Location Racing

Technical Manual

By

Philip Stafford

Table of Contents

1. Installation Instructions

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 found in the following locations:

1.1.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

1.2.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.

1.3.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

1.4.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.

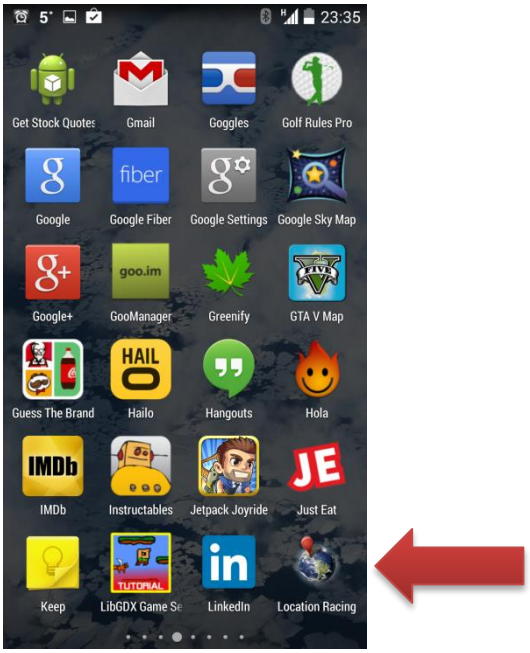
This application is only designed as a proof of concept so it is not ready to be used by the public.

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.

2. Loading a Game

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

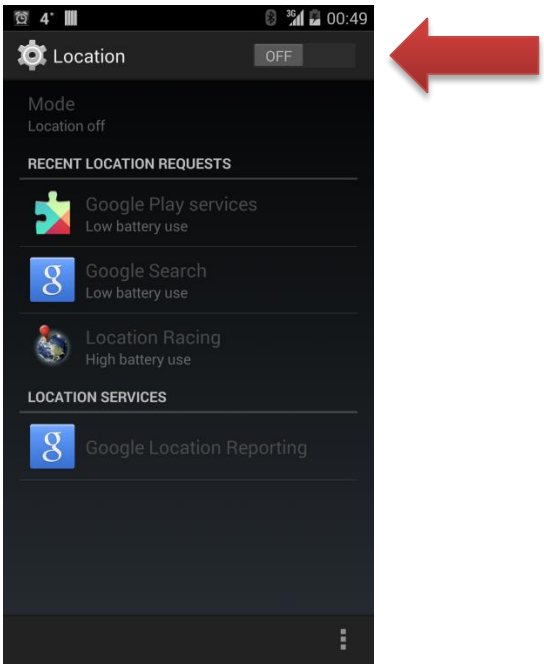
In order to use the Location Racing app the user must navigate to the Location Racing icon on their device and press it to load the application.



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.

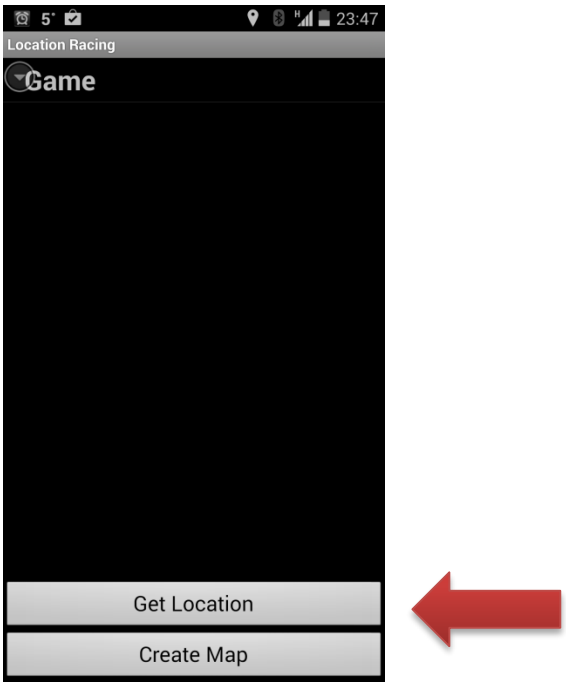


Once the user has loaded the application and the location services have been turned on then they will be presented with this screen.

