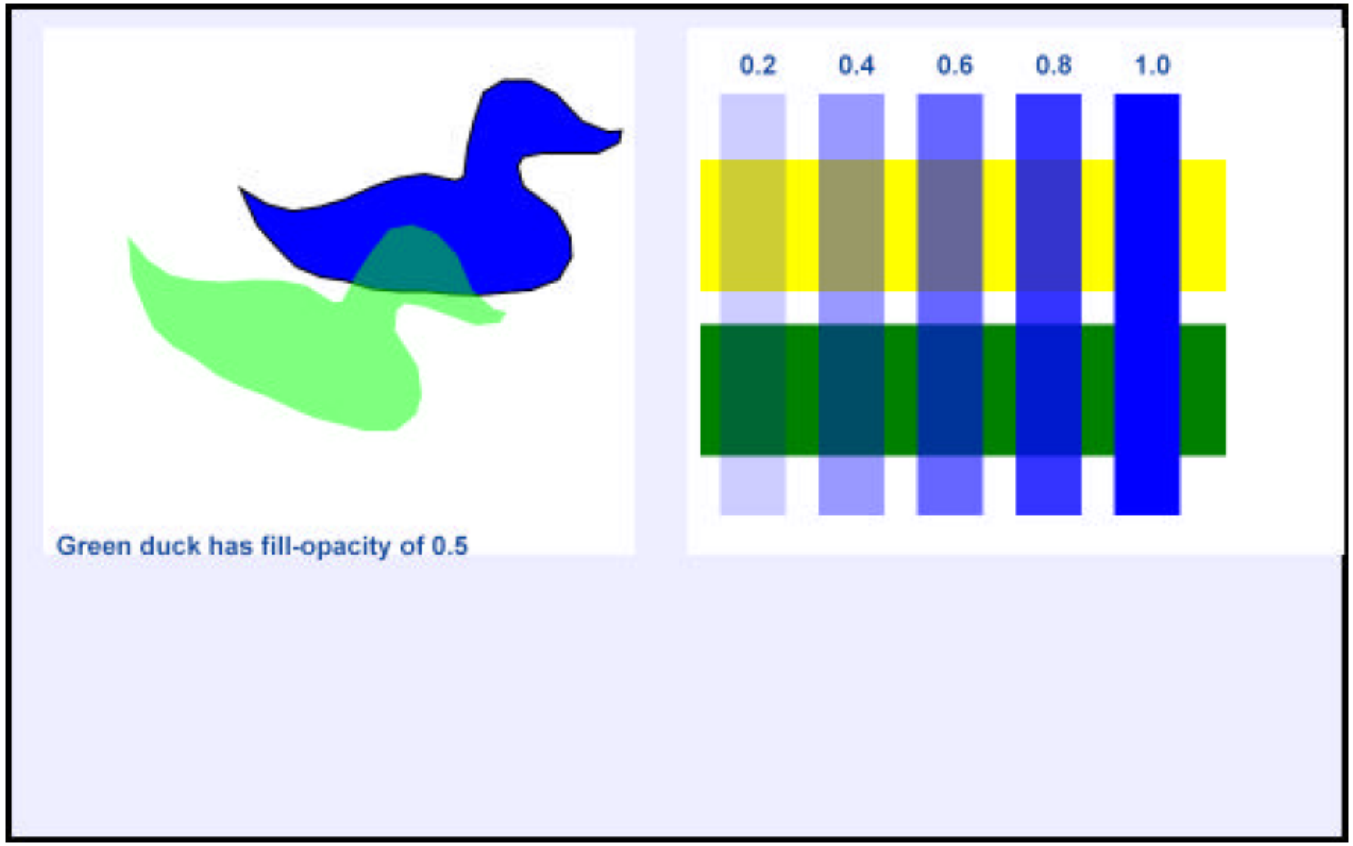
Règle de remplissage

- Pour les chemins complexes, il est parfois nécessaire de définir ce qui est intérieur ou extérieur
  - **Evenodd : si le nombre d'intersections d'une ligne au départ du point vers l'infini est impair, donc le point est interne**
  - **Non-zero : si le nombre de fois le chemin s'incline autour du point est différent de zéro, alors le point est interne**



Transparence - Opacité

- Un élément SVG peut avoir une opacité partielle entre 1 (opaque) et 0 (transparent)



Dégradé de couleurs

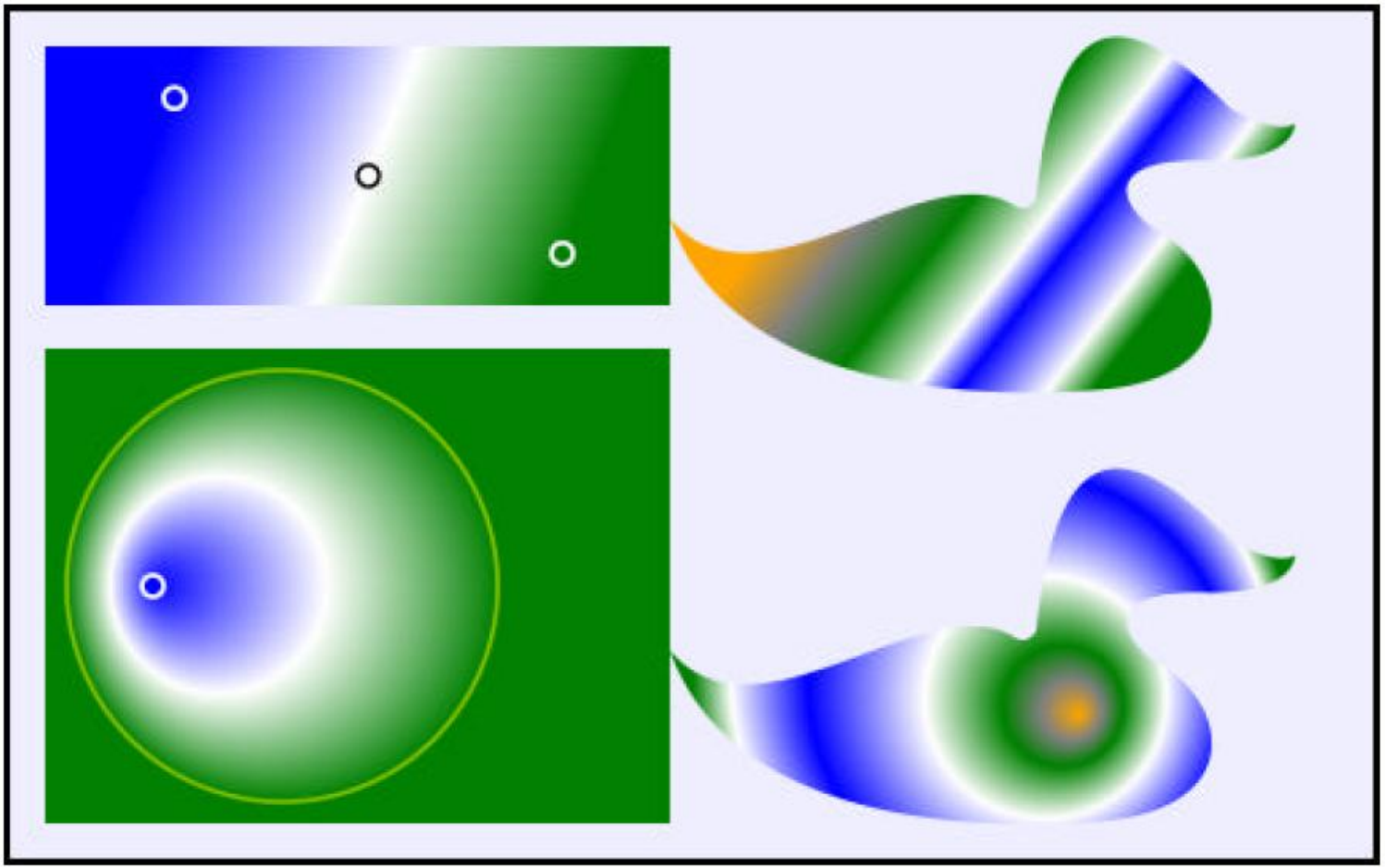
- Il est possible d'avoir des dégradés de couleurs entre 1 ou plusieurs couleurs dans un même élément. Il est aussi possible de définir des dégradé linéaires ou circulaires

```
<rect x="20" y="20" width="290" height="120"
    style="fill:url(#MyGradient)"/>
...
<linearGradient id="MyGradient" gradientUnits="userSpaceOnUse"
    x1="80" y1="44" x2="260" y2="116">
<stop offset="0" style="stop-color:blue"/>
<stop offset="0.5" style="stop-color:white"/>
<stop offset="1" style="stop-color:green"/>
</linearGradient>
```

