Capture the Campus!
Interim Report

Submitted for the BSc in
Computer Science

January 17

By
Alexander C Whitehead

Word Count: 4247

Table of Contents

Table of Figures

1 Introduction

Capture the Campus! is a game based on a combination of; the classic 80’s arcade game Qix-wherein players control a character around a box taking squares of area while avoiding enemies (System 16, 2014). This can be seen in the image below (Figure 1)-Pokémon GO (Niantic, 2016), and Ingress (Niantic, 2016).

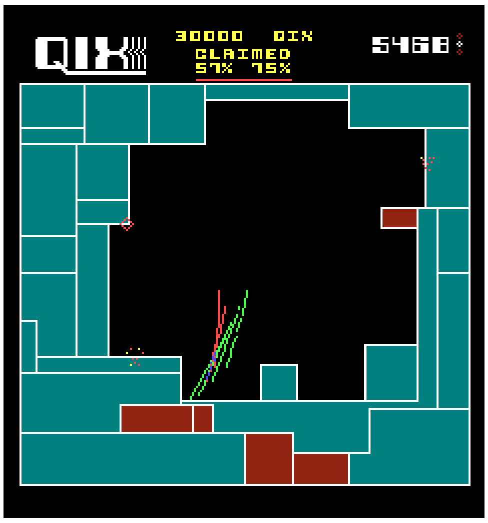


Figure 1: This is an image showing the gameplay of Qix (The International Arcade Museum, Museum of the Game, 2016).

The objective of the game will be to either in teams or individually capture parts of a definable area by physically traveling through it, the game-running on a mobile device in the player’s possession-will then track the player’s movements, this movement data will then be used to assign a score to the area captured by the player. Players can be ‘killed’ by their paths being intercepted by either another player or a computer controlled entity.
The game will end after either a predetermined total area has been captured or a time limit has been reached.
The original specification can be seen below (Appendix A:).

This is a report describing the interim design, research stages and project management issues of the project to build Capture the Campus!.
This report includes; A section covering the aims and objectives of the project, a section discussing background research conducted and creative and contextual decisions made thusly, a section further discussing the design of the project and creative and contextual decisions made regarding the system design, the UI design and the test design and a section discussing the management of the projects resources thus far.

2 Aim and Objectives

The aim of this project is to create a mobile platform based, augmented reality, Qix like, area control game

This aim will be achieved by completing the following objectives:

1. Create a Client Library
2. Create a TCP Server
3. Create a UDP Server
4. Add Multithreading to TCP Server
5. Add Logging to TCP Server
6. Create a Simple Map Based GPS Locator
7. Add None Player Entities
8. Add Single Player Gameplay
9. Add Multiplayer Gameplay
10. Add Team Based Multiplayer Gameplay

Objective 1 – Create a Client Library

Before a server can be created a basic client library must first be developed to test the prototype server with. The client should be able to send messages using UDP and TCP protocols. Development of the client shouldn't end until both it and the server are completed. The client library can be integrated into the game once completed and reused in future projects.

Objective 2 – Create a TCP Server

The TCP server will store all of the locations for each player and when traversing the playing area also their path, this information will be retrievable by other players to map their locations.
The TCP server should be able to run from a phone if necessary for local multiplayer.
This will be discussed further in the Networking (3.3) section of the Background.

Objective 3 – Create a UDP Server

The UDP server will be used as an initial contact point for the client, after receiving a UDP request the UDP server should respond with the IP address of the TCP server.
The UDP server should be able to run from a phone if necessary for local multiplayer.
This will be discussed further in the Networking (3.3) section of the Background below.

Objective 4 – Add Multithreading to TCP Server

The TCP server will have multithreading capabilities added as the TCP server should be able to accept multiple TCP requests at once to support several players simultaneously.

Objective 5 - Add Logging to TCP Server

The TCP server should be able to write a log of all connections for debugging.
The TCP server should be able to rebuild itself after a fatal error, for instance a crash, from this log file.

Objective 6 – Create a Simple Map Based GPS Locator

Before the game is created a simple map based GPS locator application should be developed, this should display the current player's location on a map. The map should translate, scale and rotate.
This application can then be transformed by the addition of gameplay into the intended product.
This will be discussed further in the Design (3.2) and Tracking (3.4) sections of the Background.

Objective 7 – Add Single Player Gameplay

Gameplay for a single player version of the game should be added including; a definable playing arena, area controlling mechanics, and scoring.

Objective 8 – Add Non-Player Entities

A non-player entity should be added that moves following a randomly generated meandering path.
This will be discussed further in the Design (3.2) and Tracking (3.4) sections of the Background.

Objective 9 – Add Multiplayer Gameplay

The client and server should be used to track the location of player in an online multiplayer session and apply killing mechanics to those who cross paths, the server should also track score.

Objective 10 – Add Team Based Multiplayer Gameplay

A game type should be implemented where gameplay is team based with score being cumulative and ending with a winning team rather than a wining individual.

3 Background

3.1 Example

Pokémon GO is a game developed by Niantic, the objective of which is to move around the real world attempting to capture virtual pets-which spawn randomly-faster than other real life people also playing the game (Niantic, 2016). The user interface of this game shows a player character which moves around a map the likes of which would work well in Capture the Campus!. Ingress-the predecessor to Pokémon GO-also uses map data and landmarks in a capture the flag style game (Niantic, 2016). This can be seen in the images below (Figure 2) (Figure 3).

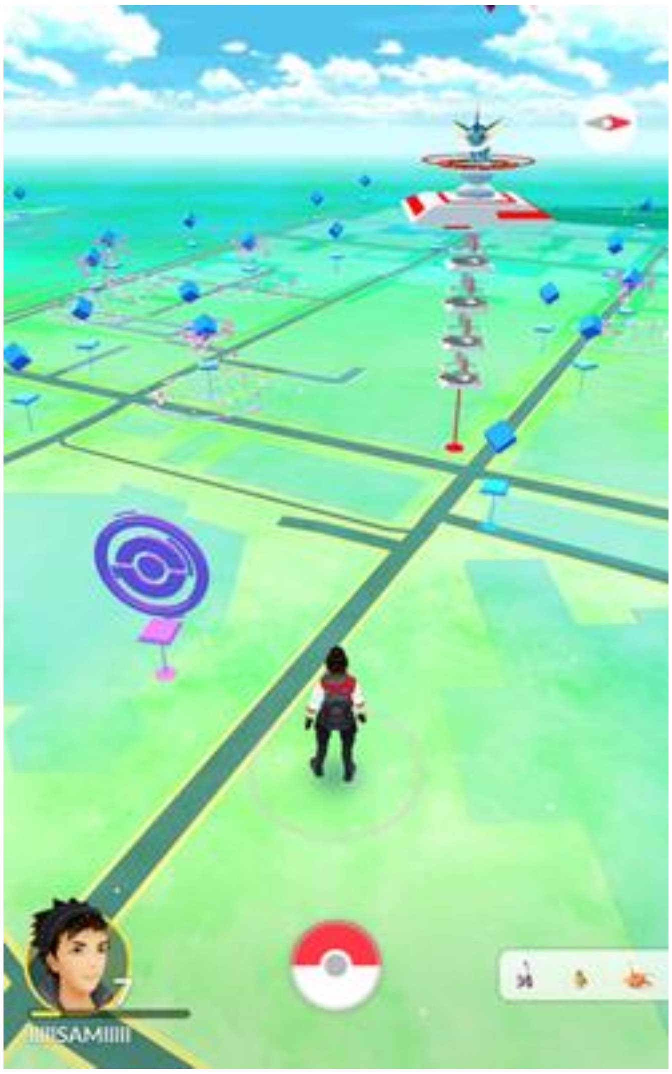
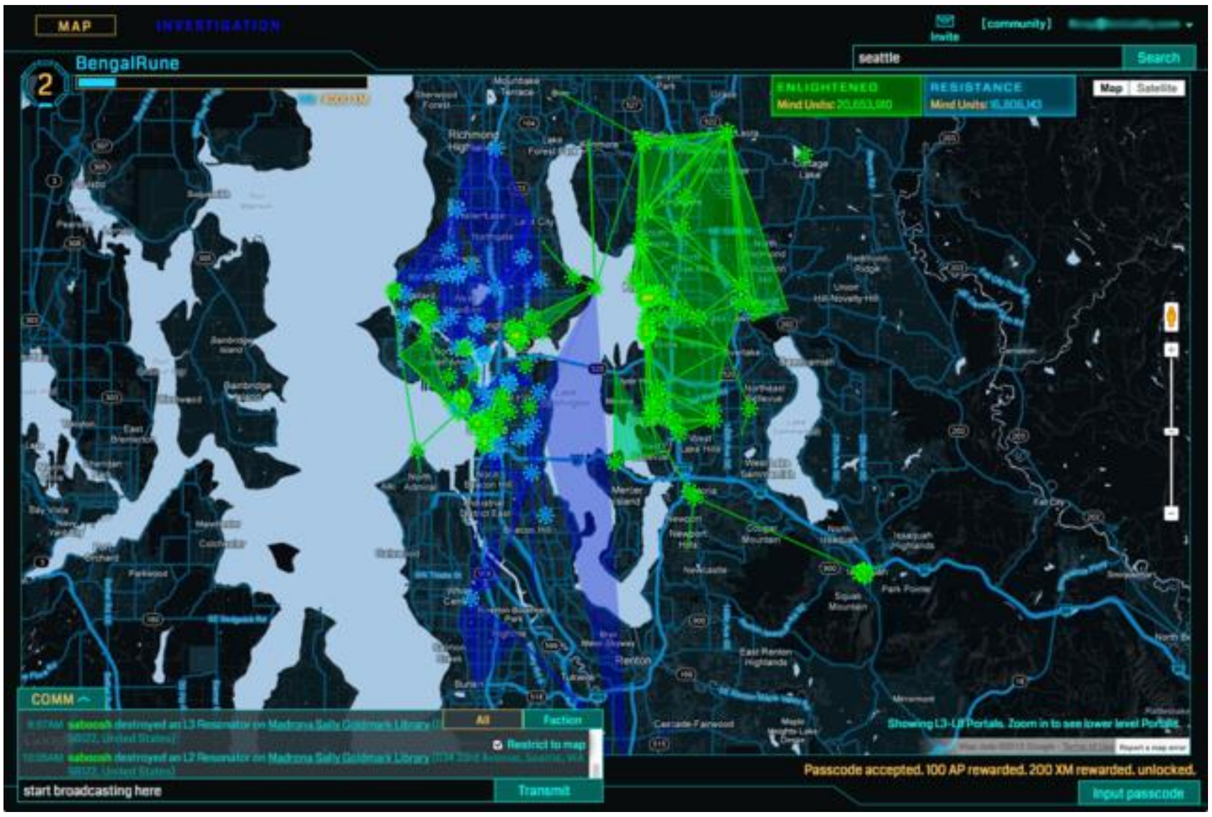


Figure 2: This image shows a standard game screen for Pokémon GO (Wikipedia, 2017).



sabocosh destroyed an L3 Resonator on Madrona Sally Goldmark Library

sabocosh destroyed an L2 Resonator on Madrona Sally Goldmark Library

start broadcasting here

Showing L3-L8 Portals. Zoom in to see lower level Portals.

Passcode accepted. 100 AP rewarded. 200 XM rewarded. unlocked.

Figure 3: This image shows a standard game screen for Ingress (Wikipedia, 2017).

One useful critique of Pokémon GO would be that even though the appeal of this game is as an augmented reality experience, there should be some none augmented reality or single player content so as to aid in accessibility for those that cannot get out and about to play the game (Credits, Extra, 2016).

UNIVERSITY OF Hull

3.2 Design

3.2.1 Platform

Firstly, it must be decided which platform the application will be run on. In this case it must be a mobile platform as it would be impossible to move something like a desktop computer around in order to play, also they do not usually contain tracking hardware.
The options for mobile platforms which contain tracking hardware are limited, they include; Android phones, iOS phones, Windows phones, and standalone trackers.

