Location Racing

Final Project Report

By

Philip Stafford

Table of Contents

1. Introduction

Location Racing is a proof of concept Android application which, when installed on a user's Android device, will allow the user to race a car around the local streets that surround the user no matter where they are in the world as long as they have a GPS signal and a data connection.

This application can be used by anyone who owns an Android device.

With over 1.3 million Android devices being activated on a daily basis the number of casual gamers is in increasing exponentially.

"…the casual games market is poised to become a €6.25 billion industry by 2014…"

(Evangelho, 2013)

With mobile device ownership growing the demand for games is growing and the potential for a game to be location aware is massive. Even though a Boeing 777 plane can disappear without a trace, almost every smartphone sold has GPS chip built into them. This is even true for even cheapest smart phones on the market these days. The Samsung Galaxy Pocket Neo smartphone retails for €39.99.

(Vodafone, 2014)

Location aware applications are becoming bigger than ever now that the technology has become so available so cheaply. Google Street View for instance has made any location that has been mapped by Google viewable at a street level by anyone in the world who has access to the internet for free.

There are no mobile applications available yet who can generate levels on the fly which are created from the user's specific location.

When the application is loaded the user is presented with a GUI which allows them to detect their location, download the map data of their surrounding area and then create a map using the data that they have just downloaded. Once all of these steps have been completed then the user is able to enjoy racing around their local area on their Android device.

1.1.Purpose of this document

The purpose of this document is to outline how the project was executed. The focus of the document will be on how the development process was carried out and it will list which features were included in the project and why and it will also list which features which were not implemented and why they were not implemented.

Also any learning outcomes from the implementation will be outlined.

2. Target Users

2.1.General Potential Users

Every Android user is a potential target user. More than 900 million Android devices have been activated to date and with 1.3 million devices being activated on a daily basis the number of devices that will need to be protected from malware is set to increase exponentially.

(Bort, 2013)

Because there are nearly 1 billion Android users out there then it is fair to say that this application will be targeted towards people who have an interest in mobile gaming.

There are statistics which state that 58% of the population of the United States of America play video games. 56% of all people on the planet own a smart phone and that 80% of the time that is spent on a mobile phone is spent playing games or using some sort of an app.

(Galarneau, 2014)

The possibilities for this game to be ported to various models are endless. This type of application could be used to train rally drivers to drive on a circuit virtually before a race. The reason for this is because they are not allowed to drive on a circuit physically before a race.

Also, this type of local level building on the fly may be ported over to action games such as Call of Duty which has a large competitive online presence of or to games like World of Warcraft which have a large online role playing user base.

By porting this idea to other game genres then the number of potential target users will increase tenfold.

2.2.Racing Game Enthusiasts

This application is targeted towards but not limited to racing game enthusiasts. The game will be easy to play while also being difficult enough to challenge the player. This will attract casual gamers, hard-core gamers and hard-core gamers.

2.3.Casual Gamers

A casual gamer is someone who likes to play the occasional game. They like to be able to pick up a game at random and play it for a small amount of time without having to worry about spending 40 minutes learning what the buttons do in order to play the game.

2.4.Hard-core Gamers

Hard-core gamers on the other hand like to get involved in a game. These players spend a lot more time playing games than casual gamers so it is crucial to ensure that these players keep coming back for more. The appeal to these players is to beat the game, unlock features and earn a reputation within the game. For these players a scoring system is crucial so that these players keep coming back to play the game.

2.5.Social Gamers

There are also social gamers. Some of these players will own a game but may have never played the single player story mode of the game. These players enjoy playing against human opponents so they will only play the online multiplayer mode of the game or the offline multiplayer mode. These players are not necessarily strictly social gamers. They may also be casual social gamers or hard-core social gamers.

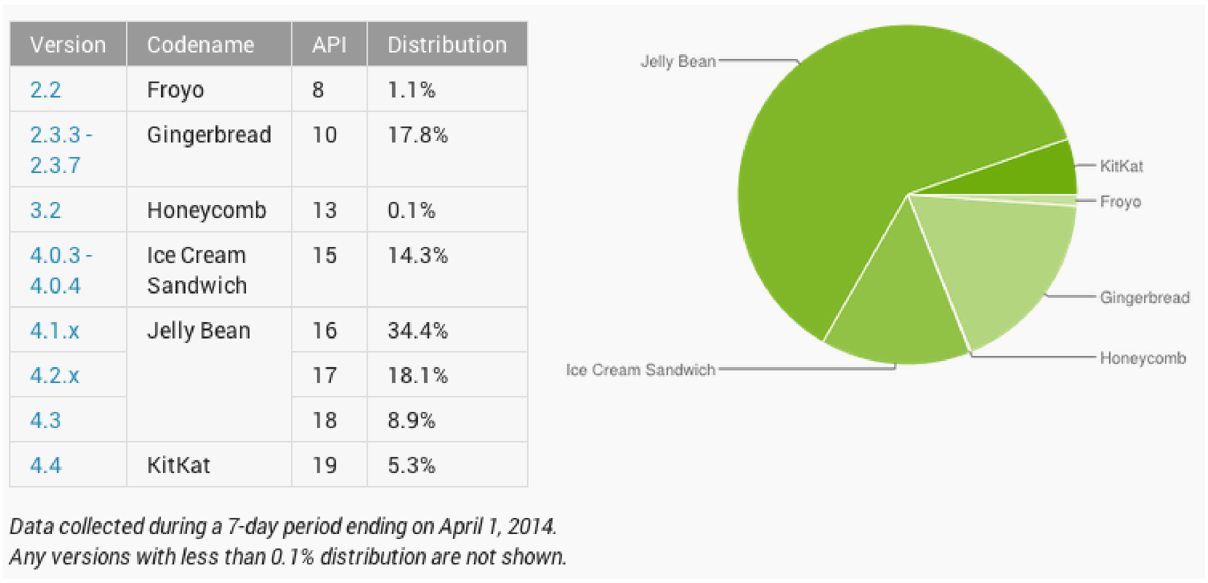
3. Scope

The Android operating system runs on many different devices. These devices are made by a large variety of different manufacturers so this means that the application will have to run on many different hardware configurations.

The lowest version of the Android operating system that this Antivirus Application will support will be Android 2.2 Froyo (API level 8).

The reason for this is that the AndEngine game engine, which the application was developer with, only supports Android devices which are running Android version 2.2 and above.

Also by targeting devices with Android version 2.2 and higher is the application will be able to run effectively on the majority of android devices. As of April 2014 the earliest version which is still running on devices in the wild is Android 2.2.x Froyo (API level 8).



Version	Codename	API	Distribution
2.2	Froyo	8	1.1%
2.3.3 - 2.3.7	Gingerbread	10	17.8%
3.2	Honeycomb	13	0.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	14.3%
4.1.x	Jelly Bean	16	34.4%
4.2.x		17	18.1%
4.3		18	8.9%
4.4	KitKat	19	5.3%

Data collected during a 7-day period ending on April 1, 2014.
Any versions with less than 0.1% distribution are not shown.

(Google, 2014)

According to these figures from Google, Location Racing will support at least 99.9% of all the Android devices which are currently in the wild.

The latest version of Android that is available on Android devices is Android 4.4 KitKat (API level 19). This version was only released on the 31st October 2013 so it hasn't been pushed out to many Android users. The update is being slowly pushed out to higher end devices.

4. Use of Application

In order for a user to play the Location Racing game they must install the application on their Android device. The device must be running Android version 2.2 (Level 8 and above). A copy of the application can be downloaded from the following locations:

4.1. Location Racing - Zoomed In Version Online

This zoomed in version is close to the car so that player can race around the streets of their area.

http://glasnost.itcarlow.ie/~softeng4/C00139462/LocationRacing/LocationRacing-ZoomedIn.apk

4.2. Location Racing - Zoomed Out Version Online

This zoomed out version allows the user to see the full map of their area on the screen.

http://glasnost.itcarlow.ie/~softeng4/C00139462/LocationRacing/LocationRacing-ZoomedOut.apk

The user must navigate to the location using the browser on their Android device. Once the file has been downloaded the user must navigate to their download folder and select the application in order to install it. Depending on the settings that are configured on the user's phone, the user may have to allow the phone to install applications from unknown sources.

