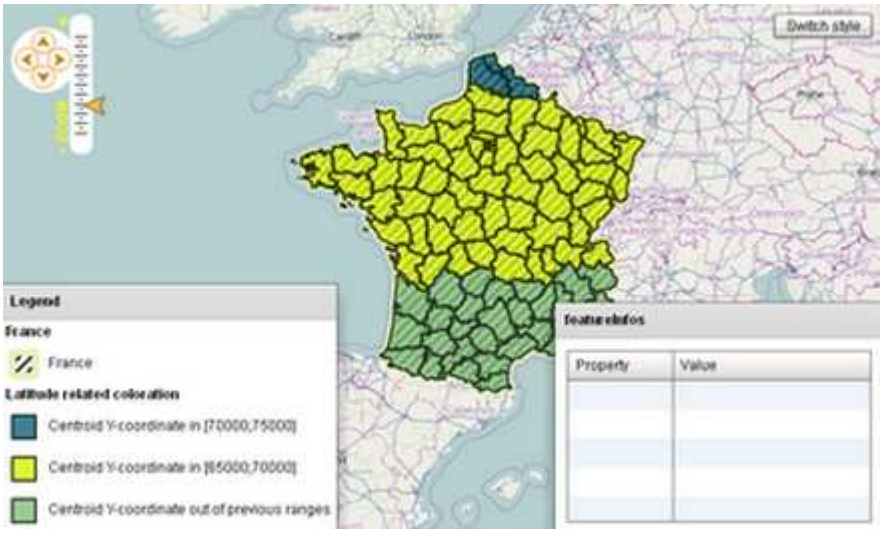
7 OpenScales

OpenScales est un framework cartographique open source écrit en ActionScript 3/Flex permettant aux développeurs de créer des **Rich Internet Mapping Application**. Dans ce tutoriel, nous allons mettre en œuvre le framework à travers une série d'exemples.

Pour commencer, télécharger **OpenScales 1.2** à l'adresse ci-dessous, puis copier les fichiers *swc* dans le dossier *libs* du projet Flash Builder.

```
http://openscales.org/downloads/index.html
```



Documentation:

```
http://openscales.org/documentation/index.html
```

Application de démonstration :

```
http://openscales.org/demo/index.html
```

L'archive téléchargée contient également les sources des exemples du démonstrateur.

8 Premiers pas

8.1 Basic OpenStreetMap

Pour commencer, nous allons créer une carte basée sur la couche Mapnik OpenStreetMap. Pour ce faire, créer un objet Map dans lequel on ajoute une couche Mapnik (on parle de « layer »). Ces composants proviennent de la librairie OpenScales et il est nécessaire d'ajouter le namespace http://openscales.org.

BasicOsm.mxml

```
<?xml version="1.0" encoding="utf-8"?>
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
               xmlns:s="library://ns.adobe.com/flex/spark"
               xmlns:mx="library://ns.adobe.com/flex/mx"
               minWidth="955" minHeight="600"
               xmlns:os="http://openscales.org">
    <os:Map
        id="fxmap"
        width="100%"
        height="100%">
        <os:Mapnik
            name="Mapnik"
            proxy="http://openscales.org/proxy.php?url="/>
    </os:Map>
</s:Application>
```

8.1.1 Résultat

L'application affiche une carte figée dans laquelle il n'est pas possible de se déplacer/zoomer :

Le composant de couche Mapnik permet de charger des tuiles cartographiques prégénérées sur toute une zone à différents niveaux de zoom depuis un cache serveur.

8.2 MouseControls, MousePosition et PanZoom

Nous allons maintenant ajouter des ***MouseControls*** permettant de naviguer dans la carte, à savoir:

- Zoomer à l'aide de la molette de la souris (WheelHandler)
- Déplacer la carte à l'aide du Drag and Drop. (DragHandler)

```
<os:WheelHandler/>
<os:DragHandler/>
```

Nous allons également ajouter un composant ***MousePosition*** permettant d'afficher les coordonnées géographiques correspondant à la position de la souris.

```
<os:MousePosition
    x="10"
    y="{fxmap.height-20}"
    displayProjection="EPSG:4326"/>
```

Pour en savoir plus sur les codes EPSG :

```
http://fr.wikipedia.org/wiki/Syst%C3%A8me_de_coordonn%C3%A9es_g%C3%A9or%C
3%A9f%C3%A9renc%C3%A9es#Les_codes_EPSG
```

Puis nous allons ajouter l'outil ***PanZoom*** à la carte. Pour cela, il est nécessaire d'ajouter du code ActionScript permettant de récupérer l'instance associée au composant MXML Map et de la stocker dans une variable « bindée ». Cette méthode `initMap()` est appelée à l'initialisation de l'application.

```
<fx:Script>
    <![CDATA[
        import org.openscales.core.Map;
        [Bindable] private var map:Map = null;
        private function initMap():void
        {
            map = fxmap.map;
        }
    ]]>
</fx:Script>
```

Puis il faut créer un composant `PanZoom` basé sur l'instance `map` (variable « bindée » créée précédemment).

