SVG
Scalable Vector Graphics

Pierre-Alain Reynier
http://www.lif.univ-mrs.fr/~preynier/XML/

Plan du cours

1 - Introduction

2 - Eléments graphiques de base

3 - Structuration

4 - Transformations

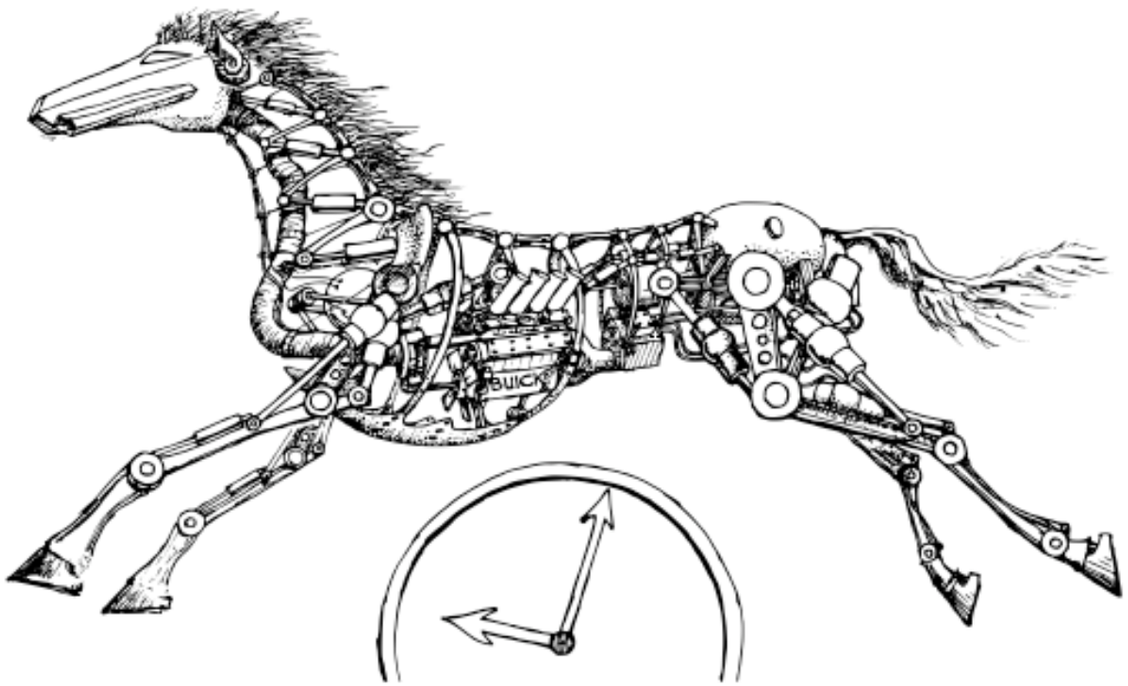
5 - Conclusion

1 - Introduction

SVG

- SVG est un standard W3C du 14/01/2003 : http://www.w3.org/TR/SVG/
- SVG est un langage de description de graphiques 2D en XML
- Trois types d'objets graphiques :
  - formes vectorielles (traits, courbes...)
  - images
  - texte
- Objets dynamiques, interactifs

SVG : Quelques exemples



SVG : Principales applications

SVG est concurrencé par Flash, bien implanté sur Internet, et donc plus destiné à

- visualisation de contenus (économiques, processus, cartes...) au format XML
- association à Javascript + DOM ou XSLT
- interface utilisateur pour certaines applications internet (Firefox)
- interfaces pour des outils de consultation avec un faible bande passante
- dessins dans le monde éducatif

Pourquoi SVG ?

Avantages liés au vectoriel :

- adaptation de l'affichage à des media variés et à des tailles différentes
- possibilité d'appliquer des styles
- possibilité d'indexer le texte du graphisme
- taille de l'image après compression
- facilité d'édition

Pourquoi SVG ?