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.

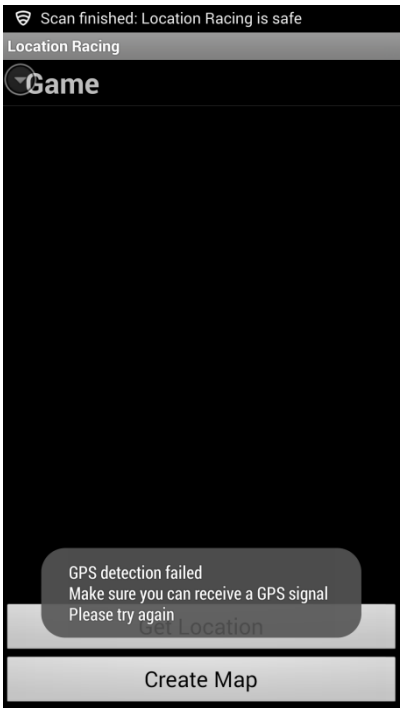


When the 'Get Location' button is pressed the app will attempt to get the location of the device.

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.

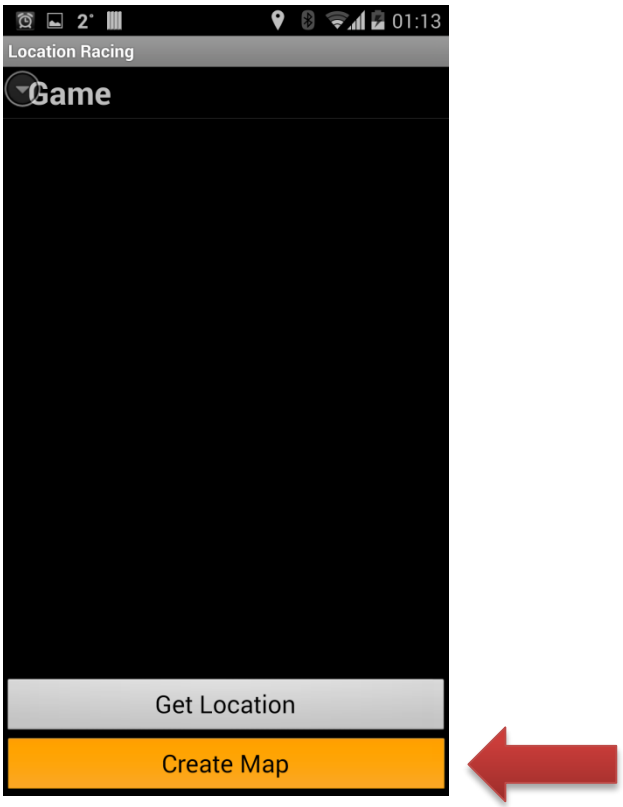
This will ensure that the device will detect its location.



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.

