OPEN STREET SOUNDS

(as of September 2014)

Bachelors project

by Michael Macek - 0504524

e0504524@student.tuwien.ac.at

Vienna University of Technology - TU Wien

Abstract

Open Street Sounds is an application which allows the user to browse through the map of Vienna and experience the sounds and noises of the city.
The map is made up of tiles supported by OpenStreetMap in various zoom levels.
At any time, it is possible to place down a pin with the left mouse button. Following a placement, the application will find informations about the surroundings and will play back fitting sounds.

Descrition

When the user opens the application for the first time, there will be some files missing, which will be downloaded automatically from their respective servers. First, the application will alert the user that the “filtered osm xml file” is missing, which should be downloaded afterwards by clicking OK. (More on that file later.) Since this file is quite large (50mb unfiltered), and also subject of constant change, it makes more sense to download a new one when the application is being used for the first time.
The other files are the tiles of the map, which will be downloaded only when needed. (In total, the application will need about 500mb of disk space.)

For the downloads, the cUrl library is used (see section: “Used Libraries”). The osm xml file[1] will be downloaded from the Overpass API[2]. Therefore the application will send a http request via cUrl to the API containing a query saved in the file “*...\res\overpass_api_query.txt*” which just specifies certain tags within a bounding box made up of longitude and latitude values. After the download is completed, the external application[3] “*...\res\osmfilter.exe*” is called to filter out unnecessary information, like name of a street, or the OSM user who created said street.
cUrl is also used to download the tiles from a tile server. The application just requests a 256x256 pixels png picture from the server and stores it locally.

When the application is open you will see a map of the city in its farthest zoom level. (OSM Zoom Level 12).
The dimensions of each map tile are 256 by 256 pixels. At first the application will generate a view that contains enough tiles to fill the window plus a 1 tile thick border around that view.

You can now drag the map, to move your focus around. As soon as you dragged 256 pixels in a horizontal or vertical direction (you would now see a border tile in whole), the application will generate a new “map” that will meet the previously mentioned requirements.

---

[1] http://wiki.openstreetmap.org/wiki/XML
[2] http://wiki.openstreetmap.org/wiki/Overpass_API
external API that returns osm xml data
[3] http://wiki.openstreetmap.org/wiki/Osmfilter
also see file dl_parameter.txt in the folder “.../res/” which are the parameters to the filter application.

The application comes with only the few tiles of zoom level 12. Every other tile further in will be downloaded automatically. Of course this will take a short while, during which the dragging and zooming will be a bit laggy. Once the application has been used a bit, those lags will disappear. There also is the possibility to download all of the tiles instantly via a menu option. This will take a while, since the application will need to download about 25,000 tiles (or 250mb of data).
Should there be no internet connection, the application will use a tile in a further zoom level and display portions of that tile instead.

The user can change between zoom levels 12 to 17 (by OSM standards[4]) by using the mouse wheel. By doing so, the application calculates the position of the cursor in longitude and latitude, generates a new map in the corresponding zoom level and refocuses the map, so that the saved longitude and latitude are on the same position as the mouse cursor is.

If you click with the left mouse button, the “real magic” starts. In the background the application has an beforementioned xml file, filtered to the applications needs. When you set down a pin, the first half of that file will be searched for nodes that are in hearing range of the pin. Those nodes are stored and in the second half of the file the applications searched for tags (definitions) of those nodes. The library used to search through the xml is libxml (see section: “Used Libraries”).
For all the nodes that were found, the application also saves the distance to the pin in relation to the maximum distance as well as the longitude and latitude.
The results will be stored in a map of a custom class[5] sorted by distance of which a vector with the closest eight will be send to the Audio Manager to play back their respective sounds.

The Audio Manager[6] will use specific sounds for each type of node and will use the distance to modify the volume and the angle as balance for the sound.
This all is done within OpenAL (see section: “Used Libraries”) which is a library for 3D sound environments. There is one listener object and multiple source objects. The listener object represents the pin, the user has put down. It also has an view vector which points north so sounds west from it will be played from the left speaker. It will be placed down with its longitude as x coordinate and latitude as y coordinate. Because the map of Vienna is a quite small area, longitude and latitude can represent a valid part of a coordinate system with straight axis. The Z value will be zero for each and every object in the audio space.

Each source represents a found node. They also will receive the longitude and latitude as their positions in X-Y space. They will also receive to values for their volume attenuation. The first value is the reference distance. Everything closer will result in the sound being played at full volume.
The second value is the maximum distance; objects farther away will be silent, and should not even be a source because of above mentioned measurements. This changeable distance

[4] http://wiki.openstreetmap.org/wiki/Zoom
Any zoom level greater than 17 would imho need more disk space then it would bring use.
[5] OssXmlResult.hpp
[6] OssAudioManager.hpp

(hearing range) is predefined as 100 meters (The same distance as the search radius of the pin).
Each source will buffer an *.wav file, which has to be named the same way as the node tag has been defined. For instance: the wave file for the tag value “place_of_worship” should be “place_of_worship.wav”.