Dégradé de couleurs

```
<radialGradient id="MyGradient2" gradientUnits="userSpaceOnUse"
    cx="130" cy="270" r="100" fx="70" fy="270">
<stop offset="0" style="stop-color:blue"/>
<stop offset="0.5" style="stop-color:white"/>
<stop offset="1" style="stop-color:green"/>
</radialGradient>
...
<rect x="20" y="160" width="290" height="220"
    style="fill:url(#MyGradient2)"/>
```

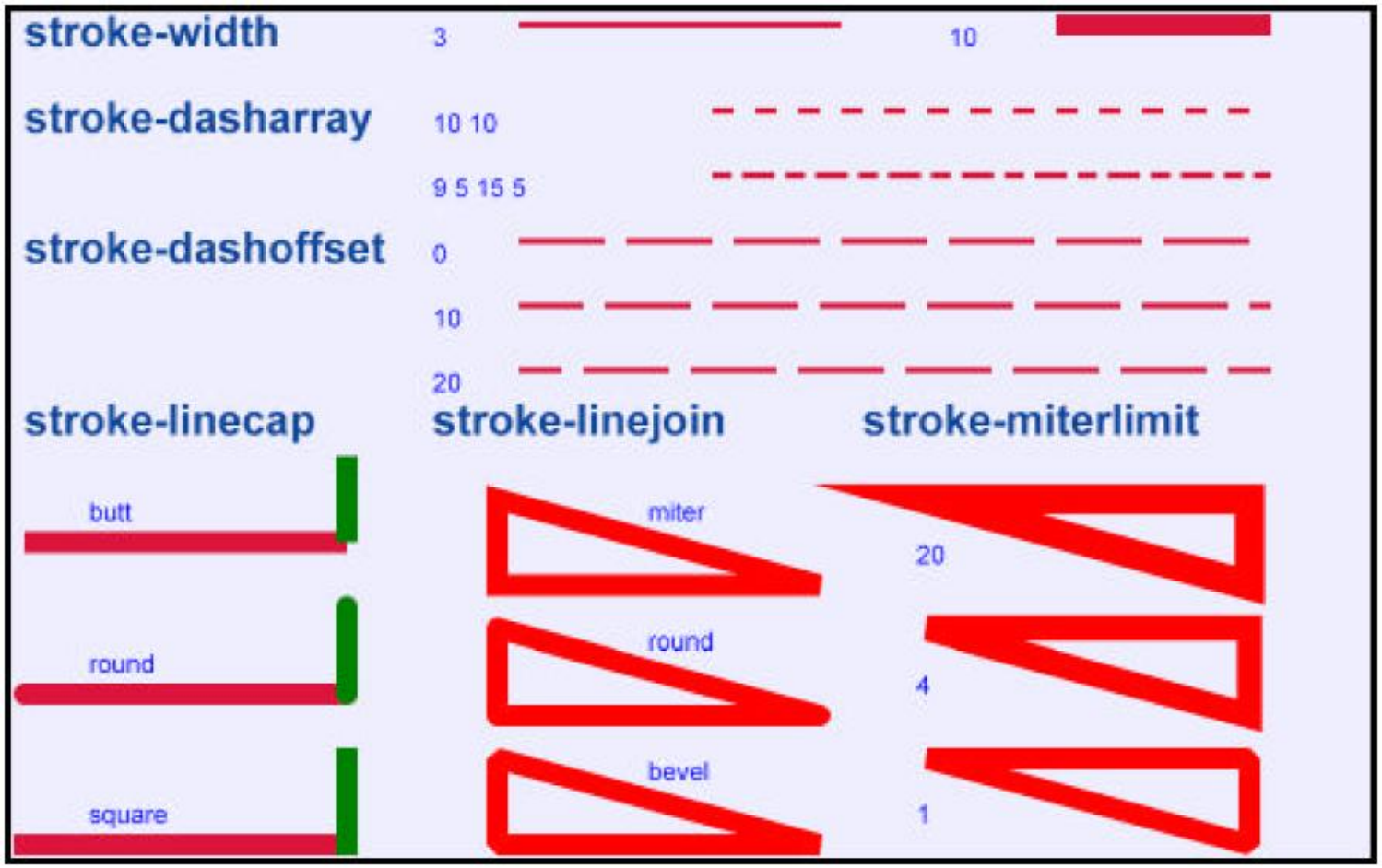
Dégradé de couleurs



Styles de lignes

- Plusieurs attributs permettent de définir le style des lignes, par exemple :
  - **stroke: la couleur ou le gradient de la ligne.**
  - **stroke-width: largeur de la ligne.**
  - **stroke-dasharray: style de ligne (dotted, solid, dashed).**
  - **stroke-dashoffset: début du pointillé.**
  - **stroke-linecap: la fin de ligne.**
  - **stroke-linejoin: le rendu d'une intersection de ligne.**

