SVG et javascript

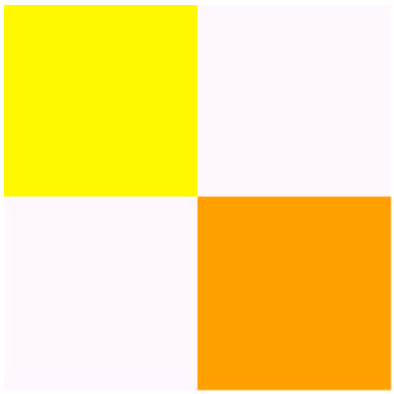
Le fichier SVG: plugin dans la page html

```
<embed width="600" height="488" src="europe.svg" name="carte" align="left">
```

Le fichier SVG est repéré par son nom, ici "carte"

Le système de coordonnées

Par défaut, l'angle supérieur gauche a pour coordonnées 0;0 et l'angle inférieur droit width;height.
On peut redéfinir ces coordonnées avec viewBox="-2760 0 5559 4524"
Attention: toutes les coordonnées sont **entières**



On peut définir de nouveaux cadrages avec les balises <svg>
<svg width="200" height="200">
<rect style="fill:yellow" width="100" height="100" /> par défaut x=0,y=0
<svg x="100" y="100" width="100" height="100">
<rect style="fill:orange" width="100" height="100" /> par défaut x=0,y=0
</svg>
</svg>

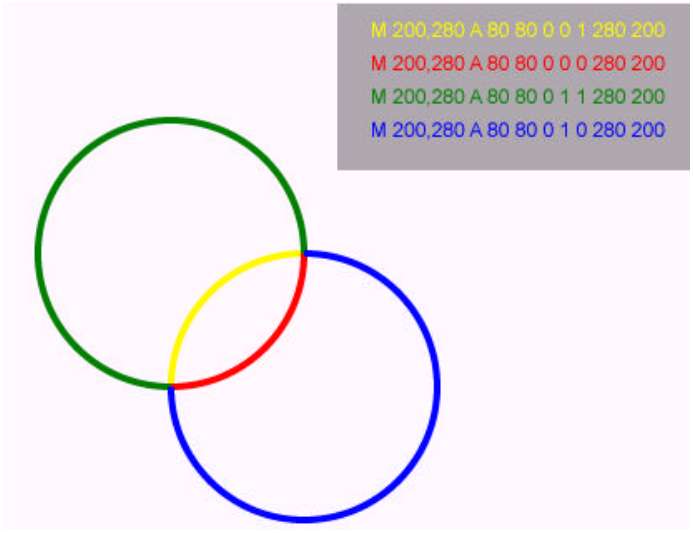
Les objets géométriques

Suite de tracés enchaînés: <path d="……"/>

Commandes
Majuscules: coordonnées absolues
minuscules: coordonnées relatives

A a Arc d'ellipse
tracé à partir du point courant
paramètres: demi-axe horizontal, demi-axe vertical, 3 fois 0 ou 1 pour définir l'arc à tracer, coordonnées du point d'arrivée

Comme sont définis le point de départ, le point d'arrivée et les axes de l'ellipse, il y a 4 arcs possibles.



C c	courbe de Bézier cubique	C -445,2375 -1595,3600 483,2226
H h	ligne horizontale	H 100
L l	joint le point	L 50 100 L est implicite dans la liste
M m	va au point sans tracer	M 100 200
Q q	courbe de Bézier quadratique	
S s	courbe lissée	
T t	courbe de Bezier quadratique lissée	
V v	ligne verticale	V 50
Z z	ferme la ligne polygonale	z

Rectangle: <rect id="bkgrnd" x="0" y="0" width="400" height="400" style="fill:#E0E0E0"/>
x et y coordonnées de l'angle supérieur gauche, width est la largeur et height la hauteur

Ellipse: <ellipse id="forme" cx="200" cy="300" rx="50" ry="30" style="fill:#AACC00"/>
cx et cy sont les coordonnées du centre, rx et ry les demi-axes

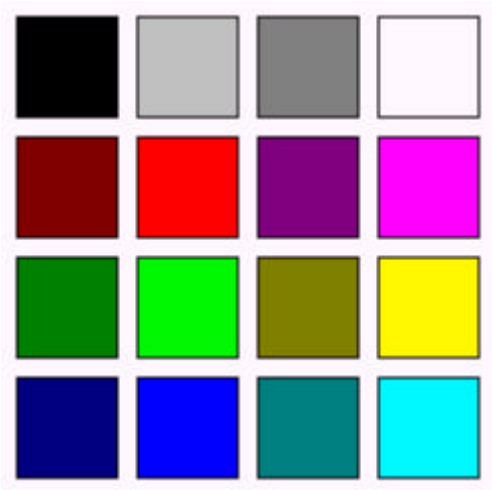
Cercle: <circle id="forme" style="stroke:black; fill:none" cx="50" cy="50" r="10"   />
cx et cy sont les coordonnées du centre, r le rayon

