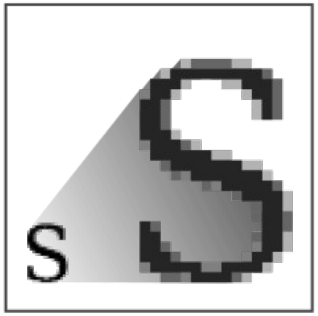
Les formats vectoriels

Exemple : le format SVG

SVG : Scalable Vector Graphics

SVG ?

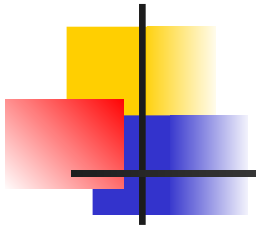


Matriciel .jpeg .gif .png

Vectoriel .svg

- Langage de **description** d'images 2D = format vectoriel,
  - contient aussi :
    - du contenu **dynamique** / des **animations**
    - du contenu **interactif** (zones cliquables pouvant être zoomées ou enrichies d'une explication, …)
    - se déclenchent de façon **déclarative** (éléments incorporés dans le contenu SVG) ou par un **script** (JavaScript)
  - basé sur du XML,
  - fait pour le Web (construit par le W3C),
  - équivalent public du format flash.
- Recommandations du W3C :
  - 2011 (SVG 1.1 Second Edition) : https://www.w3.org/TR/SVG11/
  - 2018 (SVG 2) : https://www.w3.org/TR/2018/CR-SVG2-20181004/

Utilisations les plus intéressantes (actuellement)

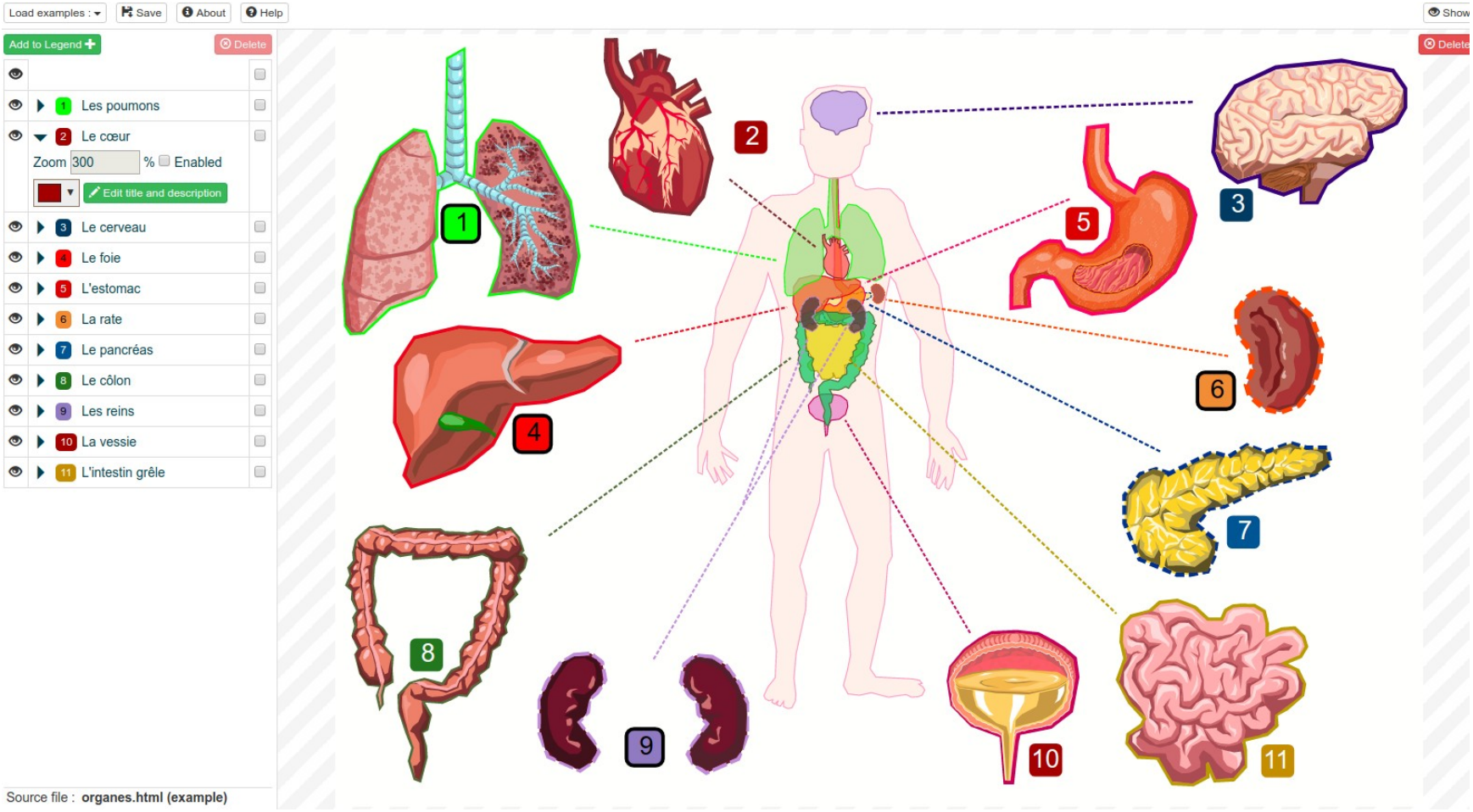


- visualisation de contenus
  - économiques, processus, cartes, etc.
- interface utilisateurs (IHM) pour certains types d'applications Internet
- dessins statiques, animés ou même interactifs dans le monde de l'éducation
- animations multimédia de contenus formalisables
  - chimie, maths, etc.