Limitations

First, while programming and planning I only looked at Vienna as the region of choice. It is hard-coded that only tiles in this area will be shown.
The effect of that is, that the longitude and latitude are seen as straight axes and used in a Cartesian coordinate system, which the earth clearly does not have.
If one could put the pin into the Atlantic Ocean at -48.19856/-16.37222[7] he would still hear the sounds of the Resselpark.

The OSM-XML file only stores nodes and ways. The detection of the surroundings can only be accomplished with nodes.
A way can be a line made up of multiple nodes or it can be an area. If the nodes that define that area (or line) are far apart and the detection area (hearing range) of the pin is too small to reach any of the nodes, it will simply not detect the area, whilst being in the middle of it.

To reduce the stress on the sound “hardware” and on the ears of the listener, the application has been limited to play a maximum of 8 different sounds.
If the pin would be placed in front of a church, which by definition of OSM is mapped as an way made out of multiple nodes, we wouldn’t only want to hear just 8 identical church bells.
Because of that, the applications only plays one sound source of a kind.
The penalty of that is, even when there are, for example, two tramway stations in the detection area, only the closer one will result in a sound source.

Underground objects like rivers are listed in the XML file. If the user should happen to place the pin at the Naschmarkt, the user will still hear the sound of a river, although it is clearly not audible in real life.

Furthermore a tramway can only be heard if the pin is close to a tramway stop.

Because of the way the Windows mouse click messages are handled[8], double click to zoom in is not implemented.
Usually Windows sends the mouse messages in this order: left mouse button down, then left mouse button up, then left mouse button double click and again left mouse button up. Because I intended to use a single left click to set the pin and start playing sounds, and also usually use the

[7] http://www.openstreetmap.org/#map=7/-48.199/-16.372
[8] http://msdn.microsoft.com/en-us/library/aa931259.aspx

mouse wheel to zoom in map applications, I did not try and implement a zoom in via double click.

Inserting new content

Inserting content is possible without changing any code. There are two steps: updating the database and adding a sound file.

To update the “database” the user must first know what kind of node to add.
For this tutorial let’s imagine we want to add a tag where key is “amenity” and the value is “biergarten”.
From the Open Street Maps wiki[9] we can see that a beer garden could be either a node or a way. Since a simple node seems useless we’ll only concentrate on having a beer garden as a way. Otherwise we would need to enter a second line for the node, like “bus_station”

First we will need to edit the file “overpass_api_query.txt” found in the “res”-folder.
We will need to add the line:
“`way["amenity"="beergarden"](48.1183,16.1827,48.3231,16.5786);`”
above the line that only says “`);`”. Then we save and close that file.

Next we need to open the file “dl_parameters.txt”. Since there is already a line for amenities: “`amenity=bus_station =place_of_worship`” we will just have to add a “`=biergarten`” to the end of that line (leave a space between the previous tag value).
Again we save and close that file.

Then we start up the application OpenStreetSound and in the “XML” menu we select “Download XML” and wait for a bit. The application will now download a new xml file with the new parameters from the api, will filter the downloaded xml file with the other parameters.

While the application finishes the first step, we need to put a suiting “*.wav” file into the “res\Sounds” folder. It would be best to edit the file with your favorite audio editor first.
In Audacity I recommend setting the sample to mono by going to “Tracks->Stereo Track to Mono”, the Project Rate to 44100 Hz (in the bottom left corner of the window) and normalizing it to -9.0 dB via “Effect->Normalize”. Then we need to export the file as a “WAV (Microsoft) signed 16 bit PCM” into the folder “res\Sounds” and name it “biergarten.wav”.

After we have placed the sound file and the application has finished updating the xml file, we can browse to an area where the tag appears on the map click the left mouse button there and enjoy our new sound!

---

[9] http://wiki.openstreetmap.org/wiki/Map_Features#Amenity

Controls

Hold LMB and drag - move map
LMB click - set Pin (load local information)
RMB click - remove Pin and stop sound
Esc - Exit
F5 - refresh map

Used libraries and external applications

libxml - http://xmlsoft.org/ - http://www.zlatkovic.com/libxml.en.html
and its dependencies: iconv and zlib
for xml parsing

libcurl - http://curl.haxx.se/libcurl/
for downloading the open street maps tiles from the official servers
(server "b.tile.openstreetmap.org" is being used)

OpenAL - http://www.openal.org/
for playing audio files in a 3D environment.

freeALUT - https://github.com/vancegroup/freealut
free Audio Library Utility Toolkit for making life with OpenAL easier.

osmfilter.exe - http://wiki.openstreetmap.org/wiki/Osmfilter
for filtering an osm xml file. to make it smaller and therefore faster to look through

File structure

File OpenStreetSound.cpp - the main class which handles the window and the user input.