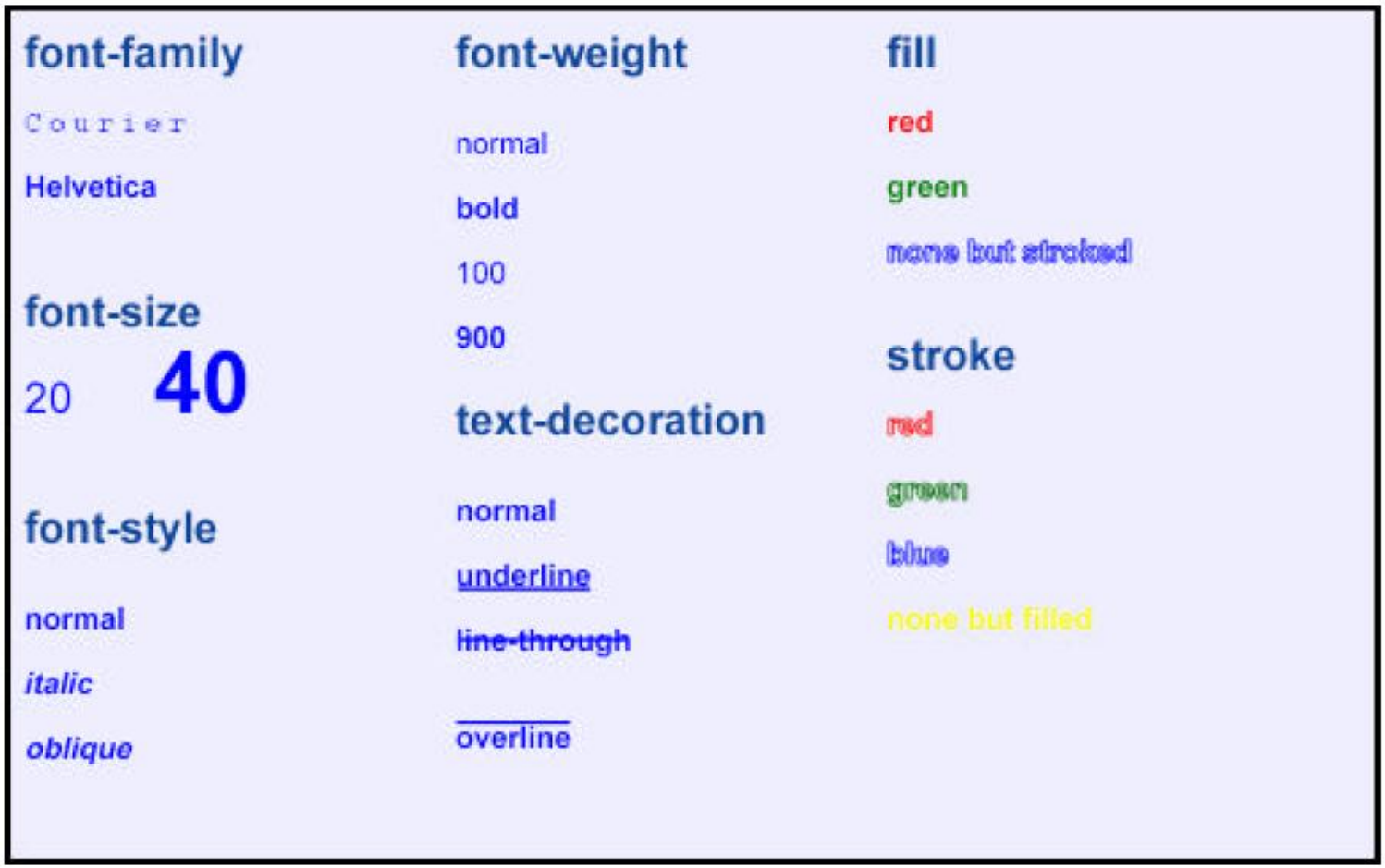
Styles de lignes



Styles de textes

- Les textes sont caractérisés principalement par 3 attributs :
  - **Font-family : taille de font**
  - **Font-style : style de font – bold – italic - …**
  - **Font-weight : comme pour les css**
  - **Text-decoration : comme pour les css**

Styles de textes



Animation

- SVG offre un mécanisme pour créer des animations
- Le principe consiste à inclure des éléments d'animation à l'intérieur des éléments graphiques
- Les animations permettent de
  - **déplacer ou déformer (pivoter, étirer, rétrécir, ...)**
  - **changer sa couleur**
  - **déplacer sur une trajectoire**
- Les fonctions d'animation sont reprises de SMIL2 (Synchronized Multimedia Integration Language, level 2)

Animation d'un attribut

- L'élément <animate …> spécifie un attribut et comment sa valeur évolue au cours du temps
  - **attributeName="..." spécifie le nom de l'attribut**
  - **from="..." et to="..." fixent les bornes des valeurs**
  - **begin="..." et end="..." (ou dur="...") déterminent le début et la fin (resp. la durée) de l'animation**
  - **fill="freeze" permet à la fin de geler l'animation**

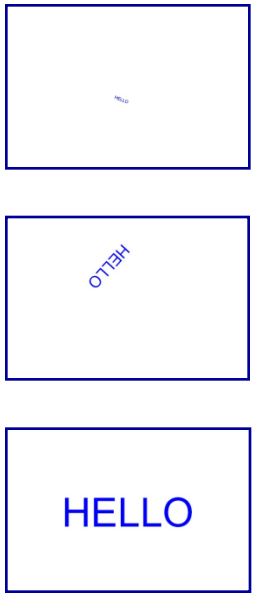
- <animateColor …> permet de d'animer les couleurs

- <animateTranform …> et <animateMotion …> permettent des mouvements complexes

Animation avec les transformations

- L'élément <animateTranform …> avec attributeName="transform" type="…" permet l'application d'une transformation géométrique

```
<svg width="600" height="400">
  <text x="300" y="240" text-anchor="middle"
   font-size="20" fill="blue">
    HELLO
    <animate attributeName="font-size"
     from="5" to="100" dur="2s" fill="freeze"/>
    <animateTransform
     attributeName="transform" type="rotate"
     from="0 300 200" to="360 300 100" dur="2s"/>
  </text>
</svg>
```



Mouvement le long d'un chemin

- L'élément <animateMotion …> permettent un déplacement le long d'un chemin défini par path="…"

```
<svg width="600" height="400">
  <text text-anchor="middle" font-size="60" ...>
    HELLO
    <animateMotion path="M 50 50 Q 150 300 250 200
     Q 350 100 450 300" dur="4s" fill="freeze"/>
  </text>
</svg>
```

- Par contre, il est pratiquement impossible d'animer une courbe définie par un élément <path> !

Liens

- Il est possible dans SVG d'utiliser les éléments a pour définir des liens

```
<a xlink:href="http://www.w3.org">
<rect width="200" height="40" />
<text x=100" y="30" style="text-anchor:middle">My button</text>
</a>
```

Images

- Avec l'élément <image xlink:href="url" …>, il est possible d'importer des images en format JPEG, PNG ou un autre fichier SVG
  - **x=…, y=…, permettent de positionner l'image**
  - **width=… et height=… permettent de dimensionner l'image**
- Il est également possible d'appliquer des filtres

Masque

```
<linearGradient id="Gradient" gradientUnits="userSpaceOnUse" x1="0"
    y1="0" x2="500" y2="0">
<stop offset="0" style="stop-color:white; stop-opacity:0"/>
<stop offset="1" style="stop-color:white; stop-opacity:1"/>
</linearGradient>
<rect x="0" y="0" width="500" height="60" style="fill:#FF8080"/>
<mask maskContentUnits="userSpaceOnUse" id="Mask">
<rect x="0" y="0" width="500" height="60" style="fill:url(#Gradient)" />
</mask>
<text x="250" y="50" style="font-family:Verdana; font-size:60; text-
    anchor:middle;fill:blue;mask:url(#Mask)">
MASKED TEXT
</text>
<text x="250" y="50" style="font-family:Verdana; font-size:60; text-
    anchor:middle;fill:none;stroke:black; stroke-width:2">
MASKED TEXT
</text>
```