Exemple de visualisation de contenu = carte

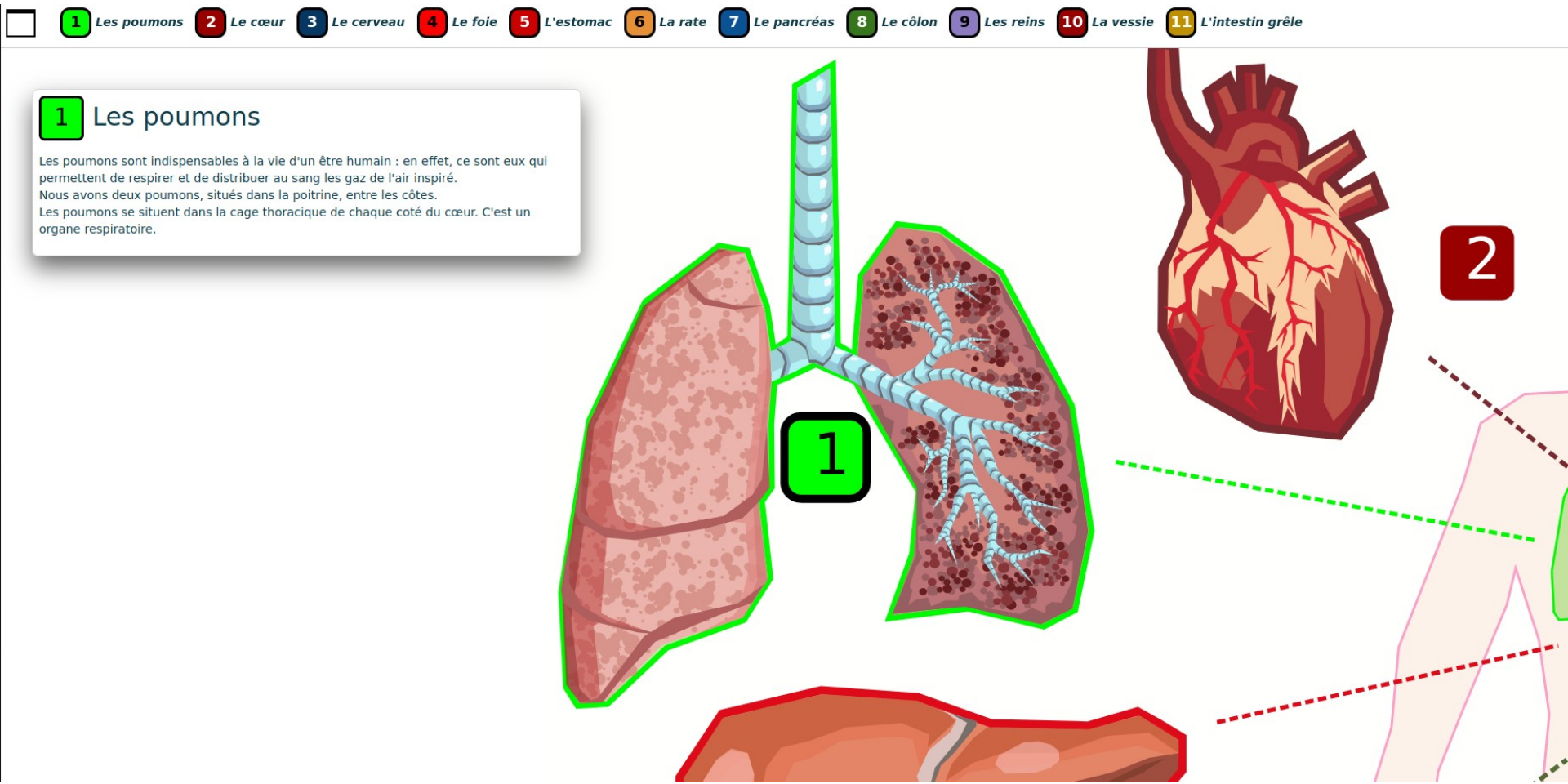
https://edutechwiki.unige.ch/fr/Fichier:Countries_of_Europe.svg

Exemple de contenu interactif

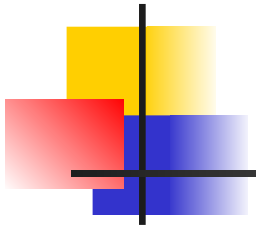


https://mothsart.github.io/labo/frontend/edit_interactive_svg/

Exemple de contenu interactif (2)