Out of the choice of mobile platforms standalone trackers can be disregarded almost immediately as the software should ideally be easily distributed, in order to run the software with a dedicated standalone tracker the tracker would probably have to come bundled with the software which is infeasible and expensive.

This leaves the choice of the three main mobile phone operating systems; Android, iOS, and Windows. Windows can be eliminated because it has such a small market share (Statista, 2016) This can be seen in the image below (Figure 4).

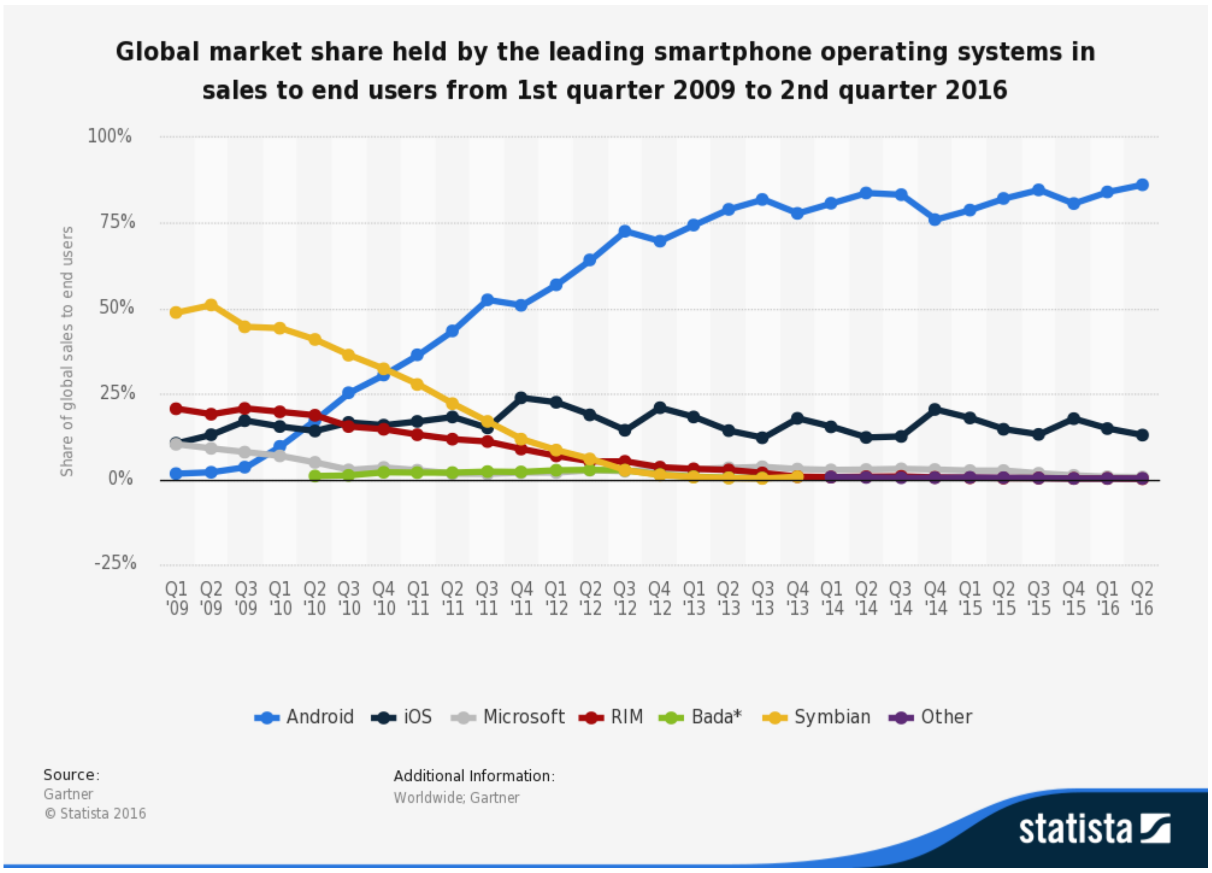


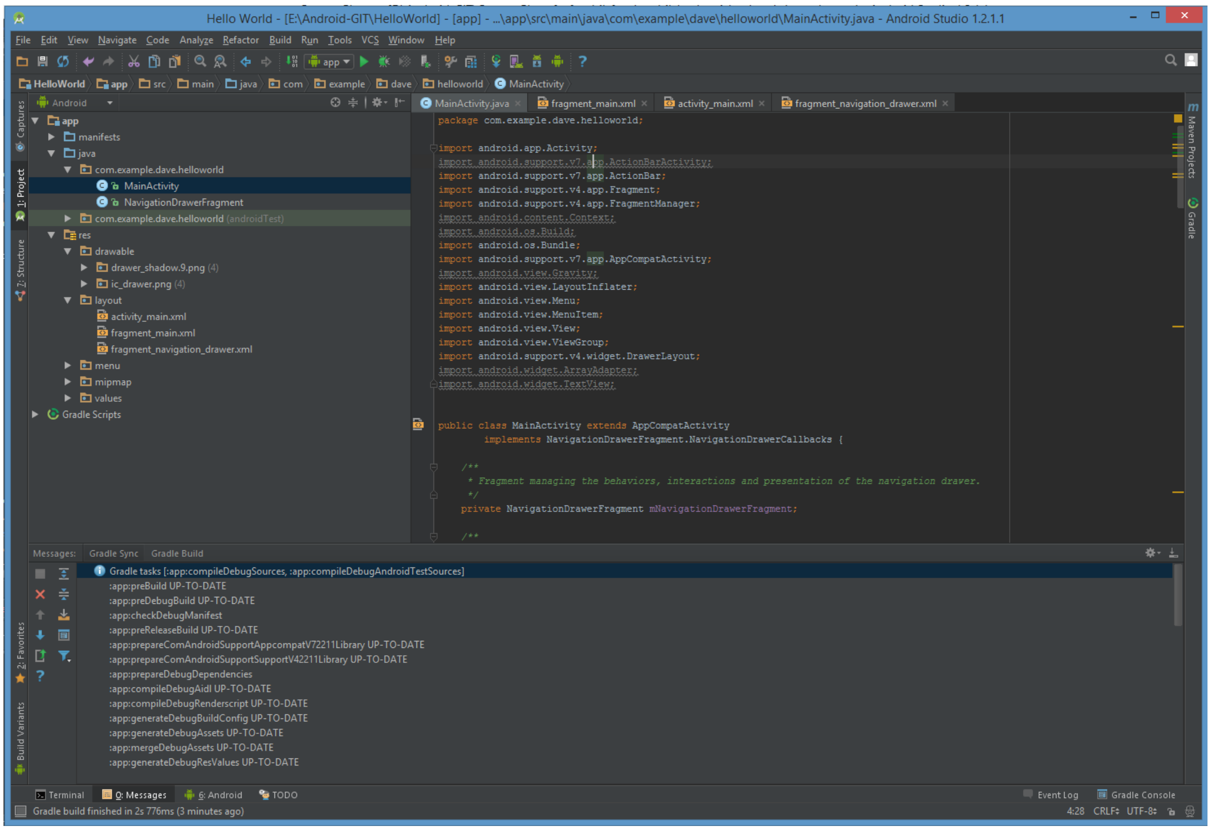
Figure 4: This image shows the current market share of mobile operating systems (Statista, 2016).

This leaves Android and iOS.
The market share for Android is greater than iOS. However, iOS users tend to on average spend more money on content (Sinicki, 2016). If the goal of this project was mass

distribution, then Android would be the best choice. However, if the goal was to make money, then iOS would be the best choice.
It can be said that Android is much easier to design, develop and publish for because; the terms used to convey design ideas in Android are less convoluted, and the Google play store is easier to publish games to, as anyone can upload their Android application package and be approved in minutes, whereas Apple insists on testing all apps on real-life humans which takes time (Sinicki, 2016).
Currently the features of both operating systems are too close to call a superior platform.

3.2.2 Language

In order to begin designing a project a language must be chosen to develop it in, for Android there is the option of developing in Java using Android Studio, in C# via Xamarin using Visual Studio or in a multitude of none native solutions. This can be seen in the images below (Figure 5) (Figure 6). for iOS there is the option of developing in Swift using Xcode or in C# via Xamarin using Visual Studio. This can be seen in the images below (Figure 7) (Figure 6).

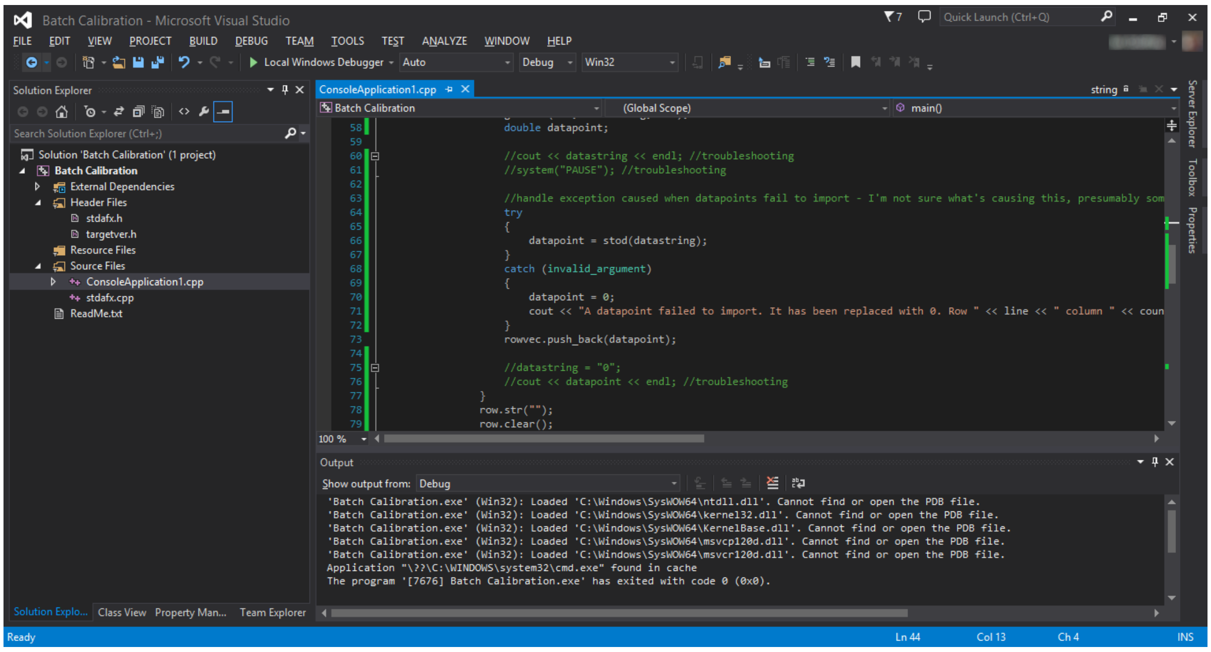

Hello World - [E:\Android-GIT\HelloWorld] - [app] - ...\app\src\main\java\com\example\dave\helloworld\MainActivity.java - Android Studio 1.2.1.1
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
HelloWorld app src main java com example dave helloworld MainActivity
Android
app
manifests
java
com.example.dave.helloworld
MainActivity
NavigationDrawerFragment
com.example.dave.helloworld (androidTest)
res
drawable
drawer_shadow.9.png (4)
ic_drawer.png (4)
layout
activity_main.xml
fragment_main.xml
fragment_navigation_drawer.xml
menu
mipmap
values
Gradle Scripts
MainActivity.java
fragment_main.xml
activity_main.xml
fragment_navigation_drawer.xml
Messages: Gradle Sync Gradle Build
Terminal 0: Messages 6: Android TODO Event Log Gradle Console
4:28 CRLF UTF-8

```
package com.example.dave.helloworld;

import android.app.Activity;
import android.support.v7.app.ActionBarActivity;
import android.support.v7.app.ActionBar;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.content.Context;
import android.os.Build;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.support.v4.widget.DrawerLayout;
import android.widget.ArrayAdapter;
import android.widget.TextView;


public class MainActivity extends AppCompatActivity
        implements NavigationDrawerFragment.NavigationDrawerCallbacks {

    /**
     * Fragment managing the behaviors, interactions and presentation of the navigation drawer.
     */
    private NavigationDrawerFragment mNavigationDrawerFragment;

    /**
```

```
Gradle tasks [:app:compileDebugSources, :app:compileDebugAndroidTestSources]
    :app:preBuild UP-TO-DATE
    :app:preDebugBuild UP-TO-DATE
    :app:checkDebugManifest
    :app:preReleaseBuild UP-TO-DATE
    :app:prepareComAndroidSupportAppcompatV72211Library UP-TO-DATE
    :app:prepareComAndroidSupportSupportV42211Library UP-TO-DATE
    :app:prepareDebugDependencies
    :app:compileDebugAidl UP-TO-DATE
    :app:compileDebugRenderscript UP-TO-DATE
    :app:generateDebugBuildConfig UP-TO-DATE
    :app:generateDebugAssets UP-TO-DATE
    :app:mergeDebugAssets UP-TO-DATE
    :app:generateDebugResValues UP-TO-DATE
```