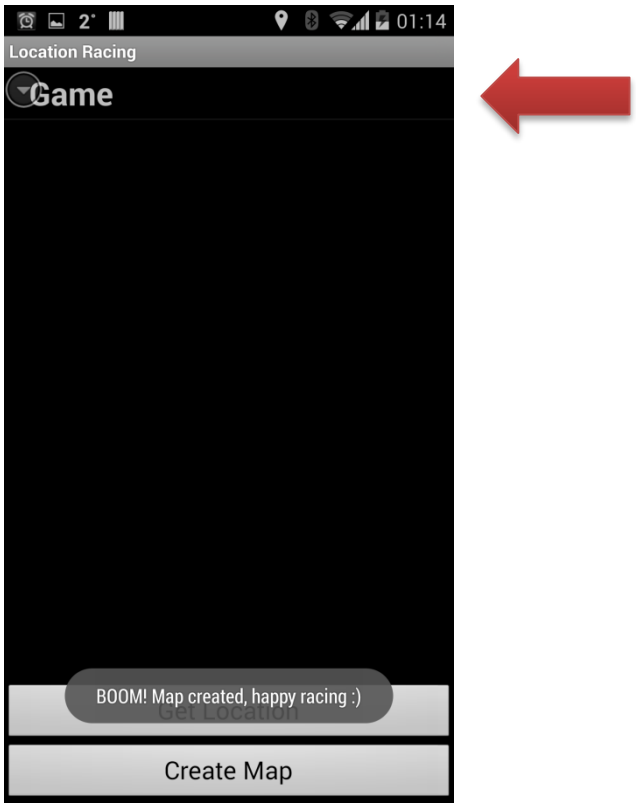
Once the user presses this button the application will take between 30 to 60 seconds to create the map. The 'Create Map' button will turn yellow and stay yellow until the map has been created.



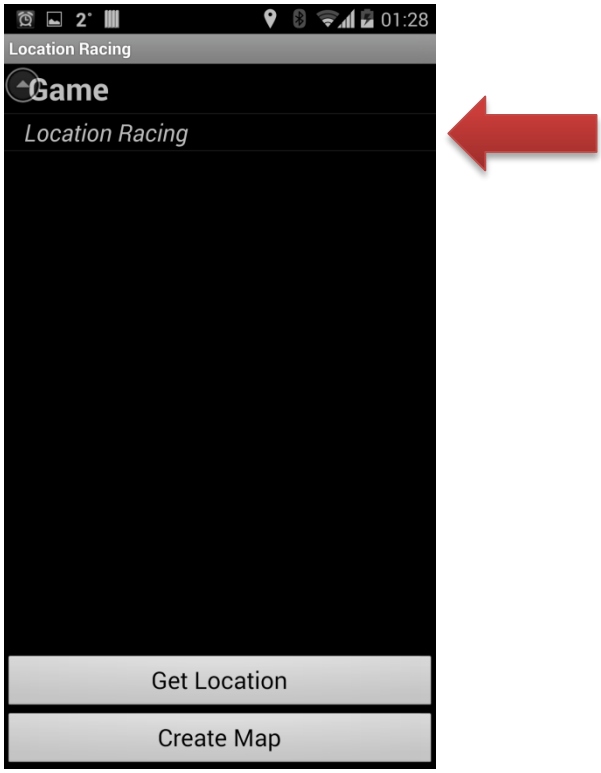
When the map has been successfully created a message will be displayed to the user to inform them that the map has been created successfully.

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

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



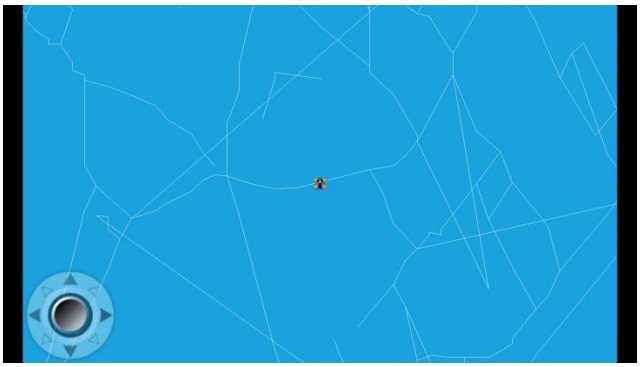
This is the last button press. In order to play the game the user has to press the 'Location Racing' button. Once this is pressed the map will begin to be drawn onto the screen.



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.



3. Algorithms & Data Structures

3.1.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.

3.2.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%2E4515992399999999%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)

3.3.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.

3.4.XML Parsing

3.4.1. 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.

3.4.2. 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.

3.5.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.

References

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/