

Développer une véritable application Open Data avec Processing

Conditions	Travail en groupe Durée : 2x2h	Moyens	<ul style="list-style-type: none"> • Poste informatique sous Windows • Connexion internet • Processing 1.5 • Notepad++
Prérequis	Etre initié à Processing	Niveau	Classe de Terminale S spécialité ISN
Compétences	<ul style="list-style-type: none"> • Compétences C2 : Concevoir et réaliser une solution informatique en réponse à un problème 		
Eléments du programme	<ul style="list-style-type: none"> • Représentation de l'information <ul style="list-style-type: none"> ○ Formats ○ Structuration et organisation de l'information • Algorithmes <ul style="list-style-type: none"> ○ Algorithmes simples (Rechercher un élément dans un tableau) • Langages et programmation <ul style="list-style-type: none"> ○ Types de données ○ fonctions 		

La Ville de Nantes, Nantes Métropole et leurs partenaires ont lancé dernièrement un appel à projets innovants "Rendez-moi la ville + facile". Les collectivités ont invité chacun à réutiliser les données publiques pour inventer de nouveaux services. L'appel à projets avait pour but de mettre en avant la créativité et l'ingéniosité de tous : citoyens, entreprises, start-up, étudiants, développeurs, créatifs, associations, collectifs... Cet appel à projets Open Data a suscité la créativité et a permis de faire émerger de nombreux projets de qualités dont le prix du jury : [Statiophone](#)



[Statiophone](#) est une application mobile iPhone permettant de connaître en temps réel les places disponibles dans les parkings publics de Nantes. Simple et facile, l'application vous géolocalise, affiche et dicte les parkings les plus proches de vous.

On se propose dans cette activité, de réaliser une application du même type (sans géolocalisation) à l'aide de Processing 1.5.

Comment afficher une carte de Nantes ? Comment récupérer les coordonnées géographiques des différents parkings de Nantes ? Comment localiser les parkings sur la carte ? Comment rendre l'application interactive ?

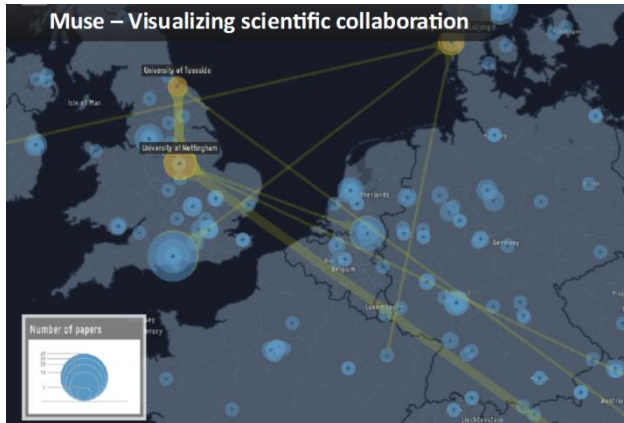
Table des matières :

1. Afficher la carte de Nantes	2
2. Récupérer les coordonnées géographiques d'un parking.....	5
3. Localiser un parking sur la carte	6
4. Rendre l'application interactive.....	7
5. Réalisation de l'application finale.....	7

1. Afficher la carte de Nantes

La cartographie avec Processing

Pour afficher et travailler sur des cartes dans Processing, il existe plusieurs méthodes : soit l'utilisation d'un logiciel de création de carte [TileMill](#), soit l'utilisation de la librairie [Unfolding](#). Nous allons travailler sur la deuxième solution car cette librairie est très utilisée à l'heure actuelle dans de nombreuses applications :



Carte présentant la collaboration entre les scientifiques (par rapport au nombre d'articles scientifiques publiés)



Carte présentant la localisation des crimes et le type de crimes commis dans une ville durant une période donnée.

Installation de la librairie Unfolding dans Processing

Télécharger la dernière version de la [bibliothèque Unfolding](#). Décompresser l'archive et copier le dossier extrait dans le dossier des librairies de Processing

Remarques : Pour trouver l'emplacement du dossier Processing sur votre ordinateur, ouvrez la fenêtre Préférences de l'application Processing et identifiez le « Sketchbook location ». Vous aurez besoin de créer le dossier « librairies » si c'est la première librairie que vous installez. Redémarrer Processing.

Attention : les cartes sont récupérées via internet, il faut donc être connecté.