Gradle build finished in 2s 776ms (3 minutes ago)

Figure 5: This image shows an example of a standard Android Studio integrated development environment window (Wikipedia, 2017).



```
            double datapoint;

            //cout << datastring << endl; //troubleshooting
            //system("PAUSE"); //troubleshooting

            //handle exception caused when datapoints fail to import - I'm not sure what's causing this, presumably som
            try
            {
                datapoint = stod(datastring);
            }
            catch (invalid_argument)
            {
                datapoint = 0;
                cout << "A datapoint failed to import. It has been replaced with 0. Row " << line << " column " << coun
            }
            rowvec.push_back(datapoint);

            //datastring = "0";
            //cout << datapoint << endl; //troubleshooting
        }
        row.str("");
        row.clear();
```

```
'Batch Calibration.exe' (Win32): Loaded 'C:\Windows\SysWOW64\ntdll.dll'. Cannot find or open the PDB file.
'Batch Calibration.exe' (Win32): Loaded 'C:\Windows\SysWOW64\kernel32.dll'. Cannot find or open the PDB file.
'Batch Calibration.exe' (Win32): Loaded 'C:\Windows\SysWOW64\KernelBase.dll'. Cannot find or open the PDB file.
'Batch Calibration.exe' (Win32): Loaded 'C:\Windows\SysWOW64\msvcp120d.dll'. Cannot find or open the PDB file.
'Batch Calibration.exe' (Win32): Loaded 'C:\Windows\SysWOW64\msvcr120d.dll'. Cannot find or open the PDB file.
Application "\??\C:\WINDOWS\system32\cmd.exe" found in cache
The program '[7676] Batch Calibration.exe' has exited with code 0 (0x0).
```

Figure 6: This image shows an example of a standard Visual Studio integrated development environment window (Wikipedia, 2017).

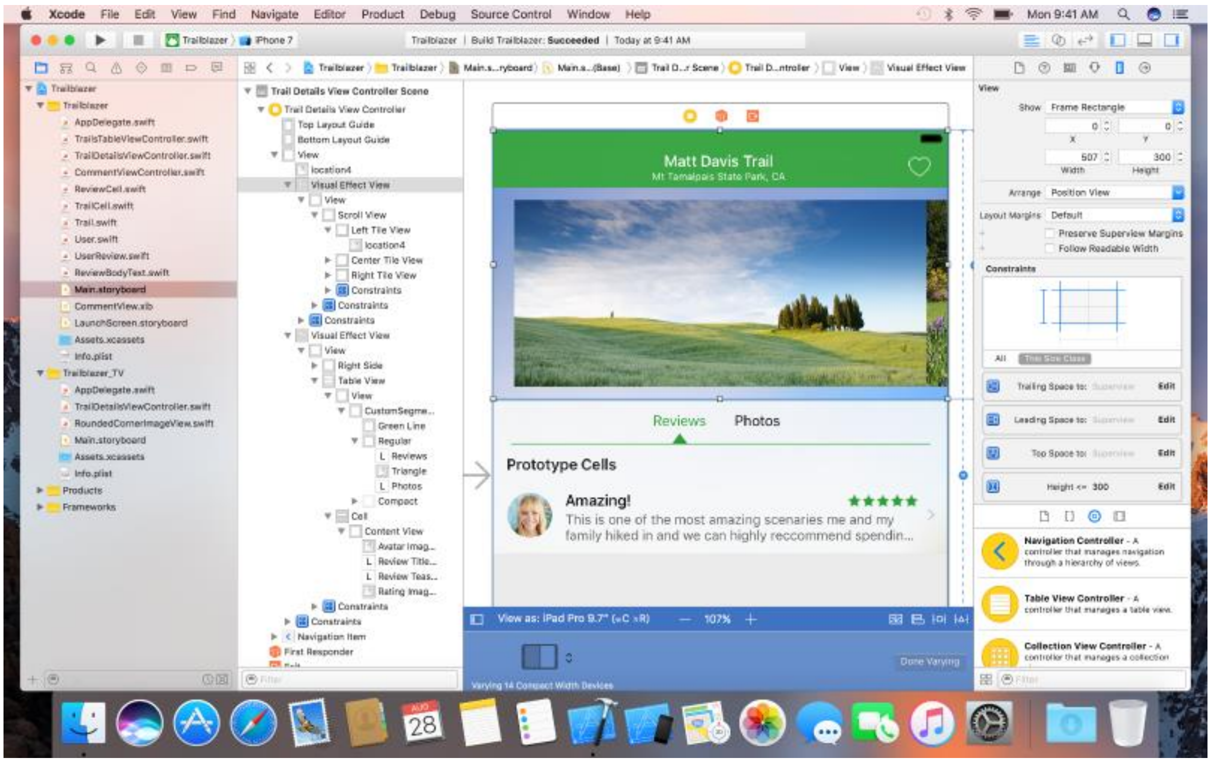

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Mon 9:41 AM
Trailblazer
iPhone 7
Build Trailblazer: Succeeded
Today at 9:41 AM
Trail Details View Controller Scene
Matt Davis Trail
Mt Tamalpais State Park, CA
Reviews
Photos
Prototype Cells
Amazing!
This is one of the most amazing scenaries me and my family hiked in and we can highly reccommend spendin...
View as: iPad Pro 9.7" (wC hR)
Navigation Controller
Table View Controller
Collection View Controller

Figure 7: This image shows an example of a standard Xcode integrated development environment window (Apple, 2017).

Both Java and Xamarin take advantage of native features of the Android operating system (Google, n.d.) (Xamarin Inc., 2016) and are therefore more suitable than none native solutions but no more suitable than each other for native Android development. However, Xamarin does allow for the use of standard .NET libraries (Montemagno, 2016) thus increasing the speed and efficiency of work flow due to developer knowledge of the standard .NET libraries.

Both Swift and Xamarin take advantage of native features of the iOS operating system (Apple, 2016) (Xamarin Inc., 2016) and are therefore no more suitable than each other for native iOS development. However, as previously discussed Xamarin allows for the use of standard .NET libraries (Montemagno, 2016).

Although as Xamarin allows for the porting of its code to both Android and iOS devices there is no need to choose one platform over another as the project can be created with both operating systems in mind thus opening up the theoretical user base that the product can appeal to from 86.2% to 99.1% (Xamarin Inc., 2016) (Statista, 2016). Therefore, C# via Xamarin using Visual Studios is the superior language in this use case.

3.3 Networking

To implement online multiplayer game functionality a networking component should be created. The network component will be created using Xamarin as it allows for the development of a cross platform solution that is easily debugged for rapid prototyping (Xamarin Inc., 2016). Xamarin will also allow for the network component to be run from both a desktop or mobile device, this is desirable as it allows the use of a predefined external server or alternatively a local server run from the client's mobile device.

There are a number of models of how and where a network component can be implemented and how and what will send and receive information where. Models include; peer-to-peer whereby each instance on a network shares its files equally with each other, with no central storage or authentication (Posey, 2000), and client/server where there are separate dedicated servers and clients (Posey, 2000). This can be seen in the image below (Figure 8).

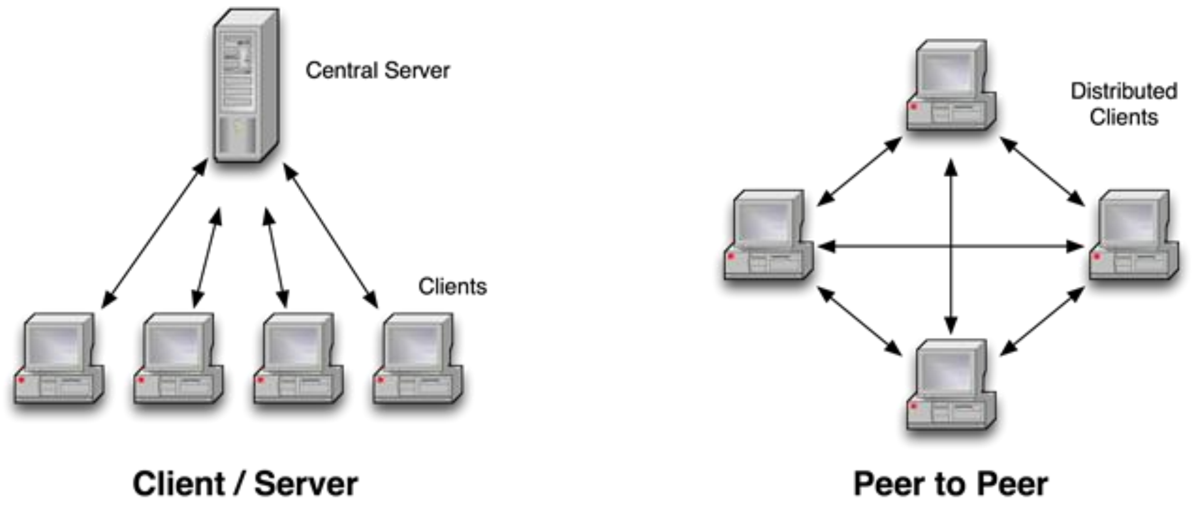


Figure 8: This image shows a visual model of the difference between a client/server model on the left and a peer-to-peer model on the right (Philips, 2014).

However, Peer-to-peer handles large numbers of users badly because, the whole network is bottlenecked by the slowest connection (Baccelli, et al., 2013). Conversely client/server handles larger numbers of users with ease as each person's connection is only as slow as their own connection to the server, but for this to work it is usually required that an offsite server or servers be dedicated to the game 24/7.

For this project a client/server model would be most efficient as there could possibly be an infinite number of players at any one time and the lag caused by this would be astronomical under a peer-to-peer architecture, also most people will be connecting to the network using mobile data so drops in connection will occur regularly causing others gaming experience to be affected if a peer-to-peer model was adopted.

There are also a number of protocols that the client and server could be written to accept, each with its own advantages and disadvantages in a given scenario. Protocols include; UDP (User Datagram Protocol) a low latency, low reliability protocol (TechTarget, 2015) (Seguin, 2014), and TCP (Transmission Control Protocol) a high latency (compared to UDP), high reliability connection-oriented protocol (TechTarget, 2014) (Diffen, n.d.).

Because of its low latency and general broadcast ability UDP is perfect for gaining an initial connection to a network component (Mey, n.d.), TCP is then useful once the initial connection has been verified to pass sensitive game data.

3.4 Tracking

Imperative to the implementation of this project would be some form of user tracking, in this case the Global Positioning System (GPS) (Brown & Sturza, 1995) will be used, this is because almost all modern mobile devices contain some form of GPS function (Zhao, 2002).