4.3. Location Racing - Zoomed In Version CD

This copy of the app can be found on the provided CD in the folder

/ Location Racing APK Files/LocationRacing-ZoomedIn.apk

4.4. Location Racing - Zoomed Out Version CD

This copy of the app can be found on the provided CD in the folder

/ Location Racing APK Files/LocationRacing-ZoomedOut.apk

If the user chooses this option they must load the specific APK file on to the Android device. Once the file has been loaded onto the device the user must navigate to location where the app has been saved and select the application in order to install it. Depending on the settings that are configured on the user's phone, the user may have to allow the phone to install applications from unknown sources.

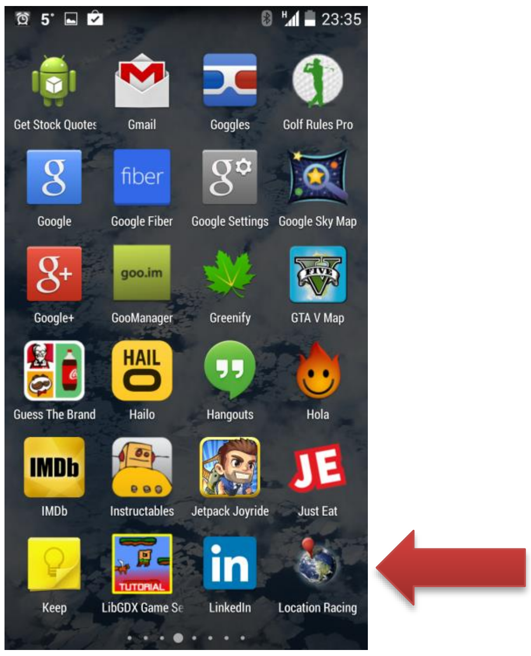
This application is only designed as a proof of concept so it is not ready to be used by the public.

The Location Racing game can be loaded by the user and they will be able to see a map on the screen of their surrounding area and they will be able to drive a car around the roads on the screen.

5. Project Description

Once the user has installed Location Racing the application will be available to the user.

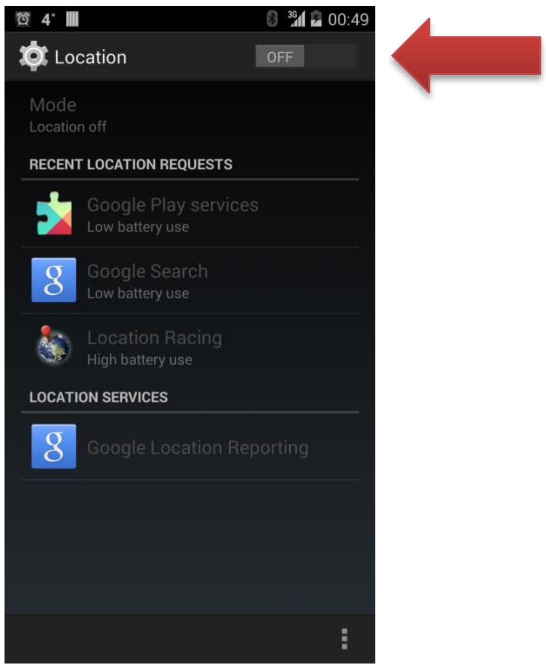
In order to use the Location Racing app the user must navigate to the Location Racing icon on their device and press it to load the application.



Once the user has loaded the app it will check to see if it has access to the location services.

If the location service is turned off on the device then the user will be brought to the location settings where they can turn on the location services.

The user will have to turn the location service on.


00:49

Location OFF

Mode
Location off

RECENT LOCATION REQUESTS

Google Play services
Low battery use

Google Search
Low battery use

Location Racing
High battery use

LOCATION SERVICES

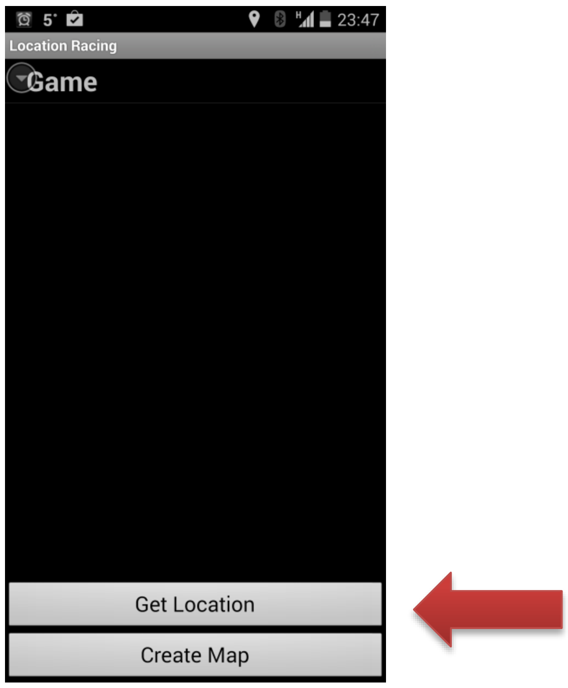
Google Location Reporting

Once the user has loaded the application and the location services have been turned on then they will be presented with this screen.

This screen presents the user with all of the options that the user needs in order to play a game of Location Racing.

To begin a game the device must first determine its location so that it can download the correct map data so that a local level can be loaded for the user to race on.

To determine the location the user must press the 'Get Location' button.

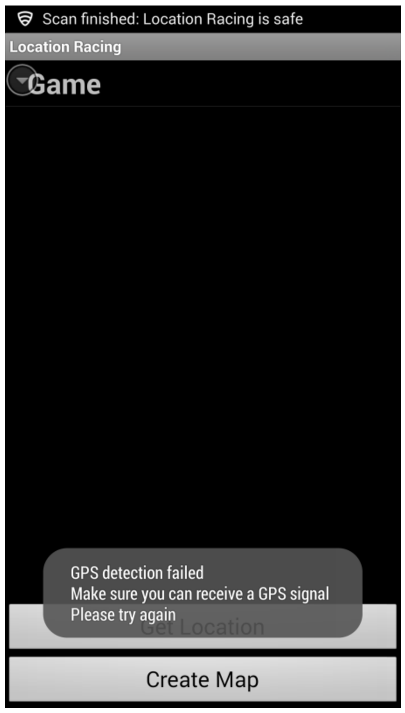


When the 'Get Location' button is pressed the app will attempt to get the location of the device.

If the device can't retrieve the last known location a message will be displayed to the user informing them that the GPS location detection has failed. It will then inform the user to make sure that a GPS signal can be received by the device and that the process may have to be tried again.

The 'Get Location' button may have to be pressed a few more times while the user moves to an area where the device has access to a direct line of site to the sky.

This will ensure that the device will detect its location.

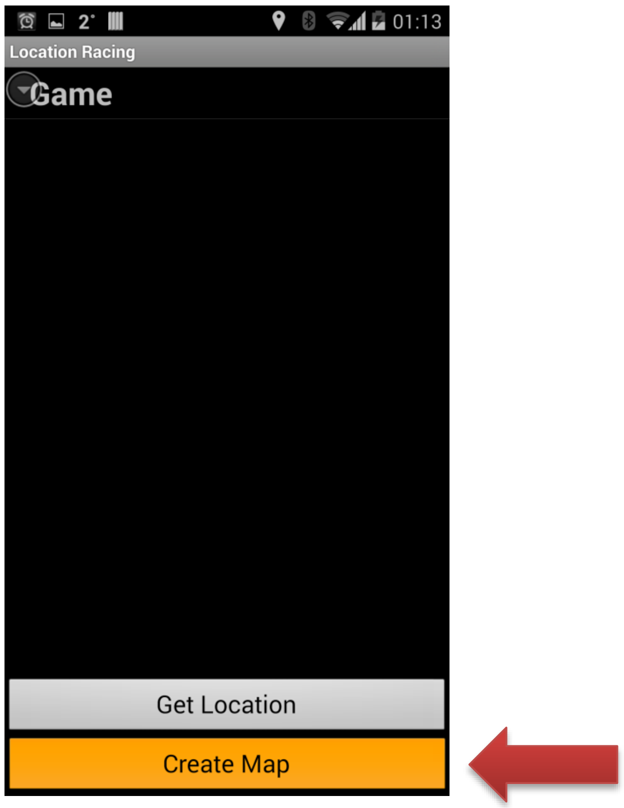

Scan finished: Location Racing is safe

GPS detection failed
Make sure you can receive a GPS signal
Please try again

When the location has been successfully detected and the map data for the surrounding area has been downloaded then the app is ready to create the map which the player will later race around.