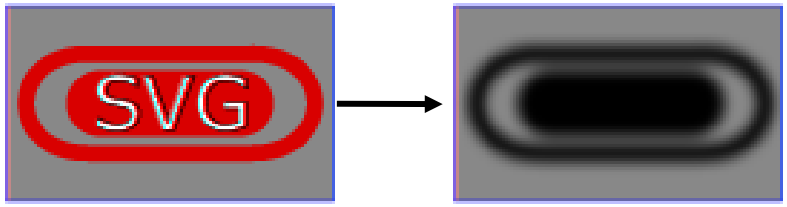
Avantages des graphiques vectoriels

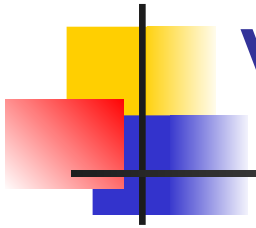


- rendu correct dans de multiples média et à différentes tailles (adaptation),
- possibilité d’appliquer des styles (CSS, XSLT),
- possibilité d’indexer le texte qui fait partie du graphisme,
- taille de l’image (après compression),
- facilités d’édition : les éléments sont des objets, hiérarchies, etc.

Avantages particuliers de SVG

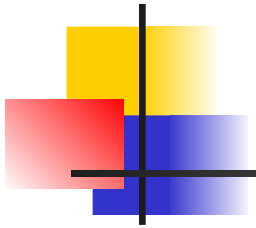
- Insertion dans le monde XML/XHTML :
  - génération de SVG avec XSLT/PHP... à partir de données XML
  - intégration dans XHTML, ...
  - utilisation de CSS (feuilles de style),
  - scriptable avec JavaScript via le DOM *(Document Object Model)*.
- Modèle de couleurs sophistiqué, **filtres** comme dans Photoshop (ex : flou),
- spécification claire,
- en **HTML5** → balise <svg>





Visualisation

- La plupart des navigateurs interprètent une partie de la norme SVG
  - MS Internet explorer
  - Opera
  - Chrome
  - Firefox 9.1
  - Safari
  - ...



Rappel : Principe des formats XML

- XML = eXtensible Markup Language
- Balises :
  - Ex : <html> </html> ou
- Attributs :
  - Ex : <p **align="center"**>ce texte est centr&eagut;</p>
- DTD = Document Type Definition

Structure d'une simple page SVG

- Une déclaration XML standard, par exemple :

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

- Pour un document non "standalone", il faut indiquer la **DTD** :

  ```
  <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
  ```

- Racine d'un contenu SVG :

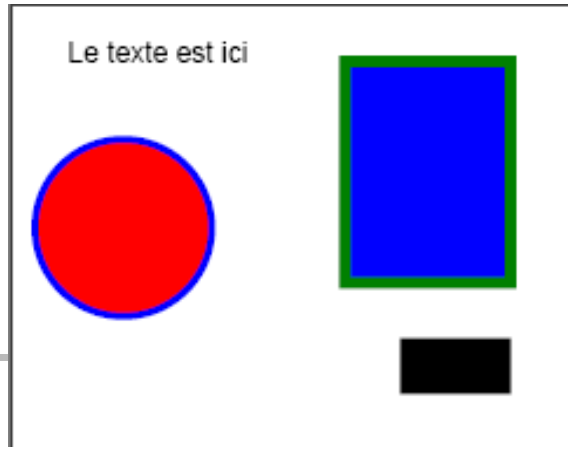
  ```
  <svg>
  ...
  </svg>
  ```

- Déclarer des **name spaces** à la racine (si ce n'était pas déjà fait dans un parent) :

```
<svg xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink" >
  ...
</svg>
```

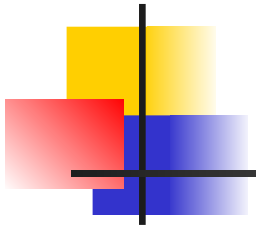
← pour que les liens internes ou avec d'autres fichiers svg fonctionnent

Exemple simple




```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
      "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg  width="5cm"
       height="4cm"  xmlns="http://www.w3.org/2000/svg">
<text x="0.5cm" y="0.5cm" width="2cm" height="1cm">Le texte est ici</text>
<circle cx="1cm" cy="2cm" r=".8cm" fill="red " stroke="blue" stroke-width="3"/>
<rect x="3cm" y="0.5cm" width="1.5cm" height="2cm" fill="blue" stroke="green"
       stroke-width="5"/>
<rect x="3.5cm" y="3cm" width="1cm" height="0.5cm" />
</svg>
```

Types de données de base pour les valeurs d'attributs



- **Angle** = entier suivi de *deg* (degrés), *grad* (grades), *rad* (radians)
- **Couleur**, spécification d'une couleur RGB comme en HTML :
  - *#rrggbb*
  - un mot-clé dans la liste : *aqua, black, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, yellow*
- **Entier** : entre - 2147483648 et 2147483647
- **Réel** :
  - soit en notation décimale,
  - soit en notation scientifique (x.yyy e (ou E)± nn) entre - 3.4e+38 et 3.4e+38
- **Longueur** = nb suivi d'un identifiant CSS : *mm, cm, m ou px (pour pixel)*
- **Temps** = nb suivi de *ms* (millisecondes) ou de *s* (secondes)