Ligne: <line x1="50" y1="50" x2="100" y2="50" />

Ligne brisée: <polyline points="50,50,75,150,100,50,125,150,150,50,175,150" />

Polygones: <polygon points="99,50,143,125,56,124,99,50" />

Les styles



Couleurs
Couleurs prédéfinies:

black	silver	gray	white
maroon	red	purple	fuchsia
green	lime	olive	yellow
navy	blue	teal	aqua

Couleur RVB donnée par ses composantes (0 à 255)
en hexadécimal "#A0B1C2"
en décimal rgb(255,0,0)
en pourcentage rgb(100%,0%,0%)

Couleur de tracé: style="stroke:black "
Couleur de remplissage: style="fill:black"

Opacité et transparence
style="fill:blue;fill-opacity:0.2" le coefficient va de 0 (rien n'est dessiné à 1 opaque)

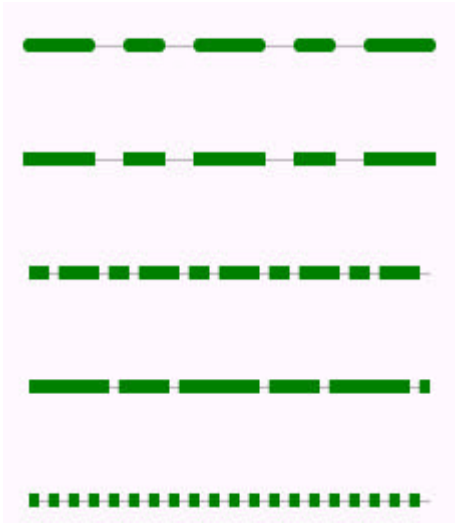
Attention: les objets sont dessinés dans l'ordre du fichier, le dernier de la liste est au-dessus des autres

Epaisseur de ligne: style="stroke-width:1"

Styles pour joindre les lignes:

<path style="stroke-linejoin:miter" d="……."/>
<path style="stroke-linejoin:round" d="……."/>
<path style="stroke-linejoin:bevel" d="……"/>

Styles de lignes



<path d="M300, 50h200" style="stroke-linecap:round; stroke-dasharray:30 20 15 20" />

<path d="M300,100h200" style="stroke-linecap:square; stroke-dasharray:30 20 15 20" />

<path d="M300,150h200" style="stroke-linecap:butt; stroke-dasharray:10 5 20 5" />

<path d="M300,200h200" style="stroke-linecap:butt; stroke-dasharray:40 5 25 5" />

<path d="M300,250h200" style="stroke-linecap:butt; stroke-dasharray:5 5" />

Le texte

Polices: <style="font-family:Times Roman" > ou < style="font-family:Helvetica;font-style:italic">

Taille: < style="font-size:18">

Couleur: <style="fill:red">

Alignement sur la position définie par x et y
<text x="280" y="70" >Texte aligné à gauche par défaut</text>
<text x="280" y="100" style="text-anchor:start">Texte aligné à gauche</text>
<text x="280" y="130" style="text-anchor:middle">Texte centré</text>
<text x="280" y="160" style="text-anchor:end">Texte aligné à droite</text>

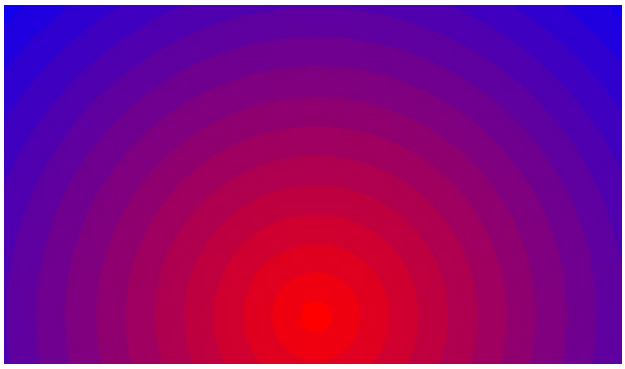
Pour changer la couleur d'un mot au milieu d'une phrase:

<tspan style="fill:red">rouge</tspan>

Les textures

Exemple de texture avec un gradient circulaire
Définition de la texture

```
<defs>
<radialGradient id="grad1" cx="250" cy="250" r="300"
fx="250" fy="250">
<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"/>
 </radialGradient>
</defs>
```