To do this the player must press the 'Create Map' button.

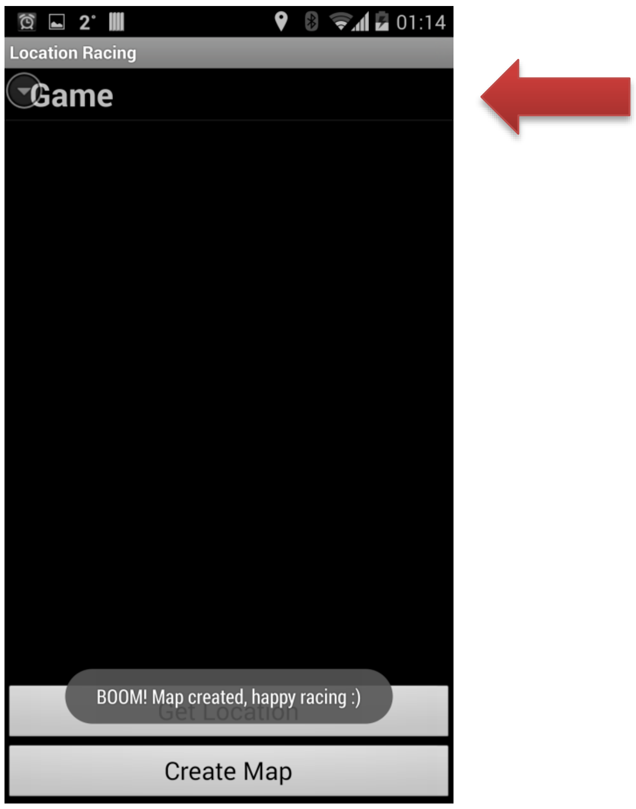
Once the user presses this button the application will take between 30 to 60 seconds to create the map. The 'Create Map' button will turn yellow and stay yellow until the map has been created.



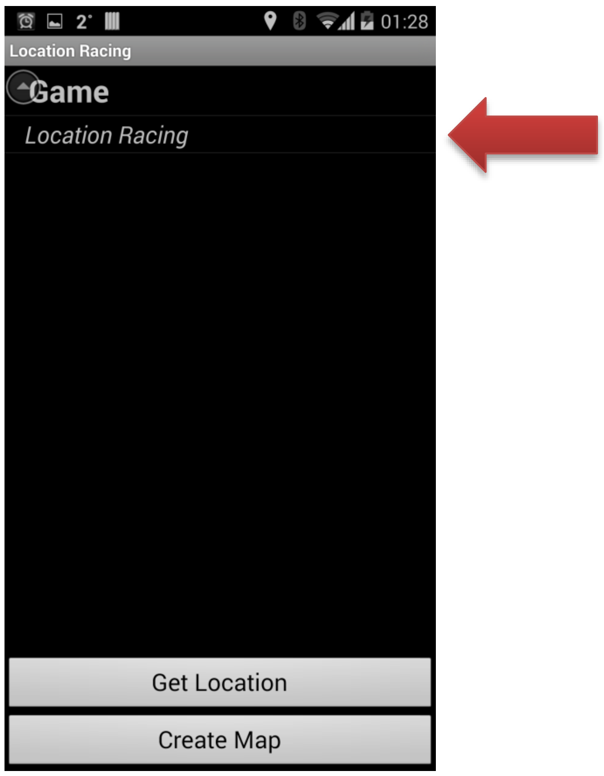
When the map has been successfully created a message will be displayed to the user to inform them that the map has been created successfully.

Next the user presses the 'Game' drop down list.

When this is pressed an option to load the 'Location Racing' game will appear.



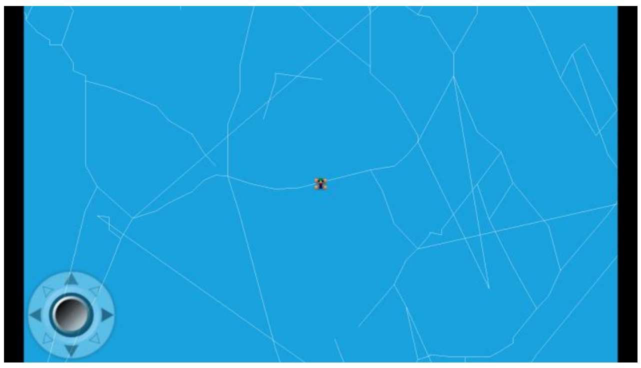
This is the last button press. In order to play the game the user has to press the 'Location Racing' button. Once this is pressed the map will begin to be drawn onto the screen.



When the game loads the user will be presented with the actual racing game.

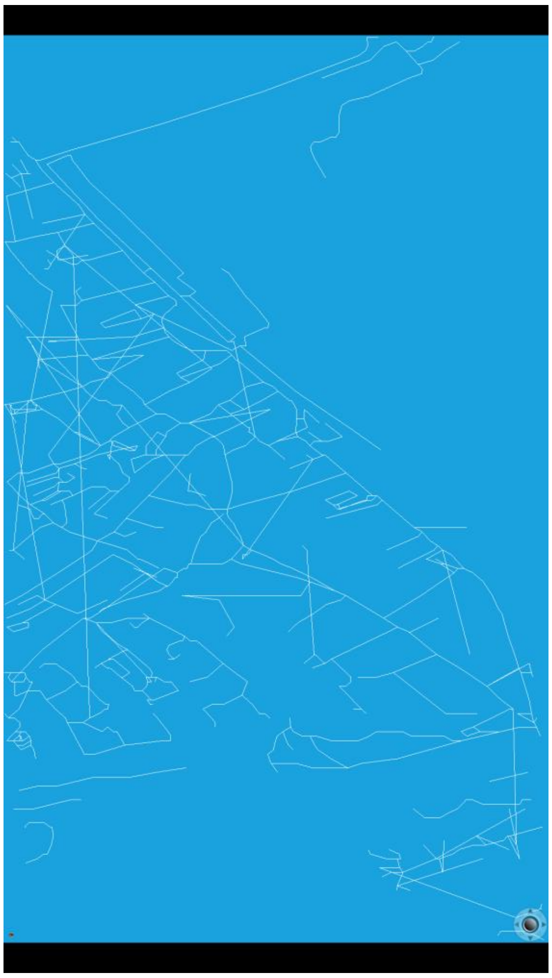
There will be a controller in the bottom left corner and the car will be displayed in the centre of the screen.

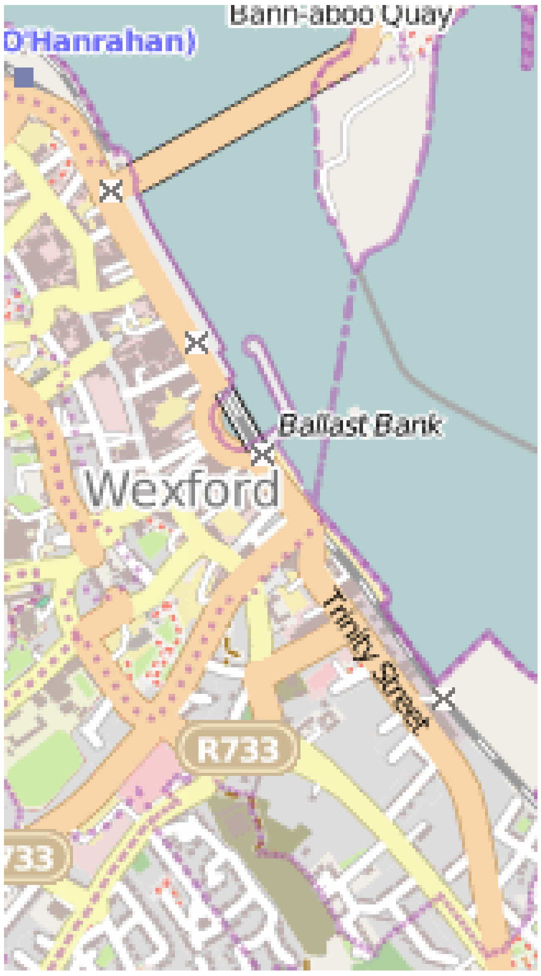
To move the car the user just has to use the controls.



