**Capture the Campus!**

**Final Report**

Submitted for the BSc in

Computer Science

April 17

By

**Alexander C Whitehead**

Word Count:

# Table of Contents

[1 Table of Contents 2](#_Toc480329885)

[2 Table of Figures 4](#_Toc480329886)

[3 Introduction 5](#_Toc480329887)

[4 Aim and Objectives 7](#_Toc480329888)

[Objective 1 – Create a Client Library and Test Wrapper 7](#_Toc480329889)

[Objective 2 – Create a TCP Server and Test Wrapper 7](#_Toc480329890)

[Objective 3 – Create a UDP Server and Test Wrapper 8](#_Toc480329891)

[Objective 4 – Add Multithreading to the TCP Server 8](#_Toc480329892)

[Objective 5 - Add Logging to the TCP Server 8](#_Toc480329893)

[Objective 6 - Create a Watchdog or Heat Beat Application for the TCP Server and a Test Wrapper 8](#_Toc480329894)

[Objective 7 - Create a mobile application with a main menu 8](#_Toc480329895)

[Objective 8 - Add a Simple Map Based GPS Locator to the Mobile Application 8](#_Toc480329896)

[Objective 9 - Create a Method to Set the Play Area 9](#_Toc480329897)

[Objective 10 - Create a Method to Redefine the Play Area by the Player’s Movements 9](#_Toc480329898)

[Objective 11 - Add Scoring and Game Ending Logic to the Game 9](#_Toc480329899)

[Objective 12 - Add a Game Over Screen 9](#_Toc480329900)

[Objective 13 - Add a Representation for the Enemy to the Game 9](#_Toc480329901)

[Objective 14 - Create a Method That Can Randomly Generate the Enemy a New Location 9](#_Toc480329902)

[Objective 15 - Create a Method That Allows the Enemy to Randomly Wander Within the Play Area 10](#_Toc480329903)

[Objective 16 - Create a Method That Deals With an Intersection between the Enemy and the Player 10](#_Toc480329904)

[Objective 17 - Add a Lobby for Local Multiplayer 10](#_Toc480329905)

[Objective 18 - Create a Method of Connection between Host and Players 10](#_Toc480329906)

[Objective 19 - Add Logic to Allow Local Multiplayer 10](#_Toc480329907)

[5 Background 11](#_Toc480329908)

[5.1 Problem Context 11](#_Toc480329909)

[5.1.1 Alternative Solutions 11](#_Toc480329910)

[5.1.2 Tracking 13](#_Toc480329911)

[5.2 Comparison of Technologies 15](#_Toc480329912)

[5.2.1 Platform 15](#_Toc480329913)

[5.2.2 Language 17](#_Toc480329914)

[5.2.3 Networking 20](#_Toc480329915)

[5.3 Comparison of Algorithms 22](#_Toc480329916)

[5.3.1 Polygon Clipping 22](#_Toc480329917)

[5.3.2 Polygon Tessellation 23](#_Toc480329918)

[5.3.3 Wandering 24](#_Toc480329919)

[6 Technical Development 25](#_Toc480329920)

[6.1 System Design 25](#_Toc480329921)

[6.1.1 CaptureTheCampus Design 25](#_Toc480329922)

[6.1.2 Client Design 26](#_Toc480329923)

[6.1.3 TCPServer Design 27](#_Toc480329924)

[6.1.4 Tests Design 28](#_Toc480329925)

[6.1.5 ToClient Design 29](#_Toc480329926)

[6.1.6 ToTCPServer Design 30](#_Toc480329927)

[6.1.7 ToUDPServer Design 31](#_Toc480329928)

[6.1.8 ToWatchdog Design 32](#_Toc480329929)

[6.1.9 UDPServer Design 33](#_Toc480329930)

[6.1.10 Watchdog Design 34](#_Toc480329931)

[6.2 User Interface Design 35](#_Toc480329932)

[6.3 Test Design 39](#_Toc480329933)

[6.4 System Implementation 40](#_Toc480329934)

[6.4.1 Polygon Clipping 40](#_Toc480329935)

[6.4.2 Polygon Tessellation 41](#_Toc480329936)

[6.4.3 Wandering 42](#_Toc480329937)

[7 Evaluation 43](#_Toc480329938)

[7.1 Project Achievements 43](#_Toc480329939)

[7.1.1 New Task List 44](#_Toc480329940)

[7.1.2 New Time Plan 47](#_Toc480329941)

[7.1.3 Further Work 49](#_Toc480329942)

[8 Conclusion 50](#_Toc480329943)