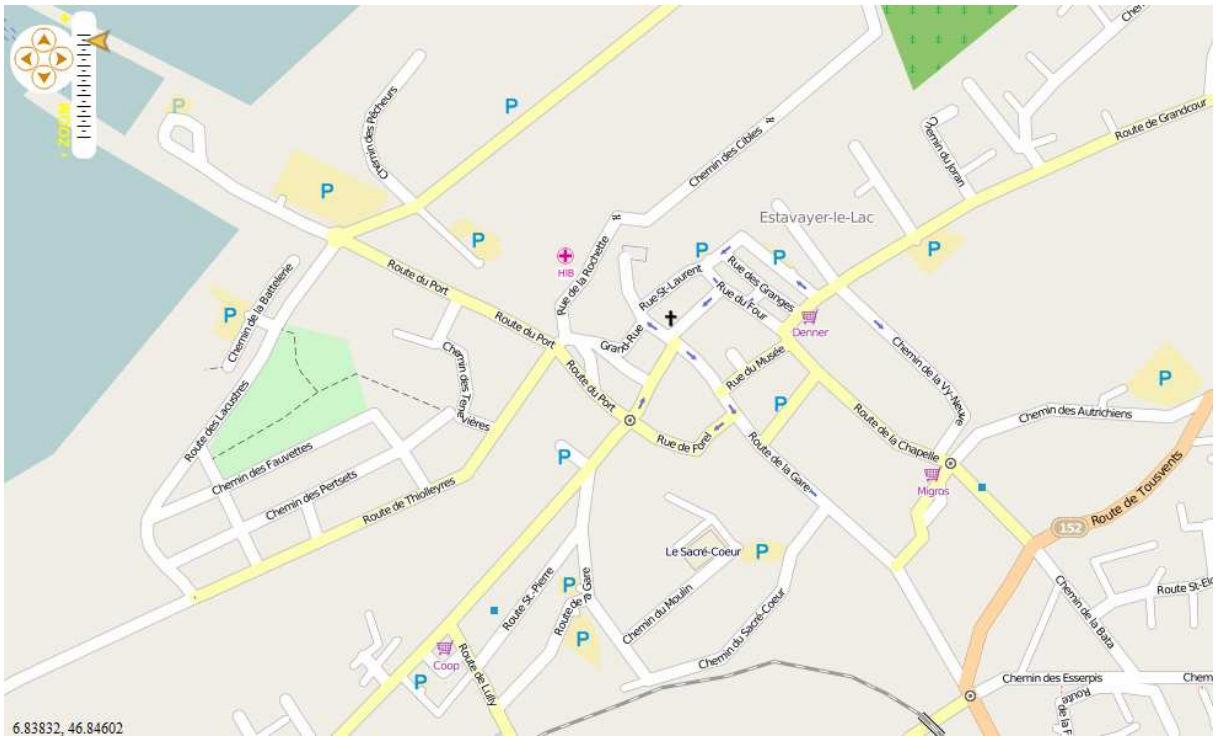
```
<os:PanZoom
    map="{map}"
    x="{fxmap.x+10}"
    y="{fxmap.y+10}"/>
```

BasicOsm_Controls.mxml

```
<?xml version="1.0" encoding="utf-8"?>
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
               xmlns:s="library://ns.adobe.com/flex/spark"
               xmlns:mx="library://ns.adobe.com/flex/mx"
               minWidth="955" minHeight="600"
               xmlns:os="http://openscales.org"
               creationComplete="initMap()">
    <os:Map
        id="fxmap"
        width="100%"
        height="100%">

        <os:Mapnik
            name="Mapnik"
            proxy="http://openscales.org/proxy.php?url="/>

        <!-- Mouse controls -->
        <os:DragHandler/>
        <os:WheelHandler/>

        <os:MousePosition
            x="10"
            y="{fxmap.height-20}"
            displayProjection="EPSG:4326"/>

    </os:Map>

    <os:PanZoom
        map="{map}"
        x="{fxmap.x+10}"
        y="{fxmap.y+10}"/>

    <fx:Script>
        <![CDATA[
            import org.openscales.core.Map;
            [Bindable] private var map:Map = null;
            private function initMap():void
            {
                map = fxmap.map;
            }
        ]]>
    </fx:Script>

</s:Application>
```

8.2.1 Résultat

Il est maintenant possible de se déplacer sur la carte à l'aide de la souris ou directement depuis l'outil de navigation situé en haut à gauche. Quant aux coordonnées géographiques de la position de la souris, elles sont affichées en bas à gauche de la carte.



8.3 OpenStreetMap centré sur Yverdon

Nous allons maintenant centrer la carte sur Yverdon. Pour ce faire, ajouter les propriétés **center** et **zoom** à notre composant Map :

- **center** correspond à la coordonnée géographique du centre de la carte.
- **zoom** correspond au niveau de zoom à utiliser

BasicOsm_Controls_Center.mxml

```
<?xml version="1.0" encoding="utf-8"?>
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
               xmlns:s="library://ns.adobe.com/flex/spark"
               xmlns:mx="library://ns.adobe.com/flex/mx"
               minWidth="955" minHeight="600"
               xmlns:os="http://openscales.org"
               creationComplete="initMap()">
    <os:Map
        id="fxmap"
        width="100%"
        height="100%"
        center="6.64263,46.79094"
        zoom="15">
        <os:Mapnik
            name="Mapnik"
            proxy="http://openscales.org/proxy.php?url="/>

        <!-- Mouse controls -->
        <os:DragHandler/>
        <os:WheelHandler/>

        <os:MousePosition
            x="10"
            y="{fxmap.height-20}"
            displayProjection="EPSG:4326"/>

    </os:Map>

    <os:PanZoom
        map="{map}"
        x="{fxmap.x+10}"
        y="{fxmap.y+10}"/>

    <fx:Script>
        <![CDATA[
            import org.openscales.core.Map;
            [Bindable] private var map:Map = null;
            private function initMap():void
            {
                map = fxmap.map;
            }
        ]]>
    </fx:Script>

</s:Application>
```

8.3.1 Résultat

La carte est maintenant centrée sur Yverdon.