Masque

Interaction

```
<svg width="500" height = "500">
<script language="JavaScript"
type="text/javascript">
<![CDATA[
function onmouseover(evt) {
var elem = evt.target;
var style = elem.getStyle();
style.setProperty("fill-opacity", 0.5);
}
function onmouseout(evt) {
var elem = evt.target;
var style = elem.getStyle();
style.setProperty("fill-opacity", 1.0);
}
]]></script>
<rect x="20" y="20" width="250" height="250" style="fill:red; stroke:none"
onmouseover="onmouseover(evt)" onmouseout="onmouseout(evt)" />
<rect x="210" y="210" width="250" height="250" style="fill:green; stroke:none"
onmouseover="onmouseover(evt)" onmouseout="onmouseout(evt)" />
</svg>
```

Interaction

- Il est possible de réagir aux évènements :
  - **onclick**
  - **onactivate**
  - **onmousedown**
  - **onmouseover**
  - **onmousemove**
  - **onmouseout**
  - **onload**

- Et définir des traitements sur des éléments SVG :
  - **getElementById**
  - **getStyle**
  - **setProperty**
  - **setAttribute**
  - **getAttribute**
  - **cloneNode**

Utilisation de style en SVG

- Il est possible d'utiliser des styles en SVG :

```
<svg viewbox= "0 0 500 300" >
<style type="text/css"> <![CDATA[
rect {stroke:black; fill:yellow}
rect.different {stroke:red; fill:none}
]]>
</style>
...
<rect x="20" y="30" width="300" height="200"/>
<rect class="different" x="20" y="330" width="300" height="200"/>
</svg>
```

Objectifs du projet SMIL

- Format textuel de documents pour l'intégration d'objet média (le html du multimédia).
- Utiliser les technologies du web pour le multimédia : XML, Namespaces, ... DOM
- Promotion de la notion de documents temporisés et de synchronisation à l'échelle des standards du web
- Neutralité vis-à-vis des protocoles d'accès réseaux et formats des médias RTP, RTSP, Mpeg,...
- Rassembler les industriels du multimédia et du web autour d'un format ouvert (le défi)

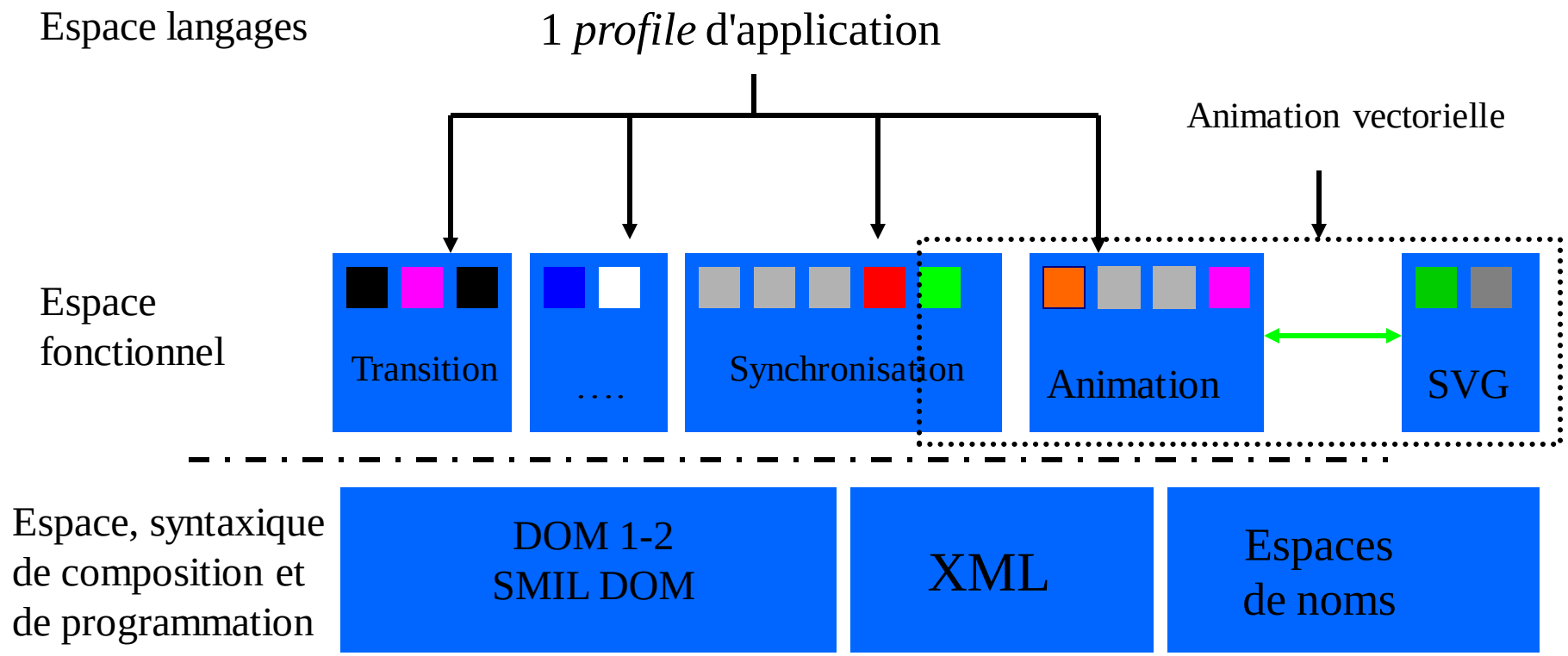
Organisations et entreprises impliquées

- Les principaux développeurs d'applications
  - **Oratrix, Real Networks, Microsoft, IBM, Macromedia, Intel, Philips, Panasonic, Nokia Produits**
  - **Institution publiques: INRIA, CWI, NIST, WGBH**

- Les forces de SMIL
  - **Version 1.0 est un succès relatif …**
  - **Très simple à apprendre et à utiliser**
  - **Plus en plus d'intégration avec les autres standards du web**

SMIL 2.0 : Principe de conception

- Méta-langage qui permet de décrire le document multimédia du plus simple au plus sophistiqué



SMIL 2.0 : espaces fonctionnels

- Les fonctionnalités couvertes dans SMIL 2.0 sont :
  - **Layout -- positionnement sur l'écran et sur les canaux audio**
  - **Content Control -- sélection du contenu, adaptation, optimisation**
  - **Structure -- la colle pour les autres modules**
  - **Metainformation -- méta-données sur le contenu**
  - **Timing and synchronisation -- le cœur de la bête**
  - **Linking -- navigation hypermédia**
  - **Media object -- média de base intégré dans une présentation**
  - **Time manipulations -- accélérateur du temps**
  - **Transition effects -- rendus, effets visuels ...**