Premier programme Processing avec Unfolding

Pour commencer, il faut importer les différentes librairies dans le code :

```
import processing.opengl.*;
import de.fhpotsdam.unfolding.*;
import de.fhpotsdam.unfolding.geo.*;
import de.fhpotsdam.unfolding.utils.*;
```

Créer un objet carte :

```
UnfoldingMap carte ;
```

Dans `setup()`, spécifier la taille de la fenêtre avec `size()`, puis initialiser le nouvel objet “carte” et ajouter les fonctions par défaut (double-clique, zoom, glissé, ...).

```
void setup() {
  size(800, 600);
  carte = new UnfoldingMap(this);
  MapUtils.createDefaultEventDispatcher(this, carte);
}
```

Enfin dans le `draw()`, il faut commander l’affichage de la carte :

```
void draw() {
  carte.draw();
}
```

1. Coder ce programme dans processing et reporter le résultat :

Résultat :



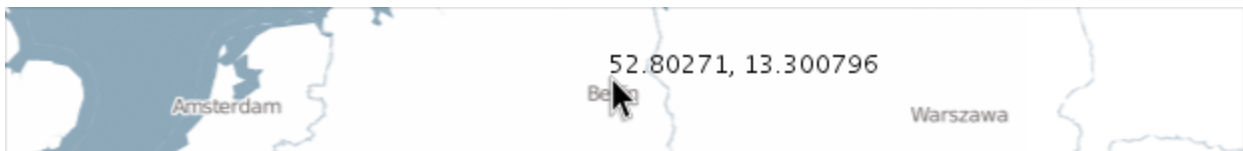
Voir `test1_unfolding.pde`

Les fonctions de la librairie Unfolding

Récupérer la géolocalisation d’un point sur la carte

On peut facilement convertir la position du curseur de la souris sur l’écran en coordonnées géographiques avec la fonction `getLocation()` :

```
void draw() {
  map.draw();
  Location location = map.getLocation(mouseX, mouseY);
  fill(0);
  text(location.getLat() + ", " + location.getLon(), mouseX, mouseY);
}
```



Modifier le style de carte

On peut modifier le style de carte en ajoutant un second paramètre lors de sa création :

```
map = new UnfoldingMap(this, new Microsoft.AerialProvider());
```

Il est nécessaire d'importer au préalable la librairie provider.

```
Import de.fhpotsdam.unfolding.providers.*;
```

Ci-dessous les différents styles de carte :



Unfolding Default



Google Terrain



Microsoft Aerial



Midnight Commander
(CloudMade)



Watercolor
(Stamen)



Buildings
(OpenStreetMap + TileMill)

Zoomer et glisser

Les utilisateurs peuvent interagir avec la carte. Ils peuvent déplacer la carte en la faisant glisser avec la souris, ou en utilisant les touches fléchées du clavier. Il est possible de zoomer en utilisant la molette de la souris ou encore en appuyant sur les touches + ou - du clavier.

Pour délimiter la visualisation sur zone géographique, il est possible de paramétrer le niveau de zoom et la zone de déplacement.

Ici, nous avons un zoom de Berlin permettant de visualiser l'ensemble de la ville.

```
map.zoomAndPanTo(new Location(52.5f, 13.4f), 10);
```

Pour limiter le glissé de l'utilisateur à 30 km autour d'un point :

```
Location berlinLocation = new Location(52.5f, 13.4f);
map.zoomAndPanTo(berlinLocation, 10);
float maxPanningDistance = 30; // en km
map.setPanningRestriction(berlinLocation, maxPanningDistance);
```

La carte de Nantes

2. Trouver sur le web les coordonnées géographiques de la ville de Nantes

Nantes est située à 47.2172 de latitude et -1.591 de longitude

3. Réaliser un programme sous Processing permettant d'afficher la carte de Nantes (vue satellite) avec un glisser maximum de 10km.

Voir [nantes_unfolding.pde](#)

2. Récupérer les coordonnées géographiques d'un parking

Lors de l'activité précédente, l'analyse des données XML renvoyées par l'API de Nantes montre que les données de géolocalisation des parkings ne sont pas fournies :

```
<Groupe_Parking>
  <Grp_identifiant>3</Grp_identifiant>
  <Grp_nom>TOUR DE BRETAGNE</Grp_nom>
  <Grp_statut>5</Grp_statut>
  <Grp_pri_aut>0</Grp_pri_aut>
  <Grp_disponible>268</Grp_disponible>
  <Grp_complet>20</Grp_complet>
  <Grp_exploitation>645</Grp_exploitation>
  <Grp_horodatage>04/02/2013 23:24:40</Grp_horodatage>
  <IdObj>299</IdObj>
</Groupe_Parking>
```

Cependant, chaque parking possède un identifiant <IdObj> au même titre que tous les équipements de la ville de Nantes. C'est grâce à cet identifiant que l'on peut retrouver les données de géolocalisation contenues dans un fichier CSV disponible sur le service Open Data de Nantes, le fichier « Equipements_publics_deplacement.csv ».