8.4 Composant WMS

Cet exemple met en œuvre l'utilisation du composant WMS qui permet de charger une couche depuis un serveur cartographique conforme OGC WMS. Pour ce faire, nous allons faire appel à un tel service conforme du SITN (Système d'Information du Territoire Neuchâtelois).

Pour en savoir plus sur WMS (Web Map Service):

```
http://fr.wikipedia.org/wiki/Web_Map_Service
```

Service WMS ImageOne2006 du SITN :

```
http://www.ne.ch/neat/site/jsp/rubrique/rubrique.jsp?StyleType=bleu&DocId
=33442
```

Le composant WMS permet de préciser l'url d'un tel service, la couche à charger, ainsi que le format d'image demandé :

```
<os:WMS
    name="SITN WMS - Ortho layer"
    url="http://sitn.ne.ch/ogc-sitn-open/wms"
    layers="ortho"
    format="image/jpeg"
    proxy="http://localhost/proxy.php?url="/>
```

BasicWms.mxml

```
<?xml version="1.0" encoding="utf-8"?>
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
               xmlns:s="library://ns.adobe.com/flex/spark"
               xmlns:mx="library://ns.adobe.com/flex/mx"
               xmlns:os="http://openscales.org"
               minWidth="955" minHeight="600">

    <os:Map
        id="fxmap"
        width="100%"
        height="100%">

<os:WMS
    name="SITN WMS - Ortho layer"
    url="http://sitn.ne.ch/ogc-sitn-open/wms"
    layers="ortho"
    format="image/jpeg"
    proxy="http://localhost/proxy.php?url="/>

        <os:Extent
            west="6.30" south="46.80"
            east="7.13" north="47.20"
            projection="EPSG:4326" />

        <os:MousePosition
            x="10"
            y="{fxmap.height-20}"
            displayProjection="EPSG:4326"/>

        <os:DragHandler/>
        <os:WheelHandler/>

    </os:Map>
</s:Application>
```

L'utilisation d'un tel service requiert de préciser la zone cartographique demandée et ce par la définition d'une enveloppe géographique (Extent, avec les valeurs west, south, east, north).

8.4.1 Résultat

L'application affiche une orthophoto du canton de Neuchâtel :

8.5 CurrentExtent

L'intérêt d'un tel framework est de pouvoir définir des interactions cartographiques personnalisées. Ainsi il est possible de gérer des événements spécifiques (MapEvent, LayerEvent).

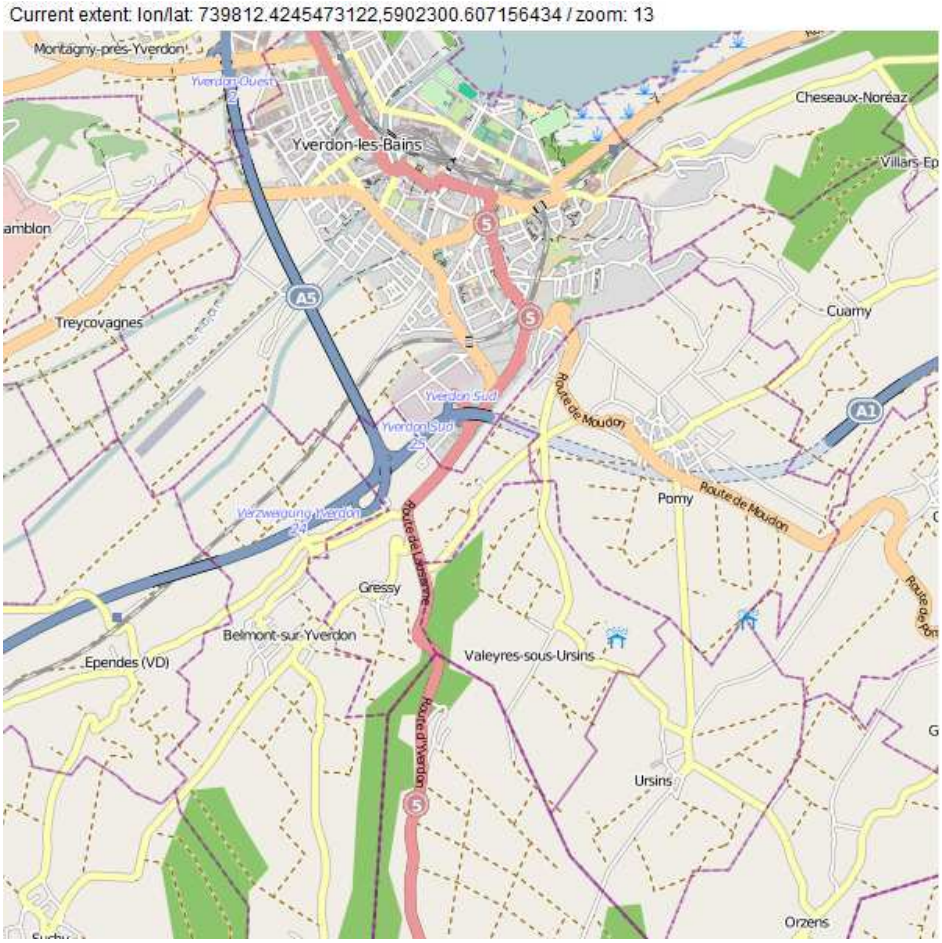
Le code ci-dessous ajoute un listener sur un MapEvent.MOVE_END (fin de déplacement de carte). Il s'agit à chaque déplacement d'afficher les nouvelles caractéristiques de la vue cartographique (map.center et map.zoom).

