M A S - I C T

C O U R S S V G

Bienvenue

Participants

Ambord Patrick
Blanc Grégoire
Byrne Simon
Chassot Frédéric
Chevre Sébastien
Dennig Sylvain
Gomez Alonso David
Hauenstein Christina
Iriarte Pablo
Kouch Rémi
Müller Claude
Peter Serge
Sehgelmeble Neil
Stauffer Sébastien
Yeddou Yazid M'Hamed

Programme

Jeudi 2 octobre 2014:

XML, DTD, SVG, Géométrie et Evénements

Jeudi 9 octobre 2014:

DOM, Interaction, Animation, Formulaire

Jeudi 16 octobre 2014:

Inkscape et SVG, Projet complet

Samedi 1er novembre 2014 : examen

Horaire

Matin salle A501

9h00 – 10h30 : cours / exercices

10h45 – 12h15 : cours / exercices

Après-midi salle A404

13h00 – 14h00 : cours

14h00 – 16h00 : laboratoire

Bibliographie

Livres :

- SVG, *J. David Eisenberg*, O'REILLY 2001
- SVG unleashed, *collectif*, SAMS 2003

Internet :

- www.w3.org/TR/SVG/
- http://www.yoyodesign.org/doc/w3c/svg1/

Plan du 2 octobre

- Document XML
- Document SVG
- Dessin : formes de base
- Dessin : tracés de chemin
- Transformations géométriques
- Evénements avec Javascript

XML

eXtensible Markup Language

XML est un *'méta-langage'* à balise formé de:

- Un prologue:

  <?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>

- Des éléments:

  <balise> ……. </balise> ou <balise/>

- Des commentaires:

  <! - - *Ceci est un commentaire* - ->

XML valide

Un document XML est **bien formé** si:

- Il ne contient qu'une balise racine
- Toute balise ouverte est refermée
- Les éléments (balises) sont imbriqués

  pas de chevauchement de balises.

En outre, un document XML est **valide** s'il est accompagné d'une **DTD**.

Exemple XML

```
<?xml version="1.0" standalone="yes"?>
<movie>
<title>Bleu</title>
<star>Reno</star>
<star>Rosa</star>
<cinema>Leone</cinema>
<cinema>Fellini</cinema>
</movie>
```

DTD

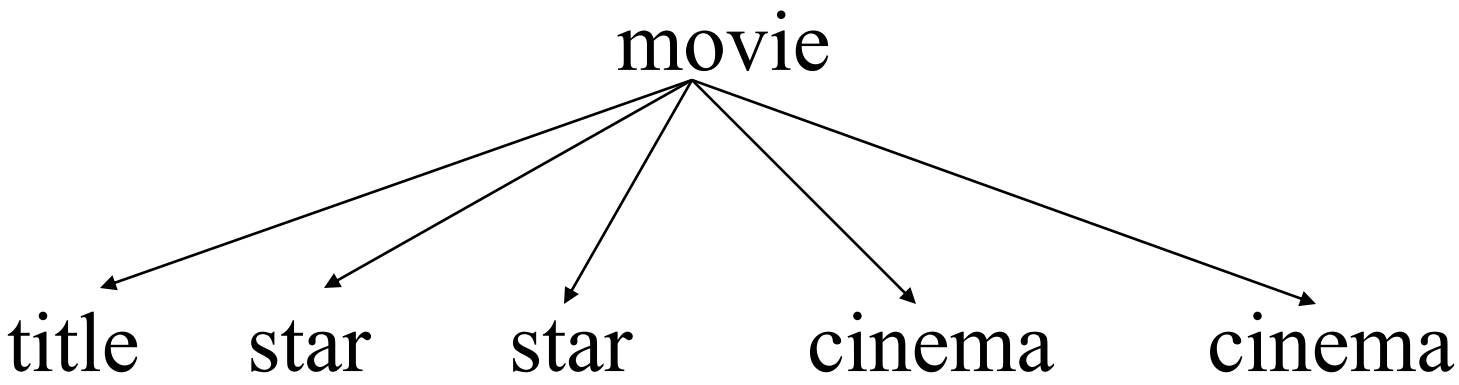
Document Type Definition

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE movie [
<!ELEMENT movie (title,star*,cinema+)>
<!ELEMENT title CDATA>
<!ELEMENT star CDATA>
<!ELEMENT cinema CDATA>
]>
```

DOM

Document Object Model

Le DOM est une représentation hiérarchique (sous forme d'arbre) du document XML.



SVG

Scalable Vector Graphic

Graphiques vectoriels adaptables

Recommandation (DTD) du W3C pour l'édition de graphiques en deux dimensions.

Scalable : les graphiques peuvent être adaptés (changement d'échelle) sans perte de qualité.

Vector : les éléments du graphique sont définis par leur forme plutôt que point par point (bitmap).

Viewer SVG

Un document XML doit être transformé pour être visualisé correctement.

XML ⇔ XSLT ⇔ Affichage

XML ⇔ XML-FO ⇔ Impression

SVG ⇔ viewer ⇔ Affichage

Un document SVG doit être interprété, il doit donc respecter scrupuleusement sa DTD.

Exemples

Quelques petits exemples

Plan de quartier pour personne aveugle

Document SVG

Un document SVG est formé d'un prologue suivi d'un élément racine: <svg>

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
"http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg xmlns="http://www.w3.org/2000/svg"
     xmlns:link="http://www.w3.org/1999/xlink">
(...)
</svg>
```

