Partie 2

COURS - Images Matricielles et Vectorielles

2.1 Préparation au TP Animation SVG

2.2 Introduction

Dans cette séance, nous allons découvrir deux grandes familles d'images numériques : - Les **images matricielles** (bitmap) : photos, images en pixels - Les **images vectorielles** : logos, icônes, tracés mathématiques

Nous nous concentrerons ensuite sur le **format SVG** (Scalable Vector Graphics) que vous utiliserez dans le prochain TP pour créer des animations.

2.3 1. Images Matricielles vs Images Vectorielles

2.3.1 1.1 Images Matricielles (Bitmap)

Définition : Une image composée de **pixels** (petits carrés colorés) organisés en grille.

Exemples de formats : `.jpg`, `.png`, `.gif`, `.bmp`

Caractéristiques : - Chaque pixel a une couleur précise - Qualité fixe liée au nombre de pixels - Perd en qualité si agrandie (pixellisation)

Exemple :

```
Image 4×4 pixels :
 
 
 
 
 
Si on double la taille → image floue/pixelisée
```

Utilisation : Photos, images complexes avec beaucoup de couleurs

---

2.3.2 1.2 Images Vectorielles

Définition : Une image définie par des **formules mathématiques** (points, lignes, courbes).

Exemples de formats : `.svg`, `.ai`, `.eps`

Caractéristiques : - Définie par des coordonnées et équations - Qualité infinie (peut être agrandie sans perte) - Fichiers légers pour des formes simples - Modifiable facilement

Exemple :

```
Cercle vectoriel :
Centre : (100, 100)
Rayon : 50
Couleur : rouge

Peut être affiché à n'importe quelle taille sans perte de qualité
```

Utilisation : Logos, icônes, schémas, animations, tracés

2.3.3 1.3 Comparaison

Critère	Matricielle	Vectorielle
Composition	Pixels	Formules mathématiques
Agrandissement	☒ Perte de qualité	☒ Sans perte
Taille fichier	Grande (photos)	Petite (formes simples)
Modification	Difficile	Facile
Adapté pour	Photos, images complexes	Logos, schémas, animations

---

2.4 2. Partie 1 : Calcul d'un Point sur un Segment

2.4.1 2.1 Le Principe Mathématique

Considérons deux points **A** et **B** dans un repère 2D : - Point A de coordonnées (x_a, y_a) - Point B de coordonnées (x☒, y☒)

Question : Comment trouver un point **M** situé sur le segment [AB] ?

Solution : Utiliser un paramètre **p** ∈ **[0, 1]**

2.4.2 2.2 La Formule Vectorielle

Formule :

```
OM = OA + p × AB
```

Où : - **O** = origine du repère (0, 0) - **OM** = vecteur position de M - **OA** = vecteur position de A - **AB** = vecteur de A vers B = OB - OA - **p** = paramètre entre 0 et 1

En développant :

```
OM = OA + p × (OB - OA)
OM = OA + p × OB - p × OA
```

```
OM = (1 - p) × OA + p × OB
```

2.4.3 2.3 En Coordonnées

Formule finale :

```
M.x = (1 - p) × A.x + p × B.x
M.y = (1 - p) × A.y + p × B.y
```

Ou de manière équivalente :

```
M.x = A.x + p × (B.x - A.x)
M.y = A.y + p × (B.y - A.y)
```

2.4.4 2.4 Interprétation de p

Valeur de p	Position de M	Calcul
p = 0	M est en A	M = A
p = 0.25	M à 25% de A vers B	M = 0.75A + 0.25B
p = 0.5	M au milieu de [AB]	M = 0.5A + 0.5B
p = 0.75	M à 75% de A vers B	M = 0.25A + 0.75B
p = 1	M est en B	M = B

2.4.5 2.5 Exemple Numérique

Données : - A(100, 50) - B(300, 200) - p = 0.5 (milieu du segment)

Calcul :

```
M.x = 100 + 0.5 × (300 - 100)
M.x = 100 + 0.5 × 200
M.x = 100 + 100 = 200

M.y = 50 + 0.5 × (200 - 50)
M.y = 50 + 0.5 × 150
M.y = 50 + 75 = 125

Résultat : M(200, 125)
```

Vérification : Le milieu de A(100, 50) et B(300, 200) est bien M(200, 125)