CurrentExtent.mxml

```
<?xml version="1.0" encoding="utf-8"?>
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
               xmlns:s="library://ns.adobe.com/flex/spark"
               xmlns:mx="library://ns.adobe.com/flex/mx"
               xmlns:os="http://openscales.org"
               minWidth="955" minHeight="600"
               creationComplete="initMap()">

    <fx:Script>
        <![CDATA[
            import org.openscales.core.Map;
            import org.openscales.core.events.MapEvent;

            [Bindable] private var map:Map = null;
            [Bindable] private var currentExtent:String;

            private function initMap():void {
                map = fxmap.map;
                getCurrentExtent();
                map.addEventListener(MapEvent.MOVE_END,
                                     getCurrentExtent);
            }

            private function getCurrentExtent(event:MapEvent = null):void
            {
                currentExtent = "lon/lat: "+map.center.lon+","
                               +map.center.lat+" / zoom: "+map.zoom;
            }

        ]]>
    </fx:Script>

    <os:Map id="fxmap"
            width="600" height="600"
            y="20"
            zoom="9" center="6.8,46.9">
        <os:Mapnik name="base"/>
        <os:MousePosition x="10" y="{fxmap.height-20}"
                          displayProjection="EPSG:900913"/>
        <os:DragHandler/>
        <os:WheelHandler/>
    </os:Map>

    <s:Label x="5" y="5" text="Current extent: {currentExtent}"/>

</s:Application>
```

8.5.1 Résultat

Le centre de la map ainsi que le niveau de zoom sont affichés après chaque déplacement.



8.6 Exercice

Mettre en place une application cartographique qui enregistre les différentes vues durant la navigation. Les caractéristiques des vues sont historisées dans une DataGrid. L'utilisateur peut revenir à tout moment à une vue précédente de l'historique par sélection dans la DataGrid.

9 Superpositions (overlay)

9.1 Basic WMS overlay

Cet exemple met en œuvre l'utilisation d'une couche WMS affichée par-dessus la couche de base Mapnik. Pour ce faire, nous allons faire appel au service WMS du serveur OGO.

- Url du service : `http://ogo.heig-vd.ch/geoserver/wms`
- Couche : `ogo:g4districts98`

Etant donné que la couche de base Mapnik utilise une projection EPSG:900913, il faut spécifier explicitement que l'on désire obtenir la couche WMS dans cette même projection. Cette couche s'ajoute donc à la carte comme suit :

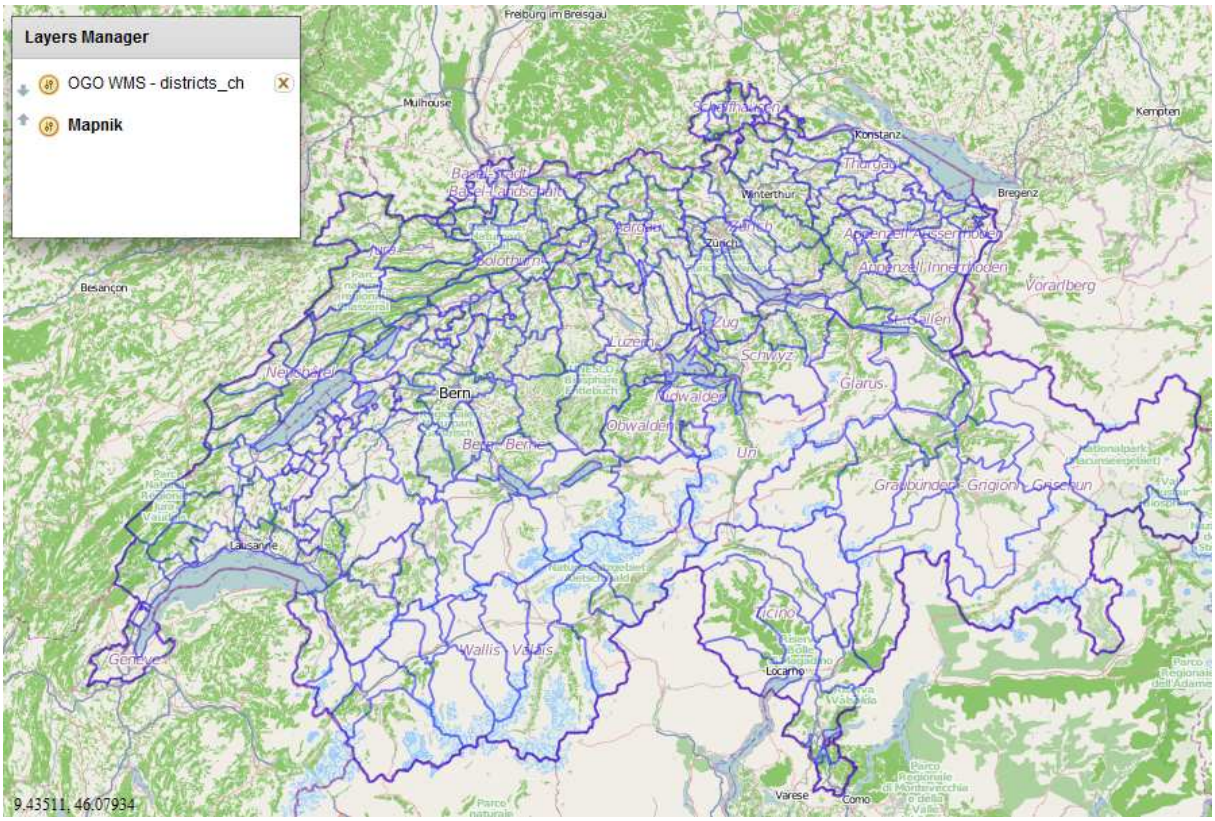
```
<os:WMS
    name="OGO WMS - districts_ch" alpha="0.5"
    url="http://ogo.heig-vd.ch/geoserver/wms"
    layers="ogo:districts_ch" projection="EPSG:900913"
    transparent="true" format="image/png"  styles="line"/>
```