SMIL 2.0 : espaces langagiers

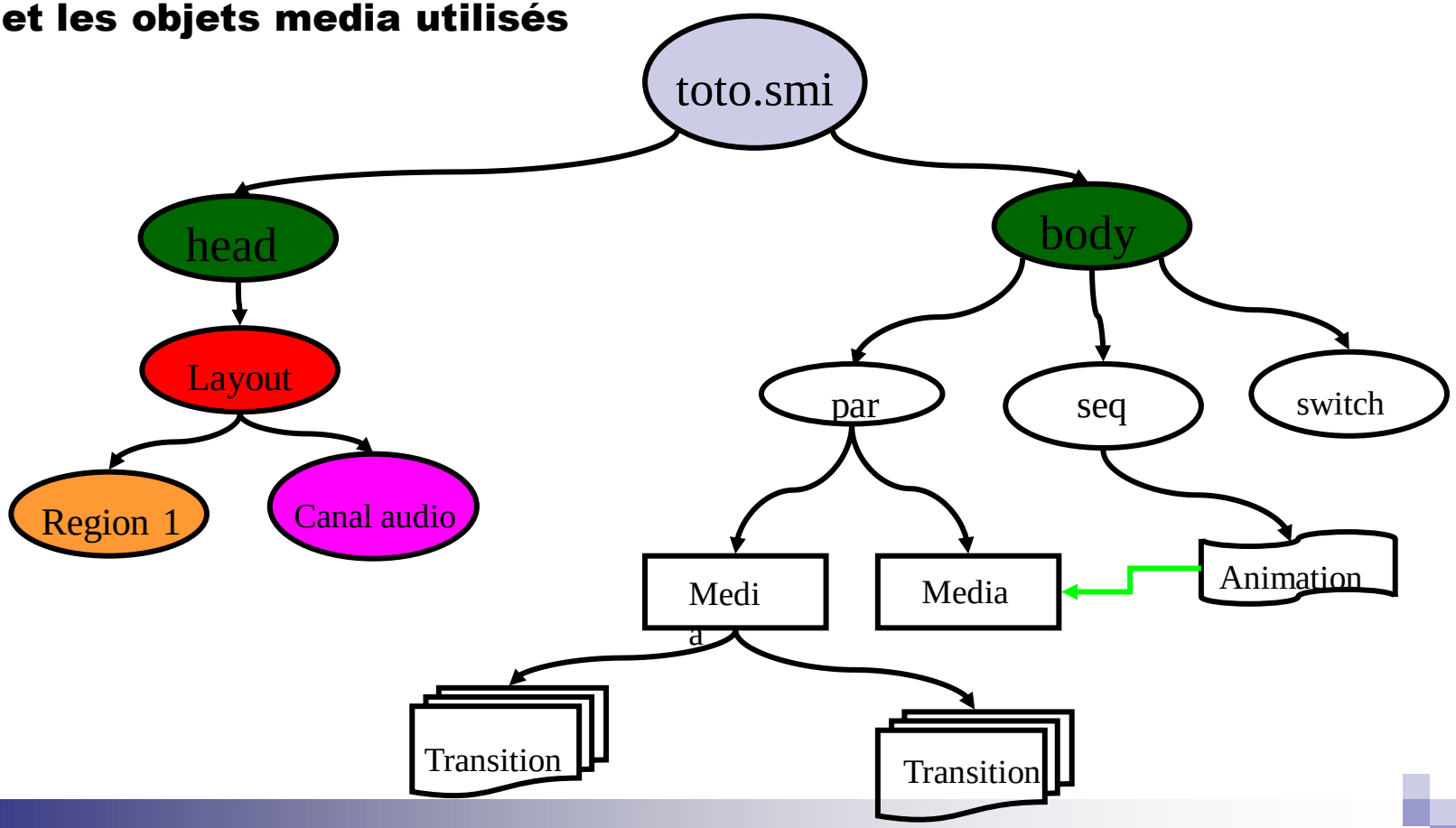
- Un profile :
  - **Langage qui correspond à un type d'applications (DTD, Schema)**
  - **Composition de l'espace fonctionnel (modules)**
  - **Intégration avec des modules extra-SMIL (Animation SVG)**
- SMIL 2.0 Language Profile (SMIL Profile) :
  - **Successeur de SMIL 1.0 (compatibilité ascendante)**
  - **Langage XML, une syntaxe et une sémantique**
  - **Composition de la plupart des fonctionnalités de SMIL 2.0**
- SMIL 2.0 Basic Language Profile :
  - **Langage pour les téléphones et PDA ….**
  - **Mécanisme d'extensibilité pour prendre en compte l'hétérogénéité**
- XHTML + SMIL
  - **Médias de base sont les éléments de XHTML**
  - **Fusion (grâce aux espaces de noms) des deux langages**

Documents SMIL typiques

- Un ensemble de “composants” accessibles via des urls,
  - **le contenu n’est pas inclus dans SMIL**
- Ces composants peuvent avoir des types de médias différents :
  - **audio, vidéo, texte, image, etc.**
- Synchronisation : intra- inter-objets et clip - sync
- Interactions des utilisateurs

Organisation d'un document SMIL

- Deux parties :
  - **Entête : contient des informations du niveau du document**
  - **Corps : contient le scénario temporel, les animations, les transitions et les objets media utilisés**



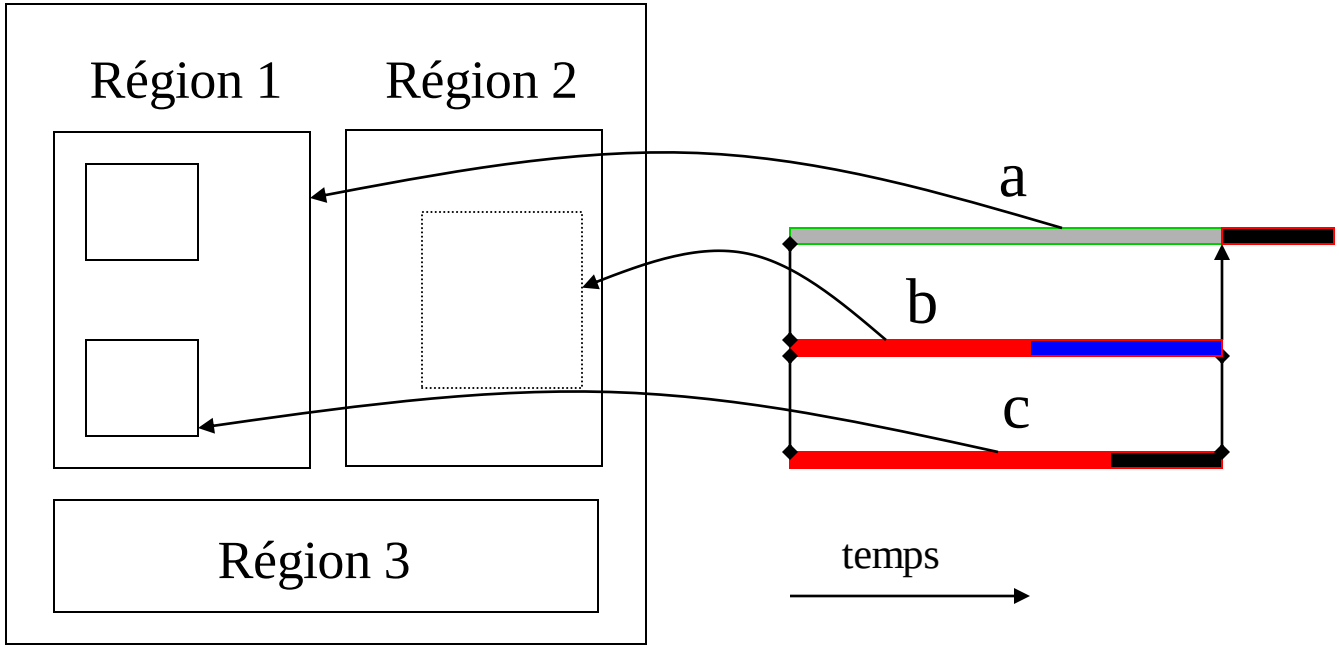
```
<smil xmlns="http://www/w3.org/2000/SMIL20/Language">
    <head>
      <layout type="text/smil-basic">
                <region id="left-video" left="20" top="50" z-index="1"/>
                <region id="left-text" left="20" top="120" z-index="1"/>
                <region id="right-text" left="150" top="120" z-index="1"/>
      </layout>
     </head>

    <body>
      <par>
      <seq>
                          <img src="graph" region="left-video" dur="45s"/>
                          <text src="graph-text" region="left-text"/>
                </seq>
                <par>
                          <a href="http://www.w3.org/People/Berners-Lee">
                                    <video src="tim-video" region="left-video"/>
                          </a>
                          <text src="tim-text" region="right-text"/>
                </par>
                <seq>
                          <audio src="joe-audio"/>
                          <video id="jv" src="joe-video" region="right-video"/>
      </seq>
       </par>
     </body>
</smil>
```