Structure d'un document SVG

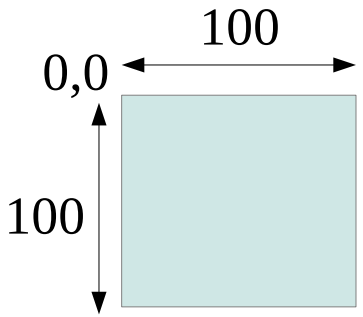
- Il se compose d'un ou plusieurs fragments délimités par la balise <svg> qui définit l'espace utilisateur
- Attributs de la balise <svg> :
  - x = "x0" ; *position en x du coin supérieur gauche*
  - y = "y0" ; *position en y du coin supérieur gauche*
  - width = "w0" ; *largeur*
  - height = "h0" ; hauteur
- **par défaut les valeurs sont en pixel (px)**
- le repère initial lié au canevas est en (0,0)
  - les x de gauche à droite
  - les y de haut en bas

Exemple CoordonnéesInitiales

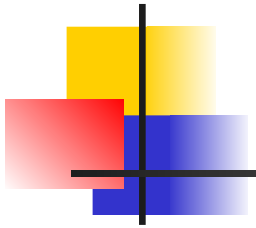
Système de coordonnées, transformations et unités

- Les objets graphiques sont construits dans un espace appelé **canevas (infini)**,
- mais seulement une région rectangulaire de celui-ci sera affichée = **viewport**
- La viewport est définie par défaut à partir des attributs *width* et *height* de la balise svg :

<**svg** whidth="100" height="100">



Système de coordonnées, transformations et unités

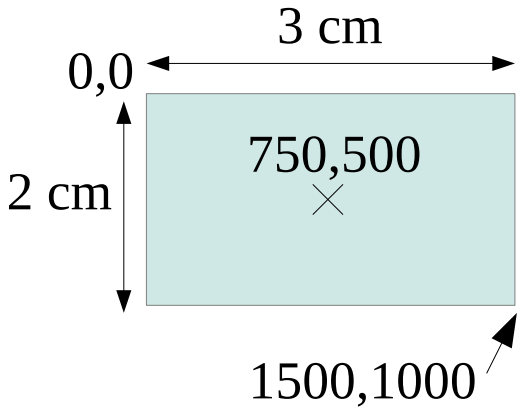


- Possibilité de préciser l'**unité de longueur** utilisée pour tous les attributs de position et longueur (par défaut elle correspond à l'unité de l'écran = pixel : px)
- Systèmes relatifs :
  - ***em*** resp. ***ex*** relatifs à la taille de la police resp. la hauteur de la police,
  - ***%*** = pourcentage de la taille de la « viewport ».
- Systèmes absolus (sauf si redéfinition du système de coordonnées) : mm, cm, in
- L'attribut « **transform** » redéfinit le système de coordonnées en transformant le repère initial

Attribut svg « viewBox »

- Redéfinit le système de coordonnées
- Exemple :

```
<svg width="3cm" height="2cm"
     viewBox="0 0 1500 1000"
     preserveAspectRatio="none" >
```



- Paramètres de viewBox :

" <pos-x>, <pos-y>, <largeur> et <hauteur> "

le point (750 ; 500) se trouve à (1.5cm ; 1cm)

Unité utilisateur et unité de l’écran : exemples

***** Par défaut, l'unité utilisateur correspond au pixel :

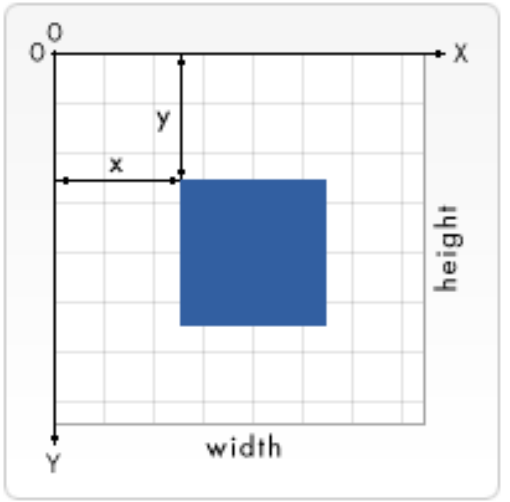
<svg width="100" height="100"> : élément SVG de 100px par 100px

*** <svg width="200" height="200" viewBox="0 0 100 100">**

ici l'image SVG fait 200px par 200px, toutefois, l'**attribut viewBox** définit que cet élément de 200 par 200 :

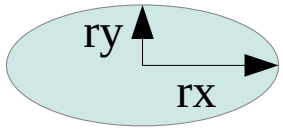
- commence au point (0,0),
- et s'étend sur une **grille de 100 unités sur 100 unités** vers la droite et vers le bas de l'écran,
- **100 unités** représentant 200 pixels, chaque unité vaut deux pixels : cela permet de doubler la taille de l'image.