[Appendix A: Capture the Campus! 51](#_Toc480329944)

[Appendix B: First Task List 52](#_Toc480329945)

[Appendix C: First Time Plan 54](#_Toc480329946)

[Appendix D: Second Task List 56](#_Toc480329947)

[Appendix E: Second Time Plan 59](#_Toc480329948)

[Appendix F: Risk Analysis 61](#_Toc480329949)

[Appendix G: Code Map 64](#_Toc480329950)

[References 65](#_Toc480329951)

# Table of Figures

[Figure 1: This is an image showing the gameplay of Qix (The International Arcade Museum, Museum of the Game, 2016). 6](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc480329952)

[Figure 2: This image shows a standard game screen for Pokémon GO (Wikipedia, 2017). 11](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc480329953)

[Figure 3: This image shows a standard game screen for Ingress (Wikipedia, 2017). 12](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc480329954)

[Figure 4: This image shows the orbits of some GPS satellites 13](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc480329955)

[Figure 5: This image shows a visualisation of the act of GPS trilateration (openclipart, 2014) 14](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc480329956)

[Figure 6: This image shows the current market share of mobile operating systems (Statista, 2016). 15](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc480329957)

[Figure 7: This image shows an example of a standard Android Studio integrated development environment window (Wikipedia, 2017). 17](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc480329958)

[Figure 8: This image shows an example of a standard Visual Studio integrated development environment window (Wikipedia, 2017). 18](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc480329959)

[Figure 9: This image shows an example of a standard Xcode integrated development environment window (Apple, 2017). 18](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc480329960)

[Figure 10: 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). 20](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc480329961)

[Figure 11: This image shows two examples of menu styles used in mobile application (Vogel, 2016) 35](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc480329962)

[Figure 12: This image shows examples of the user interface for the Android launcher Arrow Launcher by Microsoft (Funk, 2015) 36](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc480329963)

[Figure 13: This image shows an example of a retro arcade style menu . (Park Productions, 2015) 36](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc480329964)

[Figure 14: This image shows a standard game screen for the Google Maps API based game War2Map (Dempsey, 2013) 37](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc480329965)

[Figure 15: 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) 64](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc480329966)

# Introduction

This is a report describing the final research and development stages of the project to build Capture the Campus!.

This report includes:

1. A section introducing the project and covering its main aims and objectives.
2. A section discussing background research conducted. Including a subsection describing the context of the project, a subsection introducing alternative solutions to similar projects, a subsection comparing relevant development technologies and a subsection comparing known solutions to anticipated problems.
3. A section discussing the design and development of the project. Including a subsection describing the design of each part of Capture the Campus!, a subsection describing the design of the user interface, a subsection discussing any tests implemented and a subsection discussing the implementation of the solutions discussed in section 2.
4. A section evaluating the project’s success in hitting the aims outlined in section 1 and a subsection describing any further work that could be performed to complete unachieved goals.
5. A section summarising the project as a whole.

Capture the Campus! is a GPS based mobile game influenced by the classic 80’s arcade game Qix (System 16, 2014) and to a lesser extent Pokémon GO (Niantic, 2016), and Ingress (Niantic, 2016).

In the game of Qix the objective is to steer a player character around the periphery of a square play area before then traveling across this play area all while avoiding multiple computer-controlled enemies.

Once the player character has crossed the play area the play area is then redefined as the larger of the two polygons that are created by splitting the play area into two along the path travelled by the player character. A score is awarded to the player character based on the size of the polygon removed from the play area.

The enemies mentioned above wander randomly around within the play area, if an enemy intersects the player character’s path the player character’s path is reset and they lose a life.

An example of a standard Qix game screen can be seen below, including showing multiple scoring polygons that have been split by the player character away from the main play area (Figure 1).

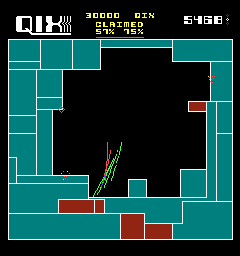


Figure 1: This is an image showing the gameplay of Qix (The International Arcade Museum, Museum of the Game, 2016).

The objective of Capture the Campus! will be similar to that of Qix, as described above. However, rather than controlling a player character with something like a mouse and keyboard or a game controller the player character will be controlled by the player’s physical movement in the real world.

The play area will be defined using latitude and longitude coordinates and to capture parts of this play area the player must move physically from one side of it to the other.

The game will be run on a mobile device in the player’s possession and their location will be tracked using GPS data from the devices GPS sensor.