Balise <svg>

La balise svg contient les éléments suivants:

- x et y : position du coin supérieur gauche du document SVG dans la fenêtre dans laquelle il a été ouvert
- weidth : largeur du document
- height : hauteur du document
- viewBox : système de coordonnées

Balise <svg> 2

Exemple :

```
<svg x ="0" y="0"
    width="300px" height="200px"
    viewBox="0 0 1500 1000"
    preserveAspectRatio="none"
```

Permet d'ouvrir un graphique de 300 pixels sur 200 gradué en 5 pour 1 pixel

Formes de base

Les formes de base sont les lignes, les lignes polygonales, les rectangles, les cercles et les polygones dont certains attributs sont :

id: un identificateur

stroke: couleur du trait

stroke-width: épaisseur du trait

fill: couleur de remplissage

Ligne

Une ligne est en fait un segment de droite défini par ses deux extrémités (x1,y1) et (x2,y2):

```
<line x1 y1 x2 y2/>
```

Exemple de ligne rouge définie entre (10,20) et (30,50) traitillée :

```
<line x1="10" y1="20" x2="30" y2="50"
 stroke="red" stroke-dasharray="9,5"/>
```

Rectangle

Un rectangle est défini par son coin supérieur gauche, sa largeur et sa longeur :

```
<rect x y width height/>
```

Exemple de rectangle rouge avec contour bleu défini en (10,20) de largeur 30 et longueur 50:

```
<rect x="10" y="20"
 width="30" height="50"
 stroke="blue" fill="red"/>
```

Cercle et ellipse

Un cercle est défini par son centre et son rayon:

```
<circle cx cy r/>
```

Une ellipse est définie par son centre et ses deux rayons:

```
<ellipse cx cy rx ry/>
```

Polygone

Un polygone est défini par une suite de points définis par deux nombres, il est fermé:

```
<polygon points/>
```

Une ligne polygonale (polyline) est définie de la même manière, elle ouverte.

Exemple de polygone rouge d'opacité 50%:

```
<polygon points="10 20, 30 40, 30 70"
 fill="red" opacity="0.5"/>
```

polygone fermé auto

Groupage

Il est possible de regrouper différents éléments de dessin dans un élément g.

Les éléments g peuvent être imbriqués.

```
<g id="tete"
  <circle cx="100" cy="100" r="10"
  <circle cx="200" cy="100" r="10"
  ..........
</g>
```

La balise <defs>

L'éléments <defs> est un conteneur qui permet de définir des éléments référencés:

```
<defs>
    <rect id="MonRect" width="60" height="10"/>
</defs>
```

Référencé à l'aide d'une balise <use> et d'un lien (xlink:href)

```
<use x="20" y="10" xlink:href="#MonRect" />
```

Transformations du plan

Les transformations géométriques **translation, rotation, changement d'échelle et inclinaison des axes,** sont obtenues par l'utilisation de l'attribut transform qui doit être placé dans une balise.

Il est recommandé de placer cet attribut dans un groupe <g>

coordonnées homogènes

Translation

Par 'translate':

```
<g transform="translate(tx,ty)">
  figure à translater
</g>
```

Par matrice:

```
<g transform="matrix(1,0,0,1,tx,ty)">
  figure à translater
</g>
```

Où tx est la translation en x et ty en y

Rotation 1

Par ‘rotate’, il est possible d’effectuer une rotation autour dun point donné:

```
<g transform="rotate(alpha,cx,cy)">
  figure à faire tourner
</g>
```

alpha est l’angle de rotation en degrés

cx et cy sont les coordonnées du centre de rot.

Rotation 2

Par une matrice de rotation :

```
<g transform="matrix(ca,sa,-sa,ca,0,0)">
  figure à faire tourner
</g>
```

Où **ca** est le cos(*alpha*)

sa est le sin(*alpha*)

L'effet est une rotation d'un angle *alpha* des axes du système de coordonnées.

Changement d'échelle

Par 'scale':

```
<g transform="scale(sx,sy)">
  figure à modifier
</g>
```

Par matrice:

```
<g transform="matrix(sx,0,0,sy,0,0)">
  figure à modifier
</g>
```

Où sx est le changement en x et sy en y

Homothétie

Une homothétie est un changement d'échelle pour lequel une seule valeur est donnée:

Par 'scale':

```
<g transform="scale(h)">
   figure à modifier
</g>
```

h est le facteur d'homothétie

Composition

Par une suite de transformations imbriquées:

```
<g transform="scale(2)">
 <g transform="rotate(45)">
  <g transform="translate(5,10)">
   <!-- Les éléments graphiques -->
</g> </g> </g>
```

Ou par la multiplication matricielle

Tracés

Une figure géométriques peut s'obtenir par le dessin de son contour à l'aide de la balise path

Un chemin ('path') est défini par une suite de points reliés par des morceaux de courbe.