It is relatively simple to implement accurate GPS tracking through the use of the chosen language (Xamarin, 2017). The Google Play Services SDK allows access to both the Google Location Services API and the Fused Location Provider API. The Google Location Services API is used to attain the location of a device through the use of mobile and local internet connections (Google, 2017), this can be further refined through the use of the Fused Location Provider API which will take the output from the Google Location Services API and augment it with GPS location information to attain a highly accurate location without draining the mobile devices battery (Google, 2016).

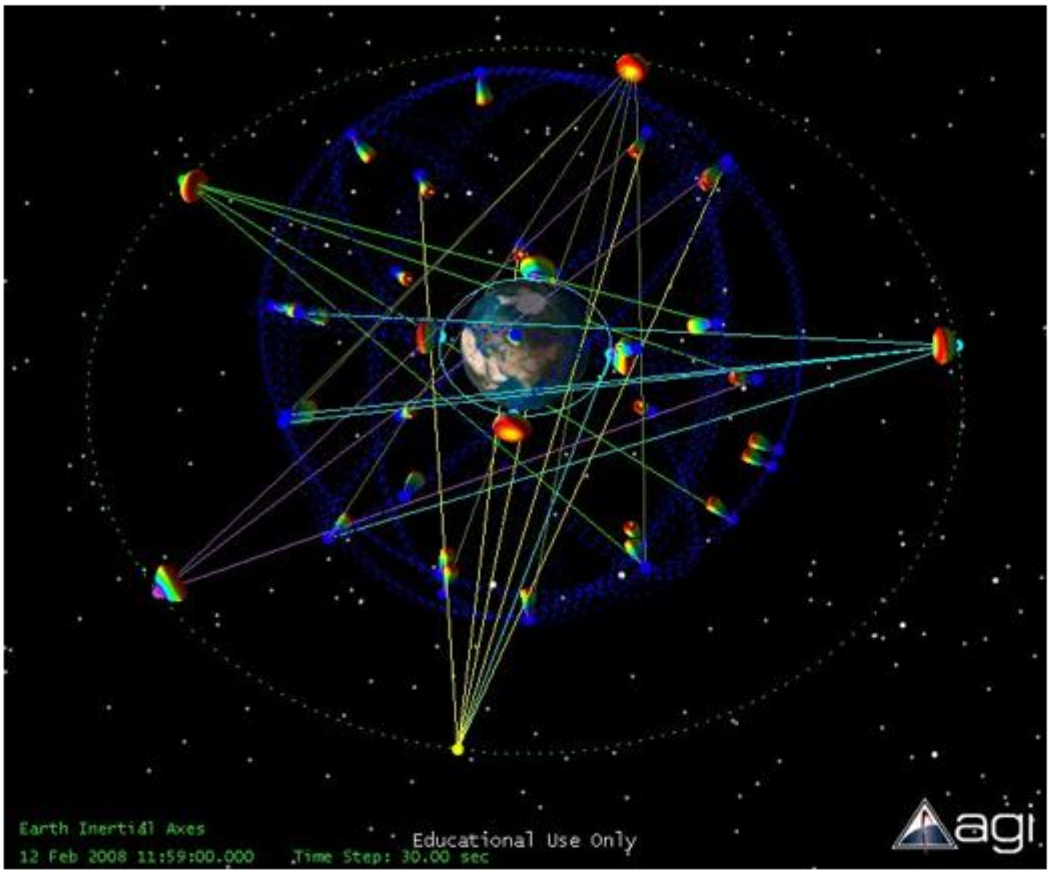


Figure 9: This image shows the orbits of some GPS satellites

GPS is usually accurate regardless of weather conditions wherever there is an unobstructed line of sight to at least four GPS satellites. Four satellites are required because one satellite is needed for every dimension of space plus time (Crato, 2010). GPS will operate under these conditions because it works not through the use of any mobile or internet connection but via a direct connection to satellites containing atomic clocks orbiting the Earth (Kaplan & Hegarty, 2006). This can be seen in the image above .(Figure 9)
Modulated electromagnetic waves carrying a pseudorandom code used to verify the identity of the sender and of the receiver and timing information regarding the exact time the message was sent are broadcasted to the satellites orbiting the Earth, these satellites then

respond with a similarly modulated signal containing the time that their internal clock believes it to be and their position (Mooney, 1985).

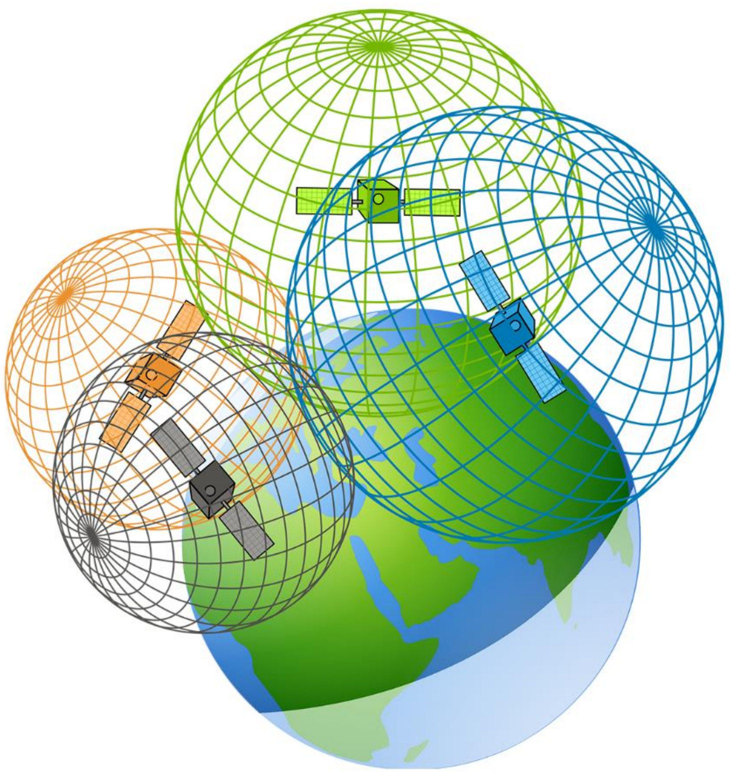


Figure 10: This image shows a visualisation of the act of GPS trilateration (openclipart, 2014)

When the sender receives responses from all satellites in range it uses the discrepancy in the time contained in all responses plus the location of these satellites to calculate a time of flight for each signal and therefore a spherical area around each satellite where the receiver could possibly be, the point where all spheres intersect is where the receiver is in three dimensional space (Ackermann, 1994). This can be seen in the image above (Figure 10).

4 Designs

4.1 System Design

Because of issues encountered during the initial design and implementation sections of the project only the server sections of the project have been fully designed and realised thus far, for more information regarding these issues please see the Project Management Review section of the report (5).
A broad overview of the entire system design can be seen below (Appendix E:).

4.1.1 Client Design

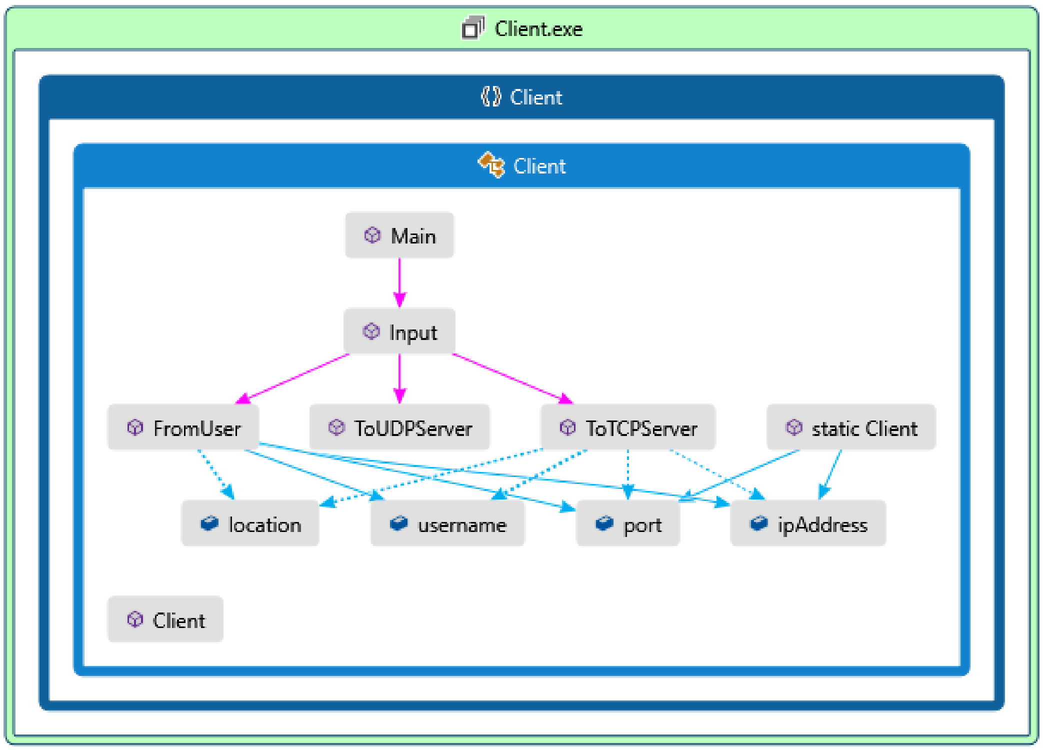


Figure 11: This image shows a diagram that describes the structure of the Client system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016)

The diagram above (Figure 11Figure 13Figure 13) shows the classes in the solution for the Client. As can be seen there is one class in this solution this is:

One which is used to communicate with both the TCP and UDP servers.
The methods contained within this class include the Input method which is the entry point into the class, the FromUser method which deals with taking the input from the user and parsing the relevant information into the variables within the class, the ToTCPServer method

which deals with taking the variables from within the class and creating relevant TCP requests to send to the TCP server and the ToUDPServer method which deals with sending and receiving UDP broadcasts to the UDP server.
This class is called Client.

These classes and their contents are private as their contents have no need to be edited from anywhere but from within themselves, being private allows for the application of encapsulation (Noble, 2002) (Berard, 2015).

4.1.2 TCP Server Design

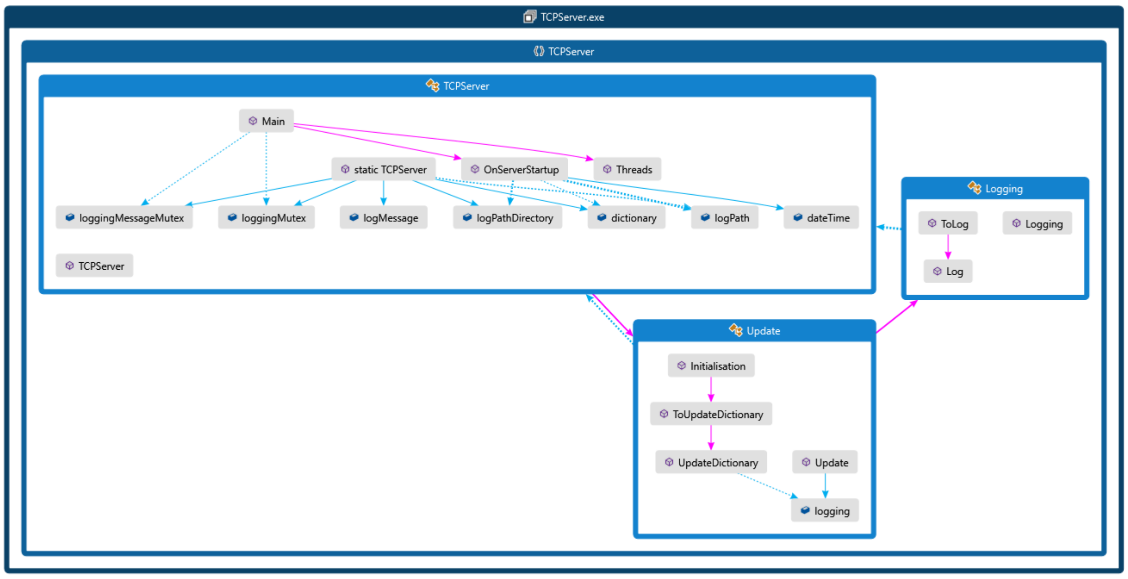


Figure 12: This image shows a diagram that describes the structure of the TCP server system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016)

The diagram above (Figure 12) shows the interaction between the classes in the solution for the TCP server. As can be seen there are three classes in this solution these are:

Firstly, one which contains the input to the server along with its database and control variables, some of these variables are used to perform a lock on the logging class to prevent issues with concurrency, this is known as enforcing a mutual exclusion concurrency control policy (Bernstein & Goodman, 1981).
The methods contained within this class include the Main method, the OnServerStartup method which deals with initialising the TCPServer including rebuilding the server if a log file exists or creating the log file if it does not and the Threads method which creates threads to deal with TCP requests.
This class is called TCPServer.