2.5 3. Partie 2 : Normalisation et Facteurs d'Échelle SVG

2.5.1 3.1 Le Problème

Un SVG possède **deux systèmes de coordonnées** :

1. **Système interne (viewBox)** : Coordonnées mathématiques définies par le créateur
2. **Système d'affichage (pixels)** : Taille réelle à l'écran

Exemple :

```
<svg width="800px" height="600px" viewBox="0 0 400 300">
```

- Le SVG s'affiche en **800×600 pixels** à l'écran
- Mais son système de coordonnées interne est **400×300 unités**

2.5.2 3.2 Le viewBox

Syntaxe :

```
viewBox="min-x min-y largeur hauteur"
```

Exemple :

```
viewBox="0 0 300 200"
```

- Coin supérieur gauche : (0, 0)
- Largeur interne : 300 unités
- Hauteur interne : 200 unités

Interprétation :

```
  0                    300
0

      Espace SVG

200
```

2.5.3 3.3 Normalisation

Pour passer du système SVG au système pixels, on utilise la **normalisation** :

Étape 1 : Convertir en pourcentage (0 à 1)

```
percentX = point.x / viewBox.width
percentY = point.y / viewBox.height
```

Étape 2 : Appliquer au conteneur en pixels

```
pixelX = percentX × container.width
pixelY = percentY × container.height
```

2.5.4 3.4 Exemple Complet

Données : - viewBox SVG : `0 0 300 200` - Taille d'affichage : 600×400 pixels - Point SVG : (150, 100)

Calcul :

Étape 1 - Normalisation :

```
percentX = 150 / 300 = 0.5   (50%)
percentY = 100 / 200 = 0.5   (50%)
```

Étape 2 - Conversion en pixels :

```
pixelX = 0.5 × 600 = 300 px
pixelY = 0.5 × 400 = 200 px
```

Résultat : Le point (150, 100) du SVG s'affiche à (300, 200) pixels à l'écran.

2.5.5 3.5 Facteur d'Échelle

On peut aussi calculer directement les **facteurs d'échelle** :

```
scaleX = container.width / viewBox.width
scaleY = container.height / viewBox.height
```

Dans notre exemple :

```
scaleX = 600 / 300 = 2
scaleY = 400 / 200 = 2
```

Application directe :

```
pixelX = point.x × scaleX = 150 × 2 = 300 px
pixelY = point.y × scaleY = 100 × 2 = 200 px
```

2.6 4. Partie 3 : Les Chemins SVG (path)

2.6.1 4.1 Structure d'un Path

Un **path** SVG est défini par l'attribut `d` (data) qui contient une série de commandes.

Exemple simple :

```
<path d="M 50,50 L 150,50 L 100,150 Z" />
```

2.6.2 4.2 Les Commandes de Base

M - Move To (Aller à) **Syntaxe** : `M x,y`

Signification : Déplace le "curseur" au point (x, y) **sans tracer**

Exemple :

```
<path d="M 100,100" />
```

→ Place le point de départ en (100, 100)

Note : Toujours la première commande d'un path

L - Line To (Ligne vers) **Syntaxe** : `L x,y`

Signification : Trace une **ligne droite** du point actuel vers (x, y)

Exemple :

```
<path d="M 50,50 L 150,50 L 150,150 L 50,150 Z" />
```

Explication :

```
M 50,50      → Aller en (50, 50)
L 150,50     → Ligne vers (150, 50)    [ligne horizontale]
L 150,150    → Ligne vers (150, 150)   [ligne verticale]
L 50,150     → Ligne vers (50, 150)    [ligne horizontale]
Z            → Fermer le chemin (retour au début)
```

Résultat : Un carré de 100×100

C - Cubic Bézier Curve (Courbe de Bézier cubique) **Syntaxe** : `C x1,y1 x2,y2 x,y`

Signification : Trace une **courbe** du point actuel vers (x, y) avec deux points de contrôle

Paramètres : - **(x1, y1)** : Premier point de contrôle - **(x2, y2)** : Deuxième point de contrôle - **(x, y)** : Point d'arrivée

Exemple :

```
<path d="M 50,150 C 50,50 150,50 150,150" />
```

Explication :

```
M 50,150           → Départ en (50, 150)
C 50,50            → Point de contrôle 1 : (50, 50)
  150,50           → Point de contrôle 2 : (150, 50)
  150,150          → Arrivée en (150, 150)
```