`<path d>`

d est la description du chemin

Les attributs de rendu sont les mêmes que pour les formes de base.

Chemin

L'attribut **d** contient les points ainsi que les commandes:

- M, m : *moveto* (déplacement sans dessin)
- L, l : *line* (dessin d'un segment, voir H et V)
- C, c : *curve* (Bézier cubique)
- Q, q : *curve* (Bézier quadratique)
- A, a : *arc* (arc de cercle)
- Z, z : *closepath*.

Majuscule: absolu
Minuscule: relatif à ma page

Chemin 2

M 10 20: déplacement absolu au point (10,20)

m 10 20: déplacement relatif

de 10 horizontalement à droite

de 20 verticalement a bas

Ceci est valable pour toutes les commandes

Exemple de triangle rouge de contour bleu:

```
<path d="M 100 100 L 300 100 L 200 300 Z"
      fill="red" stroke="blue"/>
```

Arc elliptique

La commande :

```
A rx ry rotation arc-large balayage x y
```

permet de dessiner un arc de cercle défini en valeurs absolues.

Du point courant au point (x,y)

De rayons rx et ry

D'inclinaison d'axe x 'rotation'

arc-large balayage

Bézier quadratique

La commande :

```
Q x1 y1 x y
```

permet de dessiner une courbe de Bézier quadratique.

Du point courant au point (x,y)

Par une poignée placée au point (x1,y1)

Présentation de Bézier quadratique

Bézier cubique

La commande :

```
C x1 y1 x2 y2 x y
```

permet de dessiner une courbe de Bézier quadratique.

Du point courant au point (x1,y1)

Par une poignée placée au point (x,y)

Exemple de Bézier cubique

Ecmascript

Normalisation par l'organisme suisse ECMA du langage Javascript

Javascript a été créé par Netscape pour développer des applications Internet et, par la même occasion, pour étendre les possibilités du HTML.

Il s'agit d'un langage léger et orienté objet, permettant d'écrire des scripts interprétés par le navigateur et incorporés dans le code SVG.

Script dans SVG

La balise <script> permet d'insérer un script dans un document SVG :

```
<script type="text/ecmascript">
  <![CDATA[
      contenu du script :
      déclarations et fonctions
  ]]>
</script>
```

Fonctions

Entête de la fonction:

function *nom* (*paramètres*)
un paramètre particulier : evt
Il permet d'associer un élément SVG
exemple : evt.target

Corps de la fonction:

{suite d'instructions séparées par ';'}

Appels

L'éxécution d'une fonction est déclenchée par un événement sur un élément SVG

Ces événements sont déclarer comme des attributs

Exemple: <rect onclick="fct()" …>

La fonction fct() est appelée lorsque l'on clique dans le rectangle.

Evénements 1

Evénements sur le document :

onload, onunload, onabort, onerror, onresize, onzoom.

Evénements sur la balise <svg>: onload

Exemple : <svg onload="init()">

Exécute la fonction init() au chargement du document SVG

Evénements 2

Evénements sur un élément graphique :

onclick : click de souris

onmousedown : bouton appuyé

onmouseup : bouton relâché

onmouseover : souris sur l'élément

onmousemove : la souris bouge

onmouseout : la souris sort.

Evénements 3

Exemples :

```
<circle onclick="circle_click(evt)" …….>
<rect onmouseover="rect_over()" …….>
<g onmousemove="g_move(1)" …….>
<path onmouseout="path_end()" …….>
```

Evénements sur un élément d'animation :

onbegin, onend, onrepeat

variables

Déclaration des variables soit globalement, soit localement (dans une fonction)

```
var nom_variable  ( = 21);
```

Un tableau de 3 valeurs numériques:

```
var nom_tableau = new Array (1, 2, 9);
```

Un élément du document SVG:

```
var nom_objet = evt.target;
var nom_objet = 'getElementById';
```

Elements 1

Exemple d'accès par identificateur :

```
function toto() {
var element =
   svgDocument.getElementById("nom_id");
……….}

<g id="nom_id">
……….
</g>
```

Elements 2

Exemple d'accès par paramètre 'evt' :

```
function toto(evt) {
    var element = evt.getTarget();
    ……}

<g id="nom_id"  onclick="toto(evt)">
    ……
</g>
```

Attributs

Accès aux attributs d'un élément (balise):

var objet; (accès à balise par target ou par id)

objet.setAttribute("nom", "valeur");

Attribue la valeur à l'attribut "nom".

Var valeur = objet.getAttribute ("nom");

Place la valeur de l'attribut 'nom' dans la variable 'valeur'.

Clonage

Création d'un élément par copie (cloneNode):

<element id="nom" …….>

```
var oldobjet =
svgDocument.getElementById("nom");
var newobjet =
oldobjet.cloneNode(true);
newobjet.setAttribute("??","??");
parent.appendChild(newobjet);
```

'parent' est l'élément (balise) parent de 'nom'