Namespace tools inherits all the other namespaces as well as the class Console, which only loads in Debug mode, and displays a Console window with debug information. Also a helper class which contains all the MessageBoxes lies here.

Namespace *audio* contains everything that has to do with playing sounds and noises which the user will hear, whenever he sets the pin down. The most important class in here is the OssAudioManager class, which loads and plays all the different sounds and noises.

Namespace *data* handles the xml management. The most important class in this namespace is the OssXmlReader, which parses the downloaded osm xml file and finds the information about

the surroundings out of that file. The class OssXmlLoader is used when downloading the xml file from the Overpass api and to filter the file. An instance of OssXmlResult would be an object that has been found in the proximity of the user placed pin, which contain information necessary for synthesizing.

Namespace *visual* contains everything about the displaying the map, from zooming and moving the map to downloading tiles. The most important class in this namespace is OssMap which generates the 2d array of tiles which will be shown in the application and controls all other management aspects of the tiles as well. The class OssPin is the class which controls the pin the user can set on the map with the press of the left mouse button and remove from the map with a press of the right mouse button. OssPoint is just a custom 2-dimensional integer class. OssTile represents a single tile, which is contains the pointer to a downloaded or other picture as well as the coordinates to where the picture is shown and other informations. OssTileFetcher is the class which is responsible for downloading tiles from the tile servers.

The folder "*res*" contains all the external resources:

- folder "*Tiles*" - contains all the locally stored tiles
- folder "*Sounds*" - contains all the sounds for the application.
- *osmfilter.exe* (necessary) - an external program which is used to filter the downloaded osm xml file to remove overhead (i.e. tags that are not used from the programm and therefore will only slow the parsing down).
- *dl_parameters.txt* (necessary) - a text file that contains the filter parameters for the osmfilter.exe which is called whenever the application downloads a new xml file which will afterwards be filtered automatically.
- *pin.ico* (necessary) - the pin icon.
- *overpass_api_query.txt* (necessary) - the query the application will send to the overpass api to fetch an osm xml file of vienna with only nodes and ways that include the tags we want to synthesize.
- *resources.res* (necessary) - a resource file from Visual Studio which contains the information of where the application finds the icon.
- *default.png* (necessary in the folder "Tiles") - the default tile, which will be shown, if no other tile was found, or if the user looks outside of the "valid" area.

The folder "*external*" contains all the files from the external libraries like libcurl and others.

Used Sounds

"highway"="motorway" - http://www.freesound.org/people/schafferdavid/sounds/165017/
"highway"="primary" - see “motorway”
"highway"="secondary" - http://www.freesound.org/people/inchadney/sounds/173154/
"highway"="tertiary" - see “secondary”
"highway"="residential" -http://www.freesound.org/people/OSH37/sounds/140820/
"highway"="pedestrian" - http://freesound.org/people/inchadney/sounds/149816/
"highway"="living_street" - see “residential”
"leisure"="playground" - http://freesound.org/people/foongaz/sounds/209901/
"leisure"="park" - http://www.freesound.org/people/pawsound/sounds/154835/
"leisure"="pitch" - http://www.freesound.org/people/tonant/sounds/237345/
“railway"="tram_stop" - http://www.freesound.org/people/notchfilter/sounds/43695/
"railway"="station" - http://freesound.org/people/inchadney/sounds/159465/
"amenity"="bus_station" - http://freesound.org/people/ninebilly/sounds/173006/
"amenity"="place_of_worship" - http://www.freesound.org/people/JarredGibb/sounds/219044/
“amenity”=”marketplace” - http://freesound.org/people/eckel/sounds/34615/
"waterway"="river" - http://www.freesound.org/people/Glaneur%20de%20sons/sounds/24511/
"natural"="water" - http://freesound.org/people/Owl/sounds/156525/
"power"="plant" - http://freesound.org/people/Zabuhailo/sounds/167673/
"power"="generator" - http://freesound.org/people/paulw2k/sounds/244420/
"landuse"="forest" - http://freesound.org/people/costaipsa/sounds/114021/
"landuse"="meadow" - http://freesound.org/people/inchadney/sounds/163617/
"landuse"="cemetary": “*Music for Funeral Home - Part 11*” from http://incompetech.com/music/royalty-free/?keywords=funeral
"landuse"="vineyard" - http://freesound.org/people/FreqMan/sounds/24979/
"landuse"="industrial" - http://freesound.org/people/Micronin/sounds/184888/
"shop"="mall" - http://freesound.org/people/ultradust/sounds/160578/ and “*Local Forecast - Elevator*” from http://incompetech.com/music/royalty-free/index.html?genre=Jazz