As in Qix there should be computer-controlled enemies, these enemies will be able to kill the player but rather than removing a life, points will be deducted from their score.

The game will end after a predetermined area has been captured.

The original specification for Capture the Campus! can be seen below (‎Appendix A:).

# 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 and Test Wrapper
2. Create a TCP Server and Test Wrapper
3. Create a UDP Server and Test Wrapper
4. Add Multithreading to the TCP Server
5. Add Logging to the TCP Server
6. Create a Watchdog or Heat Beat Application for the TCP Server and a Test Wrapper
7. Create a mobile application with a main menu
8. Add a Simple Map Based GPS Locator to the Mobile Application
9. Create a Method to Set the Play Area
10. Create a Method to Redefine the Play Area by the Player’s Movements
11. Add Scoring and Game Ending Logic to the Game
12. Add a Game Over Screen.
13. Add a Representation for the Enemy to the Game
14. Create a Method That Can Randomly Generate the Enemy a New Location
15. Create a Method That Allows the Enemy to Randomly Wander Within the Play Area
16. Create a Method That Deals With an Intersection between the Enemy and the Player
17. Add a Lobby for Local Multiplayer
18. Create a Method of Connection between Host and Players
19. Add Logic to Allow Local Multiplayer

### Objective 1 – Create a Client Library and Test Wrapper

Before a server can be created, a basic client library must first be developed to test the TCP and UDP servers with.

The client can be built first as examples of working TCP and UDP servers are easy to come by online to test the client’s functionality against.

The client should be able to send messages using TCP and UDP protocols. The client should be able to send TCP messages using any IP address or port.

Development of the client should not end until both it and the servers are completed.

The client library can be integrated into the game once completed and reused in future projects.

A wrapper for the client should be created to allow it to be used as both a library and as an application for integration and testing.

### Objective 2 – Create a TCP Server and Test Wrapper

The TCP server will store the vertexes of the initial play area, the position of the enemy and the current location of each player.

This information will be used to synchronize all instances of the current local multiplayer game.

The TCP server should be able to run from a phone to allow multiple concurrent instances of multiplayer games.

The TCP server should be able to be opened on any port.

A wrapper for the TCP server should be created to allow it to be used as both a library and as an application for integration and testing.

This will be discussed further in the Networking (‎5.2.3) section of the Background.

### Objective 3 – Create a UDP Server and Test Wrapper

The UDP server will be used as an initial contact point for the client, the UDP server should broadcast a coded message containing the IP address of the TCP server. The coded message should help to eliminate malicious attacks on the TCP server.

The UDP server should be able to run from a phone to allow multiple concurrent instances of multiplayer games.

The UDP server should be able to be opened on any port.

A wrapper for the UDP server should be created to allow it to be used as both a library and as an application for integration and testing.

This will be discussed further in the Networking (‎5.2.3) section of the Background below.

### Objective 4 – Add Multithreading to the TCP Server

The TCP server should have multithreading capabilities added, the TCP server should be able to accept multiple TCP requests simultaneously to support several players concurrently.

### Objective 5 - Add Logging to the TCP Server

The TCP server should be able to write a log of all messages for debugging purposes.

The log should be able to be created at any file directory.

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 Watchdog or Heat Beat Application for the TCP Server and a Test Wrapper

A Watchdog or Heart Beat application should be created; this application will check to see if the game has crashed and clean up, if it has, by stopping the servers.

A wrapper for the Watchdog or Heart Beat application should be created to allow it to be used as both a library and as an application for integration and testing.

### Objective 7 - Create a mobile application with a main menu

An initial application should be created to build the games functionality upon.

Before the menu is launched, all dependencies for the game should be checked and the application should not run if a dependency is found to be missing.

The main menu should have the capability to start a single player game, start a local multiplayer game as a host or join an already existing local multiplayer instance.

### Objective 8 - Add a Simple Map Based GPS Locator to the Mobile Application

Before the game logic is added to the application a simple map based GPS locator application should be developed, this should display the device’s location on a map. The map should translate, with the device’s movement.

This application can then be transformed by the addition of gameplay into the intended product.

### Objective 9 - Create a Method to Set the Play Area

A method should be created which can be used before a game instance is initialised to set the initial play area for the game.

To be in keeping with the influence of Qix the initial play area should be a square or rectangle.

### Objective 10 - Create a Method to Redefine the Play Area by the Player’s Movements

A method should be created which can be used to track a player’s path through the play area. Once the player exits the play area it should redefine the play area as the larger of the two polygons created by splitting the play area into two along the path travelled by the player.