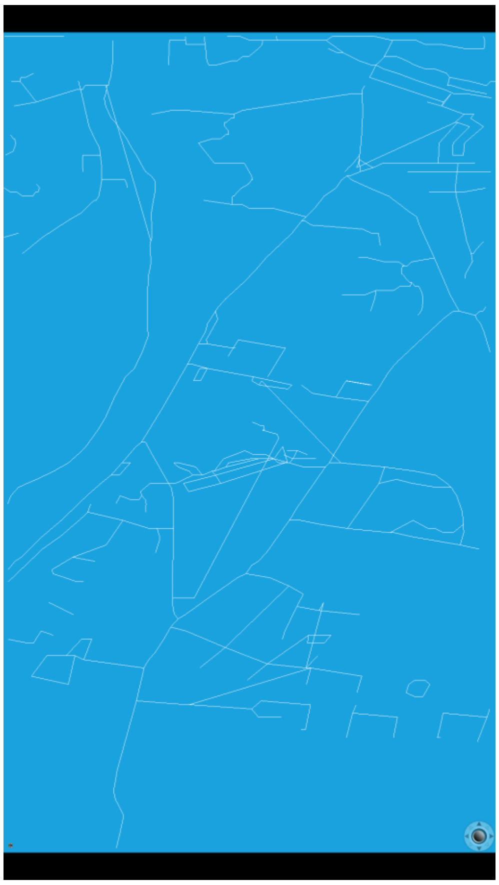
6. Screenshot & Map Comparisons

6.1. Wexford





6.2.Carlow



R430
Barrow
R448
Burren
R448
Burren

7. Conformance to Specification and Design Manual

7.1. What Was Achieved from Specification and Design

7.1.1. Location Feature

The location feature had to be broke into two different sections.

7.1.1.1. Location Detection

In order for the game to work the app has to be able to detect the location of the device.

Firstly the device checks to see if it has access to the location services. If not the app redirects the user to the location services settings so that the user can turn on location services.

If the location service is turned off on the device then the user will be brought to the location settings where they can turn on the location services.

The app will then try and retrieve the last know location of the device. If the location comes back as null the app will attempt to generate a location fix. The device will listen for a location change and when the location change happens, the device the app will retrieve the location and use it.

7.1.1.2. OSM API Call

Once the app has successfully determined the location of the device it begins to create a string which is used to retrieve map data from an online mapping service called OpenStreetMap (OSM).

OSM is…

"…An openly licensed map of the world being created by volunteers using local knowledge, GPS tracks and donated sources..."

(DMOZ.org, n.d.)

The OSM API only allows software developers to download all of the information about an area. This information includes trees, houses, shops, café's, pubs, landmarks, etc. This is all great and useful information but for the purposes of this app it was an information overload and it was unnecessary. All of this information would have to be processed which would be a waste of processor time.

A more powerful OSM API is called Overpass API. Overpass API allows complex API queries on specific areas. For the purposes of this app all the information that was needed was the GPS coordinates of all of the roads in the user's location.

Overpass API allows software developers to make calls to retrieve only the information that they regard as useful.

Once the location has been determined the URL can be built to make the API call. The user's latitude and longitude both have values added and subtracted to them in order to make a boundary box. The boundary box is an area with maximum and minimum latitude and a maximum and minimum longitude. Or in other words a box with north, south, east and west values.

When the URL has been made the application downloads the map data. The API call is made to this website http://overpass-api.de/

Anything inside of these values is returned from the API call. An example of an API URL would be…

http://overpass.osm.rambler.ru/cgi/interpreter?data=node%2852%2E3241655%2C%2D6%2E46279924%2C52%2E344165499999995%2C%2D6%2E451599239999999%29%3Bout%3B

This will be explained later in this document.

When an API call is made the map data is returned to the app in the form of an XML file. The XML file is stored on the device which can be processed at a later date to create a map on the device.

The download feature in the application is the work of Hassanpur. (Hassanpur, 2011)

7.1.2. Track Generation Feature

The track generation feature also had to be broke into two different sections.

7.1.2.1. XML Parsing

The two XML files that are downloaded from the API call contain all of the information that is required to create the map for the game.

The wayInfo.xml file contains the information for all of the individual roads and all of the nodes within each node whereas the nodeInfo.xml file contains all of the information about each node. I.e. the latitude and longitude.

The wayInfo.xml file is parsed using an XMLPullParser, which is built into Android, and all of the information regarding an individual way is stored in the SQLite database.

The nodeInfo.xml file is also parsed using an XMLPullParser and all of the information about each node is store in the SQLite database.

7.1.2.2. SQLite Database Storage

While the XMLPullParser is retrieving the relevant information from the XML files the information is being sent to the SQLite database.

All of the information that is being extracted from the nodeInfo.xml file is stored in a table called nodes. The `nodes` table has 4 fields, `entry_id`, `node_id`, `latitude` and `longitude`.

The `entry_id` is the unique key for the `nodes` table.

All of the information that is being extracted from the wayInfo.xml file is stored in a table called ways. The `ways` table also has 4 fields, `entry_id`, `way_id`, `node_a` and `node_b`.

The `entry_id` is also the unique id for the `ways` table.

7.1.3. Display Map Feature

The display map feature was implemented using a game engine called AndEngine. AndEngine handled the drawing of the race track on the screen.

7.1.3.1. AndEngine Integration

The app was developed using a game engine called AndEngine. The game engine was developed by a guy named Nicolas Gramlich.

AndEngine is freely available to anyone who wants to wants to use it.

7.1.3.2. Creating a Scene

When the player wants to load a game the app has to use and engine to draw the map onto the screen.

The map is drawn by retrieving the latitude and longitude of each node which are within the user's location. While the each node is being retrieved it is converted into an X and Y coordinate. These X and Y coordinates are used as points so that the map information can be drawn on the screen.

When all of these points are joined up correctly the user will be able to distinguish their local area on the screen. As well as the road information being drawn in the screen a car and a set or controls are also drawn on to the screen.

The AndEngine game engine provided a package full of examples. One of these examples was of a game called Racer. The Racer example was modified to create the game scene which was used in Location Racing.

7.1.4. Race Feature

AndEngine also controlled the race feature. AndEngine draws the map of the user's location, the car and the controls for the car on the screen of the device.

AndEngine also interprets the input from the user. This input is used to move the position of the car on the screen. When the car moves around the screen the camera is set to chase the car so that it is always in the centre of the screen.

7.2.What Wasn't Achieved from Specification and Design

7.2.1. From the Race Feature

Keeping a record of the time isn't implemented, collision detection and handle the physics of the game so that when the cars are driving around the track they are reacting and handling realistically.

The collision detection could not be implemented because the boarders of the race track could not be implemented correctly. This is explained more in the algorithm section.

7.2.2. Circuit Determination Feature
This feature was supposed to have 2 options. The player could choose to allow the computer to automatically generate a race circuit or the player can create their own race circuit. If the player had of allowed the computer to select a circuit then the computer would have randomly chosen a route on the map up to 2 miles long. This distance was supposed to be changeable by the player.

If the player decided to create their own race circuit then the player could have chosen a circuit with no maximum length specified. The only conditions for the creation of the circuit were that it cannot leave the map, it was supposed to contain a start point and a finishing point and it must have a minimum distance of 500 meters.

This feature would allow the user to actually race a car around their surrounding location. This feature would have allowed the user to pick their own circuit to race around or to have the computer generate a random circuit for them to race around.

This feature would have added some extra game play to the app.

7.2.3. User Account Feature
If the user account feature had of been implemented it would have added some valuable extra features to the game. If each user had their own account they could have kept track of their score and achievements easily in the cloud.

7.2.4. Multiplayer Feature
The multiplayer feature would have allowed human players who were in the same geographical location to race against each other online. This feature would have been driven by a server which was hosted in the cloud.

This feature would have allowed an online community to thrive and maybe grow into something bigger. The implementation of an online multiplayer feature would allow friends to meet up and challenge each other to race on familiar turf.

8. Technologies Used

8.1. Technologies Suggested to Use

8.1.1. LibGDX

LibGDX was the game engine that was decided on to develop the Location Racing app. The reason that LibGDX was chosen from the list of game engines that were researched is because LibGDX seemed to have a good support network and many tutorials to follow.

