R4.C.12 Initiation au GéoWeb et au Mashup

partie #2

BUT INFORMATIQUE

Webmapping

Le Webmapping (ou cartographie en ligne) est l'ensemble des solutions basées sur l'architecture web pour pouvoir produire, concevoir, traiter et publier des cartes géographiques numériques.

Cela regroupe donc trois composantes : une partie cliente (visualisation des données), une partie serveur (services web) et les données.

Panorama des solutions existantes

Il existe de nombreuses **librairies carto**, plus ou moins avancées, qui répondent plus ou moins aux besoins. En voici une liste non exhaustive :

- Leaflet : Open-source, librairie pour besoins basiques, et simple à mettre en place. De multiples plugins disponibles.
- OpenLayers : Open-source, librairie carto pour besoins plus avancés. Répond à quasiment tous les besoins (hors 3D).
- Google Maps API JS : Payante, librairie avancée 2D et 3D, avec accès simple à tous les contenus et services Google.

Panorama des solutions existantes

Il existe de nombreuses **librairies carto**, plus ou moins avancées, qui répondent à plus ou moins aux besoins. En voici une liste non exhaustive :

- ArcGis Maps for JS : Payante, librairie avancée 2D et 3D de l'écosystème ESRI.
- Mapbox GL JS : Payante, librairie avancée 2D et 3D.
- MapLibre : Open-source, fork de Mapbox GL JS. WebGL par défaut, donc performante. Avec des possibilités de 3D.
- deck.gl : Open-source, framework WebGL spécifique pour de la visualisation de données volumineuses. Intégrable dans les autres librairies.

Affichage des données cartographiques

Pour afficher des données cartographiques, ou simplement des données visuelles (= autres que textuelles), il n'existe finalement que 3 « solutions » clientes :

- des **images**
- le langage **SVG**
- un **contexte de dessin** au pixel

Ces solutions sont bien entendu utilisables conjointement. Elles sont celles utilisées dans les librairies cartographiques du marché.

Les images

On parle ici des images matricielles (raster), c'est à dire composées de pixels.

Il existe plusieurs formats adaptés au web : jpg, png, gif, ou plus récemment webp ou avif, qui gère ou non la transparence.

On utilise la balise HTML **`<img>`**, ou alors un élément HTML quelconque avec une image d'arrière-plan en CSS (background-image).

Les images

Exemple de tuile d'OpenStreetMap

Tuile Google Maps transparente (uniquement les toponymes)

Les images

Avantages :

- simples à utiliser
- performant (pas de calculs à réaliser pour le navigateur)

Inconvénients :

- les images doivent être générées en amont, c'est à dire coté serveur
- pas de personnalisation possible coté client
- pas d'interactions
- peuvent être pixelisées (écrans haute résolution)

Le langage SVG

SVG est un langage qui permet de générer des **images vectorielles**.

Basé sur XML (et donc proche de HTML), il permet de dessiner des formes simples ou avancées.

Une carte étant principalement composée d'objets vectoriels, ce format est assez utilisé, notamment pour les objets superposés aux cartes (marqueurs, lignes, etc.).

Le SVG peut être intégré directement dans un document HTML avec la balise <svg>, ou peut être une image au format .svg.

SVG fourni également une API JavaScript pour interagir avec les objets.

Le langage SVG

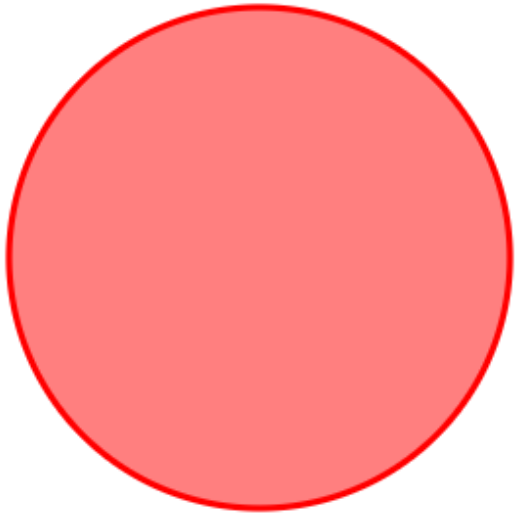
Avantages :