Wms_Overlay.mxml

```
<?xml version="1.0" encoding="utf-8"?>
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
               xmlns:s="library://ns.adobe.com/flex/spark"
               xmlns:mx="library://ns.adobe.com/flex/mx"
               xmlns:os="http://openscales.org"
               minWidth="955" minHeight="600">

    <os:Map
        id="fxmap"
        width="100%"
        height="100%"
        center="8.32,46.9"
        zoom="8">

        <os:Mapnik
            name="Mapnik"
            proxy="http://openscales.org/proxy.php?url="/>

        <os:WMS
            name="OGO WMS - districts_ch" alpha="0.5"
            url="http://ogo.heig-vd.ch/geoserver/wms"
            layers="ogo:districts_ch" projection="EPSG:900913"
            transparent="true" format="image/png" styles="line"/>

        <os:MousePosition
            x="10"
            y="{fxmap.height-20}"
            displayProjection="EPSG:4326"/>

        <os:DragHandler/>
        <os:WheelHandler/>

        <os:ControlPanel
            title="Layers Manager"
            x="10" y="10"
```

```
            width="215">
            <os:LayerManager/>
        </os:ControlPanel>
    </os:Map>
</s:Application>
```

9.1.1 Résultat



L'exemple `Wms_Overlay_Mapnik_Sitn_Ogo.mxml` présent dans les sources met en œuvre l'utilisation de trois couches provenants de différents services.

9.2 Superposition d'objets géographiques – point

Il est possible d'ajouter des objets géographiques points, lignes et polygones à notre carte (on parle de Features). Tout objet est caractérisé par une géométrie et des attributs.

9.2.1 Création de la couche d'objets

Pour ce faire, il est nécessaire de créer une nouvelle couche :

```
var featureLayer:FeatureLayer = new FeatureLayer("MyFeatures");
featureLayer.projection = new ProjProjection("EPSG:4326");
featureLayer.style = Style.getDefaultPointStyle();
```

L'ajout d'un point peut se faire de plusieurs manières:

- L'ajout d'un `PointFeature`, dont la symbologie est celle définie par défaut sur la couche (`Style.getDefaultPointStyle`).

```
var point:PointFeature = PointFeature.createPointFeature(
                                    new Location(6.65591, 46.78566));
featureLayer.addFeature(point);
```

- L'ajout d'un `Marker`, dont la symbologie est celle définie par défaut pour les marqueurs.

```
var marker:Marker = new Marker(
                new org.openscales.geometry.Point(6.64282, 46.79092));
featureLayer.addFeature(marker);
```

- L'ajout d'un `CustomMarker`, dont la symbologie peut être personnalisée.

```
var marker_2:PointFeature =
CustomMarker.createUrlBasedMarker("http://ogo.heig-
                vd.ch/ria/images/add_placemark.png",
                new Location(6.64922, 46.78637),
                {featureName:"Custom marker"});
featureLayer.addFeature(marker_2);
```

PointsAndMarkers_Overlay.mxml

```
<?xml version="1.0" encoding="utf-8"?>
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
               xmlns:s="library://ns.adobe.com/flex/spark"
               xmlns:mx="library://ns.adobe.com/flex/mx"
               xmlns:os="http://openscales.org"
               minWidth="955" minHeight="600"
               creationComplete="initMap()">
    <os:Map
        id="fxmap"
        width="800"
        height="600"
        zoom="15"
        center="6.64282, 46.79092"
        x="100"
        y="100">
        <os:Mapnik
            name="base"
            proxy="http://openscales.org/proxy.php?url="/>
        <os:MousePosition
            x="10"
            y="{fxmap.height-20}"
            displayProjection="EPSG:4326"/>
        <os:DragHandler/>
        <os:WheelHandler/>
    </os:Map>

    <os:PanZoom
        map="{map}"
        x="{fxmap.x+10}"
        y="{fxmap.y+10}"/>

    <fx:Script>
        <![CDATA[
            import org.openscales.core.Map;
            import org.openscales.core.feature.CustomMarker;
            import org.openscales.core.feature.Marker;
            import org.openscales.core.feature.PointFeature;
            import org.openscales.core.layer.FeatureLayer;
            import org.openscales.core.style.Style;
            import org.openscales.geometry.Point;
            import org.openscales.geometry.basetypes.Location;
            import org.openscales.proj4as.ProjProjection;

            [Bindable] private var map:Map = null;

            private function initMap():void{
                map = fxmap.map;

                var featureLayer:FeatureLayer = new
                            FeatureLayer("PointsFeatures");
                featureLayer.projection = new
                            ProjProjection("EPSG:4326");
                featureLayer.style = Style.getDefaultPointStyle();
                map.addLayer(featureLayer);

                // Ajout du premier PointFeature
                var point_1:PointFeature =
                      PointFeature.createPointFeature(
                                  new Location(6.65591, 46.78566),
                                  {featureName:"Point 1"});
                featureLayer.addFeature(point_1);
```

```
                // Ajout du deuxième PointFeature
                var point_2:PointFeature =
                                PointFeature.createPointFeature(
                                        new Location(6.64046, 46.79113),
                                        {featureName:"Point 2"});
                featureLayer.addFeature(point_2);

                // Ajout d'un marqueur
                var marker_1:Marker = new Marker(
                new org.openscales.geometry.Point(6.64282, 46.79092),
                                {featureName:"Default marker"});
                featureLayer.addFeature(marker_1);

                // Ajout d'un marqueur personnalisé
                var marker_2:PointFeature =
                        CustomMarker.createUrlBasedMarker("http://ogo.heig-
vd.ch/ria/images/add_placemark.png",
                                new Location(6.64922, 46.78637),
                                {featureName:"Custom marker"});
                featureLayer.addFeature(marker_2);
            }
        ]]>
    </fx:Script>
</s:Application>
```