This was supposed to help learn how to use the LibGDX game engine. Also the option to easily port the application to various other platforms was very appealing.

It turns out that a different game was chosen because it was deemed to have an easier learning curve for a beginner who had no previous knowledge of android game development, which was the case.

8.1.2. Android

Android was chosen because there is a large support community available to help with any difficulties which may have arisen during the development phase.

Also, I had some previous experience of using the Java programming language to develop software and seeing as Android development is done mainly using the Java programming language; it seemed like the best option available at the time.

Also it is very easy and free for someone to get started in Android development. Each Android device has the option to unlock Developer mode which allows the user to easily start installing and testing devices on their actual devices instead of an Android emulator which is included in the Eclipse Android Development Tools.

8.1.3. Java

The Java programming language was chosen automatically when the Android platform was decided upon. This was a good choice due to the great support available online for Java and Android development.

8.1.4. OpenStreetMap

OpenStreetMap (OSM) was selected because it allowed the developer to retrieve a huge amount of information about a specific area. Google maps did not allow the same amount of content to be downloaded as easily.

In hindsight the amount of information that was available for download became a small problem because there was actually too much information being downloaded. This was increasing the amount of CPU time which was needed to process the large amounts of information.

This problem was never conceived in the research manual so upon further research it was found that there was an API provider which allows the developer to retrieve only what information the required from a specific area.

This discovery was crucial in the development of the app. It cut the CPU time dramatically which confirmed that the decision to use OSM as the map data provider the correct choice.

8.1.5. OSM2World

OSM2World has huge potential to convert a specific location into a 3D model for the purposes of creating a location aware video game.

Soon after the research manual for the project was finalised the idea to create a 3D game was scrapped. It was thought that a 2D racing game would be more achievable given the time constraints of the project and the lack of experience that would be needed to create 3D Android app.

It was a wise choice to drop the 3D idea and opt for a 2D app.

8.1.6. Eclipse/IntelliJ

The choice of Integrated Development Environment (IDE) was not stated in the Research Manual because the decision wasn't made by the time the Research Manual was completed.

Ideally IntelliJ would be the best solution because of the quality of the IDE. It is very stable and it is far better than Eclipse but at the end of the day, Google seemed to support Eclipse with their Android Developer Tools so Eclipse became the standard for Android development.

Because of this most of the game engines and the tutorials that have been written for Android development are all support Eclipse. There is very limited support for IntelliJ. It is possible to develop Android apps and use game engines with IntelliJ it is an option but this could more than likely become a problem while looking for support with any particular issues.

Things are starting to change though because Google have now teamed up with the developers of IntelliJ, JetBrains, to create a new standard IDE for Android development. This IDE is called Android Studio.

Android Studio is still in beta so it is not entirely stable at the moment. Even though this tool will be great in the future, it was just too risky to chance using it for this project.

With all of that being said Eclipse was chosen for the IDE.

This was the best choice for logical reasons i.e. community support but by no means is Eclipse the best tool for the job.

8.1.7. Google App Engine

Google App Engine was chosen to support the backend of the app. This backend would have managed looking after user accounts, logging in and out, handling multiplayer functionality and keeping track of an online leader board.

Unfortunately none of these features were implemented during the development of the project.

8.2.Technologies Used Instead of Suggested Technologies

8.2.1. AndEngine

After more research and careful consideration a game engine called AndEngine was chosen to develop the app. This game engine wasn't even mentioned in the Research Manual because it wasn't discovered until after the Research Manual was complete.

AndEngine provides an all in one solution for Android game development. The reason AndEngine was chosen over LibGDX is because it was stated many times online that AndEngine was a better choice for a beginner to game development.

Also there is a discussion on a forum where both the developer of LibGDX, Mario Zechner and the developer of AndEngine, Nicolas Gramlich bot discuss the pros and cons of both game engines.

(Developers, 2010)

9. Problems Encountered

9.1. AndEngine Setup

Setting up the AndEngine game engine within the Eclipse IDE was difficult for someone who has zero experience setting up a game development environment.

Nicholas Gramlich, the developer of AndEngine, has a repository on Github where all of the packages that are required for AndEngine are stored.

(Gramlich, 2010)

I found a tutorial online on how to setup the AndEngine development environment. I followed all of the instruction to the letter. After about one and a half hours of setting up the environment I encountered several bugs in the AndEngine package that I could not find any solutions to online.

I ended up deleting the Environment and starting all over again. I presumed that I must have done something wrong so I followed the same tutorial again. The tutorial instructed me to download all of the packages manually, extract them and then add them the Eclipse development environment.

Once all of the packages were added to the environment they all had to be linked together. After I had completed all of that I ended up with the same errors again and I couldn't find a solution to them again.

Again I deleted all of the packages. I searched around for another tutorial to setup the AndEngine development environment.

When I found another tutorial the guy who made the tutorial, RealMayo, explained that Nicolas Gramlich's Github repos contain bugs that have not been fixed. He explained that the Nicolas constantly updates the game engine. So while you will have the latest version of AndEngine you will also have a game engine which contains bugs.

(RealMayo, 2013)

Real Mayo then went on to say that he had created branches of the AndEngine repositories and that he had fixed all of the bugs in them. I followed RealMayo's tutorial to the letter and low and behold again, there were some bugs in RealMayo's code. I went to the AndEngine forums where I saw posts of RealMayo saying that his code did not contain any bugs and to reinstall everything again. I tried many more times and still kept getting the same error.

Eventually I found an AndEngine setup tutorial from guy called Martin Varga which worked perfectly.

(Varga, 2013)

It took me 2 weeks to get this development environment setup correctly by the time I kept reinstalling AndEngine and looking for bug fixes.

9.2.AndEngine Tutorial

Firstly I started following a tutorial by a guy called Travis.

(Travis, 2012)

I spent a week following this guy's tutorial only to discover that he decided to quit making the tutorials before I learned enough about AndEngine game development to create my own full android game. This was a waste of a week.

After that tutorial fail I found another tutorial which seemed to be far more complete.

Once I had AndEngine set up I followed the tutorial from a guy called Matim. This is a very comprehensive tutorial which covers a lot of information regarding AndEngine so I had to take my time while following this tutorial to make sure I understood all of the areas of the tutorial.

(Matim, n.d.)

When I was following the tutorial the sprites that I was drawing weren't ending up I the same place as they should have been according to the tutorial. I presumed that this was because my device had a 1920 x 1080 resolution screen so I altered the X and Y coordinates of the sprites and the appeared in the correct location on the screen of the device.

After a week and a half of following the tutorial I encountered a bug that I could not explain and that was not mentioned in the tutorial. I searched around for a solution to the error and I discovered that the error was because I was using a version of AndEngine that was out of date.

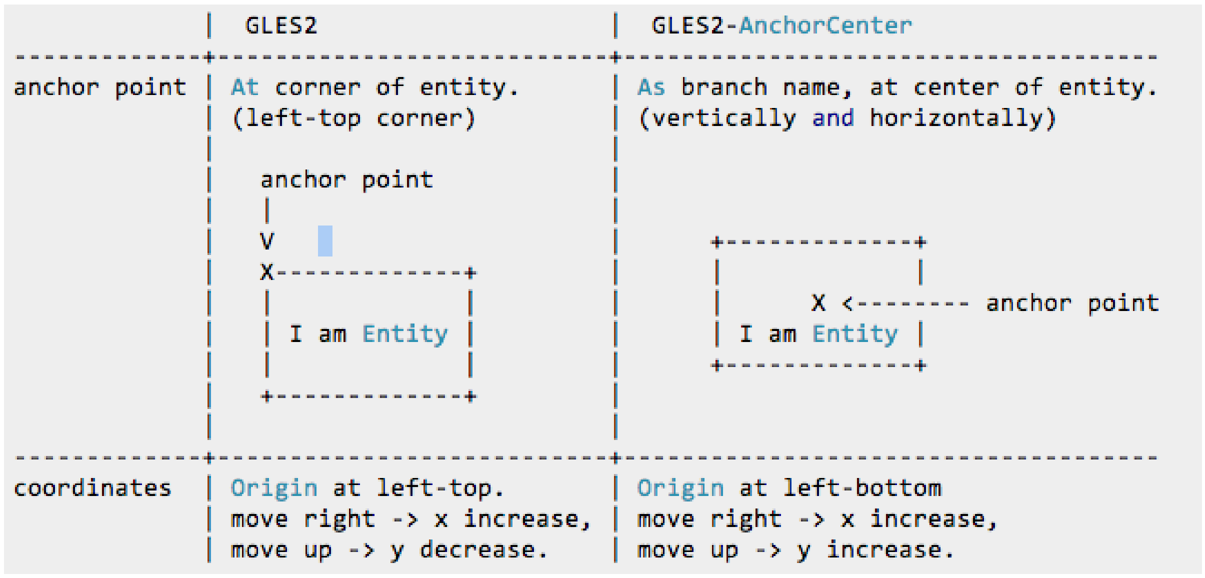
It turns out that I there are 3 different versions of AndEngine. GLES1, GLES2 and GLES2 Anchor Centre. GLES1 is the original version which has been discontinued.