Formes de base

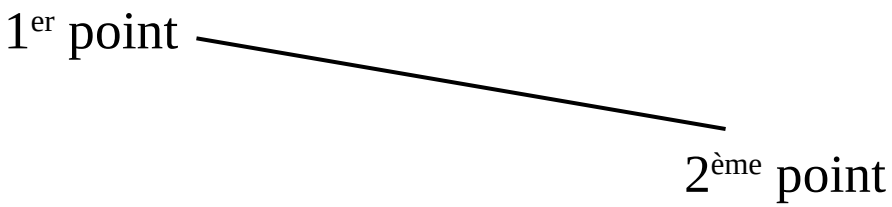


- Le rectangle
  - <**rect** x="100" y="100" width="400" height="200"… />
  - width et height : en nb de px (pixels)
  - autres attributs rx="5" ry= "10" (arrondi des coins)
- Le cercle : <**circle** cx="600" cy="200" r="100" … />
  - cx, cy = centre du cercle
  - r = rayon du cercle
- L'ellipse : <ellipse cx= "100" cy="100" rx="250" ry="100" … />

  rx, ry= rayon de l'ellipse en x et y



Formes de base (2)



- La line
  - <line x1="100" y1="300" x2="300" y2="100" … />
  - x1, y1, x2, y2 = coord. des 1er et 2ème points (extrémités)
- La polyligne
  - <polyline points="50,375 150,375 150,325 250,325" … />
  - points = liste des coordonnées (x,y) des points composant cette forme
- Le polygone = polyligne fermée
  - <polygon points="850,75 958,137.5 958,262.5 850,325 742,262.6 742,137.5" … />

Le text

- <text> et ses principaux attributs :

<**text** x="50" y="150" font-family="Verdana" font-size="55"
fill="blue">

Bonjour

<**tspan** fill="red" font-size="100"> **vous** </tspan>

</text>

- Attributs :
  - x et y : position du 1er caractère
  - font-family : police de caractère
  - font-size : taille (hauteur) de la police
  - fill : couleur de la police

Les tracés : balise path

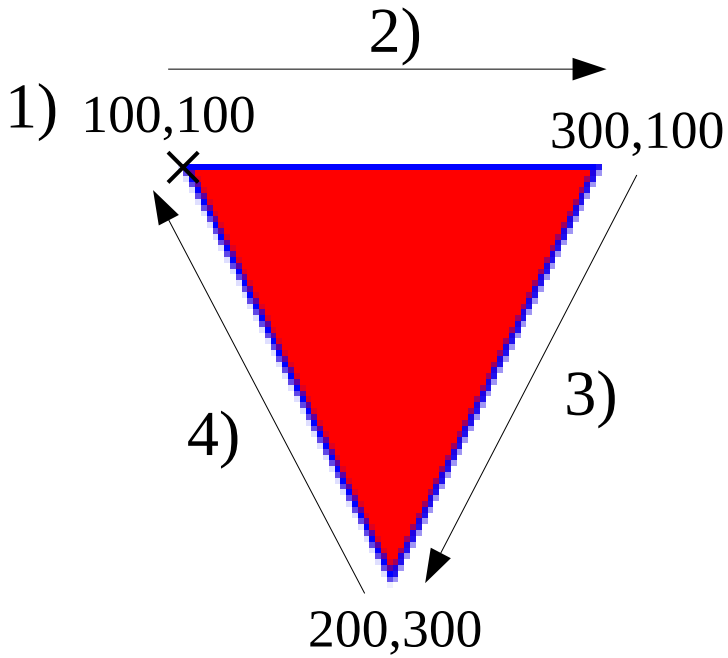
<**path** d="M 100 100 L 300 100 L 200 300 z" .../>

- **attribut d** : définit le contour d'une forme (chemin du tracé)
- défini par une succession de lettres et des valeurs x et y de déplacements (suivant le principe de la tortue)
  - Moveto
    - M => en coordonnées **absolues**
    - m => en coordonnées **relatives** (sauf si au début)
  - Lineto
    - L => **absolues**
    - l => **relatives**
    - H, h => déplacement horizontaux absolus ou relatifs
    - V, v => déplacement verticaux absolus ou relatifs

Les tracés : exemple

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

1) **Moveto** 100,100
2) **Lineto** 300,100
3) **Lineto** 200,300
4) **z** : pour fermer le tracé



Les tracés

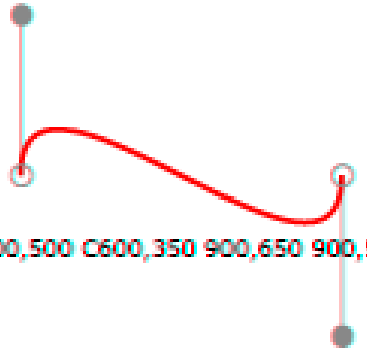
Possibilité de créer :