9.2.2 Résultat

9.3 Superposition d'objets géographiques – lignes

Il est possible d'ajouter des lignes à notre carte. Pour ce faire, il est nécessaire de créer une nouvelle couche :

```
var featureLayer:FeatureLayer = new FeatureLayer("LineFeatures");
featureLayer.projection = new ProjProjection("EPSG:4326");
featureLayer.style = Style.getDefaultLineStyle();
```

Les coordonnées de la ligne doivent être composées dans un vecteur de la manière suivante :

```
var vector_1:Vector.<Number> = new Vector.<Number>;
vector_1.push(6.65591);       //P1[x]
vector_1.push(46.78566);      //P1[y]
vector_1.push(6.64922);       //P2[x]
vector_1.push(46.78637);      //P2[y]
vector_1.push(6.64046);       //P3[x]
vector_1.push(46.79113);      //P3[y]
vector_1.push(6.64282);       //P4[x]
vector_1.push(46.79092);      //P4[y]
```

La création d'un objet de type ligne (`LineStringFeature`) se fait au travers d'une géométrie (`LineString`) à partir du vecteur créé ci-dessus.

```
var line_1:LineString = new LineString(vector_1);
var lineFeature:LineStringFeature = new LineStringFeature(line_1,
                                                {featureName:"Line 1"});
featureLayer.addFeature(lineFeature);
```

LineString_Overlay.mxml

```
<?xml version="1.0" encoding="utf-8"?>
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
               xmlns:s="library://ns.adobe.com/flex/spark"
               xmlns:mx="library://ns.adobe.com/flex/mx"
               xmlns:os="http://openscales.org"
               minWidth="955" minHeight="600"
               creationComplete="initMap()">
    <os:Map
        id="fxmap"
        width="800"
        height="600"
        zoom="15"
        center="6.64282, 46.79092"
        x="100"
        y="100">
        <os:Mapnik
            name="base"/>
        <os:MousePosition
            x="10"
            y="{fxmap.height-20}"
            displayProjection="EPSG:4326"/>
        <os:DragHandler/>
        <os:WheelHandler/>
    </os:Map>
```

```
    <os:PanZoom
        map="{map}"
        x="{fxmap.x+10}"
        y="{fxmap.y+10}"/>

    <fx:Script>
        <![CDATA[
            import org.openscales.core.Map;
            import org.openscales.core.feature.LineStringFeature;
            import org.openscales.core.layer.FeatureLayer;
            import org.openscales.core.popup.Anchored;
            import org.openscales.core.style.Style;
            import org.openscales.geometry.LineString;
            import org.openscales.proj4as.ProjProjection;

            [Bindable] private var map:Map = null;

            private var currentPopup:Anchored;

            private function initMap():void{
                map = fxmap.map;
                var featureLayer:FeatureLayer = new
                                    FeatureLayer("MyFeatures");
                featureLayer.projection = new
                                    ProjProjection("EPSG:4326");
                featureLayer.style = Style.getDefaultLineStyle();

                // Création du vecteur de la LineString
                var vector_1:Vector.<Number>=new Vector.<Number>;
                vector_1.push(6.65591);    //P1[x]
                vector_1.push(46.78566);   //P1[y]
                vector_1.push(6.64922);    //P2[x]
                vector_1.push(46.78637);   //P2[y]
                vector_1.push(6.64282);    //P2[x]
                vector_1.push(46.79092);   //P2[y]
                vector_1.push(6.64046);    //P3[x]
                vector_1.push(46.79113);   //P4[y]

                // Création de la LineString
                var line_1:LineString = new LineString(vector_1);
                var lineFeature:LineStringFeature = new
                       LineStringFeature(line_1, {featureName:"Line 1"});
                featureLayer.addFeature(lineFeature);

                map.addLayer(featureLayer);
            }
        ]]>
    </fx:Script>
</s:Application>
```

9.3.1 Résultat

9.4 Superposition d'objets géographiques – polygones

Il est possible d'ajouter des polygones à notre carte. Pour ce faire, il est nécessaire de créer une nouvelle couche :

```
var featureLayer:FeatureLayer = new FeatureLayer("PolygonFeatures");
featureLayer.projection = new ProjProjection("EPSG:4326");
featureLayer.style = Style.getDefaultSurfaceStyle();
```