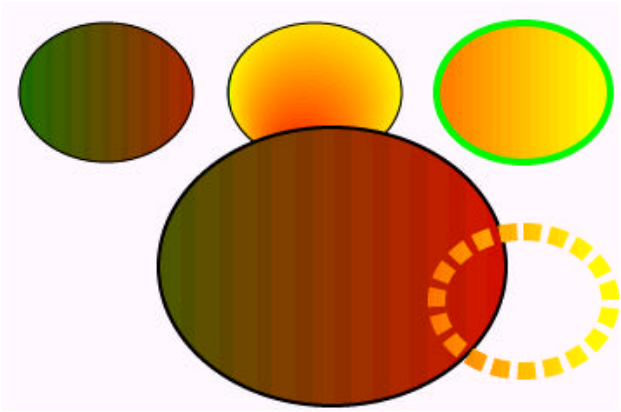


Utilisation de la texture ainsi définie pour remplir un rectangle:

```
 <rect style="fill:url(#grad1)" x="50" y="50" width="400" height="400" />
```

Autres exemples de gradients:

```
<linearGradient id="lingrad1" x1="0" y1="0" x2="350"
y2="0">
<stop offset="0" style="stop-color:green"/>
<stop offset="0.5" style="stop-color:red"/>
<stop offset="1" style="stop-color:yellow"/>
</linearGradient>
```



Les transformations

Des transformations peuvent être appliqués aux objets (y compris le texte)
Sont définies
la translation: <rect height="100" width="30" transform="translate(130 50)"/>
Le rectangle dont l'angle de départ avait pour coordonnées (100;30) sera translaté de 130 en x et 50 en y, son angle de départ sera alors (230;80)
Attention pour les transformations, le centre est obligatoirement (0;0) et les ordonnées sont orientées du haut vers le bas …..

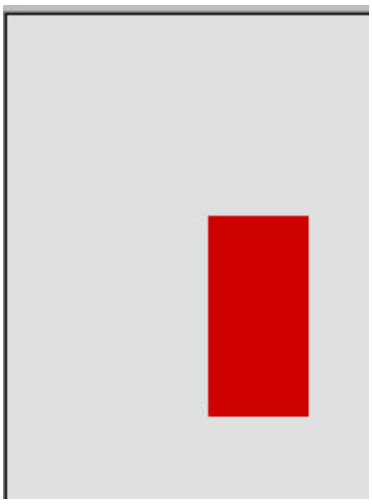
la rotation:
<rect height="100" width="30" transform="rotate(60)"/>
l'angle est en degrés, orienté dans le sens trigonométrique pour le repère et donc le sens contraire pour la représentation habituelle du plan.

les symétries par rapport aux axes:
<rect height="100" width="30" transform="skew-x(40)"/>
<rect height="100" width="30" transform="skew-y(40)"/>

l'homothétie généralisée avec un rapport en x et un autre en y:
<rect height="100" width="30" transform="scale(2 3)"/>

et toute transformation à partir d'une matrice:
<rect height="100" width="50" transform="matrix(0 1 1 0 80 50)"/>

Sur cet exemple, l'angle du rectangle de départ est à (0;0) valeurs par défaut de x et y.
La matrice (0 1 1 0) est celle de la symétrie axiale par rapport à la 1ère bissectrice, l'autre pour le plan habituel, et une translation de (100;100) est appliquée, d'où le résultat:



Avec le plugin d'Adobe, le résultat le plus fiable est obtenu à partir des matrices des transformations

Les Animations

Un exemple, un disque tourne en s'étirant et se rétractant:

```
<svg>
<g transform="translate(160 150)">
<circle id="circle1" style="fill:#3333ff;stroke-width:1;opacity:.8" cx="0" cy="0" r="30">
<animateTransform attributeName="transform" type="rotate" values="0;360" additive="sum" dur="3s"
repeatDur="indefinite"/>
<animateTransform attributeName="transform" type="scale"  values="4 1;0.5 1;4 1"   additive="sum" dur="4s"
repeatDur="indefinite"/>
</circle>
</g>
</svg>
```

Notons que les transformations ayant (0;0) pour centre, on définit le disque centré en (0;0), puis on lui applique une translation (160 150) pour qu'il soit visible.

On peut évidemment commander une animation avec la souris en cliquant sur un objet.

Voir plus loin la gestion des actions de la souris sur le dessin.

Accès aux objets:

Un exemple d'objet: une ligne polygonale fermée

```
<g id="France" style="stroke:#131516;stroke-width:3;stroke-linejoin:round;fill:#DEECB3">
  <path d="M-1006 3752 l 25 -36 8 -31 13 -12 0 -102 -17 -17 -17 21 0 16 -58 19 -10 77 20 36 27 29 9 0 z"/>
 </g>
```

Appels javascript depuis le fichier HTML

Récupérer la couleur du remplissage

```
a=document.carte.getSVGDocument().getElementById('France').getAttribute('style');
b=a.indexOf('fill:');couleur=a.substring(b+5,b+12);
```

(b vaut dans notre exemple "stroke:#131516;stroke-width:3;stroke-linejoin:round;fill:#DEECB3 ")