- personnalisation de l'affichage (rendu + accessibilité)
- interactions faciles (événements JS + API spécifique)
- rendu non pixelisé

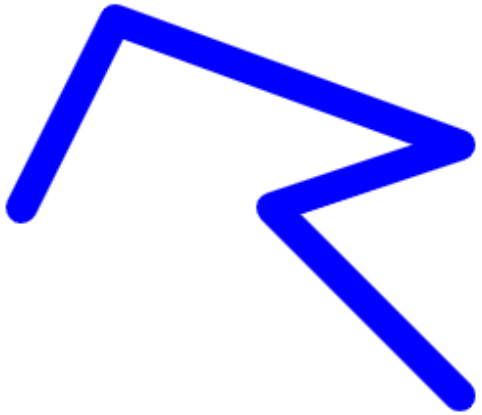
Inconvénients :

- problème de performance quand beaucoup d'objets
- pas vraiment adapté à des fonds de carte

Le langage SVG



Cercle SVG



Polyligne SVG

Le langage SVG

```
<svg viewBox="0 0 100 100" width="256" height="256">
  <circle r="40" cx="50" cy="50" fill="rgba(255,0,0,.5)“ stroke="red"></circle>
</svg>
```

```
<svg viewBox="0 0 100 100" width="256" height="256">
  <path d="m10,50L25,20L80,40L50,50L80,80" stroke-width="5" fill="none"
   stroke="blue" stroke-linejoin="round" stroke-linecap="round"></path>
</svg>
```

Contexte de dessin au pixel

Le **contexte de dessin** s'utilise à l'aide de la balise HTML <canvas>, liée à plusieurs APIs JavaScript spécifiques.

L'objectif est de dessiner les pixels sur ou de la carte, à la volée, grâce à JS.

On obtient un gain de performance d'affichage (par rapport à SVG), tout en permettant une personnalisation coté client (contrairement aux images seules).

Cependant, les interactions deviennent plus complexes, et on perd un peu en temps d'affichage, notamment sur mobile.

Contexte de dessin au pixel

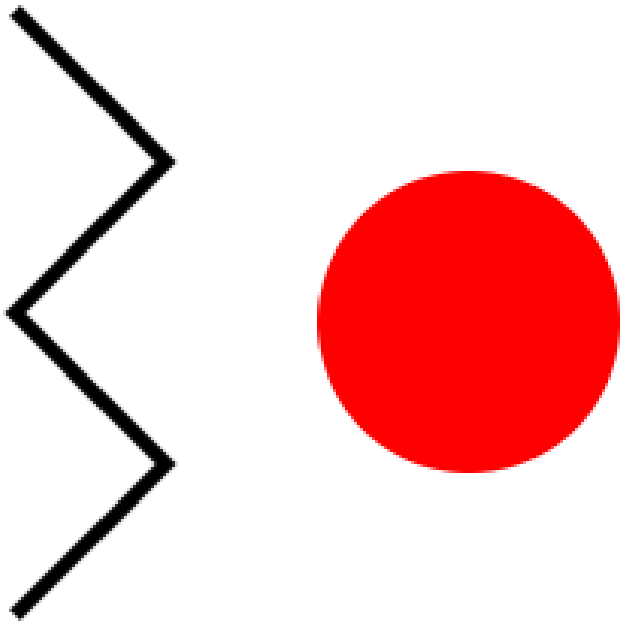
Avantages :

- personnalisation de l'affichage (rendu)
- performance accrue
- 3D possible (WebGL)

Inconvénients :

- interactions plus complexes (pas d'objets, que des pixels)
- perte d'accessibilité (par rapport à SVG)

Contexte de dessin au pixel



```
<canvas id="dessin" width="256" height="256"></canvas>
<script>
        let ctx = dessin.getContext('2d');
        ctx.width=ctx.height=256;
        ctx.moveTo(25,25);
        ctx.lineTo(75,75);
        ctx.lineTo(25,125);
        ctx.lineTo(75,175);
        ctx.lineTo(25,225);
        ctx.lineWidth = 5;
        ctx.stroke();
        ctx.beginPath();
        ctx.arc(175, 128, 50, 0, 2 * Math.PI);
        ctx.fillStyle = 'red';
        ctx.fill();
</script>
```

Tuiles cartographiques

Pour afficher une image de la Terre entière, à tous les niveaux de zoom, de manière simple, a été inventé le concept de pyramide de tuiles.