If a player crosses their own path the method should remove the loop from the player’s path as otherwise more than two polygons would be created when splitting the play area.

The area removed from the play area should be filled with the player characters colour.

### Objective 11 - Add Scoring and Game Ending Logic to the Game

When a player removes a polygon from the play area their score should be increased by an amount relative to the polygon’s size.

The game should end once the play area’s size has been reduced by a certain amount.

### Objective 12 - Add a Game Over Screen

A screen should be created that is displayed to the player once a game has ended.

The game over screen should display the player’s score and allow them to return to the main menu to play again.

### Objective 13 - Add a Representation for the Enemy to the Game

Something that can be used to represent the computer-controlled enemy should be added to the play area.

If the enemy finds itself outside of the play area, for instance, if the enemy is located within a polygon removed from the play area then the enemy should try to move itself to be within the play area again.

### Objective 14 - Create a Method That Can Randomly Generate the Enemy a New Location

A method should be created which can find a random location within any given play area, this is because no point within the initial play area can be assumed to always exist within the redefined play area

This should be used as an initial location for the enemy or to relocate the enemy if it finds itself outside of the play area.

### Objective 15 - Create a Method That Allows the Enemy to Randomly Wander Within the Play Area

To be in keeping with the influence of Qix the computer-controlled enemy should follow a wandering path like the eponymous Qix enemy from Qix.

This path should follow a cone of possible moves relative to the normal vector of the previous move of the enemy, this will force the enemy to move around the play area rather than, as would be possible otherwise, vibrating back and forth eternally.

### Objective 16 - Create a Method That Deals With an Intersection between the Enemy and the Player

A method should be added that removes the player’s path and deducts an amount from their score if the enemy intersects any point along their path.

A lives system shall not be implemented as lives systems were originally developed to encourage people to pay to play arcade games more frequently (Spikejumper2, 2017).

A lives system would also clash with the ethos of a multiplayer game as some players could end up without any more lives well before their peers, thus rendering them out of the game.

### Objective 17 - Add a Lobby for Local Multiplayer

A lobby should be created where players wait for their local multiplayer game to start, while the servers on the backend connect and synchronise their devices.

In case the UDP server fails in establishing an initial connection the option to input the IP address of the target host should be allowed, thus the IP address of the host should also be displayed on the host device.

### Objective 18 - Create a Method of Connection between Host and Players

A backend method should be created which while the players wait in the lobby synchronises their devices with the vertexes of the initial play area, the position of the enemy and the current location of each player.

This method should also respond to the Watchdog or Heart Beat application to stop it from prematurely closing the TCP server.

### Objective 19 - Add Logic to Allow Local Multiplayer

The gameplay logic should be adapted to allow and track multiple simultaneous players.

Each instance of the local multiplayer game should also by synchronised with the vertexes of the initial play area, the position of the enemy and the current location of each player.

The game should also respond to the Watchdog or Heart Beat application to stop it from prematurely closing the TCP server.

# Background

## Problem Context

## Alternative Solutions

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).

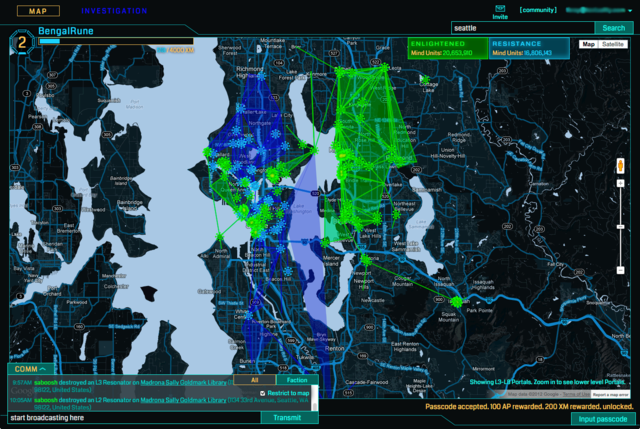


Figure 3: This image shows a standard game screen for Ingress (Wikipedia, 2017).

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 are not mobile (Credits, Extra, 2016). However, this kind of gameplay would be beyond the scope of this project.

