Gestion de carte géographique avec Qt

Géomatique
: La géomatique regroupe l'ensemble des outils et méthodes permettant d'acquérir, de représenter, d'analyser et d'intégrer des données géographiques.

OpenStreetMap
: OpenStreetMap (OSM) est un projet qui a pour but de constituer une base de données géographiques libre du monde (permettant par exemple de créer des cartes sous licence libre), en utilisant le système GPS et d'autres données libres.

La bibliothèque qmapcontrol

`QMapControl` est un widget Qt qui permet d'utiliser les données de carte dans une application Qt.

Avec QMapControl il est possible utiliser différents fournisseurs de carte comme OpenStreetMap. Ensuite, vous pouvez modifier la vue de votre QMapControl, écriture d'une application de géolocalisation quand avoir un récepteur GPS ou tout simplement parcourir les cartes. Il est possible d'avoir plus d'une couche, de sorte que vous pouvez mélanger des cartes de différents serveurs. Pour enrichir les cartes que vous pouvez dessiner votre propre objet à coordonnées.

Liste des fonctionnalités :

- Compatible avec de multiples fournisseurs de cartes (serveur de tuiles et WMS serveur)
- Dessiner des objets personnalisés dans la carte (lignes, points, images, autres widgets Qt)
- La carte de navigation est personnalisable
- Analyseur de coordonnées GPS

Lien : ~~www.medieninf.de~~

Exemple :

```
// create MapControl
MapControl* mc = new MapControl(QSize(480,640));
//mc->showScale(true);

// create MapAdapter to get maps from
//MapAdapter* mapadapter = new OSMMapAdapter();
TileMapAdapter* mapadapter = new TileMapAdapter("a.tile.openstreetmap.fr", "/osmfr/%1/%2/%3.png", 256, 0, 17);

// create a map layer with the mapadapter
Layer* l = new MapLayer("Custom Layer", mapadapter);

// add Layer to the MapControl
mc->addLayer(l);
```

Ce simulateur permet de charger les coordonnées GPS d'un parcours et d'envoyer les trames NMEA 183 sur un port série RS232. On peut visualiser le parcours du bus sur une carte.

Code source : simulateur-siv-carte.zip

Il existe un portage de `QMapControl` pour Qt5 : https://github.com/TheDZhon/QMapControl

Viualiser une carte avec QWebView

Pour simplement visualiser une carte dans une application Qt, il est possible d'utiliser un `QWebView` qui permet de charger une URL.

Pour cela, il faut :

- installer `libqt5webkit5-dev` : `sudo apt install libqt5webkit5-dev`
- ajouter le module **webkitwidgets** dans le fichier `.pro` : `QT += core gui widgets webkitwidgets`
- instancier un `QWebView` et charger une URL :
  `webView->load(QUrl("https://www.openstreetmap.org/export/embed.html?bbox= 4.81696%2C43.9483,4.81696%2C43.9483&marker= 43.9483,4.81696")`

Code source : TestQWebView.zip

Le type QML Map et QQuickWidget

Lien : Cours QML

Qt fournit un ensemble de ressources (classes C++ et composants QML) pour la géolocalisation et l'affichage de cartes :

- une API de positionnement
- Maps and Navigation (QML)
- Maps and Navigation (C++)

L'affichage d'une carte est effectué à l'aide du type QMLMap. Pour accéder aux données cartographiques qui seront affichées dans un objetMap, QML fournit un Plugin de service.

Les cartes peuvent également contenir des objets de superposition qui sont utilisés pour afficher des informations. Il existe un ensemble d'objets de superposition prédéfinis de base : MapCircle, MapRectangle, MapPolygon, MapPolyline et MapQuickItem.

Il est possible d'utiliser des interfaces en QML dans une application Qt Widget en utilisant la classe `QQuickWidget` . On utilise alors la méthode `setSource()` pour charger un fichier `.qml` dans le widget. L'association peut se faire aussi dans Qt Designer.

En résumé, il faut :

- ajouter les modules **qml quickwidgets** dans le fichier `.pro` : `QT += core gui location qml quickwidgets`
- créer un fichier QML en y ajoutant le `Plugin` et la `Map`
- associer le fichier QML au `QQuickWidget`

Exemple de fichier `.qml` intégrant une `Map` et le `Plugin` pour OpenStreetMap :

```
import QtQuick 2.0
import QtQuick.Controls 2.2
import QtLocation 5.3
import QtPositioning 5.0

Item
{
    Plugin
    {
        id: mapPlugin
        locales: "fr_FR"
        name: "osm" // OpenStreetMap
    }

    Map
    {
        id: map
        anchors.fill: parent
        plugin: mapPlugin
        center: QtPositioning.coordinate(43.95, 4.8167) // Avignon
        zoomLevel: 14
    }

    MouseArea
    {
        anchors.fill: map
        hoverEnabled: true
        property var coordinate: map.toCoordinate(Qt.point(mouseX, mouseY))
        Label
        {
            x: parent.mouseX - width
            y: parent.mouseY - height - 5
            text: "lat : %1 - lon : %2".arg(parent.coordinate.latitude).arg(parent.coordinate.longitude)
        }
    }
}
```

Code source : TestMap.zip

Le géocodage consiste à associer des coordonnées géographiques (longitude/latitude) à une adresse postale. Qt fournit le type QMLGeocodeModel pour rechercher ces information géographiques. Il permettra de connaître l'adresse postale complète à partir de ses coordonnées géographiques (longitude/latitude). Le géocodage inverse est aussi possible grâce à `GeocodeModel`.

Le type QML RouteModel est utilisé pour extraire des itinéraires géographiques d'un fournisseur (cf. `Plugin`). Les itinéraires incluent des données sur les itinéraires entre deux points, des itinéraires avec plusieurs points de passage et divers autres concepts similaires. Il s'utilise avec des vues telles que `MapItemView`.

Voir aussi : http://tvaira.free.fr/dev/qt-android/qt-android-geolocalisation.html

Ce deuxième exemple montre l'interaction possible entre QML et C++ :

- utilisation d'un MapQuickItem
- appel d'une fonction QML à partir de C++
- signal/slot entre QML et C++

Code source : TestMap-v2.zip

Retour au sommaire