Secondly, one which is used to update the server database and to access the logging class. The methods contained within this class include the Initialisation method which is the entry point into the class, the ToUpdateDictionary method which allows access to the dictionary through a mutual exclusion concurrency control policy (Bernstein & Goodman, 1981) and the UpdateDictionary method which deals with updating the server database.
This class is called Update.

Finally, one which is used to update the server log file, the server log will be used to debug the code if necessary and can be used by the server to rebuild itself if it suffers from a fatal error.

The methods contained within this class include the ToLog method which allows access to the log file through a mutual exclusion concurrency control policy (Bernstein & Goodman, 1981) and the Log method which deals with updating the server log file.
This class is called Logging.

These classes and their contents are private as their contents have no need to be edited from anywhere but from within themselves, being private allows for the application of encapsulation (Noble, 2002) (Berard, 2015).

4.1.3 UDP Server Design

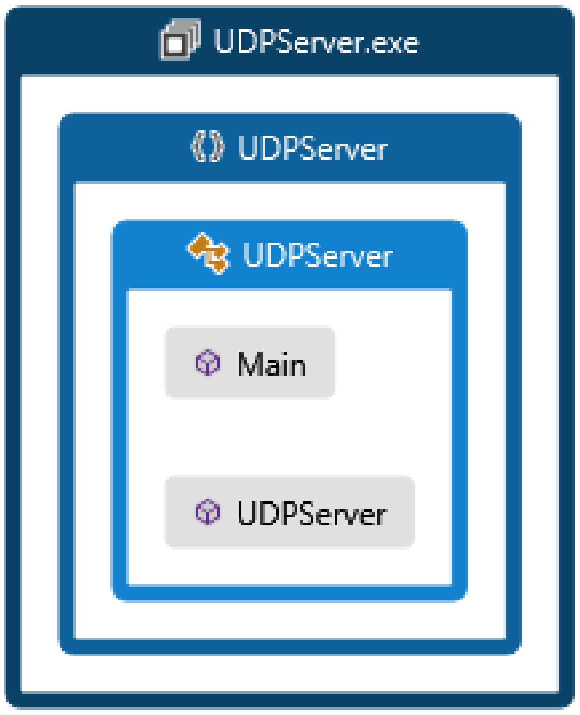


Figure 13: This image shows a diagram that describes the structure of the UDP server system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016)

The diagram above (Figure 13Figure 13) shows the classes in the solution for the UDP server. As can be seen there is one class in this solution this is:

One which contains the input to the server.
The methods contained within this class include the Main method and the UDPServer method which deals with receiving and transmitting UDP broadcasts.
This class is called UDPServer.

This class and its contents are private as its contents have no need to be edited from anywhere but from within itself, being private allows for the application of encapsulation (Noble, 2002) (Berard, 2015).

4.1.4 Watchdog Design

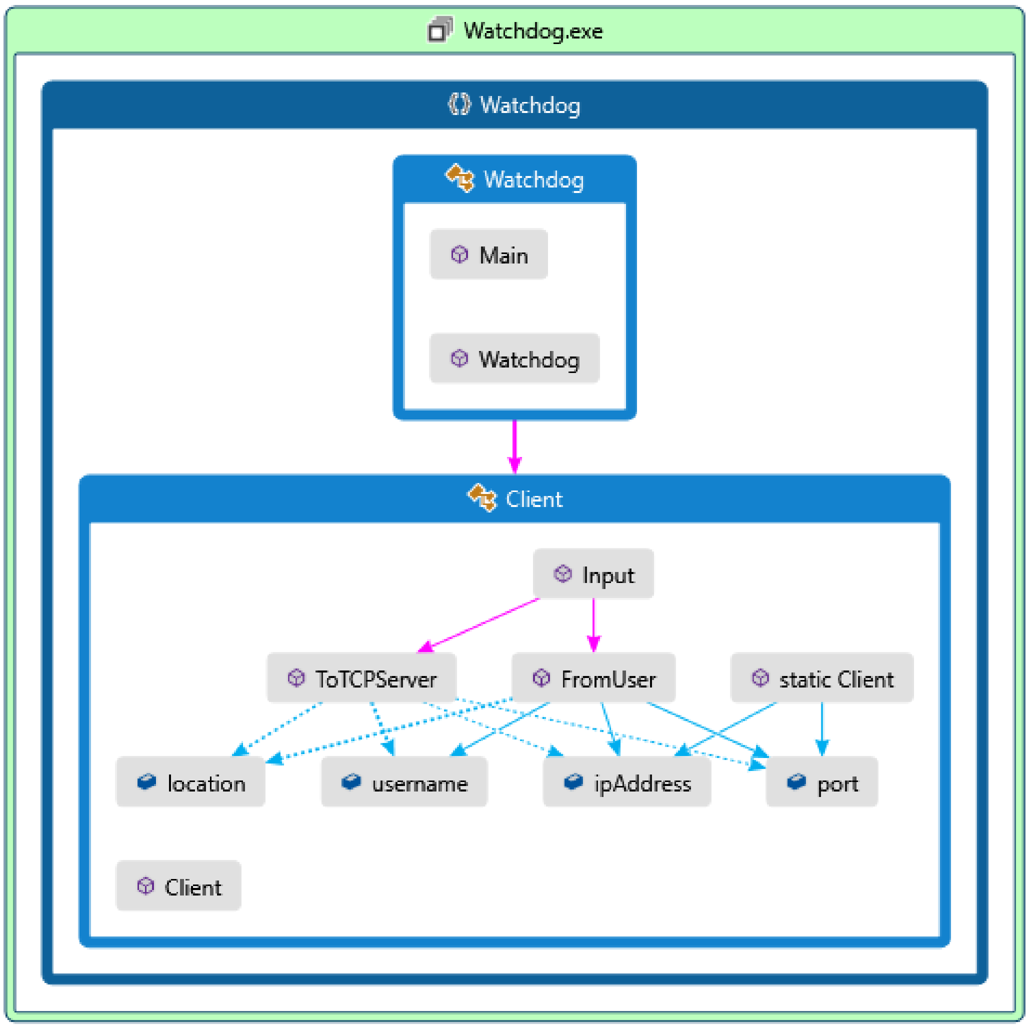


Figure 14: This image shows a diagram that describes the structure of the Watchdog and Client system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016)

The diagram above (Figure 14) shows the interaction between the classes in the solution for the Watchdog. As can be seen there are two classes in this solution these are:

Firstly, one which contains the input to the watchdog
The methods contained within this class include the Main method and the Watchdog method which deals with starting and maintaining both the TCP and UDP servers.
This class is called Watchdog.

Secondly, one which is used to communicate with the TCP server.
The methods contained within this class include the Input method which is the entry point into the class, the FromUser method which deals with taking the input from the user or a directly inserted input and parsing the relevant information into the variables within the class and the ToTCPServer method which deals with taking the variables from within the class and creating relevant TCP requests to send to the TCP.

These classes and their contents are private as their contents have no need to be edited from anywhere but from within themselves, being private allows for the application of encapsulation (Noble, 2002) (Berard, 2015).

4.2 User Interface Design

Because of issues encountered during the initial design and implementation sections of the project a formal user interface design document has not been created as the limitations of the development software to create the user interface have not yet been realised, for more information regarding these issues please see the Project Management Review section of the report (5).

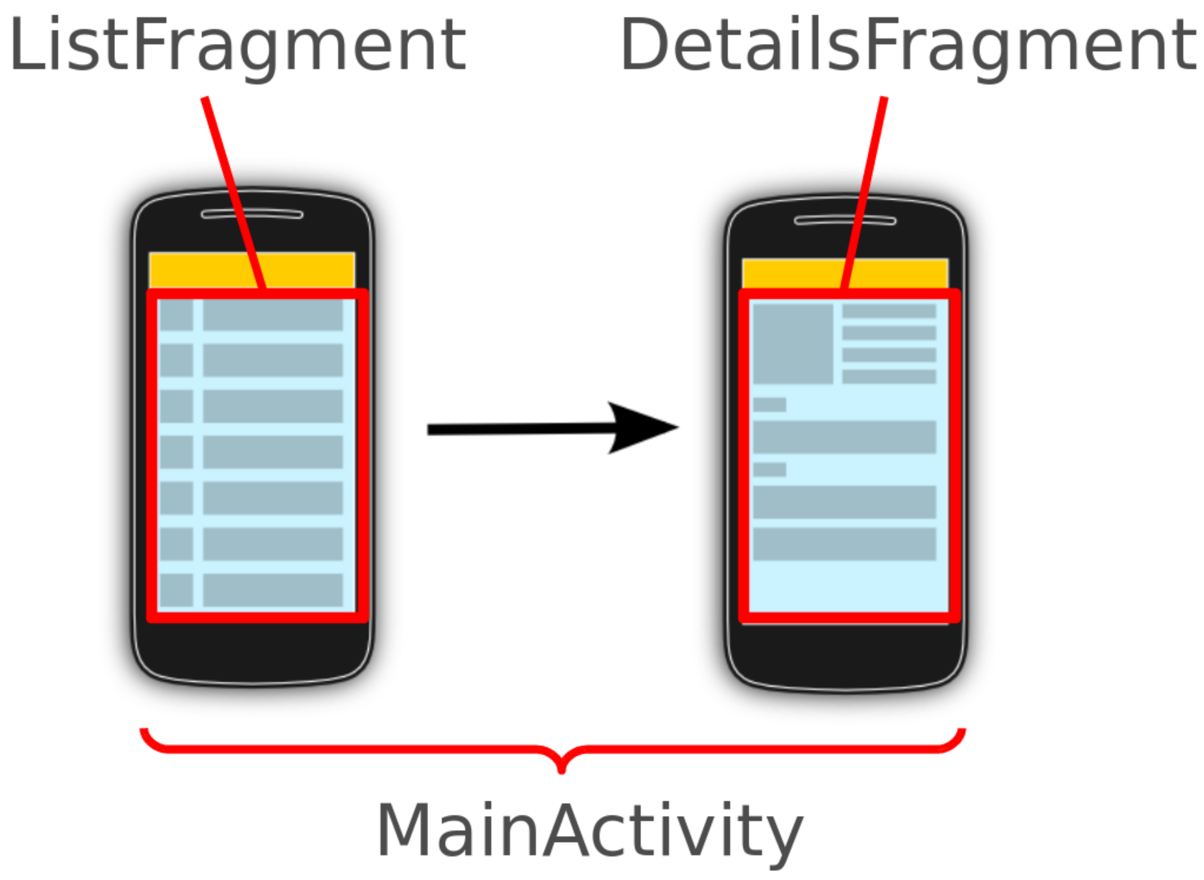


Figure 15: This image shows two examples of menu styles used in mobile application (Vogel, 2016)

The image above (Figure 15) shows two representations of menu styles used in mobile applications, design ideas can be taken from this image to aid in the visualisation of the projects user interface.