GLES2 is the second edition and GLES2 Anchor Centre is the third edition. Apparently GLES2 Anchor Centre edition will merge back into GLES2 eventually.

The difference between GLES2 and GLES2 Anchor Centre is explained by StackOverFlow member 正宗白布鞋:

"Every Sprite (actually Entity) has an anchor point. When you place a Sprite on Scene position (x, y), it means you stick the anchor point of Sprite on the coordinates (x, y).

In AndEngine GLES2 and GLES2-AC branches, both anchor point and coordinates system are different."



```
             |  GLES2                      |  GLES2-AnchorCenter
-------------+----------------------------+------------------------------------
anchor point | At corner of entity.        | As branch name, at center of entity.
             | (left-top corner)           | (vertically and horizontally)
             |                             |
             |   anchor point              |
             |   |                         |
             |   V                         |     +-------------+
             |   X-------------+           |     |             |
             |   |             |           |     |      X <-------- anchor point
             |   | I am Entity |           |     | I am Entity |
             |   |             |           |     +-------------+
             |   +-------------+           |
             |                             |
-------------+----------------------------+------------------------------------
coordinates  | Origin at left-top.         | Origin at left-bottom
             | move right -> x increase,   | move right -> x increase,
             | move up -> y decrease.      | move up -> y increase.
```

“Note: The anchor point is also used when Entity rotates, skews and scales. So, in GLES2 branch, when entity is rotating, people often think the entity is also moving, but it doesn't, the anchor point is always fixed at coordinates (x, y).”

(正宗白布鞋, 2013)

I was using GLES2 where Matim was using GLES2 Anchor Centre. I had to go back to the drawing board again and install the GLES2 Anchor Centre version of AndEngine.

It took me another week to get AndEngine GLES2 Anchor Centre installed and all of the bugs fixed.

Now I had to fix all of the sprite positions for the tutorial in order to draw them in their correct locations. Once I had that fixed I could carry on with the tutorial.

After two and a half weeks following and a week trying to install the correct version of AndEngine I ended up with a fully functional Android game which was able to load a level from an XML file. I thought I could use this would be the way to load in the information for Location Racing.

I began to detect the location, retrieve map data of the area that surrounded the user and create databases to store the information for the user’s map information. I then figured out how to create an XML file which would load all of the map data into the game scene.

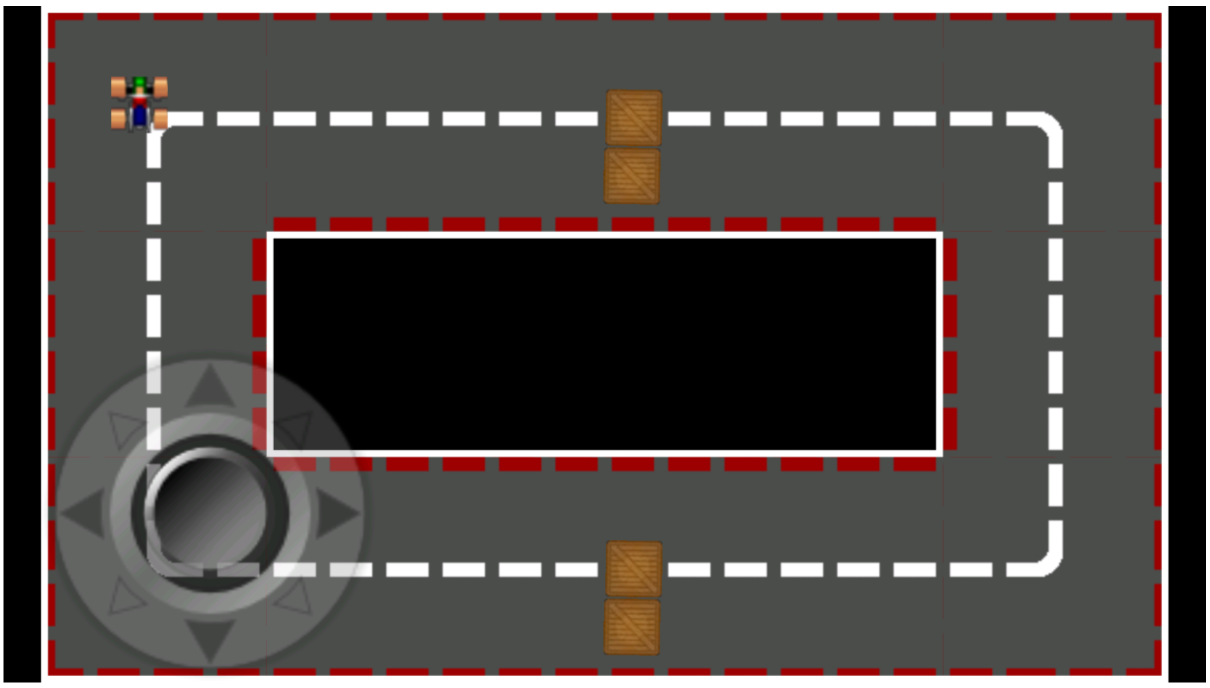
When the XML file was loaded into the game scene it drew a 10 x 10 pixel sprite at every X and Y coordinate that the XML file contained.

The XML file loading method would only allow a sprite to be drawn to the screen at a specific X and Y coordinate. I realised that this was not going to be any good for my app which would have to draw lines between each X and Y coordinate.

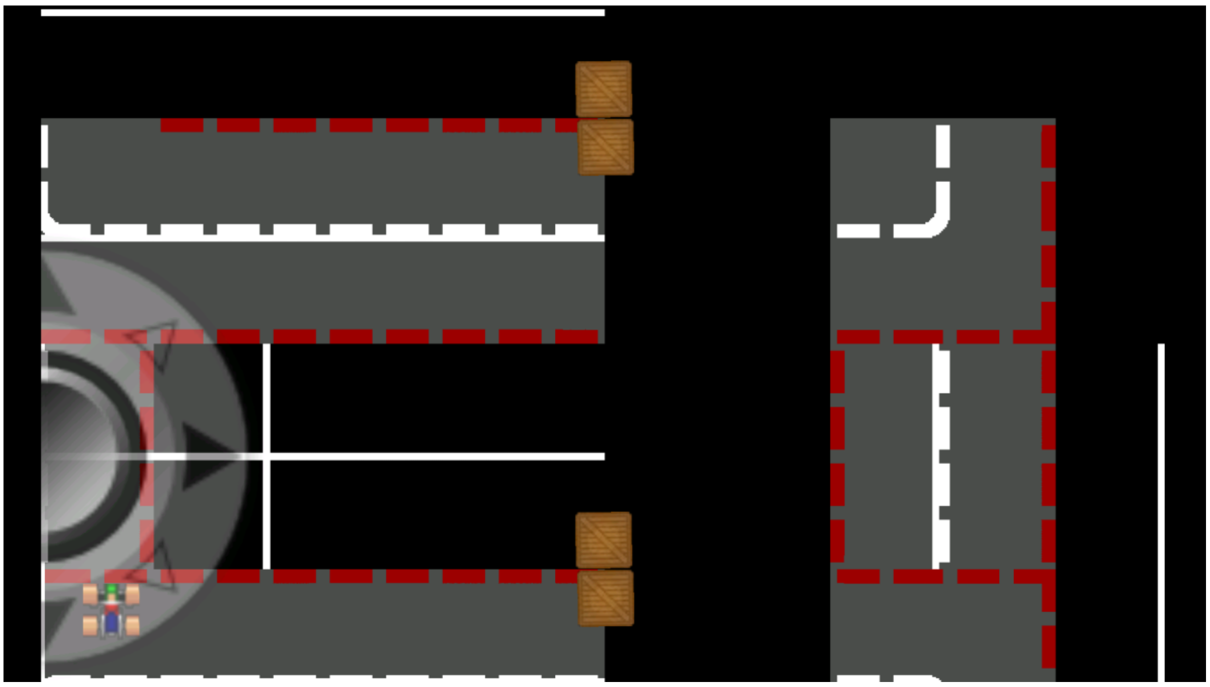
After wasting three and a half weeks on this I had to abandon that plan and go back to the drawing board.