Avantages liés au format SVG :

- insertion dans le monde XML :
  - génération via XSLT à partir de XML
  - intégration dans XHTML, viewers SMIL
  - utilisation de CSS
  - scriptable avec Javascript via DOM
- modèles de couleur complexes, filtres graphiques
- indexage par les moteurs de recherche
- partage de code (format libre)

Structure d'un document

```
<?xml version="1.0" standalone="no"?>
```

déclaration XML standard

```
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
"http://www.w3.org/TR/2001/PR-SVG-20010719/DTD/svg10.dtd">
```

indication de la DTD pour un document non standalone

```
<svg width="400" height="250"
xmlns="http://www.w3.org/2000/svg">
```

déclaration du namespace et des dimensions du document SVG.
On est ici à la racine du contenu SVG

```
...
</svg>
```

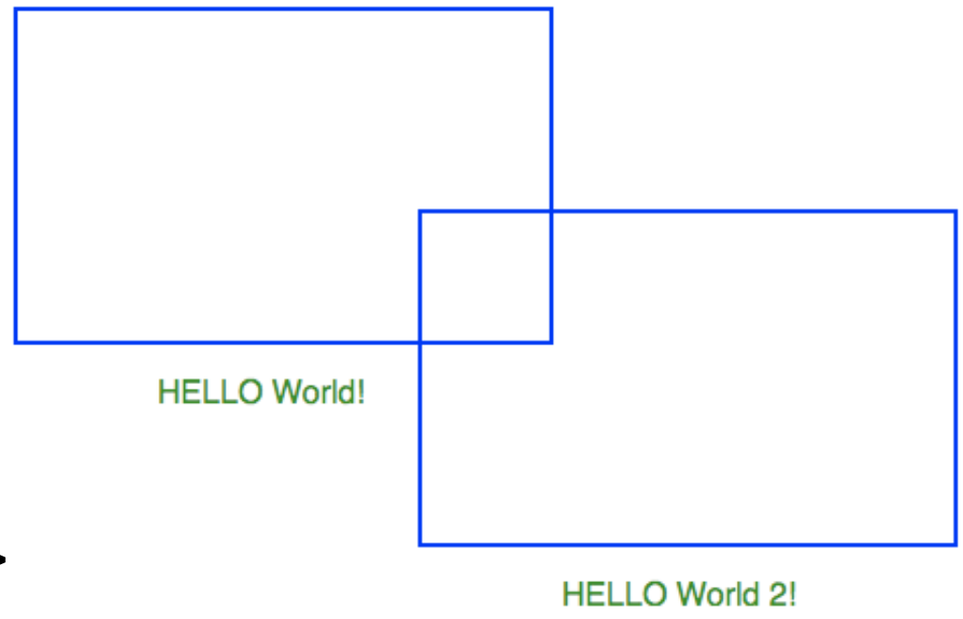
Structure d'un document

- Un document SVG se compose de un ou plusieurs éléments <svg>
- <svg> est la racine d'un graphisme SVG
  - on peut imbriquer les éléments <svg>
  - chaque <svg> crée un nouveau système de coordonnées
- Attributs de l'élément <svg> :
  - x,y : coordonnées
  - width, height : dimensions

Structure d'un document

Premier exemple :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20001102//EN"
"http://www.w3.org/TR/2000/CR-SVG-
20001102/DTD/svg-20001102.dtd">
<svg xmlns="http://www.w3.org/2000/svg">
    <rect x="5" y="5" width="265" height="165"
    style="fill:none;stroke:blue; stroke-width:2" />
    <text x="75" y="200" style="font-size:18;
    font-family:Helvetica; fill:green">
    HELLO World! </text>
    <svg width="300" height="200" x="200" y="100">
        <rect x="5" y="5" width="265" height="165"
        style="fill:none;stroke:blue;stroke-width:2" />
        <text x="75" y="200" style="font-size:18;
        font-family:Helvetica; fill:green">
            HELLO World 2!</text>
    </svg>
</svg>
```



A propos des références

- Utilisation de xlink :