- des courbes de Bézier
- des enchaînements d'arcs elliptiques

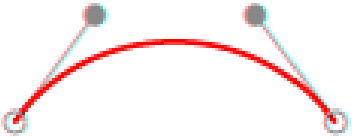
M100,200 C100,100 400,100 400,200

M600,200 C675,100 975,100 900,200

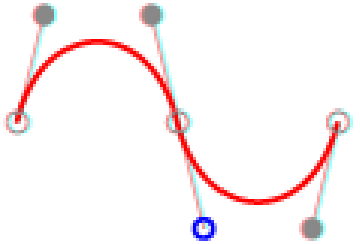
M100,500 C25,400 475,400 400,500



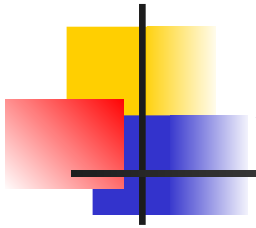
M600,500 C600,350 900,650 900,500



M100,800 C175,700 325,700 400,800



M600,800 C625,700 725,700 750,800
S875,900 900,800



Attributs communs aux formes

- Les principaux :
  - fill = peinture de remplissage
    - valeur = none ou couleur, texture, dégradé ...
  - stroke = couleur de bordure
  - stroke-width = épaisseur de la bordure
  - opacity = valeur du canal alpha
  - visibility = "hidden | visible"

Référencement

- On peut **définir un élément** et lui attribuer un identifiant : <defs>
- On l'**utilise** ensuite avec <use>
- **Attention** il faut préciser l'**espace de nom** dans la **balise svg :**

```
<svg xmlns:xlink="http://www.w3.org/1999/xlink">
   <defs>
      <circle id="anneau" fill="none" stroke-width="2" r="8.5" opacity=".5"/>
   </defs>
   <use x="20" y="20" stroke="blue" xlink:href="#anneau" />
   <use x="40" y="20" stroke="black" xlink:href="#anneau" />
   <use x="60" y="20" stroke="red" xlink:href="#anneau" />
   <use x="30" y="30" stroke="yellow" xlink:href="#anneau" />
   <use x="50" y="30" stroke="green" xlink:href="#anneau" />
   ...
```

Regroupement

- La balise <g> permet de regrouper des éléments
- En affectant un **attribut à <g>**, il sera appliqué à tous les éléments contenus dans cette balise
- Exemple :

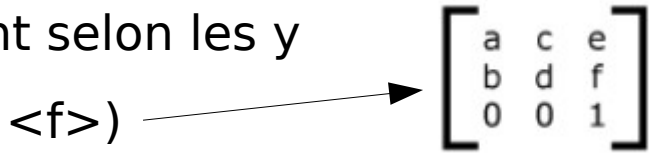
```
<g fill="red" >
   <rect x="1cm" y="1cm" width="1cm" height="1cm" />
   <rect x="3cm" y="1cm" width="1cm" height="1cm" />
</g>
<g fill="blue" >
   <rect x="1cm" y="3cm" width="1cm" height="1cm" />
   <rect x="3cm" y="3cm" width="1cm" height="1cm" />
</g>
```

Balise « transform »

- liste de transformations qui peuvent être :
  - **translate**(<tx> [<ty>]) → translation de tx et ty
  - **scale**(<sx> [<sy>]) changement d'échelle de sx et sy
  - **rotate**(<rotate-angle> [<cx> <cy>]) → rotation de <rotate-angle> (en degrés) autour du point [<cx> <cy>]

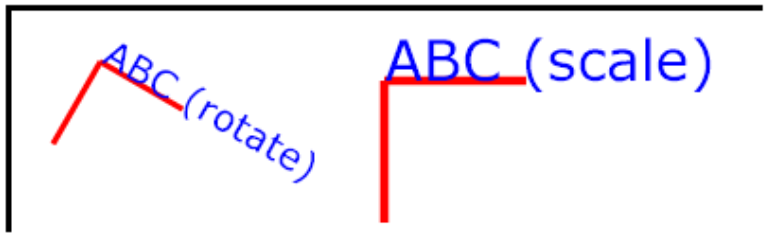
= translate(<cx>, <cy>) rotate(<rotate-angle>) translate(-<cx>, -<cy>)

  - **skewX**(<skew-angle>) = cisaillement selon les x
  - **skewY**(<skew-angle>) = cisaillement selon les y
  - **matrix**(<a>,< b>,< c>, <d>, <e>, <f>)

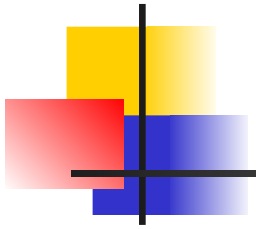


- la première transformation appliquée est la plus **à droite**

```
<!-- Établit un nouveau système de coordonnées, dont l'origine est (50,30)
     dans le système de coordonnées initial et qui est tourné de 30 degrés -->
<g transform="translate(50,30) rotate(30)">
    <g 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" font-size="20" font-family="Verdana" fill="blue" >
      ABC (rotate)
    </text>
</g>
<!-- Établit un nouveau système de coordonnées, dont l'origine est (200,40)
     dans le syst. coord. initial et qui reçoit un facteur d'échelle de 1.5 -->
<g transform="translate(200,40) scale(1.5)">
   <g 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" font-size="20" font-family="Verdana" fill="blue" >
      ABC (scale)
    </text>
</g>
```