Attribuer une couleur de remplissage à l'objet

```
document.carte.getSVGDocument().getElementById('France').getStyle().setProperty('fill', '#FF0000')
```

Montrer/cacher cet objet

```
document.carte.getSVGDocument().getElementById('France').getStyle().setProperty("display","none")}
document.carte.getSVGDocument().getElementById('France').getStyle().setProperty("display","inline")
```

Modifier un objet

Depuis la page HTML
Tous les attributs d'un objet sont modifiables
Ici le plugin est nommé "figure", l'objet est un rectangle, nous modifions sa largeur

```
document.figure.getSVGDocument().getElementById('ID_objet').setAttribute('width',200);
```

En utilisant une fonction définie dans le fichier svg

Nous voulons dessiner un triangle dont les coordonnées des sommets sont envoyées
Depuis le fichier HTML, appel en javascript:

```
window.triangle(xa,ya,xb,yb,xc,yc);
```

Dans le fichier svg:
Au chargement, appel de la fonction OnLoadEvent(evt) pour définir l'objet window.triangle

```
<svg id="root" xml:space="preserve" width="400" height="400" onload="OnLoadEvent(evt)">
……..
```

Le tracé est défini, vide au départ, il a l'ID "trace1"

```
<path id="trace1" style="stroke-width:1; stroke:blue; fill:none" d=""/>
……
<script><![CDATA[
```

Fonction triangle qui construit la chaîne d

```
function triangle(a,b,c,d,e,f)
{   var str="M";
  str+=a+" "+b+" "+c+" "+d+" "+e+" "+f+" z";
  trace1.setAttribute("d", str);}
```

Déclaration de l'objet window.triangle

```
function OnLoadEvent(evt)
{   var doc = evt.getTarget() != null ? evt.getTarget().getOwnerDocument() : null;
  if (doc != null)
  {  trace1 = doc.getElementById("trace1");
     window.triangle = triangle;}}
]]></script>
</svg>
```

Nous aurions le même effet en plus simple avec dans le fichier HTML, l'appel:

```
tri="M "+xa+" "+ya+" "+xb+" "+yb+" "+xc+" "+yc+" z";
document.figure.getSVGDocument().getElementById("trace1").setAttribute("d",tri);
```

Travailler à la souris sur le dessin

Nous pouvons récupérer les actions de la souris sur le dessin
Un exemple simple:
Quand la souris est sur le dessin, les coordonnées du pointeur sont récupérées et affichées dans un formulaire nommé "coords" de la page HTML, quand la souris sort du rectangle, l'affichage est vide ….

```
<?xml version="1.0" encoding="iso-8859-1"?>
<svg id="root" xml:space="preserve" width="400" height="400">
<script><![CDATA[
var xm;
var ym;
function coordonnees(evt)
{xm=evt.getClientX();ym=evt.getClientY();
document.coords.x.value=xm;document.coords.y.value=ym;}
function sorti(evt)
{document.ou.T1.value="";document.ou.T2.value="";}
]]></script>
<g onmousemove="coordonnees(evt)" onmouseover="coordonnees(evt)" onmouseout="sorti(evt)">
<rect id="bkgrnd" x="0" y="0" width="400" height="400" style="fill:#E0E0E0"/>
</g>
</svg>
```

Autre exemple: déplacer une figure en cliquant à la souris et en déplaçant la souris:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<svg id="root" xml:space="preserve" width="400" height="400" onload="OnLoadEvent(evt)">
<script><![CDATA[
var appui=false;
var xm;
var ym;
function accrocher(evt)
{if (evt.getTarget().getAttribute("id")=="bkgrnd")
{xm=evt.getClientX();ym=evt.getClientY();appui=true;
rectangle.setAttribute("x",xm);
rectangle.setAttribute("y",ym);}}
function bouger(evt)
{if (appui==true)
{xm=evt.getClientX();ym=evt.getClientY();
rectangle.setAttribute("x",xm);
rectangle.setAttribute("y",ym);}}
function lacher(evt)
{if (evt.getTarget().getAttribute("id")=="rectangle")
{appui=false;}}
function OnLoadEvent(evt)
{ var doc = evt.getTarget() != null ? evt.getTarget().getOwnerDocument() : null;
  if (doc != null)
  { rectangle=doc.getElementById("rectangle");}}
]]></script>
<g onmousemove="bouger(evt)" onmousedown="accrocher(evt)" onmouseup="lacher(evt)">
<rect id="bkgrnd" x="0" y="0" width="400" height="400" style="fill:#E0E0E0"/>
<rect id="rectangle" x="0" y="0" width="100" height="50" style="fill:#CC0000"/>
</g>
</svg>
```

Quand on clique, le rectangle est collé à la souris et suit ses mouvements tant que le bouton est pressé.