Au départ, le monde entier tient dans un carré de 256x256 pixels.

C'est le niveau de zoom 0, et l'image a pour coordonnées x=0, y=0 et z=0.

Tuiles cartographiques

Au niveau supérieur, la largeur et hauteur sont doublées, et on peut donc représenter le niveau 1 avec 4 images de 256x256 pixels.

Chaque image a pour coordonnées x=0/y=0, x=1/y=0, x=0/y=1, x=1/y=1 et z=1.

Tuiles cartographiques

Et ainsi de suite….

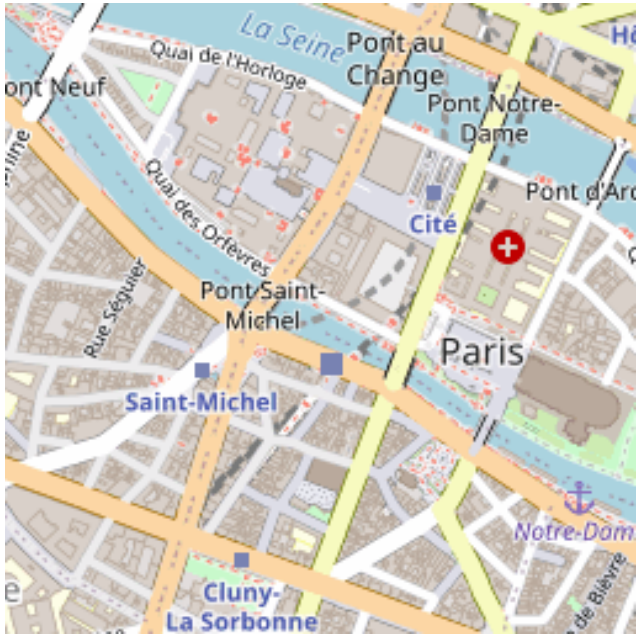
Tuiles cartographiques

Toutes les images sont précalculées coté serveur.

Ces images sont fournies par de multiples sources (ou providers), basées sur de multiples données, et sont requêtées sous forme d'une URL plus ou moins normalisée.

En voici une liste importante sur Leaflet.

Tuiles cartographiques



Tuile/Données OpenStreetMap

Tuile/Données Google Maps

Tuile Stamen/Données OpenStreetMap

Tuiles cartographiques

Plusieurs standards :

- WMS (Web Map Service) est un service qui permet de générer des cartes à la demande sous forme d'images (par exemple, PNG ou JPEG).Lorsqu'une requête est envoyée, le serveur WMS génère l'image à partir de données géospatiales et la renvoie au client. Les cartes sont donc générées dynamiquement, chaque image étant créée en réponse à une requête spécifique, ce qui permet une grande flexibilité.
- TMS (Tile Map Service) et WMTS (Web Map Tile Service), standards OGC qui définissent des requêtes contenant les paramètres x, y et z. Avec la possibilité d'obtenir des métadonnées (souvent assez verbeux). TMS définit le y du bas vers le haut, WMTS le contraire.
- XYZ est un standard « de fait » car très populaire et plus simple à déployer. Les requêtes contiennent les paramètres x, y et z. Il n'y a pas de métadonnées associées. Le y est défini du haut vers le bas.

formats de données géographiques

Les formats sont multiples et permettent de stocker, d'échanger, et d'analyser des données.

Certains formats sont adaptés au web, d'autres pas forcément. On peut catégoriser les formats en deux types : **raster** ou **vecteur**.