9.3.AndEngine Versions

I decided to try and modify the Racer example. I loaded up the racer example which was supposed to look like this



But instead it looked like this and it was totally unplayable. The entire screen was screwed up.



After uninstalling AndEngine and reinstalling and engine a load of times, by using all of the tutorials I mentioned previously, and trying to eliminate all of the bugs each time I reinstalled AndEngine I realised that it must have been GLES2 Anchor Centre which was the problem.

Previously my problem was that I was using GLES2 when I should have been using GLES2 Anchor Centre. Now it turns out that I should be using GLES2 instead of GLES2 Anchor Centre. Beam me up!

After wasting two days uninstalling and reinstalling different versions of AndEngine I finally got a working version working with all of the bugs fixed. It took me another 2 days to get all of the code I wrote during the previous tutorial to work in the Racer example.

9.4.AndEngine Support

During the development of this app I came across many programming problems. I was led to believe that the AndEngine community is a thriving little community but I found it difficult to get definitive answers on some of the programming problems I had.

9.5.Eclipse Logcat Bug

In my experience the Eclipse IDE is a very buggy piece of software. The logcat output for the android would constantly crash thus forcing me to restart Eclipse.

On other occasions the code development windows would freeze forcing me to close Eclipse also.

10. Algorithms

10.1. Location Detection

The location detection checks to see if a location has been determined. If not, the app will try to generate a location.

When a location is determined a specified value is added and subtracted to the latitude to create a north and south value and a specified value is added and subtracted to the longitude to create an east and west value. These four values are stored and they make up the north, south, east and west boundaries for the overpass API call.

When these values are determined they are used to create the API URL.

After the boundaries are calculated they are stored in an object for use at a later time.

10.2. OSM API Call

This is a full API URL:

http://overpass.osm.rambler.ru/cgi/interpreter?data=node%2852%2E3241655%2C%2D6%2E46279924%2C52%2E344165499999995%2C%2D6%2E451599239999999%29%3Bout%3B

This is the beginning of the API URL:

http://overpass.osm.rambler.ru/cgi/interpreter?data=node%28

Each north, south, east and west boundary is sent to a method to be converted into the URL.

The value is first checked to see if it's positive or negative. If it is negative it converted it is converted to positive by multiplying by negative 1 and the result is stored in a temporary value. The value is then converted from a double to a string. The value is then sent off to build a negative part of the URL.

The negative part of the URL is built up by concatenating the ASCII value for a minus sign, then concatenating the part before the decimal point of the value, then concatenating the ASCII value for a full stop, then concatenating the decimal part of the value and finally concatenating the ASCII value for a comma to the URL.

If the value is positive the value is converted from a double to a string. The value is then sent off to build a positive part of the URL.

The positive part of the URL is built up by concatenating the part before the decimal point of the value, then concatenating the ASCII value for a full stop, then concatenating the decimal part of the value and finally concatenating the ASCII value for a comma to the URL.

This process is executed 4 times until the URL is created.

This part of the URL is common to both the node API call and the way API call.

The common part of the API URL is then concatenated to each of the node and way API URL’s to make up two individual API URL’s.

Each of the URL’s is than passed individually to a separate thread to make the API call and download the map data XML file relating to the URL.

The download feature in the application is the work of Hassanpur. (Hassanpur, 2011)

10.3. Check if Database Exists

Try to open the database. If the attempt is true set the exists flag to true.

If it fails the catch block will catch the error and set the exists flag to false.

10.4. XML Parsing

10.4.1. parseWayXML()

Before the XML parsing begins the application checks to see if there is a database already created for this application. If a database exists it is deleted of all of its data.

After this database check the application will attempt to open the wayInfo.xml file. The name of the wayInfo.xml file is passed into a method which will use an XMLPullParser to retrieve the information from the file.

An XMLPullParser is created and the location of the file is detected. The XMLPullParser searches through the XML file looking for starting tags with the value ‘way’. If it finds a new way tag set the newWayStarted flag to true. When it finds one it knows that a road has started.

Now check if it’s the first node in the road and if it is it sets the node id to nodeA and adds nodeA to the validHighwayNodes array list. If the value is not the first value in the road then it sets the node id to nodeB and adds nodeB to the validHighwayNodes array list.

It then checks if nodeA and nodeB both have values. If they do it increments an index, this is used as the primary key. Then it adds the index to an index array list, it adds the way id to a way id array list, it adds the nodeA to a nodeA array list and it adds the nodeB to a nodeB array list. Then it swaps the value in nodeB to nodeA.

Then it checks for a tag to see if it is a valid highway. If the highway tag is found the highwayFound flag is set to true.

If an end tag is found and the end tag is a ‘way’ tag then check if the highwayFound tag is true then send all of the array lists to the database.

Then reset the values of the array lists to empty.

If the highwayFound tag is false then reset the values of the array lists to empty.

After either of those options reset the highwayFound to false, newWayStarted to false and set the index, way id, nodeA and nodeB to zero.

This process is repeated until the wayInfo.xml file is fully parsed, the end document tag is found and the all the relevant edges have been stored in the database.

10.4.2. parseNodeXML()

The name of the nodeInfo.xml file is passed into a method which will use an XMLPullParser to retrieve the information from the file.

An XMLPullParser is created and the location of the file is detected.

Retrieve an array list of all of the valid highway nodes form the database. Calculate the size of the list to use as an index.

The XMLPullParser searches through the XML file looking for starting tags with the value 'node. If it finds a new node tag it gets value of the node id. This value is sent to the database of valid highway nodes and the database is queried to see if the node id exists in the valid highway nodes database.

If the node id exists in the database then the latitude and longitude of the node are stored in the nodes database.

This process is repeated until the nodeInfo.xml file is fully parsed, the end document tag is found and the all the relevant nodes have been stored in the database.

10.5. Scene Creation

Get an array list of all of the ways from the database and store it in wayArrayList. Then calculate the size of it to use as a loop condition.

Get the north, south, east, and west boundaries for the GPS coordinates from the object that was created earlier on. If the startOfNewRoad flag is true then retrieve the values form wayId, nodeA and nodeB and set the startOfNewRoad flag to false.

Else if the next condition check checks if the loop index +1 is not greater than the size of the array list. This means that it is the last element in the wayArrayList. And that the wayId of the current element is the same value of the next element. This means that this isn't the last node in the current way.

If the conditions are both true then retrieve the wayId, set the value of nodeB to nodeA and retrieve a new value of nodeB.

Else set the startOfNewRoad flag back to true because this node is that last node in the current way. Set the value of nodeB to nodeA and retrieve a new value of nodeB.

Send the id number of nodeA and the name of the column 'node_id' to the database to retrieve a hash map which contains the entry_id, node_id, latitude and longitude of the nodeA.

Send the id number of nodeB and the name of the column 'node_id' to the database to retrieve a hash map which contains the entry_id, node_id, latitude and longitude of the nodeB.

If the value for nodeA is not empty retrieve the latitude and longitude of nodeA and convert them into X and Y coordinates to be drawn on the screen at a later date.

If the value for nodeB is not empty retrieve the latitude and longitude of nodeB and convert them into X and Y coordinates to be drawn on the screen at a later date.

Check that the X and Y coordinates for nodeA and that the X and Y coordinates for nodeB all have values. If they do have values then use them to draw a line and then attach them to the scene.

11. Description of Learning

11.1. Technical

By developing this Android application I learned a lot of information about how the Android system operates. I learned how an application can manipulate all of the underlying hardware features that the Android device has at its disposal.