Entête

Corps = scénario

Aspect Spatial



Régions hiérarchiques et sous-régions pour le placement spatial

Corps du document : synchronisation

- Contient le scénario temporel du document
- Un scénario est défini de façon récursive : Schedule elements
  - **Schedule = Parallel | Seq | Excl |**
    | Media object
    | ancres (dep/arr)
    | Switch
    | priorityClass
    | Prefetch

Objets média de base

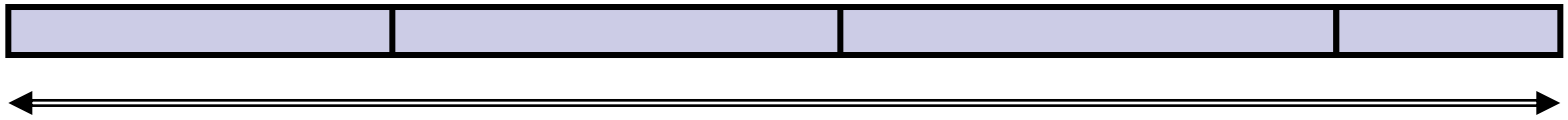
- Objets média marqués avec les balises :
  - **Audio, Video, Text, Img, Textstream, Animation, Ref, Param, et … Prefetch**

- Attributs :
  - **Src : localise le fichier du media de base (URL) rtsp://rtsp.example.org/video.mpg**
  - **Type : type mime (eg. video/mpeg)**
  - **Region : identifiant d'une surface d'affichage**
  - **Dur : durée de l'objet média**

Attributs de synchronisation

- L'attribut dur (duration)
  - **"intrinsic": la durée est celle du média (la durée du fichier externe).**
  - **"explicit": la durée est spécifiée dans le document (dur= "15 s")**
- L'attribut repeat
  - **RepeatCount="3"**
  - **la durée est celle du média (la duree du fichier externe).**

- RepeatDur="12 s" :



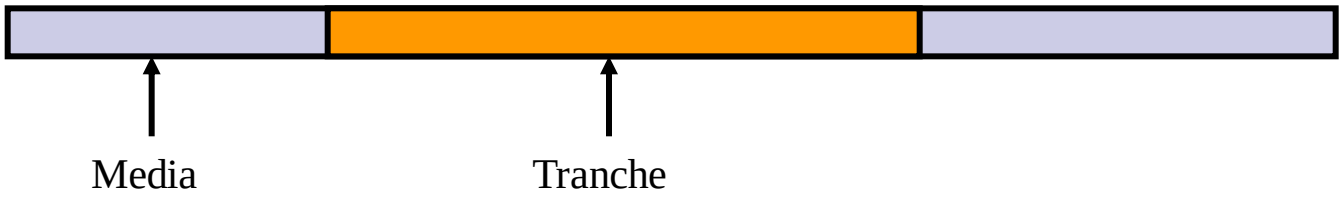
Attributs de synchronisation

- L'attribut begin, end
  - **Valeur (begin= "13 s") : décalage par rapport à l'élément père.**
  - **Référence à une autre horloge : (begin= "e2.end + 5 s ")**
  - **Référence au temps absolu :**
    (begin= "wallclock(2001-01-01Z)"
  - **Référence à un événement asynchrone (interactivité):**
    (begin= "bouton.click")

Découpage des media

- Clipping spatial effectué à travers les régions et sous-régions
- Clipping temporel effectué avec les attributs clip-begin et clip-end (médias sont externes)

```
<video id="a" src="attendre.mpg"
  clip-begin="smpte=00:01:45"
  clip-end="smpte=00:01:55"
  …
/>
```



L’élément séquentiel : seq

- Sémantique : jouer en séquence un ensemble d’objets
- Attributs
  - **Fill : utilisé pour la « persistance » sur l’écran**
  - **Remove : effacer de l’écran dès la terminaison**
  - **Freeze : garder la dernière image après terminaison**

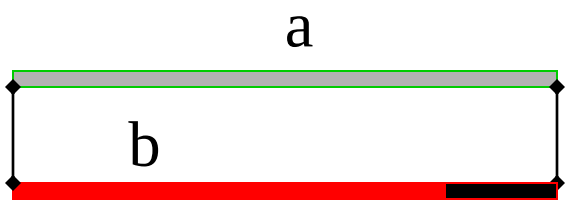
```
<seq>
  <image id="a" regionName="x" src="attendre.gif"
     fill="freeze"/>
  <video id="b" regionName="x" src="video.au dur="20 s" />
</seq>
```

L’élément parallèle : par (1)

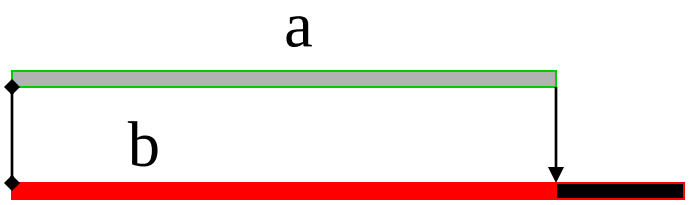
- Sémantique :
  - **Jouer en parallèle un ensemble d’objets**
  - **Terminaison : date maximale du dernier**
- Attributs :
  - **endSync : Last (Rendez-vous)**
  - **Dur : Horloge de référence du par(Wall clock)**
  - **Begin/End : Arc de synchronisation**

L’élément parallèle : par (2)