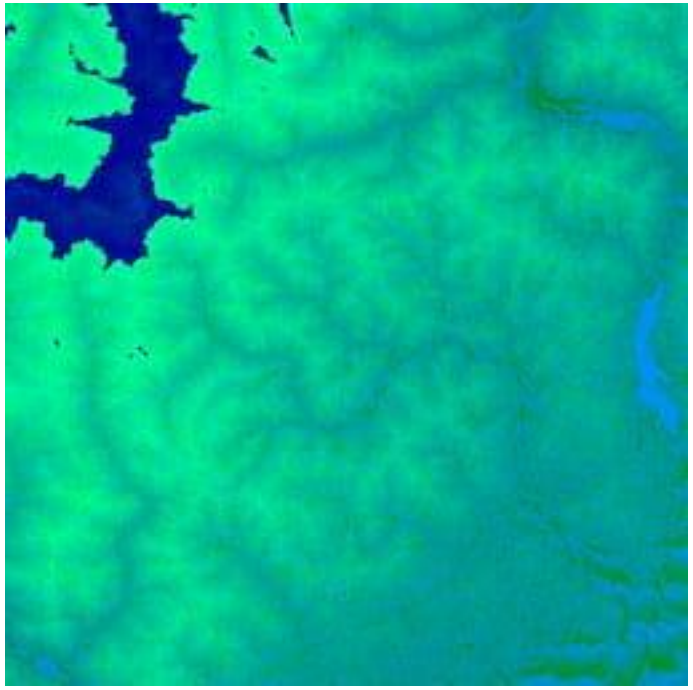
formats de données géographiques

Formats Rasters

Les principaux formats pour les fonds de cartes ou les **orthophotos** sont les formats d'images classiques (png, jpg, webp, etc.)

Une image composée de pixels peut contenir d'autres informations. C'est le cas par exemple des données d'élévations, où chaque pixel représente une hauteur en fausses couleurs.

De plus, il existe des formats d'images géoréférencées, comme GeoTIFF ou JPEG2000.



Tuile représentant l'élévation

formats de données géographiques

Formats Vecteurs

Le format vecteur le plus utilisé est certainement le Shapefile. Ce format n'est pas vraiment adapté à la communication client/serveur, donc au web en général.

Les **formats d'échanges** courants entre client et serveur sont basés sur les formats textes classiques : JSON, XML ou TXT.

formats de données géographiques

Formats Vecteurs

Formats basés sur **JSON** :

- **GeoJSON** permet d'encoder les structures classiques de données géographiques (point, lignes, surfaces), de manière très légère, tout en respectant la syntaxe objet JavaScript (JSON). C'est donc certainement le format le plus utilisé sur le web.
- **TopoJSON** est similaire à GeoJSON, mais stocke les données de manière topologique (maillage), et donc permet un gain de poids non négligeable pour des données jointes. Cependant, ce format est moins bien supporté par défaut.

formats de données géographiques

Formats Vecteurs

Exemple de données au format GeoJSON


```
{
 "type": "FeatureCollection",
 "features": [
  {
   "type": "Feature",
   "geometry": {
    "type": "Point",
    "coordinates": [ 2.3522, 48.8566 ]
   },
   "properties": {
    "name": "Paris"
   }
  }
 ]
}
```

formats de données géographiques

Formats Vecteurs

Formats basés sur **XML** :

- **GPX** est un format spécifique aux traces GPS.
- **KML** est un format conçu par Google pour ses propres produits.
- **GeoRSS** est format basé sur RSS (donc XML). Il permet d'ajouter une dimension géo dans un flux RSS.

```
<?xml version="1.0" encoding="UTF-8"?>
<gpx version="1.1">
  <wpt lat="48.8566" lon="2.3522">
    <name>Paris</name>
  </wpt>
</gpx>
```

Exemple de GPX avec waypoint

formats de données géographiques

Formats Vecteurs

Formats basés sur **TXT** :

- **WKT** (Well-Known Text) est un format de représentation textuelle des géométries spatiales.
- **WKB** (Well-Known Binary) une version binaire de WKT.

```
POINT (2.3522 48.8566)

POLYGON ((10 10, 20 10, 20 20, 10 20, 10 10))
```

Exemple de WKT