I also learned how to make an API call to a system or server and how to retrieve the information and store it on the device.

I learned a great deal about how to parse XML files and how to process and manipulate data to into a data structure that I want. Also when I had turned the data into a data structure that I wanted I had to figure out a way to make the data do exactly what I wanted it to do.

I didn’t have very much experience working with Android, XML, SQLite, OpenStreetMap or a game engine like AndEngine.

I was a great opportunity to get exposure to all of these technologies.

I also would have never been comfortable working with Array Lists or Hash Maps but because I had to use them in my project I would not have a problem working with them again.

11.2. Personal

Personally this is a project I have been thinking about for nearly 20 years. Back when I first thought if the idea it was impossible to make this kind of an application happen. But with the massive breakthroughs in technology I was able to create an application that a team of developers would not have been able to create 20 years ago.

I also learned how to deal with errors and problems. Between first year to third year I would not have been able to deal with problems or figure out solutions to complicated programming difficulties. When I used to go and look for a solution to a problem I wouldn’t have a clue how to interpret the solution to a problem to a question I had.

Even though the solution to the problem would be written on the screen, I would get overwhelmed by the complexity of the code. Now I have learned to break down the code and to use the parts of the code that would benefit me.

Also researching is a skill that I feel more confident in completing now. Also, I feel strongly that it is much easier to do something that you enjoy rather than something that you are force to do. I thoroughly enjoyed the actual development time of this project. I wish I had more time to contribute to the project.

12. Review of Project

Looking back on the project I am pretty happy with the way it turned out. Before I started the project I had great visions of how the finished project would look. I imagined that it would look fantastic. Looking at it now, it is not the best looking app out there but it does exactly what it set out to do.

The plan was to develop a proof on concept app which would allow the user to download the map data for their surrounding area, generate a map and to drive around that area. While the application still has a very long way to go in terms of visuals, for a proof of concept app it does exactly what it says on the tin.

12.1. What Went Wrong

I had a lot of problems with the AndEngine game engine. I feel that if I hadn't have had all of the issues I had with all of the bugs of the outdated versions of AndEngine and having to change from GLES2 to GLES2 Anchor Centre and from GLES2 Anchor Centre I may have progressed further in the project and I could have put more time into enhancing the rendering of the map on the screen.

12.2. What is Still Outstanding

There is definitely a call for the whole social aspect of the original game idea to be implemented into this app. For me it wasn't one of the must have features of the game but I would be nice to see it implement.

I said from the start that I would be happy if I could display a map of a local area on screen and drive a car around the local streets that I would be happy and that is exactly what has been achieved.

All of the multiplayer support, world ranking leader boards and user accounts would be great extras to have in the game but as for the original proof of concept I think it is mission accomplished.

12.3. Attempting the Project Again

If I were to attempt the project again I would forget about trying to follow endless tutorials in order to develop a game. I spent week trying to follow those tutorials and in the end they were all a waste of time because I did not learn anything from them that I could implement in the app.

I would also try to do more research into the game engine. I also lost a lot of time while jumping back and forward from one version of the game engine to another.

Even though Eclipse gave me some difficulties I would probably use it again to develop this type of application. It's not because it is the best IDE out there, it's just that in today's Android game development world it seems to be the most popular.

12.4. Advice for Someone Else Attempting a Similar Project

I would advise them to spend a lot of time researching the technologies that they are going to use because all of the technologies that I researched thoroughly we're excellent. The really worked well together. I knew that they would because I put a lot of effort into selecting them.

The one technology that I had the most trouble with was AndEngine. The reason for this is because I had never even heard of it while I was researching game engines. I discovered it late into the development of the project.

The reason I decided to use it was I could see its potential of how easy it would be to use. If I had have spent the time to researching this game engine I feel that I could have prevented around 4 weeks of delays during the development of the project.

I know I would have had higher standard of a project that what I have now.

12.5. Technology Choices

I feel like I definitely made all of the right choices when it comes to the technologies that I used during this project. The finished project is proof in itself.

Even though I had a lot of issues with AndEngine I do believe that it was the best option that saw available to me.

13. Acknowledgements

I suggested to someone nearly twenty years ago that it would be cool if you could play a racing game where you could drive around your local streets. When I said the idea to someone I was told “Don’t be so stupid.”

So I would like to say huge thank you to Joseph Kehoe for all his help and guidance while developing this project. It has long been a dream of mine to see this application being made. Who would have thought the nearly twenty years on that I would have been the person who actually developed the application!

Also I would like to thank Nicolas Gramlich for his awesome work on developing AndEngine. (Gramlich, 2010)

I would like to thank Travis, RealMayo, Smartus and Matim for their work putting together tutorials for the likes of me to follow.

(Travis, 2012) (RealMayo, 2013) (Varga, 2013) (Matim, n.d.)

I’d like to thank for explaining the difference between GLES2 and GLES2 Anchor Centre.

(正宗白布鞋, 2013)

And finally to Hassanpur who created an awesome file downloader for android.

(Hassanpur, 2011)

Bibliography

Bort, J. (2013, May 15). *Google: There Are 900 Million Android Devices Activated*. Retrieved April 2014, from BusinessInsider: http://www.businessinsider.com/900-million-android-devices-in-2013-2013-5

Developers, A. (2010, December 22). *Libgdx vs. andengine*. Retrieved January 2014, from Groups.Google.com: https://groups.google.com/forum/#!topic/android-developers/xR6e9JNpwWM

DMOZ.org. (n.d.). *DMOZ - Reference: Maps: OpenStreetMap*. Retrieved April 2014, from dmoz.org: http://www.dmoz.org/Reference/Maps/OpenStreetMap/

Evangelho, J. (2013, 01 02). *2012's Massive Casual Gaming Footprint: Is The 'Hardcore' Audience Disappearing?* Retrieved April 2014, from Forbes.com: http://www.forbes.com/sites/jasonevangelho/2013/01/02/2012s-massive-casual-gaming-footprint-is-the-hardcore-audience-disappearing/

Galarneau, L. (2014, January 16). *2014 Global Gaming Stats: Who's Playing What, and Why?* Retrieved April 2014, from BigFishGames: http://www.bigfishgames.com/blog/2014-global-gaming-stats-whos-playing-what-and-why/

Google. (2014, April). *Dashboards*. Retrieved April 2014, from Developer.Android.com: http://developer.android.com/about/dashboards/index.html?utm_source=ausdroid.net

Gramlich, N. (2010, May 25). *nicolasgramlich (Nicolas Gramlich) · GitHub*. Retrieved January 2014, from Github.com: https://github.com/nicolasgramlich/

Hassanpur, M. (2011, April). *Android Development: Downloading a file from the web*. Retrieved April 2014, from Hassanpur.com: http://www.hassanpur.com/blog/2011/04/android-development-downloading-a-file-from-the-web/

Matim. (n.d.). *Full Game Tutorial - part 1 - introduction.* Retrieved February 2014, from Matim-Dev.com: http://www.matim-dev.com/full-game-tutorial---part-1.html

RealMayo. (2013, January 15). *AndEngine Tutorial 2013*. Retrieved February 2014, from YouTube.com: https://www.youtube.com/watch?v=lQW1WQOCri0

Travis. (2012, December 25). *1. Installing AndEngine GLES2 Tutorial | Android Game Development*. Retrieved February 2014, from http://www.mybringback.com/: http://www.mybringback.com/tutorial-series/12714/installing-andengine-tutorial/

Varga, M. (2013, September 08). *AndEngine Tutorial - Extensions and Examples*. Retrieved February 2014, from android.kul.is: http://android.kul.is/2013/09/tutorial-andengine-extensions-and-examples.html

Vodafone. (2014, April). *Samsung Galaxy Pocket Neo | Pay as you go | Vodafone Ireland*. Retrieved April 2014, from Vodafone.ie: http://shop.vodafone.ie/shop/phones-and-plans/phones/pay-as-you-go/phone-details/samsung-galaxy-pocket-neo-payg-1title

正宗白布鞋. (2013, November 05). *AndEngine - how to know the position for placing an object on a scene*. Retrieved March 2014, from stackoverflow.com: http://stackoverflow.com/questions/19712710/andengine-how-to-know-the-position-for-placing-an-object-on-a-scene