For this project a main menu is required to select the game mode to be played either single player, free for all, team death match or to open a server instance.
There are two main mobile application user interface design ideologies one which states that menus can follow lists called ListFragment and one that states that menus can follow clusters of information called DetailsFragment (Balagtas-Fernandez, et al., 2009) (Vogel, 2016).

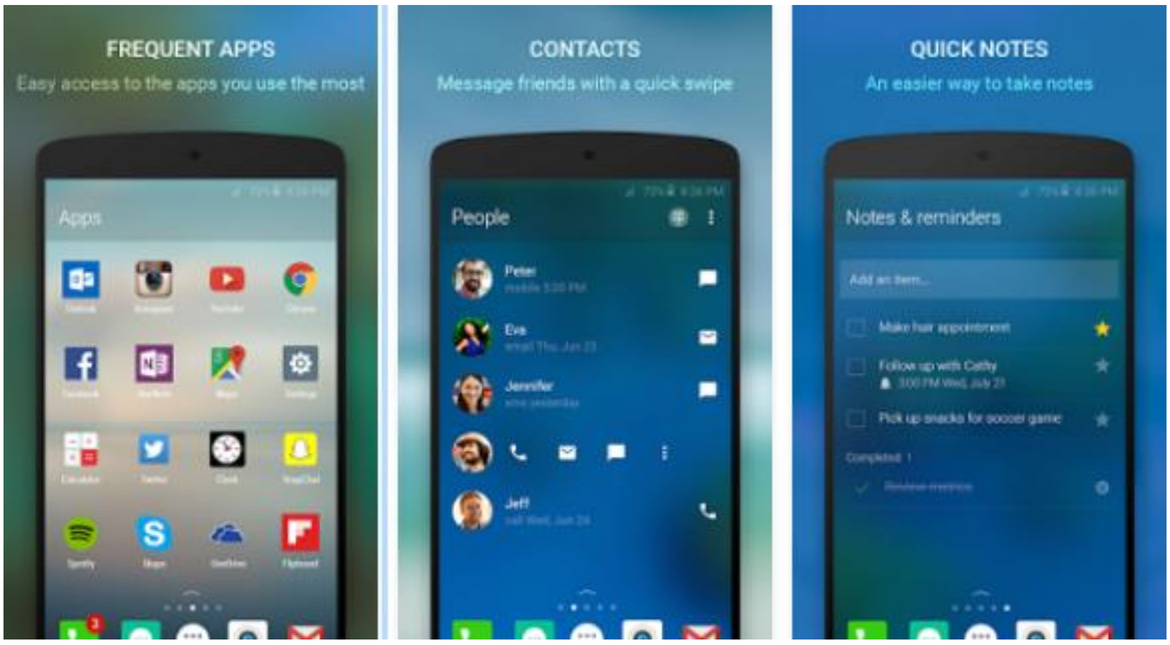
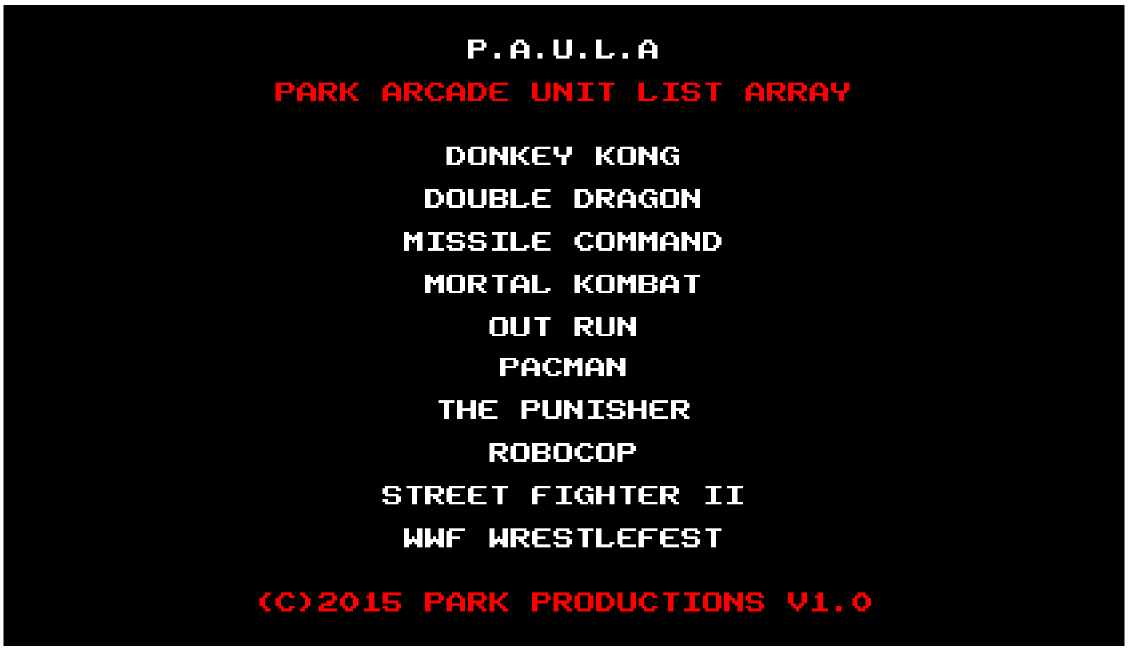


Figure 16: This image shows examples of the user interface for the Android launcher Arrow Launcher by Microsoft (Funk, 2015)

The image above (Figure 16) shows an example of both types of menu user interface, on the left there is an example of a DetailsFragment menu with boxes representing objects within the menu and in the centre and on the left there are ListFragment menus showing ordered lists.



P.A.U.L.A

PARK ARCADE UNIT LIST ARRAY

DONKEY KONG

DOUBLE DRAGON

MISSILE COMMAND

MORTAL KOMBAT

OUT RUN

PACMAN

THE PUNISHER

ROBOCOP

STREET FIGHTER II

WWF WRESTLEFEST

(C)2015 PARK PRODUCTIONS V1.0

Figure 17: This image shows an example of a retro arcade style menu . (Park Productions, 2015)

For this project in order to emulate Qix and other retro arcade games a ListFragment style ordered list menu will be adopted, an example of this can be seen in the image above (Figure 17).

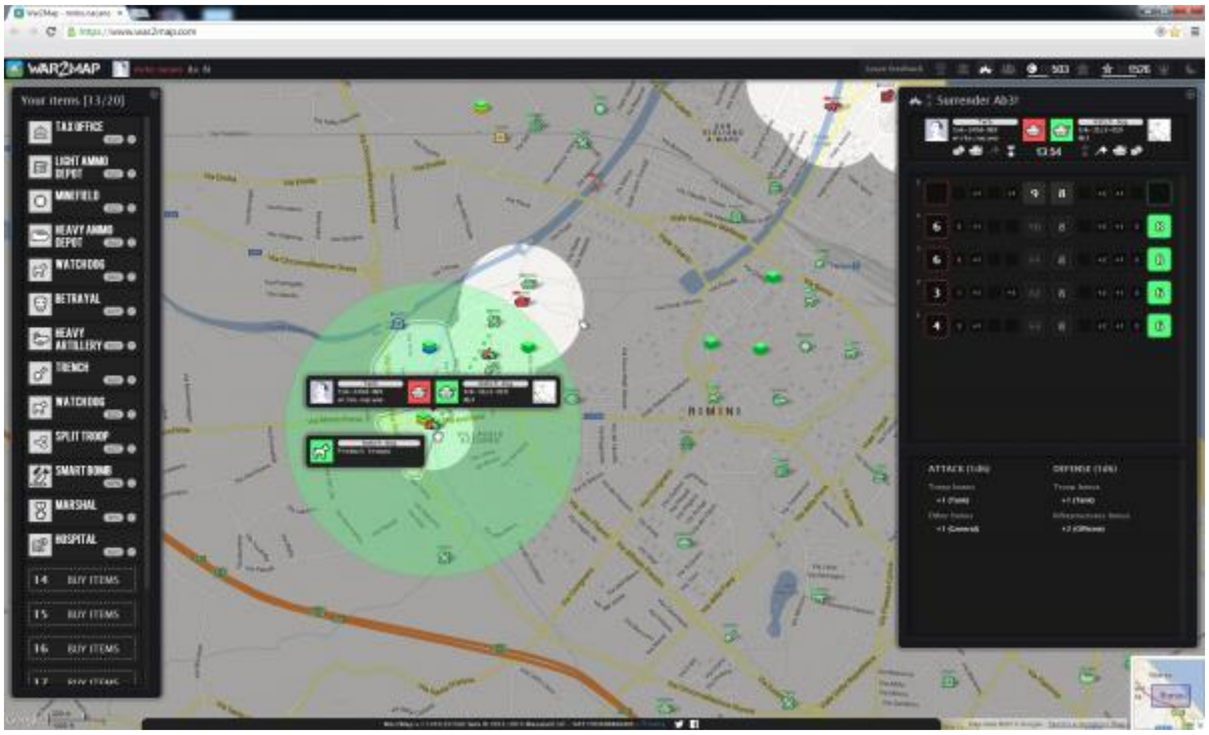


Figure 18: This image shows a standard game screen for the Google Maps API based game War2Map (Dempsey, 2013)

The image above (Figure 18Figure 18Figure 18) shows a representation of a game being played using the Google Maps API, design ideas can be taken from this image to aid in the visualisation of the projects user interface.

Firstly, dotting the map there are multiple instances of tokens representing game objects, this shows that the Google Maps API offers the function to draw images upon the map. The ability to draw images over the map could be used in this project to represent either player characters or non-player entities.

Secondly, a large green shape can be seen, this shows that the Google Maps API offers the function to draw polygons upon the map. The ability to draw polygons over the map could be used to in this project to represent either the play area and/or to represent the areas of the play area controlled by a player.

Thirdly, a dotted line can be seen spanning the distance between two avatars, this shows that the Google Maps API offers the function to draw a lines connecting two coordinates upon the map. The ability to draw lines connecting two coordinates over the map could be used in this project to represent the paths taken by each player character crossing the playing area.

In addition, down the left hand side of the screen a ListFragment ordered list has been drawn, this shows that the Google Maps API offers the function to draw a ListFragment ordered list upon the map. The ability to a ListFragment ordered list over the map could be used in this project to represent the score achieved by each player.

Finally, it can be assumed that the map retains its panning and scaling features.

4.3 Test Design

Because of issues encountered during the initial design and implementation sections of the project unit testing has not been implemented, for more information regarding these issues please see the Project Management Review section of the report (5).

An example of some unit tests that could be implemented would be:

#	Test	Result
1	Start Watchdog	Starts UDP server and TCP server
2	Wait more than 10 seconds	Closes Watchdog, UDP server and TCP server
3	Start Client	Starts Client
4	Start UDP server	Starts UDP server
5	Send broadcast to UDP server	Receive UDP server IP and port
6	Close UDP server	Closes UDP server
7	Start TCP server	Starts TCP server
8	Send username to TCP server	Receive error message
9	Send username followed by location to TCP server	Receive acknowledgement message
10	Send username to TCP server	Receive location
11	Close TCP server	Closes TCP server
12	Start TCP server	Starts TCP server
13	Send username to TCP server	Receive location
14	Close TCP server	Closes TCP server
15	Close Client	Closes Client

Tests 1-2 deal with testing the functionality of the Watchdog including; starting up and shutting down both the UDP and TCP servers.

Test 3 starts the debugging client.

Tests 4-6 deal with testing the functionality of the UDP server; including sending and receiving UDP broadcasts.

Tests 7-14 deal with testing the functionality of the TCP server; including throwing errors, recording usernames and locations, recalling a given location for a given username and rebuilding the database after shutting down.