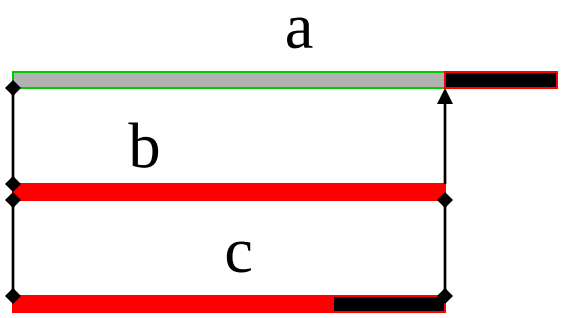
Last



First

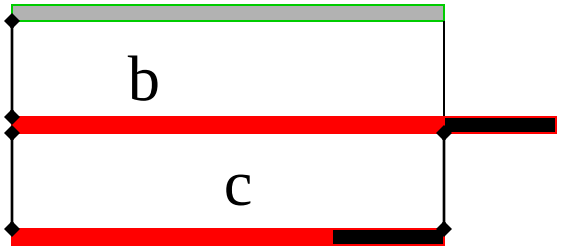


Master (b)



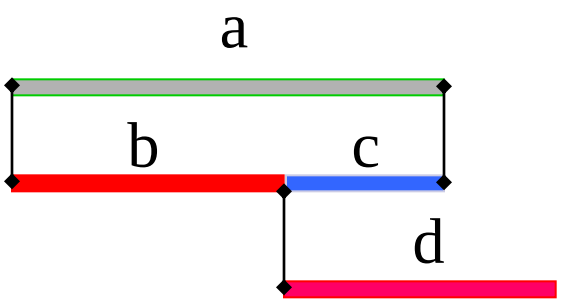
Wall clock

par



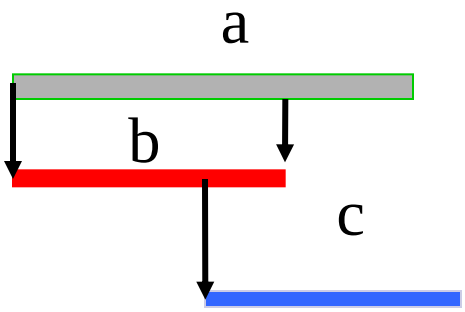
Arcs de synchronisation et événements

- Permet de construire des structures de graphe :




```
...
<par>
      <audio id="a" src="audio.au" begin="b.begin" />
      <video id="b" src="video.au end="c.begin" />
      <text id="c" src="text" begin="d.begin"
                                end=a.end  />
      <image id="d" src="image.gif" begin="b.end/>
</par>
...
```

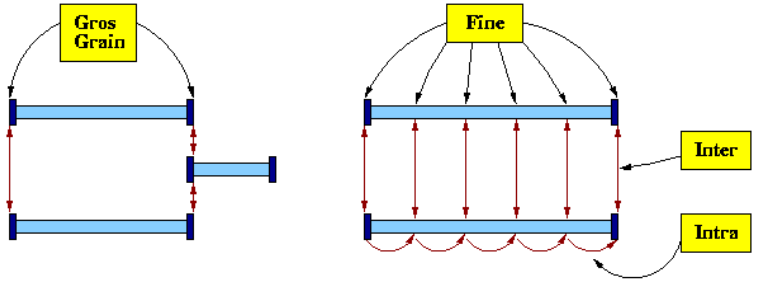
Déclenchement sur événements :

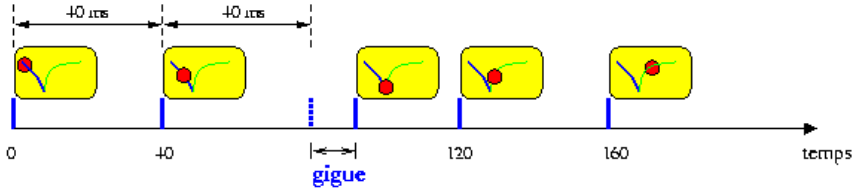



```
...
<par>
 <img id="a" src="image" />
 <video id="b" src="video.mpg" begin="a.activateEvent"
    end="a.activateEvent />
 <text id="c" src="text" end="b.focusInEvent" />
</par>
...
```

syncBehavior et syncTolerance

- syncBehavior
  - **canSlip : la synchro est faible, l'élément fils peut se décaler par rapport au père**
  - **locked : la synchronisation est forte, montant du décalage toléré (syncTolerance).**
  - **Independent : synchro complètement indépendante**
- syncTolerance =“nombre de secondes de décalage du gigue”
- syncMaster=“true” élément métronome de l'élément par





L’élément switch

- Un élément à choisir parmi un ensemble d ’éléments alternatifs
- Choix est basé sur des valeurs d’attributs
  - **language, screen size, depth, bitrate, systemRequired**
  - **…et des préférences de l’utilisateur**

```
...
<par>
   <text .../>
   <switch>
      <par bitrate="40000">
       ...
      </par>
      <par bitrate="24000">
       ...
      </par>
       ........
   </switch>
</par>
...
```

```
...
<switch>
       <audio src="joe-audio-better-quality" language="fr"/>
       <audio src="joe-audio" language="en"/>
</switch>
...
```

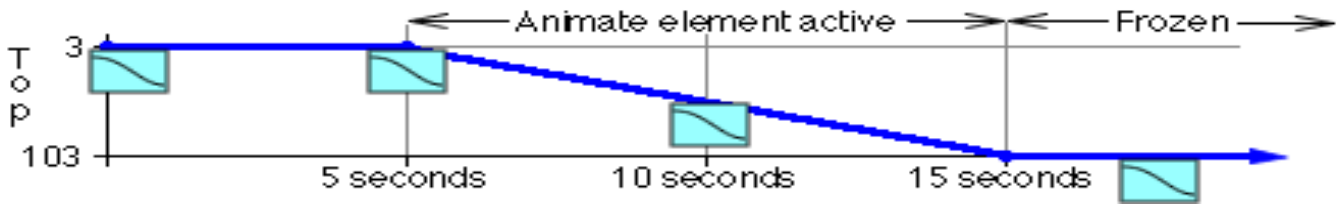
Animations

- Définition :
  - **Un ensemble d'attributs cible de l'animation**
  - **Une fonction (mode de calcul) qui fait évoluer ces attributs**
  - **Un contrôle sur les instants d'application des changements**