```
<svg xmlns:xlink="http://www.w3.org/1999/xlink"> ... </svg>
```

- Attribut id : permet de nommer un élément
- Appel à l'aide de href et #nom_variable :

```
xlink:href="#nom_variable"
```

2 - Eléments graphiques de base

- Introduction
- Mécanismes généraux
- Rendu graphique
- Formes géométriques simples
- Formes arbitraires
- Textes

Eléments graphiques : introduction

- Textes : text
- Formes géométriques de base :
  rect, circle, ellipse, line, polyline
- Formes arbitraires : path

Chaque élément graphique est représenté par un élément XML paramétrable avec des attributs

Mécanismes généraux

- Attributs : identifiant (id), position (x, y), style (proche de CSS2)
- Positionnement : système de coordonnées d'origine en haut à gauche
- Transformations : possibilité de translation, rotation, redimensionnement....

Rendu graphique

- Attributs principaux : (cf CSS2)
  - stroke : forme du bord de l'objet
  - fill : objet rempli ou non
  - color : couleur...
- 2 possibilités pour donner ces attributs :
  - 1 attribut global style à la CSS2
  - 1 attribut pour chaque style

Exemples :

```
<rect x="200" y="100" width="60" height="30"
style="fill:red;stroke:blue;stroke-width:3" />

<rect x="200" y="100" width="60" height="30" fill="red"
stroke="blue" stroke-width="3" />
```

Rendu graphique : fill

- Couleur : fill=”red”
  nom, URI... de la couleur
- Opacité : fill-opacity=”0.2”
  valeur comprise entre 0 et 1

Rendu graphique : stroke

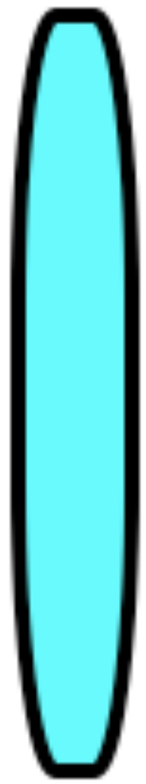
- Couleur : stroke=”red”
  nom, URI... de la couleur
- Opacité : stroke-opacity=”0.2”
  valeur comprise entre 0 et 1
- Epaisseur du trait : stroke-width
- Jonction de ligne : stroke-linejoin=
  miter (anguleux), bevel (biais), round (arrondi)
- Pointillés : stroke-dasharray=”10 5 10”

Figures géométriques : rectangle

- Elément rect
- Attributs x et y : position
- Attributs rx et ry : axes de l'ellipse utilisée pour arrondir

Exemple :