Test 15 shuts the debugging client down.

5 Project Management Review

Generally, progress on the project thus far has been stunted by multiple factors:

Firstly, the main computers that were chosen to be used to complete the project did not in fact have the required development software installed to develop the project. In addition, the development software has not been installed by the administrators of the computers as of the writing of this report, plus the administrators of the computers will not allow administrative access to the computers to allow the instillation of the development software. Furthermore, the backup computer-that was to be used in situations like this-that has the development software installed-does not have the required specifications to run the development software to an adequate performance level. This means that anything that requires this development software has not begun development yet.
However, this does mean that the extra time available has been invested to fully optimise and add additional features-that would not have otherwise been added-to the areas of the project that could be developed without this development software.

Secondly, the initial time plan (0) did not take into account other extraneous commitments when allotting time to this project. Time must be allowed to complete other compulsory projects on time, as such it appears that this project is behind schedule when in fact the initial time plan was inherently flawed and unachievable.

Thus a new task list (5.1) and time plan (**Error! Reference source not found.**5.10) has been created taking into account the issues mentioned above:

The tasks that require the specific development software mentioned above have been delayed until after this report has been completed, this should give enough time for a solution to the problem to present itself, either the instillation of the required development software on the main computers, an upgrade of the backup computer or a change of development software.

Sections of time have been blocked out in order for the completion of extraneous compulsory commitments, this includes the reworking of the remaining time allocated to this project in order to ensure the smoothest development experience possible.
The addition of a multigame server has been abandoned since the initial report stage as the scope of the project has had to be reduced because of the issues mentioned above.

Because of the delays caused by the lack of development software sections have had to be amended to the risk analysis (Appendix D:) section to reflect any risks that have been encountered that were not originally planned for. Including; a section detailing the risk of missing development software, a section detailing the risk of the backup computer being insufficient to develop on and a section detailing the risk of insufficient funds to upgrade the backup computer.

Because of the nature of this project there are practically no ethical concerns. There is no intention to conduct any kind of experiment so ethical approval is not needed and any other ethical concerns, for example, use of the project ending in the injury of members of the public have been covered in the risk analysis section (Appendix D:) and/or are beyond the scope of the project.

5.1 New Task List

#	Task Name	Description	Duration (weeks)
1	Research	Conduct research that will aid in the writing of reports and initial designing of the project	9
2	Initial report	Write the initial report deliverable	2
3	Create server client library	Create a client library that is capable of interfacing with the server	5
4	Create TCP server	Create a TCP server that can accept TCP packets from the client	2
5	Create UDP server	Create a UDP server that can accept UDP packets from the client and can be used to identify the IP of the server	1
6	Add multithreading capabilities to server	Add multithreading to the server so that it is capable of accepting more than one client request at a time	2
7	Add logging capabilities to the server	Add logging to the server so that it is capable of recovering from a fatal error	2
8	Work on other projects	Time set aside to work on other projects	6
9	Interim report	Write the interim report deliverable	3
10	Create main menu for game	Create a main menu for the game that will be displayed when the game is started and between every game instance. The main menu should display all options for game types and settings	1
11	Add game screen and assets to game	Add a game screen to the game that is displayed once the play game option is selected and also add assets to the game to be used to display player characters	2
12	Add map to game screen	Add a suitable map to the game screen	2

13	Add translations and scaling to map	Add translations and scaling to the map so that it is possible to move the map around and zoom in and out	2
14	Add player character and movements to map	Make the player character move as the player moves, this should work via GPS	2
15	Add non-player entities	Add a non-player entity that is capable of killing the player, this entity should move following a randomly generated meandering path	2
16	Final report	Write the final report deliverable	14
17	Add client calls to store and recall player positions from server	Make the game send its current location to the server at a reasonable interval and also make it so that the game requests the location of every other player	2
18	Add tracking data and bounds of playing field to game	Make it so that the game then draws all the players at the correct locations and that it is possible to create the area of play	2
19	Add taking mechanics from tracking data to game	Make it so that when a player completes a run from one side of the playing area to another they take the smallest area for their own team	2
20	Add killing mechanics from tracking data	Make it so that if a player crosses the path of another active player one of the players dies	2
21	Add scoring data to game	Make it so that the score of all players is tracked based on the area of land taken	2

22	Work on other projects	Time set aside to work on other projects	7
23	Add team mode to game	Make it so that players can play in teams	3

5.2 New Time Plan

#	Task Name	University Calendar Weeks																																
		4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
1	Research	■	■	■	■										■	■	■	■	■															
2	Initial report			■	D																													
3	Create server client					■	■	■	■	■																								
4	Create TCP server						■	■																										
5	Add UDP to server								■																									
6	Add multithreading capabilities to server								■	■																								
7	Add logging capabilities to the server								■	■																								
8	Work on other projects								■	■	■	■	■	■																				
9	Interim report																■	■	D															
10	Create main menu for game																			■														
11	Add game screen and assets to game																			■	■													
12	Add map to game screen																			■	■													
13	Add translations																			■	■													

	and scaling to map																			■	■													
14	Add player character and movements to map																				■	■												
15	Add non-player entities																				■	■												
16	Final report																				■	■	■	■	■	■	■	■	■	■	■	■	■	D
17	Add client calls to store and recall player positions from server																					■	■											
18	Add tracking data and bounds of playing field to game																						■	■										
19	Add taking mechanics from tracking data to game																							■	■									
20	Add killing mechanics from tracking data to game																								■	■								
21	Add scoring data to game																									■	■							
22	Work on other projects																									■	■	■	■	■	■	■		
23	Add team mode to game																										■	■						

Appendix A: Capture the Campus!

Capture the flag is a well-known game (sub-) genre that requires players to capture the flag. Often there are two opposing teams each of which has a flag that must be defended from the other team.

http://en.wikipedia.org/wiki/Capture_the_flag

This project requires the student to design and develop an augmented reality game for a GPS-enabled mobile device (preferably Windows Phone 7). The details of the gameplay are open to negotiation, but a suggestion is that a number of flags (or capture points) are distributed in GPS locations around the campus. The players are required to visit the location for a specified time period to capture the location. The players accumulate score based on number of capture points held and time they are uncontested for. The status of the capture points should be persistent, with the game's progress being able to be tracked over multiple (perhaps unlimited) days.

The project involves databases for storing flag locations and capture logs etc.

Project Code: DJP3

Appendix B: First Task List

#	Task Name	Description	Duration (weeks)
1	Research	Conduct research that will aid in the writing of reports and initial designing of the project	10
2	Initial report	Write the initial report deliverable	2
3	Create server client library	Create a client library that is capable of interfacing with the server	5
4	Create TCP server	Create a TCP server that can accept TCP packets from the client	2
5	Add UDP to server	Add UDP to the server that can be used to identify the IP of the server	1
6	Add multithreading capabilities to server	Add multithreading to the server so that it is capable of accepting more than one client request at a time	2
7	Create main menu for game	Create a main menu for the game that will be displayed when the game is started and between every game instance. The main menu should display all options for game types and settings	1
8	Add game screen and assets to game	Add a game screen to the game that is displayed once the play game option is selected and also add assets to the game to be used to display player characters	2
9	Add map to game screen	Add a suitable map to the game screen	3
10	Add translations and scaling to map	Add translations and scaling to the map so that it is possible to move the map around and zoom in and out	2
11	Add player character and	Make the player character move as the player moves, this should work via GPS	2

	movements to map		
12	Interim report	Write the interim report deliverable	3
13	Add client calls to store and recall player positions from server	Make the game send its current location to the server at a reasonable interval and also make it so that the game requests the location of every other player	3
14	Final report	Write the final report deliverable	14
15	Add tracking data and bounds of playing field to game	Make it so that the game then draws all the players at the correct locations and that it is possible to create the area of play	2
16	Add taking mechanics from tracking data to game	Make it so that when a player completes a run from one side of the playing area to another they take the smallest area for their own team	2
17	Add killing mechanics from tracking data	Make it so that if a player crosses the track of another active player one of the players dies	2
18	Add scoring data to game	Make it so that the score of all players is tracked based on the area of land taken	2
19	Add team mode to game	Make it so that players can play in teams	3
20	Add multigame server	Make it so that multiple game instances can run on one server	3

Appendix C: First Time Plan

#	Task Name	University Calendar Weeks																																
		4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
1	Research	■	■	■	■										■	■	■	■	■	■														
2	Initial report			■	D																													
3	Create server client					■	■	■	■	■																								
4	Create TCP server						■	■																										
5	Add UDP to server								■																									
6	Add multithreading capabilities to server								■	■																								
7	Create main menu for game										■																							
8	Add game screen and assets to game										■	■																						
9	Add map to game screen											■	■	■																				
10	Add translations and scaling to map												■	■																				
11	Add player character and movements to map												■	■																				
12	Interim report														■	■	D																	

13	Add client calls to store and recall player positions from server																			■	■	■												
14	Final report																			■	■	■	■	■	■	■	■	■	■	■	■	■	D
15	Add tracking data and bounds of playing field to game																					■	■										
16	Add taking mechanics from tracking data to game																						■	■									
17	Add killing mechanics from tracking data to game																							■	■								
18	Add scoring data to game																								■	■							
19	Add team mode to game																										■	■	■				
20	Add multigame server																													■	■	■	

Appendix D: Risk Analysis

Risk	Current Risk			How to Avoid	How to Recover	Residual Risk		
	Severity (L/M/H)	**Likelihood (L/M/H)**	**Significance (Severity, Likelihood)**			**Severity (L/M/H)**	**Likelihood (L/M/H)**	**Significance (Severity, Likelihood)**
Data loss	H	M	HM	Keep backups	Reinstate from backups	L	M	LM
Loss of backups	H	L	HL	Keep multiple backups	Use alternate backup	L	L	LL
Underestimate workload	H	M	HM	Regularly review progress against Time Plan	Invest more time into work, possible reduction of objectives	H	L	HL
Critical error in deliverable	H	M	HM	Perform adequate research	Debug code	H	L	HL
Skill Risk	M	M	MM	Perform adequate training	Invest more time into research	L	L	LL
Scope Creep	M	H	MH	Fully define scope	Define scope at current point	M	L	ML
Inefficient Program Performance	H	L	HL	Spend time testing code	Remove extraneous features	M	L	ML

Server Crashes	M	M	MM	Attempt to optimize server code	Implement alternative servers	L	L	LL
Incompatible with target device	H	L	HL	Create program with target device specifications in mind	Create alternative version for target device	L	L	LL
Medical emergency	H	L	HL	Care for developers health	Comment code regularly so that it is well understood	M	L	ML
Development software unavailable during demonstration	H	H	HH	Place key files into a backup	Download key files	L	L	LL
Development software unavailable during development	H	L	HL	Ensure development software is installed on development computer	Keep a backup computer to use in the case that the development software cannot be installed on the main computers	M	L	ML

Backup computer insufficient to develop on	H	M	HM	Ensure development software is installed on development computer so the backup computer is unnecessary	Upgrade the backup computer	H	L	HL
Insufficient funds to upgrade backup computer	H	H	HH	A way to avoid this risk would be beyond the scope of this project.	Change the development software being used	H	H	HH
User injured while using application	H	M	HM	Only allow authorized access to application while in development and provide warning messages while in play	Medical attention may be required if a user is injured while using the application	H	L	HL