Schéma conceptuel :

```
        (50,50)     (150,50)
          ↑                ↑

    Point de contrôle 1     Point de contrôle 2


      (50,150)  →courbe(150,150)
      Départ              Arrivée
```

Résultat : Une courbe en forme de “U”

Z - Close Path (Fermer le chemin) **Syntaxe** : `Z` ou `z`

Signification : Trace une ligne droite vers le point de départ (dernier M)

Exemple :

```
<path d="M 100,100 L 200,100 L 150,200 Z" />
```

→ Triangle (la commande Z ferme automatiquement en reliant au point M)

2.6.3 4.3 Exemples Complets

```
<svg viewBox="0 0 200 200">
    <path d="M 100,50 L 150,150 L 50,150 Z"
          fill="blue" stroke="black" stroke-width="2"/>
</svg>
```

Décomposition : - `M 100,50` : Départ au sommet - `L 150,150` : Ligne vers le coin bas-droit - `L 50,150` : Ligne vers le coin bas-gauche - `Z` : Retour au sommet (ferme le triangle)

```svg
<svg viewBox="0 0 300 150">
    <path d="M 0,75 C 50,25 100,25 150,75 C 200,125 250,125 300,75"
          fill="none" stroke="blue" stroke-width="3"/>
</svg>
```

Décomposition : - `M 0,75` : Départ à gauche au milieu - `C 50,25 100,25 150,75` : Première vague (montée puis descente) - `C 200,125 250,125 300,75` : Deuxième vague

---

```svg
<svg viewBox="0 0 200 300">
    <path d="M 150,50 C 150,0 50,0 50,50
             C 50,100 150,100 150,150
             C 150,200 50,200 50,250"
          fill="none" stroke="red" stroke-width="4"/>
</svg>
```

2.6.4 4.4 Variantes de Commandes

Exemple 3 : Lettre "S" stylisée Commandes relatives (minuscules) :

```
m, l, c, z  → Coordonnées relatives au point actuel
M, L, C, Z  → Coordonnées absolues
```

Exemple :

```svg
<!-- Absolu -->
<path d="M 100,100 L 200,100" />

<!-- Relatif (équivalent) -->
<path d="M 100,100 l 100,0" />
```

Autres commandes utiles : - **H x** : Ligne horizontale vers x - **V y** : Ligne verticale vers y - **S x2,y2 x,y** : Courbe de Bézier lissée - **Q x1,y1 x,y** : Courbe de Bézier quadratique

---

2.7 6. Synthèse et Préparation au TP

2.7.1 Ce que vous avez appris :

 Différence matriciel/vectoriel - Matriciel = pixels, vectoriel = formules mathématiques

 Calcul de point sur segment - Formule : M = A + p × (B - A) avec p ∈ [0, 1]

 Normalisation SVG - viewBox = système interne - Conversion en pourcentage puis en pixels

 Commandes SVG path - **M** : Déplacer le curseur (Move) - **L** : Tracer une ligne (Line) - **C** : Tracer une courbe (Curve) - **Z** : Fermer le chemin (Close)

2.7.2 Lien avec le prochain TP :

Dans le TP, vous utiliserez : 1. Un fichier SVG créé avec GIMP contenant un `<path>` 2. La méthode JavaScript `path.getPointAtLength(distance)` qui utilise le principe du paramètre p 3. La normalisation pour convertir les coordonnées SVG en pixels d'affichage

Vous êtes prêts !

2.8 7. Pour aller plus loin

2.8.1 Ressources :

- MDN - SVG Tutorial : https ://developer.mozilla.org/fr/docs/Web/SVG/Tutorial
- SVG Path Visualizer : https ://svg-path-visualizer.netlify.app/
- GIMP Documentation Chemins : https ://docs.gimp.org/

2.8.2 Commandes SVG supplémentaires :

- **Q** : Courbe de Bézier quadratique (1 point de contrôle)
- **A** : Arc elliptique
- **H** : Ligne horizontale
- **V** : Ligne verticale

2.8.3 Applications pratiques :

- Création de logos
- Animations web
- Cartographie interactive
- Data visualisation
- Interfaces graphiques