Les coordonnées du polygone sont à entrer dans un vecteur de la manière suivante :

```
var vector_1:Vector.<Number> = new Vector.<Number>;
vector_1.push(6.65591);       //P1[x]
vector_1.push(46.78566);      //P1[y]
vector_1.push(6.64922);       //P2[x]
vector_1.push(46.78637);      //P2[y]
vector_1.push(6.64046);       //P3[x]
vector_1.push(46.79113);      //P3[y]
vector_1.push(6.64282);       //P4[x]
vector_1.push(46.79092);      //P4[y]
```

La création d'un objet de type polygone (`PolygonFeature`) se fait au travers d'un vecteur de géométries (`LinearRing`) à partir du vecteur créé ci-dessus. Une géométrie `LinearRing` est une `LineString` spéciale qui est automatiquement « fermée ».

```
var ring_1:LinearRing = new LinearRing(vector_1);
var geom:Vector.<Geometry> = new Vector.<Geometry>;
geom.push(ring_1);
var polygon:Polygon = new Polygon(geom);
var polygonFeature:PolygonFeature = new PolygonFeature(polygon,
                                          {featureName:"MyPolygon"});
featureLayer.addFeature(polygonFeature);
```

Polygon_Overlay.mxml

```
<?xml version="1.0" encoding="utf-8"?>
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
               xmlns:s="library://ns.adobe.com/flex/spark"
               xmlns:mx="library://ns.adobe.com/flex/mx"
               xmlns:os="http://openscales.org"
               minWidth="955" minHeight="600"
               creationComplete="initMap()">
    <os:Map
        id="fxmap"
        width="800"
        height="600"
        zoom="15"
        center="6.64282, 46.79092"
        x="100"
        y="100">
        <os:Mapnik
            name="base"/>
        <os:MousePosition
            x="10"
            y="{fxmap.height-20}"
            displayProjection="EPSG:4326"/>
```

```
        <os:DragHandler/>
        <os:WheelHandler/>
    </os:Map>

    <os:PanZoom
        map="{map}"
        x="{fxmap.x+10}"
        y="{fxmap.y+10}"/>

    <fx:Script>
        <![CDATA[
            import org.openscales.core.Map;
            import org.openscales.core.feature.PolygonFeature;
            import org.openscales.core.layer.FeatureLayer;
            import org.openscales.core.popup.Anchored;
            import org.openscales.core.style.Style;
            import org.openscales.geometry.Geometry;
            import org.openscales.geometry.LinearRing;
            import org.openscales.geometry.Polygon;
            import org.openscales.proj4as.ProjProjection;

            [Bindable] private var map:Map = null;

            private var currentPopup:Anchored;

            private function initMap():void
            {
                map = fxmap.map;

                var featureLayer:FeatureLayer = new
                            FeatureLayer("PolygonFeatures");
                featureLayer.projection = new
                            ProjProjection("EPSG:4326");
                featureLayer.style = Style.getDefaultSurfaceStyle();
                map.addLayer(featureLayer);

                // Création du vecteur du LinearRing
                var vector_1:Vector.<Number>=new Vector.<Number>;
                vector_1.push(6.65591);    //P1[x]
                vector_1.push(46.78566);   //P1[y]
                vector_1.push(6.64922);    //P2[x]
                vector_1.push(46.78637);   //P2[y]
                vector_1.push(6.64046);    //P3[x]
                vector_1.push(46.79113);   //P3[y]
                vector_1.push(6.64282);    //P4[x]
                vector_1.push(46.79092);   //P4[y]

                // Création du Polygon
                var ring_1:LinearRing = new LinearRing(vector_1);
                var geom:Vector.<Geometry> = new Vector.<Geometry>;
                geom.push(ring_1);
                var polygon:Polygon = new Polygon(geom);
                var polygonFeature:PolygonFeature = new
                       PolygonFeature(polygon, {featureName:"MyPolygon"});
                featureLayer.addFeature(polygonFeature);
            }
        ]]>
    </fx:Script>

</s:Application>
```

9.4.1 Résultat

9.5 Interaction avec les objets

9.5.1 On Click

Nous allons à présent ajouter une info bulle lors du clic sur un objet. Pour ce faire, il est nécessaire d'ajouter un EventListener « CLICK » sur la couche de features s'occupant d'appeler la méthode showPopup( ).

```
featureLayer.addEventListener(MouseEvent.CLICK, showPopup);
```

```
private function showPopup(e:FeatureEvent):void
{
    // Supprimer l'éventuel popup ouvert
    if (currentPopup != null)
    {
        fxmap.map.removePopup(currentPopup);
        currentPopup = null;
    }

    // Ajoute le nouveau popup
    var feature : Feature = e.feature;
    currentPopup = new Anchored();
    currentPopup.htmlText = feature.data.featureName;
    currentPopup.size = new Size(measureText(currentPopup.htmlText).width
                                                            + 35, 30);
    currentPopup.feature = feature;
    fxmap.map.addPopup(currentPopup, true);
}
```

La notation `feature.data` permet d'accéder aux attributs de la feature. En effet, les attributs sont rattachés à un objet géographique grâce à l'utilisation du second paramètre des constructeurs createPointFeature, PolygonFeature, LineStringFeature. Par exemple:

```
PointFeature.createPointFeature(…,{featureName:"Point 1"});
```

On reconnait ici la syntaxe de déclaration d'un objet litéral pour définir les attributs de géométrie.