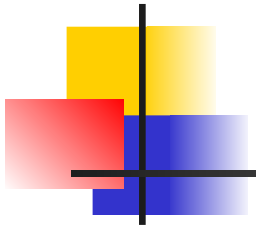


Exemple RotationEchelle



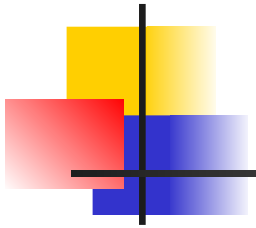
Remplissage, bordure et marqueur

- 3 types de remplissage :
  - une seule couleur,
  - un dégradé (linéaire ou radial),
  - un motif (vecteur ou image, éventuellement en mosaïque).
- On utilise les attributs '**fill**' et '**stroke**‘ qui admettent des paramètres de « peinture » (= couleurs)



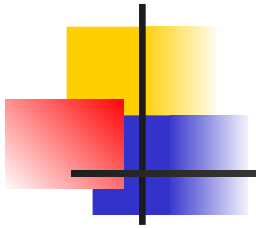
Couleurs

- Toutes les couleurs SVG sont spécifiées dans l'espace de couleur sRGB = Standard RGB
  - #f00 /* #rgb */   16 val/canal
  - #ff0000 /* #rrggbb */   256 val/canal
  - rgb(255,0,0) /* integer range 0 - 255 */
  - rgb(100%, 0%, 0%)   /* float range 0.0% - 100.0% */
- On peut utiliser des mots clés
  - Exemple : red , blue, ...



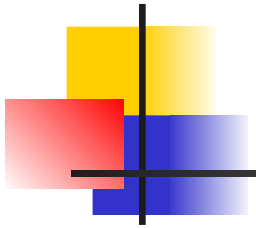
Animations

- Animation de type "XML/SMIL" avec des **balises spéciales** :
  - compatibilité correcte avec les navigateurs,
  - on peut animer pratiquement chaque **attribut**.
- Animation via le DOM de SVG avec un **script** :
  - marche avec Firefox >1.5
  - chaque **attribut** est accessible selon DOM1 & 2.
  - on écrit un petit **programme JavaScript** qui :
    - modifie les attributs d'éléments
    - ou ajoute/enlève des éléments/attributs du DOM.



L’animation SVG

- SVG utilise les éléments d’animation suivants :
  - ‘set' : permet de modifier la valeur d’un attribut au cours du temps
  - 'animate' : permet aux attributs et aux propriétés scalaires de recevoir différentes valeurs au cours du temps
  - 'animateMotion' : déplace un élément le long d'un tracé de mouvement
  - 'animateColor‘ : modifie la valeur de couleur des attributs au cours du temps
  - 'animateTransform’ : anime un attribut de transformation (translation, changement d’échelle, rotation et/ou inclinaison).

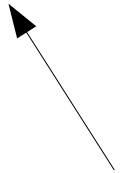


Attributs communs aux animations

- *begin* : temps de début de l’animation
- *end* : temps de fin de l’animation
- *from xx to yy* : le paramètre animé varie de xx à yy
- *dur* : durée de l’animation
- *repeatcount* : nombre de répétitions de l’animation
- *fill* : définit l’état à la fin de l’animation (freeze | remove)
- *attributeType* : "CSS | XML | auto" indique le type d’attribut qu’il faut animer

Exemple d'animation : set

```
<rect x="50" y="50" width="200" height="100“
      fill="#CCCCFF" stroke="#000099"
      visibility ="hidden" >
   <set attributeName="visibility" attributeType="XML"
      begin="4s" dur="5s" to="visible" />
</rect>
```



Variation de la visibilité
de l'objet (caché → visible)

Exemple d’animation : animate

```
<rect x="300" y="100" width="300" height="100"
      fill="rgb(255,255,0)" >
  <animate attributeName="width" attributeType="XML"
    begin="0s" dur="9s" fill="freeze" from="300" to="800" />
  <animate attributeName="height" attributeType="XML"
    begin="0s" dur="9s" fill="freeze" from="100" to="300" />
</rect>
```

Variation de la largeur/hauteur de l’objet

Attributs qui contrôlent la succession des animations

- additive = replace | sum
  - si plus d'1 transformation sont effectuées en même temps
  - **sum** = l'animation va s'ajouter à la valeur de l'attribut et des autres animations,
  - **replace** = l'animation va surclasser la valeur de l'attribut et des autres animations (valeur par défaut).
- ex : simple animation « agrandir » qui va accroître la largeur d'un rectangle de 10 pixels (1 px / s) ...

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

Attributs qui contrôlent la succession des animations

- accumulate : sum | none

  chaque **itération** (**repeatCount**) se bâtit sur la fin de la précédente (sum) ou non (none).

- Ex :

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

  - à la fin de la 1ère répétition, le rectangle a une largeur de 30 pixels,
  - à la fin de la 2ème une largeur de 40 pixels,
  - et à la fin de la 5ème une largeur de 70 pixels.

