Libosmscout Library

October 15, 2019

1. Introduction

libosmscout offers applications simple, high-level interfaces for **offline location and POI lookup**, **rendering** and **routing** functionalities based on OpenStreetMap (OSM) data.

It offers you much of the functionality you need, to implement a offline-capable routing application (or similar) based on OSM data.

The goals of libosmscout are:

- Enable OpenStreetMap and its rich data set
- Implement all feature required for a navigation app or application that want to use some subset of such features.
- Work on low-end hardware like it is commonly used for handhelds, phones and similar.
- Offer compact data storage making it possible to place data for countries and up to continents on SD cards of common size.

2. Features

All basic features for

- importing OSM data
- querying the resulting database
- rendering fancy maps nobody have ever seen
- finding locations no one else could find

3. Namespace

The main namespace is **osmscout**. Some of the core libraries include:

libosmscout/
The libosmscout base library.
libosmscout-import/
Library containing the import code, allowing conversion of *.osm and *.pbf files into the libosmscout-specific database.
libosmscout-map/
Base library for all map renderer.
libosmscout-map-qt/
Library containing the Qt render backend.

libosmscout-client-qt/
Library containing common Qt application code (shared by OSMScout2 and StyleEditor).

4. Dependencies

Libosmscout has the following optional and required dependencies:

libxml2 (optional)
libxml2 is required for parsing *.osm file during import
protobuf-c, protobuf-compiler (optional)
These are required for parsing *.pbf files
marisa (optional)
for an additional location full text search index
libagg (optional), freetype (optional)
for the agg backend. If you use agg, freetype must be available, too
cairo (optional), pango (optional)
for the cairo backend, if pango is available it will be used for complex text rendering instead of the cairo toy font rendering code
Qt5 (optional)
for the Qt5 backend.
freeglut (optional), glu (optional)
for the OpenGL demo

5. Preparations

Note, that if your distribution supports Qt4 and Qt5 make sure that you install the Qt5 packages. If you have installed Qt4 and Qt5 packages in parallel you might have to install additional packages that offer you ways to default to Qt5 or select Qt5 dynamically (e.g. qt5-default for debian).

6. General Concepts

This section tries to explain some general concepts of libosmscout.

- **Import**

Libosmscout has its own internal representation of OSM data. This is necessary to offer fast access of the data and reduce the size of the data on disk. Standard export formats like .osm or *.osm.pbf files are used as data dump from the OSM SQL database for further processing and are designed for fast, index random data access.

For importing the data the Import tool is used. The import tool generates a custom libosmsocut database from the raw data passed.

- **Database**

  The import tool creates a custom database from the import data. The database consists of a number of files. Some files hold data, other files act as index into data and still other files have index and actual data combined.

- **Services**

  To further hide the details or complexity of the low level access code a number of Services have been defined that offer a more suitable API.

  Currently the following services exist:

- MapService
- LocationService
- RoutingService
- POIService

- **BasemapDatabase**

  The basemap database is a special database that holds world wide information to be used as a background to the actual data from individual database instances.

  The basemap currently holds the following information:

- World-wide coastlines

- **Rendering backend**

  For actual drawing of maps you need a rendering backend. Since libosmscout is layered and modular the actual rendering is decoupled from database access. Services (in this case `MapService') are designed to be used by different backends. No special API is enforced.

  The following backends currently exist:

- libagg backend
- cairo backend (with and without pango support)
- Qt backend
- iOS and Mac OS backend
- DirectX backend (proof of concept)
- SVG backend (this backend is in a proof of concept state and is not activly maintained)
- OpenGL backend (current in proof of concept but will be rewritten as part of the GSoC 2017).

- **Type configration file (*.ost)**

  The type configuration file maps certain tags onto types. Each object imported from the import fil(s) will have a certain type.

  Types are later on used to assign further information to the object and are used as a filter criteria for the style sheet.

- **Style sheets (*.oss)**

  To allow configurable rendering and things like night mode libosmscout uses style sheets for configuration of the actual rendering look.

7. Qt QML API

Part of this project is libosmscout-client-qt library that aims to provide simple API for applications using user interface in QML.

8. Supported platforms

The following platforms are supported:

- Linux using gcc or clang, using meson or cmake as build system
- Mac OS X and iOS using XCode/clang
- Windows
  - Visual Studio 2013 (likely libosmscout source has to be patched slightly)
  - Visual Studio 2015
  - Visual Studio 2017
  - MinGW/MSYS2
- Android
  - Qt
  - Initial efforts to generate Java stubs by using SWIG to alloe easy access to libosmscout from normal Android applications
- Sailfish OS using Qt backend
- Ubuntu phone using Qt backend

The compiler must at least support C++14. But, in the github is written windows platform is not supported.

9. Conclusion

I think this library has all of the requirements of our project and specially is compatible with Qt. But based on the github, windows platform is not supported.