```
<rect width="30" height="200"
    x="100" y="150" rx="10" ry="75"
    stroke-width="4px" stroke="black"
    fill="cyan"/>
```



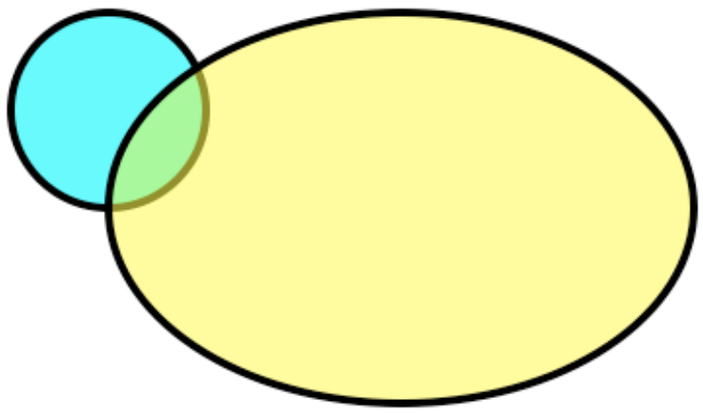
Figures géo. : cercles et ellipses

- Eléments circle et ellipse
- Attributs cx et cy : position du centre
- Forme :

  Cercle : attribut r pour le rayon

  Ellipse : attributs rx et ry pour les demi-axes

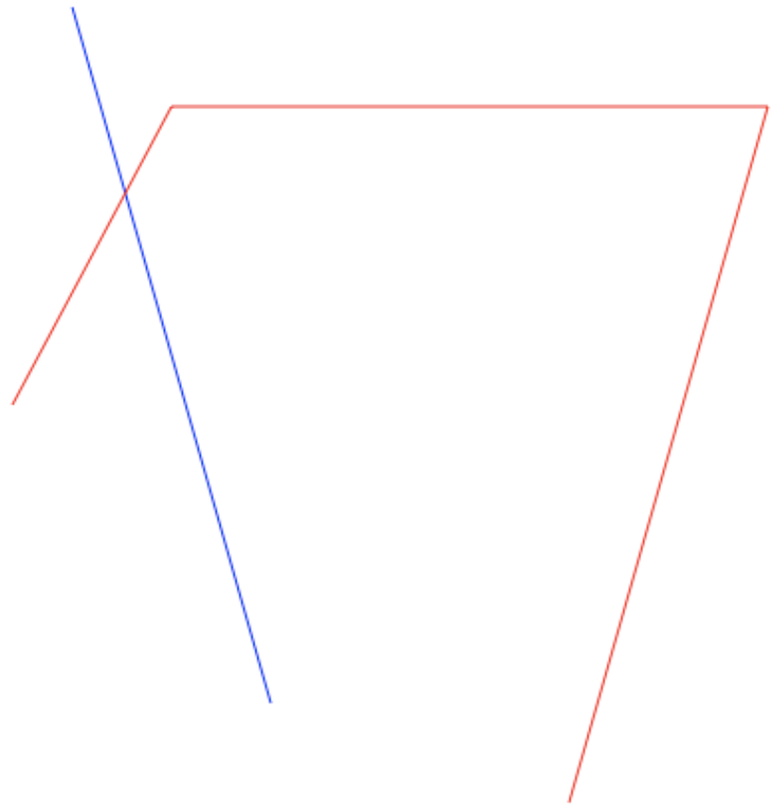
```
<g stroke-width="4px" stroke="black">
  <circle r="50" cx="100" cy="100"
      fill="cyan" fill-opacity="1"/>
  <ellipse rx="150" ry="100" cx="250"
      cy="150" fill="yellow"
      fill-opacity="0.5"/>
</g>
```



Figures géo. : lignes (brisées)

- Eléments line et polyline
- Attributs de line :
  x1, y1, x2, y2 : position des 2 extrémités
- Attribut de polyline :
  points : suite de coordonnées

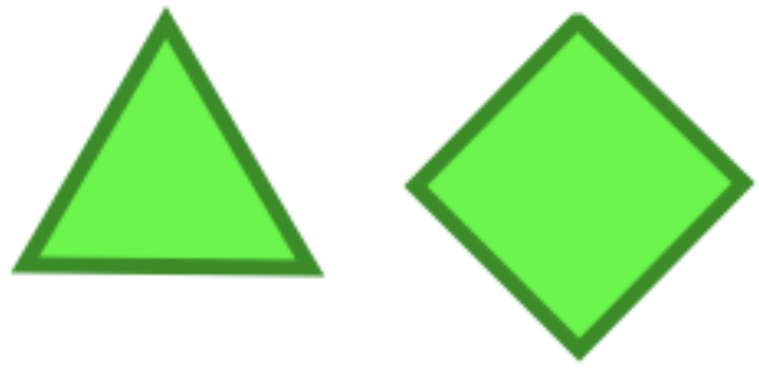
```
<line x1="50" y1="50" x2="150"
      y2="400" stroke="blue"/>
<polyline points="20,250, 100,100,
   400,100, 300,450"
   fill="none" stroke="red"
   stroke-linecap="round"/>
```



Figures géo. : polygones

- Elément polygon
- courbe fermée / courbe ouverte (polyline)
- Attribut de polygone :
  points : suite de coordonnées