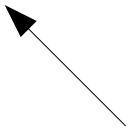
animateMotion

```
<circle cx="100" cy="100" r="40" stroke="blue"
        stroke-width="4" fill="red">
   <animateMotion dur="2s" repeatCount="indefinite"
                  path="M50,50 v200 h200 v-200 h-200" />
</circle>
```

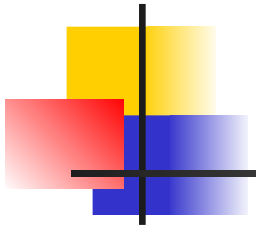
Cercle se déplaçant le long du chemin (path)

animateTransform

```
<rect x="50" cy="100" width="50" height="50"
      stroke="blue" stroke-width="4" fill="red">
  <animateTransform attributeName="transform"
                    attributeType="XML" type="translate"
from="0,0" to="300,300"  dur="2s" additive="sum" fill="freeze"/>
  <animateTransform attributeName="transform"
                    attributeType="XML"  type="rotate"
      from="0" to="90"  dur="2s" additive="sum" fill="freeze"/>
</rect>
```

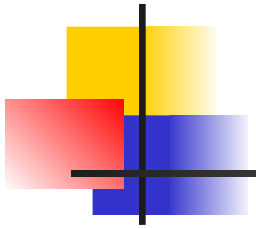


Rectangle translaté et tourné en même temps



Interactivité : évènement souris

- **click**
  - clic de souris sur l'objet (enfoncé puis relâché
  - Attribut d'événement : **onclick**
- **mousedown**
  - bouton enfoncé sur l'objet
  - Attribut d'événement : **onmousedown**
- **mouseup**
  - bouton relâché sur l'objet
  - Attribut d'événement : **onmouseup**
- **mouseover**
  - curseur au dessus de l'objet
  - Attribut d'événement : **onmouseover**
- **mousemove**
  - curseur bougé dans l'objet
  - Attribut d'événement : **onmousemove**
- **mouseout**
  - le curseur quitte l'objet
  - Attribut d'événement : **onmouseout**



Évènements animations

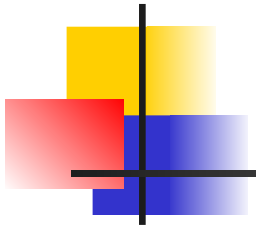
- **Begin**
  - un élément d'animation commence.
  - attribut d'événement : **onbegin**
- **endEvent**
  - un élément d'animation s'achève
  - attribut d'événement : **onend**
- **repeatEvent**
  - un élément d'animation se répète.
  - il est déclenché toutes les fois où l'élément se répète, après la première
  - attribut d'événement : **onrepeat**

Interactivité : évènement souris

Pour déclencher / arrêter des animations à partir d'évènements :

- **begin="id_objet.nom_evenement"**
- ex1 :
  - **begin ="bouton.click"**
  - **end= "carte.mouseout"**
  - **begin= "click"**
- ex2 : <**animateColor** fill="freeze" dur="0.1s" to="yellow"
  from="blue" attributeName="fill" **begin="mouseout"**/>
- ex3 :

```
<rect id="ani1" x="10" y="10" width="80" height="55" fill="blue" >
    <set begin="click" attributeName="fill" to="yellow" />
</rect>
```

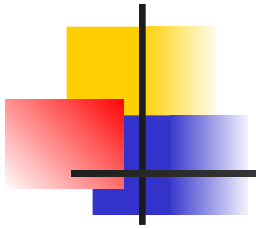


Autres évènements

- keydown (onKeydown)
  - Se produit lorsque l'utilisateur appuie sur une touche de son clavier.
- keypress (onKeypress)
  - Se produit lorsque l'utilisateur maintient une touche de son clavier enfoncée.
- keyup (onKeyup)
  - Se produit lorsque l'utilisateur relâche une touche de son clavier préalablement enfoncée.
- load (onLoad)
  - Se produit lorsque le navigateur de l'utilisateur charge la page en cours

Exemples

```
<rect id="pulser" x="40" y="25" width="120" height="50"
rx="10" ry="10" fill="blue" stroke="black" >
<animate begin="mouseover" end="mouseout;click"
    dur="2s" repeatCount="indefinite" attributeName="fill"
    from="blue" values="lightblue; blue; lightblue" fill="freeze" />
<animate begin="click" dur="1.0s" repeatCount="5"
    attributeName="fill" values="red; peachpuff;
    lightgoldenrodyellow; plum; white; red" fill="freeze" />
<animate begin="click+4.5s" dur="2.0s" attributeName="y"
    calcMode="linear" values="0; 1; 2; 4; 8; 16; 32; 64; 128; 256;"
    additive="sum" fill="freeze" /> <set begin="click+6.4s"
    attributeName="opacity" to="0" fill="freeze" />
<set begin="click+12.0s" attributeName="fill" to="blue" />
<set begin="click+12.0s" attributeName="opacity" to="1" />
</rect>
```



Scripts

- Création par code
- Interactivité

http://double.co.nz/video_test/video.svg