Les sources sont disponibles dans le fichier `PointsAndMarkers_Overlay_click.mxml`

9.5.2 On Over

Il est possible d'ajouter l'info bulle lorsque la souris est placée sur un objet. Pour ce faire, ajouter les EventListener suivants :

```
featureLayer.addEventListener(MouseEvent.MOUSE_OVER, addPopup);
featureLayer.addEventListener(MouseEvent.MOUSE_OUT, removePopup);
```

```
private function addPopup(e:FeatureEvent):void
{
    // Ajoute le nouveau popup
    var feature : Feature = e.feature;
    currentPopup = new Anchored();
    currentPopup.htmlText = feature.data.featureName;
    currentPopup.size = new Size(measureText(currentPopup.htmlText).width
                                                            + 35, 30);
    currentPopup.closeBox = false;
    currentPopup.feature = feature;
    fxmap.map.addPopup(currentPopup, true);
}
```

```
private function removePopup(e:FeatureEvent):void
{
    // Supprimer l'éventuel popup ouvert
    if (currentPopup != null)
    {
        fxmap.map.removePopup(currentPopup);
        currentPopup = null;
    }
}
```

Il en va de même pour l'interaction sur des lignes et des polygones.

Les sources sont disponibles dans le fichier `PointsAndMarkers_Overlay_over.mxml`

9.6 KML Features

Jusqu'à présent, nous avons créé des features « manuellement ». Il est possible de récupérer une couche de features provenant d'un fichier KML. Ce standard OGC est basé sur le formalisme XML et permet de décrire des objets géographiques ainsi que leur symbologie.

Pour en savoir plus sur le format KML (Keyhole Markup Language):

```
http://en.wikipedia.org/wiki/Keyhole_Markup_Language
```

Dans notre exemple, nous allons charger deux couches de features KML :

Limite administrative de la Haute-Savoie

```
<os:KML name="Haute-Savoie" url="http://ogo.heig-vd.ch/ria/data/74.kml"/>
```

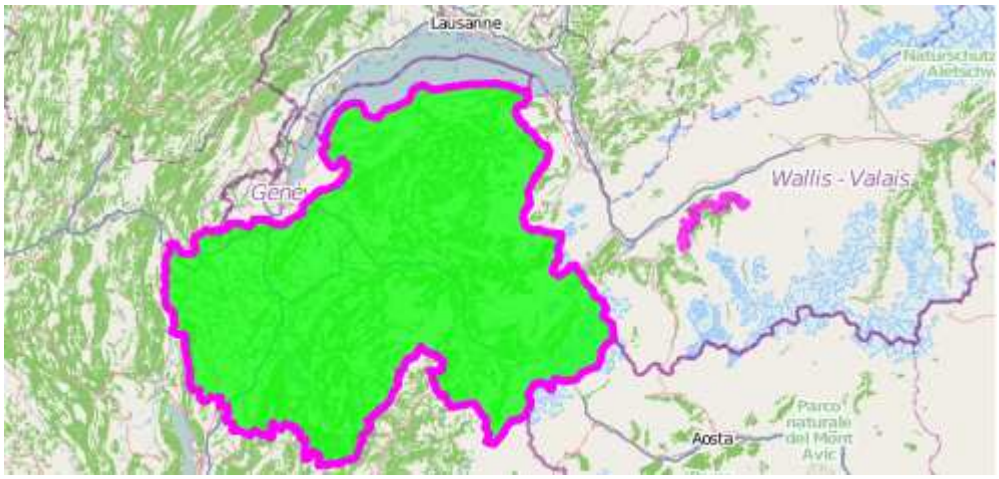
Trace GPS du Grand-Raid

```
<os:KML name="Grand Raid" url="http://ogo.heig-vd.ch/ria/data/utgtrack-
819.kml"/>
```

Kml_Overlay.mxml

```
<?xml version="1.0" encoding="utf-8"?>
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
               xmlns:s="library://ns.adobe.com/flex/spark"
               xmlns:mx="library://ns.adobe.com/flex/mx"
               xmlns:os="http://openscales.org"
               minWidth="955" minHeight="600">
    <s:VGroup>
        <os:Map id="fxmap"
                width="500"
                height="600"
                x="60"
                y="50"
                zoom="9" center="6.8,46">
            <os:Mapnik name="base"/>
            <os:KML
                name="Haute-Savoie"
                url="http://ogo.heig-vd.ch/ria/data/74.kml"/>
            <os:KML
                name="Grand Raid"
                url="http://ogo.heig-vd.ch/ria/data/utgtrack-819.kml"/>
            <os:MousePosition
                x="10" y="{fxmap.height-20}"
                displayProjection="EPSG:4326"/>
            <os:DragHandler/>
            <os:WheelHandler/>
            <os:ControlPanel
                title="Layers Manager"
                x="510" y="10" width="215">
                <os:LayerManager/>
            </os:ControlPanel>
        </os:Map>
    </s:VGroup>
</s:Application>
```

9.6.1 Résultat



9.7 Exercice

Mettre en place une application cartographique qui localise des boulangeries suisses. Les données proviennent d'un flux GML `http://ogo.heig-vd.ch/ria/data/boulangeries.xml`. L'objectif est d'afficher un point pour chacune des boulangeries du flux GML. De plus, on désire afficher le nom de la boulangerie dans un pop-up lorsque l'utilisateur clique sur un point.

Pour en savoir plus sur le format GML (Geography Markup Language):

```
http://en.wikipedia.org/wiki/Geography_Markup_Language
```

Parsing XML e4x avec namespaces

```
http://www.darronschall.com/weblog/2006/04/using-xml-namespaces-with-e4x-and-actionscript-3.cfm
```