```
<g style="stroke:green; fill:lime;
  stroke-width:5" >
  <polygon points="99,50,143,125,
    56,124,99,50" />
  <polygon  points="225,50,275,99,
    225,150,175,100,224,50" />
</g>
```



Formes arbitraires

- Elément path
- Ces formes peuvent servir de support à d'autres éléments comme du texte (cf plus loin)

- Attributs de base :
  - d : définit les path data = liste de commandes permettant de tracer le chemin
  - nominalLength : facultatif, longueur totale du chemin

Formes arbitraires : path data

- Règles générales concernant les commandes :
  - abréviations en une lettre
  - les répétitions peuvent être omises
  - coordonnées absolues (majuscule) ou relatives (minuscule)
  - les espaces peuvent être supprimés

Formes arbitraires : path data

- moveto x y (m ou M) : nouveau ss-chemin en (x,y)
- closepath (z ou Z) : ferme un ss-chemin entre le point courant et le dernier moveto
- lineto x y (l ou L) : ligne droite entre le point courant et le point (x,y)
- horizontal lineto x (h ou H)
- vertical lineto y (v ou V)

Formes arbitraires : courbes

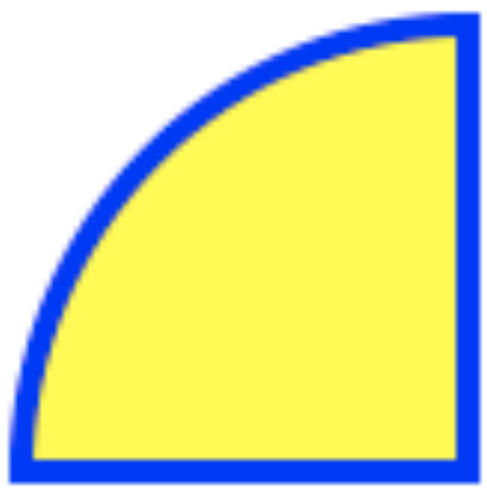
Cde	Argument	Description
A, a	rx ry x-axis-rotation large-arc sweep x y	Arc elliptique entre le point courant et le point (x,y) de rayons (rx ry), de rotation x-axis- rotation, large-arc (0 ou 1) si inf. ou sup. à 180 deg. et sweep (1 ou 0) si dans le sens positif ou non
Q, q	x1 y1 x y	Courbe de Bézier quadratique du point courant à (x y) avec le point de contrôle (x1 y1)
T, t	x y	Courbe de Bézier quadratique du point courant à (x y) avec comme point de contrôle le point courant ou celui construit par réflexion du précédent point de contrôle (Q,q)
C, c	x1 y1 x2 y2 x y	Courbe de Bézier cubique du point courant à (x y) avec les points de contrôle (x1 y1) pour le début de la courbe et (x2 y2) pour la fin
S, s	x2 y2 x y	Courbe de Bézier cubique du point courant à (x y) avec les points de contrôle (x2 y2) pour la fin, et celui construit par réflexion à partir de celui de la commande C, c précédente pour le début

Formes arbitraires : exemple 1

```
<g stroke-width="4px" stroke="blue">
  <!-- On pose le crayon (M 50 100), ensuite on tire
un trait vers le coin du bas (L 100 150) et vers le
coin en haut à droite (150 50), finalement on ferme
(z) -->
  <path d="M 50 50 L 100 150 150 50 z" fill="cyan"/>
</g>
```

Formes arbitraires : exemple 2

```
<g stroke-width="4px" stroke="blue">
  <!-- Exemple de courbe:
    On se positionne à 180, 180 (M 180, 180);
    On dessine une ligne verticale vers y=-75 (v=-75). Cela
devient le point de départ pour l'arc.
    On dessine un arc (a) avec rayon x=75 et y=75 (75,75), sans
rotation (0). Le deuxième 0 indique que l'arc se trouve du côté
"petit", le troisième 0 indique une direction de dessin
négative. Les -75, 75 indiquent l'arrivée de l'arc.
    On ferme le tout.
  -->
  <path d="M 180,180 v-75 a75,75 0 0,0 -75,75 z"
     fill="yellow"/>
</g>
```



Textes

- Elément text :
  - Attributs x et y de position
  - mise en forme réalisée à la CSS2
- Elément d'ajustement tspan :
  - Attributs x et y
  - Attributs dx et dy : décalage
  - Attribut style pour la police, la couleur...
- Possibilité d'inclure du texte référencé

Textes : un exemple