4. Récupérer le fichier « Equipements_publics_deplacement.csv » sur le site Open Data de Nantes.
5. Ouvrir le fichier avec notepad++ et analyser son contenu et sa structure.
6. Réaliser un programme Processing afin d'extraire du fichier csv les données de géolocalisation du parking « TOUR DE BRETAGNE » portant l'identifiant « 299 ». Les données doivent être placées dans deux variables lat et lon de type chaîne de caractères. L'algorithme est le suivant :

DEBUT

Extraire chaque ligne du fichier csv et placer la chaîne de caractères correspondante dans un tableau.

Pour i de 0 à nombre de lignes du fichier (c'est-à-dire la taille du tableau) **Faire**

 Décomposer la ligne en chaînes de caractère suivant le séparateur utilisé dans le fichier csv (caractère ;).

 Mettre en forme la chaîne de caractère correspondant au IdObj (exemple : «299,000 » => « 299 »)

Si IdObj du CSV = IdObj du XML **alors**

```
Récupérer la chaîne de caractère correspondant à la latitude.
Récupérer la chaîne de caractère correspondant à la longitude.
Afficher la variable lat et la variable lon
```

```
Fin du Si
Fin du Pour
FIN
```

Exemple de correction (voir lat_lon_csv.pde)

3. Localiser un parking sur la carte

L'objectif est maintenant d'afficher sur la carte l'emplacement du parking signalé par un petit icône.

Les étapes à suivre sont les suivantes :

1. Récupérer les données XML du parking Tour de Bretagne
2. Récupérer les coordonnées géographiques du parking à partir du fichier csv et de l'identifiant IdObj (données XML)
3. Afficher la carte de Nantes.
4. Placer l'icône parking sur la carte

Pour convertir les coordonnées géographiques du parking en coordonnées sur l'écran et afficher l'icône à cet emplacement, il faut utiliser le code suivant :

```
int parkingPictoTaille=20;
float xy[];

Location position = new Location(lat, lon);
// Mets à jour la position du parking sur la carte (en pixels)
xy = carte.getScreenPositionFromLocation(position);
// Affichage du picto pour ce parking
image(datanantesPictoParking, xy[0]-parkingPictoTaille/2, xy[1]-
parkingPictoTaille/2, parkingPictoTaille, parkingPictoTaille);
```

Cependant la fonction location() utilise les paramètres lat (latitude) et lon (longitude) de type float. Or dans le programme précédant les variables lat et lon sont de type String. Il faut donc convertir les variables lat et lon précédentes en variables lat et lon de type float :

⇒ L'algorithme est le suivant :

```
Décomposer la chaîne de caractère correspondant à la latitude
suivant le séparateur « , » (exemple : 47,215 => 47 et 215).
```

```
Recomposer la chaîne de caractère avec le séparateur « . »
(exemple : 47 et 125 => 47.215).
```

```
Convertir la chaîne de caractère en variable de type float
```

⇒ Le code est le suivant :

```
String[] split_lat = split(element[15], ',');
String joine_lat = join(split_lat, ".");
float lat=float(joine_lat);
String[] split_lon = split(element[14], ',');
```

```
String joine_lon = join(split_lon, ".");  
float lon=float(joine_lon);
```

7. Réaliser le programme complet en utilisant pour l'icône le fichier image fourni

[Voir Parking_TourDeBretagne_Carte.pde](#)

4. Rendre l'application interactive

L'objectif est d'afficher la disponibilité du parking Tour de Bretagne lorsque l'utilisateur clique sur l'icône parking au niveau de la carte.

8. Réaliser le programme

[Voir Parking_TourDeBretagne_Carte_interactive.pde](#)

5. Réalisation de l'application finale

L'objectif est d'afficher tous les parkings de Nantes sur la carte. Ainsi que le nombre de places disponibles lorsque l'utilisateur clique sur l'icône d'un parking.

9. Réaliser le programme

[Voir Parkings_Nantes_Carte_interactive.pde](#)

[Voir datanantes_Parkings_Carte_interractive.pde](#) avec programmation orienté objet utilisant la notion de classe