Appendix E: Code Map

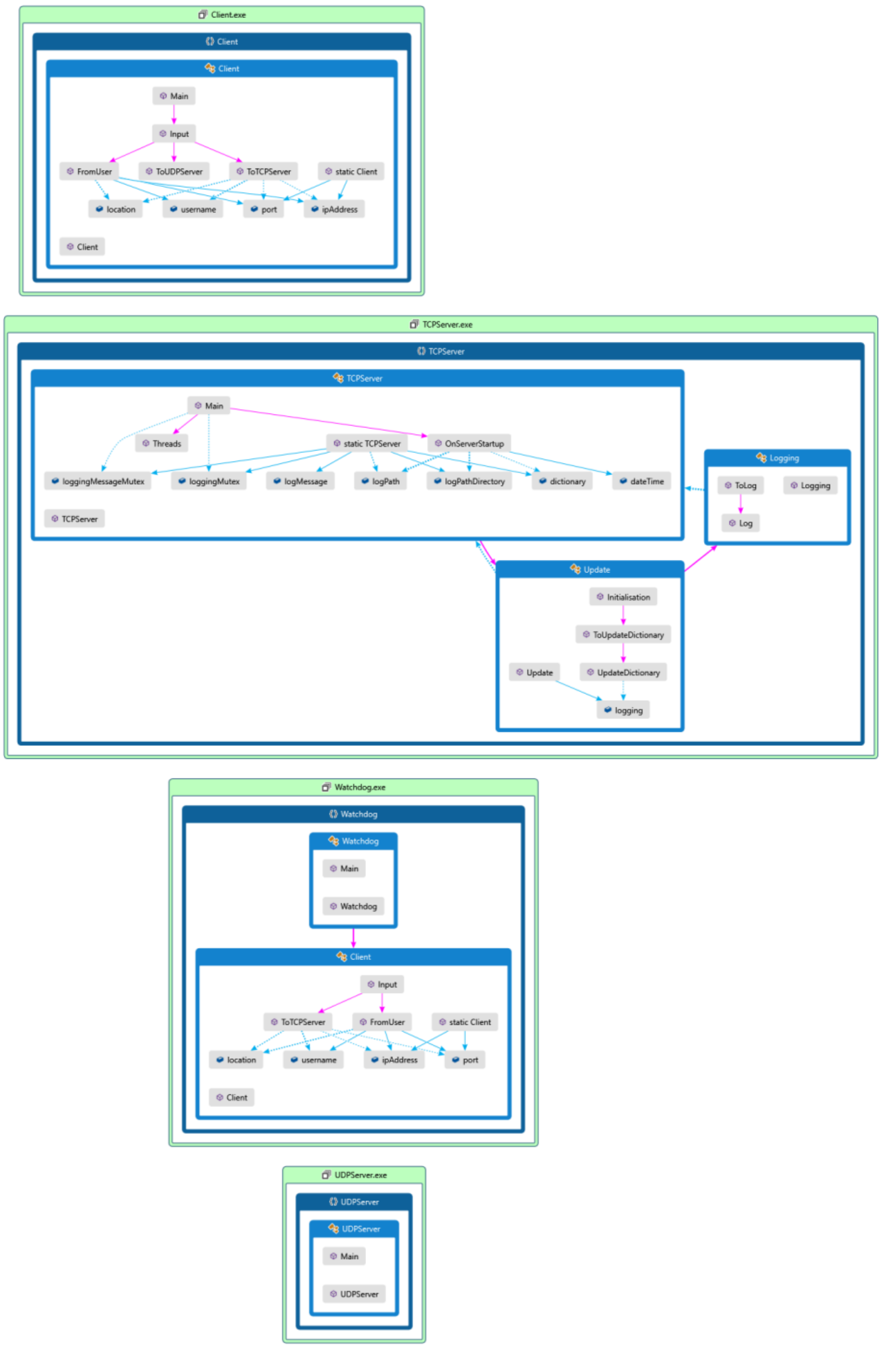


Figure 19: This image shows a diagram that describes the structure of the system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. (Wikipedia, 2016)

References

Ackermann, F., 1994. PRACTICAL EXPERIENCE WITH GPS SUPPORTED AERIAL TRIANGULATION. *The Photogrammetric Record,* 14(84), pp. 860-874.

Apple, 2016. *The powerful programming language that is also easy to learn..* [Online]
Available at: https://developer.apple.com/swift/
[Accessed 12 October 2016].

Apple, 2017. *Tools you'll love to use..* [Online]
Available at: https://developer.apple.com/xcode/ide/
[Accessed 11 January 2017].

Baccelli, F., Mathieu, F., Norros, I. & Varloot, R., 2013. *Can P2P Networks be Super-Scalable.* [Online]
Available at: https://arxiv.org/pdf/1304.6489.pdf
[Accessed 14 November 2016].

Balagtas-Fernandez, F., Forrai, J. & Hussmann, H., 2009. *Evaluation of User Interface Design and Input Methods for Applications on Mobile Touch Screen Devices.* Uppsala, Springer, Berlin, Heidelberg.

Berard, E. V., 2015. *Abstraction, Encapsulation, and Information Hiding.* [Online]
Available at: http://www.tonymarston.co.uk/php-mysql/abstraction.txt
[Accessed 17 January 2017].

Bernstein, P. A. & Goodman, N., 1981. Concurrency Control in Distributed Database Systems. *ACM Computing Surveys,* 13(2), pp. 185-221.

Brown, A. K. & Sturza, M. A., 1995. *GPS tracking system.* United States of America, Patent No. US 07/800,850.

Crato, N., 2010. How GPS Works. In: *Figuring it Out.* Lisboa: Springer-Verlag Berlin Heidelberg, pp. 49-52.

Credits, Extra, 2016. *Improving on Pokemon GO - Making Better Augmented Reality Games - Extra Credits.* [Online]
Available at: https://www.youtube.com/watch?v=94KwB205DDk
[Accessed 11 October 2016].

Dempsey, C., 2013. *War2Map – A Role-Playing Google Maps Game.* [Online]
Available at: https://www.gislounge.com/war2map-role-playing-google-maps-game/
[Accessed 16 January 2017].

Diffen, n.d. *TCP vs. UDP.* [Online]
Available at: http://www.diffen.com/difference/TCP_vs_UDP
[Accessed 11 October 2016].

Funk, B., 2015. *Microsoft's Arrow launcher personalizes the Android experience.* [Online]
Available at: http://techreport.com/news/29254/microsoft-arrow-launcher-personalizes-the-android-experience
[Accessed 18 January 2017].

Google, 2016. *FusedLocationProviderApi.* [Online]
Available at:

https://developers.google.com/android/reference/com/google/android/gms/location/FusedLocationProviderApi
[Accessed 19 January 2017].

Google, 2017. *The Google Maps Geolocation API.* [Online]
Available at: https://developers.google.com/maps/documentation/geolocation/intro
[Accessed 19 January 2017].

Google, n.d. *Getting Started with the NDK.* [Online]
Available at: https://developer.android.com/ndk/guides/index.html
[Accessed 11 October 2016].

Kaplan, E. D. & Hegarty, C. J., 2006. *Understanding GPS Principles and Applications.* 2nd ed. Norwood: Artech House.

Mey, M. D., n.d. *Network discovery using UDP Broadcast (Java).* [Online]
Available at: http://michieldemey.be/blog/network-discovery-using-udp-broadcast/
[Accessed 11 October 2016].

Montemagno, J., 2016. *.NET Standard Library Support for Xamarin.* [Online]
Available at: https://blog.xamarin.com/net-standard-library-support-for-xamarin/
[Accessed 11 October 2016].

Mooney, F. W., 1985. Terrestrial Evaluation of the GPS Standard Positioning Service. *Navigation,* 32(4), pp. 351-369.

Niantic, 2016. *Ingress.* [Online]
Available at: https://www.ingress.com/
[Accessed 11 October 2016].

Niantic, 2016. *Pokémon GO.* [Online]
Available at: http://www.pokemongo.com/en-uk/
[Accessed 11 October 2016].

Noble, J., 2002. Visualising Objects: Abstraction, Encapsulation, Aliasing, and Ownership. In: S. Diehl, ed. *Software Visualization.* Wellington: Springer Berlin Heidelberg, pp. 58-72.

openclipart, 2014. *GPS satellites - trilateration.* [Online]
Available at: https://openclipart.org/detail/191659/gps-satellites-trilateration
[Accessed 19 January 2017].

Park Productions, 2015. *P.A.U.L.A - Park Arcade Unit List Array.* [Online]
Available at: http://www.parkproductions.co.uk/area/games/games.htm
[Accessed 18 January 2017].

Philips, B., 2014. *The P2P Witch Hunt.* [Online]
Available at: http://blog.peer5.com/the-p2p-witch-hunt/
[Accessed 11 October 2016].

Posey, B., 2000. *Understanding the differences between client/server and peer-to-peer networks.* [Online]
Available at: http://www.techrepublic.com/article/understanding-the-differences-between-client-server-and-peer-to-peer-networks/
[Accessed 11 October 2016].

Seguin, K., 2014. *How unreliable is UDP?.* [Online]
Available at: http://openmymind.net/How-Unreliable-Is-UDP/
[Accessed 10 January 2017].

Sinicki, A., 2016. *Developing for Android vs developing for iOS – in 5 rounds.* [Online]
Available at: http://www.androidauthority.com/developing-for-android-vs-ios-697304/
[Accessed 11 October 2016].

Statista, 2016. *Global mobile OS market share in sales to end users from 1st quarter 2009 to 1st quarter 2016.* [Online]
Available at: https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/
[Accessed 11 October 2016].

System 16, 2014. *TAITO QIX HARDWARE.* [Online]
Available at: http://www.system16.com/hardware.php?id=633
[Accessed 11 October 2016].

Tarasenko, N., 2009. *Geostationary Satellites and the GPS Constellation.* [Online]
Available at: http://ccar.colorado.edu/asen5050/projects/projects_2009/tarasenko/
[Accessed 19 January 2017].

TechTarget, 2014. *TCP (Transmission Control Protocol).* [Online]
Available at: http://searchnetworking.techtarget.com/definition/TCP
[Accessed 11 October 2016].

TechTarget, 2015. *UDP (User Datagram Protocol).* [Online]
Available at: http://searchsoa.techtarget.com/definition/UDP
[Accessed 11 October 2016].

The International Arcade Museum, Museum of the Game, 2016. *Qix.* [Online]
Available at: http://www.arcade-museum.com/game_detail.php?game_id=9185
[Accessed 12 October 2016].

Vogel, L., 2016. *Android development with Android Studio - Tutorial.* [Online]
Available at: http://www.vogella.com/tutorials/Android/article.html
[Accessed 18 January 2017].

Wikipedia, 2016. *Class Diagram.* [Online]
Available at: https://en.wikipedia.org/wiki/Class_diagram
[Accessed 16 January 2017].

Wikipedia, 2017. *Android Studio.* [Online]
Available at: https://en.wikipedia.org/wiki/Android_Studio
[Accessed 11 January 2017].

Wikipedia, 2017. *Ingress (Video game).* [Online]
Available at: https://en.wikipedia.org/wiki/Ingress_(video_game)
[Accessed 11 January 2017].

Wikipedia, 2017. *Microsoft Visual Studio.* [Online]
Available at: https://en.wikipedia.org/wiki/Microsoft_Visual_Studio
[Accessed 11 January 2017].

Wikipedia, 2017. *Pokemon GO.* [Online]
Available at: https://en.wikipedia.org/wiki/Pok%C3%A9mon_Go
[Accessed 11 January 2017].

Xamarin Inc., 2016. *Building Cross Platform Applications.* [Online]
Available at: https://developer.xamarin.com/guides/cross-platform/application_fundamentals/building_cross_platform_applications/
[Accessed 11 October 2016].

Xamarin Inc., 2016. *Using Native Libraries.* [Online]
Available at:
https://developer.xamarin.com/guides/android/advanced_topics/using_native_libraries/
[Accessed 11 October 2016].

Xamarin, 2017. *Introduction to Location Services and the Fused Location Provider.* [Online]
Available at:
https://developer.xamarin.com/guides/android/platform_features/maps_and_location/location/
[Accessed 19 January 2017].

Zhao, Y., 2002. Standardization of mobile phone positioning for 3G systems. *IEEE Communications Magazine*, 7 August, pp. 108-116.