```
<g style="font-family:Verdana; font-size:12pt">
   <text x="1cm" y="1.5cm" style="fill:blue">
     Voici un
       <tspan style="font-weight:bold; fill:red">texte</tspan>
     écrit droit.
   </text>
   <text x="1cm" y="3cm" style="fill:blue">
     Et en voici
       <tspan dx="2em" dy="-.5cm" style="font-weight:bold; fill:red">
          un autre
       </tspan>
       <tspan dy="1cm">
          un peu décalé...
       </tspan>
    </text>
</g>
```

Voici un **texte** écrit droit.

un autre

Et en voici

un peu décalé...

Textes : le long des chemins

- On impose au texte de suivre un chemin prédéfini avec la balise textPath
- On référence ce chemin avec :
  <textPath xlink:href=”uri” />

Textes : un exple avec un chemin

```
<path id="MonChemin"
          d="M 50 100
             C 100 50 150   0 200 50
             C 250 100 300 150 350 100
             C 400 200 450 50 450 50" fill="none" stroke="none"/>

<text style="font-family:Verdana; font-size:20; fill:blue">
  <textPath xlink:href="#MonChemin">
    On monte, puis on descend, et hop une petite bosse!
  </textPath>
</text>
```

3 - Structuration

Structuration

- SVG permet de regrouper des objets dans des blocs, de les nommer, et de les réutiliser.
- SVG possède plusieurs constructions :
  - le fragment d'un document SVG : svg
  - un groupe d'éléments : g
  - définition d'objets : def
  - utilisation d'objets : use
  - insertion d'images : image
- Les objets héritent le style de leur parents

Structuration : svg et g

- L'élément <svg> est la racine d'un document SVG. Chaque svg introduit un nouveau système de coordonnées
- L'élément <g> sert à regrouper des éléments graphiques.

```
<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>
```

Structuration : defs

- L'élément <defs> autorise la définition d'objets référencés plus tard
- Au plus un <defs> par fichier .svg
- Les objets définis au sein de defs ne sont pas dessinés.

Structuration : use

- L'élément <use> permet de réutiliser les éléments suivants :
  svg, g, les éléments graphiques et use
- L'élément utilisé est appelé via href
- On peut définir une position (x y), modifier la taille (width height), le style (stroke, opacity...)
- Le comportement de use est (un peu) différent selon le type d"élément réutilisé.

Structuration : un exemple