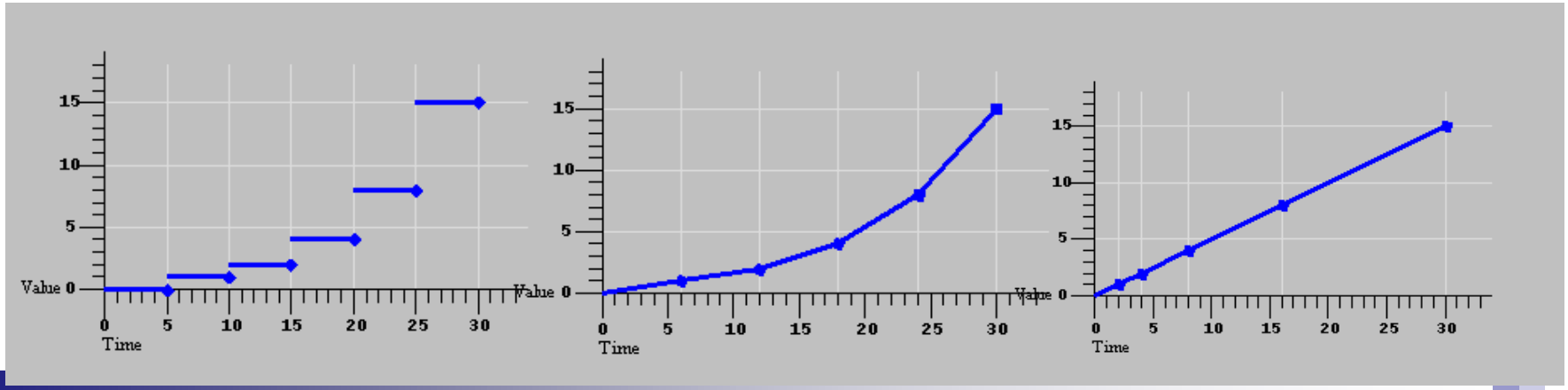
- Syntaxique
  - **animateMotion : mouvements graphique d'éléments**
  - **animate : animation générique appliquée à un attribut spécifique d'un élément from/to/by/calcMode**
  - **set : changement discret d'un attribut à un instant donné**
  - **animateColor : animation dans le domaine des couleurs**

Animations

```
<img top="3" ...>
   <animate begin= "5s" dur="10s" attributeName="top"
     by="100" repeatCount="2.5" fill="freeze"
    calcMode="linear"/>
</img>
```



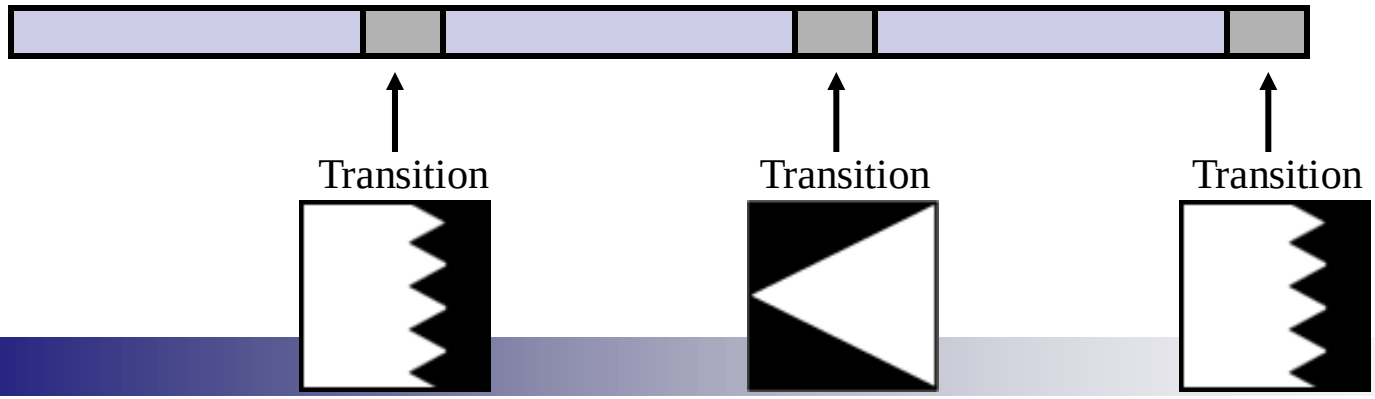
Modes de calcul : discret, liste avec interpolation linéaire, régulier



Transitions

- Elément : transition
  - **Type et Subtype (catégories de transition + variante)**
  - **Rappels par les attributs transIn et transOut**
  - example

```
...
<transition id="wipe1" type="zigZagWipWipe" subtype="leftToRight" dur="1s"/>
<transition id="wipe2" type="veeWipe" subtype="leftToRight" dur="1s"/>
...
<seq>
  <img src="butterfly.jpg" dur="5s"... />
  <img src="eagle.jpg" dur="5s" fill="transition" transIn="wipe1" ... />
  <img src="wolf.jpg" dur="5s" fill="transition" transIn="wipe2"
   transOut="wipe1" ... />
</seq>
```



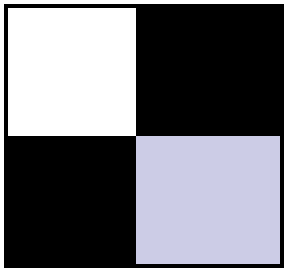
Les liens hypermédia temporisés

- Compatible avec (Xlink/Xpointer)
- Extension de la sémantique des URLs
  - **http://foo.com/path.smil#ancre(begin(id(ancre))**
  - **deux types (a: totalité, area: partie)**
  - **sauts dans l'espace et dans le temps**
- Attribut show
  - **Replace (valeur par défaut)**
  - **New (fork)**
  - **Pause (Appel de procédure)**

Liens avec du spatial et du temporel

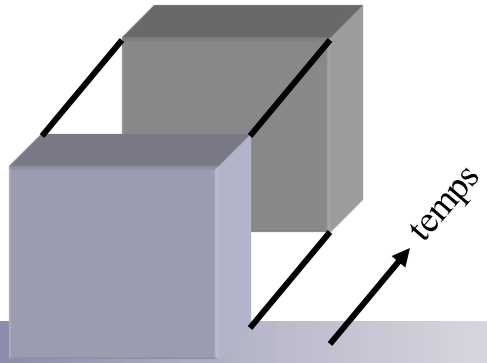
- Ancre accrochée à une partie de la surface d'un objet

```
<video src="rtsp://www.w3.org/video.mpg">
    <area href="http://www.w3.org/AudioVideo" coords="0%, 0%, 50%, 50%"/>
    <area href="http://www.w3.org/Style" coords="50%, 50%, 100 %, 100%"/>
</video>
```



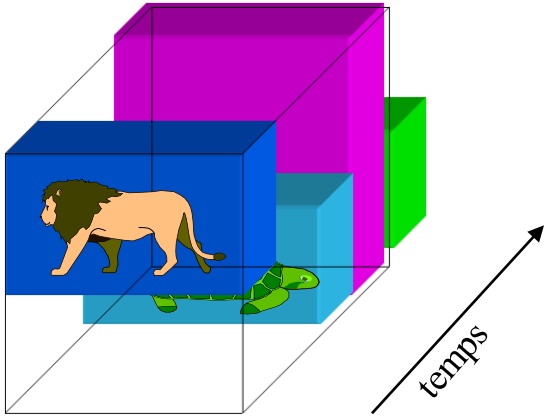
- Ancre accrochée à une sous-durée de l'intervalle d 'un objet

```
<video src="rtsp://www.w3.org/video.mpg">
    <a href="http://www.w3.org/AudioVideo" begin="0 s" end="5 s" />
    <a href="http://www.w3.org/Style" begin="10 s" end="15 s"/>
</video>
```



Combinaison des deux …

```
<video src="rtsp://www.w3.org/video.mpg">
    <area href="http://www.w3.org/Lion"
                    begin="0 s" end="5 s" coords="0%, 0%, 100%, 50%"/>
    <area href="http://www.w3.org/Tortue"
                     begin="10 s" end="15 s" coords="0%, 50%, 100 %, 100%"/>
</video>
```



Et avec l'animation de coords