```
<defs>
   <g id="bleublancrouge">
      <rect x="0" fill="blue" width="10" height="10"/>
      <rect x="10" fill="white" width="10" height="10"/>
      <rect x="20" fill="red" width="10" height="10"/>
      <rect x="0" fill="none" width="30" height="10" stroke="black"/>
   </g>
</defs>
<use x="10" y="5" xlink:href="#bleublancrouge" />
<use x="20" y="20" xlink:href="#bleublancrouge" opacity="0.3" />
```

Structuration : images

- Insertion d'images via l'élément <image>
- Formats supportés : jpg et png
- Possibilité aussi d'inclure un fichier svg
- Attributs habituels :
  - x, y width, height
  - href pour faire référence au fichier inclus
- Attribut spécial preserveAspectRatio (optionnel) permettant d'éviter une déformation de l'image

4 - Transformations

Transformations

- Introduites via l'attribut `transform`
- Valable pour les éléments de groupement (comme <g>) et pour tous les éléments graphiques (figures, texte...)
- Permet entre autres :
  - translations
  - rotations
  - redimensionnement
  - transformations via une matrice

Translations : <translate>

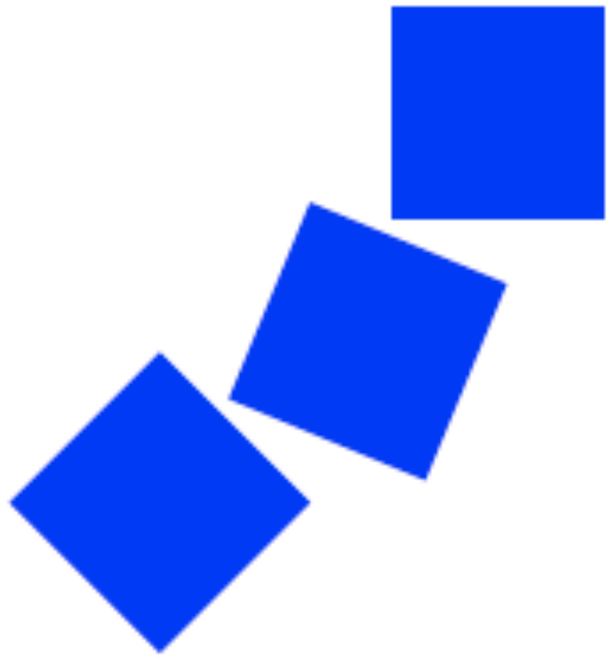
- Syntaxe : <g transform=”translate(50,20)”>
- Permet d’opérer une translation du groupe de 50 pixels vers la droite, et 20 vers le bas.
- Exemple :

```
<defs>
  <g id="carre">
    <rect x="10" y=”10” fill="blue"
       width="20" height="20"/>
  </g>
</defs>
<use xlink:href="#carre"/>
<use xlink:href="#carre" transform="translate(40,0)"/>
<use xlink:href="#carre" transform="translate(20,20)"/>
<use xlink:href="#carre" transform="translate(40,40)"/>
<use xlink:href="#carre" transform="translate(0,40)"/>
```

Rotations : <rotate>

- Syntaxe : rotate(angle [cx,cy])
- L'angle est indiqué en degrés
- Par défaut le centre de la rotation est l'origine du système de coordonnées

```
<defs>
  <g id="carre">
    <rect x="10" y=”10” fill="blue"
        width="20" height="20"/>
  </g>
</defs>
<use id=”carretranslate” xlink:href="#carre"
   transform="translate(40,0)"/>
<use xlink:href="#carretranslate"
   transform=”rotate(22,5)”/>
<use xlink:href="#carretranslate"
   transform="rotate(45)"/>
```



Chang. d'échelle : <scale>

- Syntaxe : scale(sx [,sy])
- sx représente le facteur pour l'échelle x
- Si sy n'est pas donné, par défaut sx=sy

```
<use xlink:href="#carre"
   transform="scale(0.5)"/>
<use xlink:href="#carre"
   transform="translate(5,5)"/>
<use xlink:href="#carre"
   transform="translate(15,15) scale(2)"/>
<use xlink:href="#carre"
   transform="translate(35,35) scale(4)"/>
```

Enchaînement des opérations

- On peut enchaîner les transformations en insérant des espaces
- Attention le système de coordonnées est modifié lui aussi !!

5 - Conclusion

Aller plus loin

- Seul un fragment de SVG a été présenté. Plus d'informations dans la recommandation W3C :

  http://www.w3.org/TR/SVG

- Il existe une version 2.0 en cours de développement....
- Extensions pas vues ici :
  - Animation (couleurs, déplacements....)
  - Interactivité (déclenchement au passage de la souris...)

Outils pour SVG

- Edition :
  - c'est un document XML, donc tout éditeur suffit !
  - Adobe Illustrator et Corel Draw permettent aussi d'exporter au format SVG
- Visualisation :
  - IE: plugin Adobe à installer
  - Firefox : natif
- Attention : toute la spéc. n'est pas reconnue...