## 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).

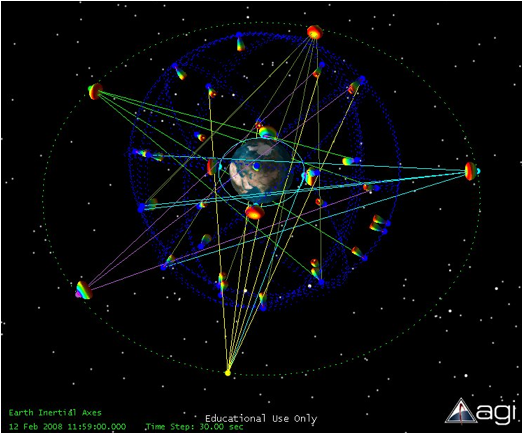


Figure 4: This image shows the orbits of some GPS satellites

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 4)

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).

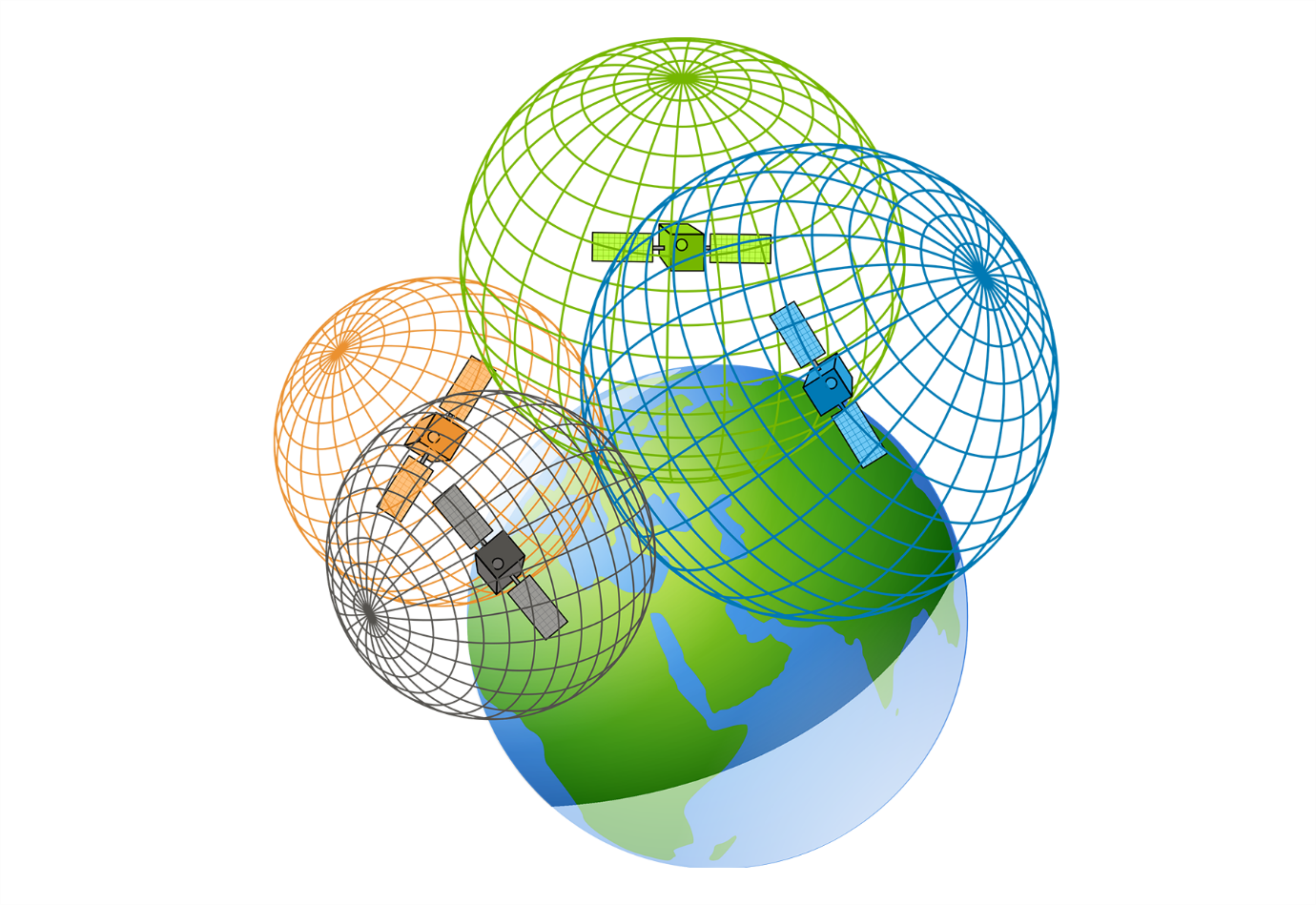


Figure 5: This image shows a visualisation of the act of GPS trilateration (openclipart, 2014)

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 5).

## Comparison of Technologies

## 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 6).

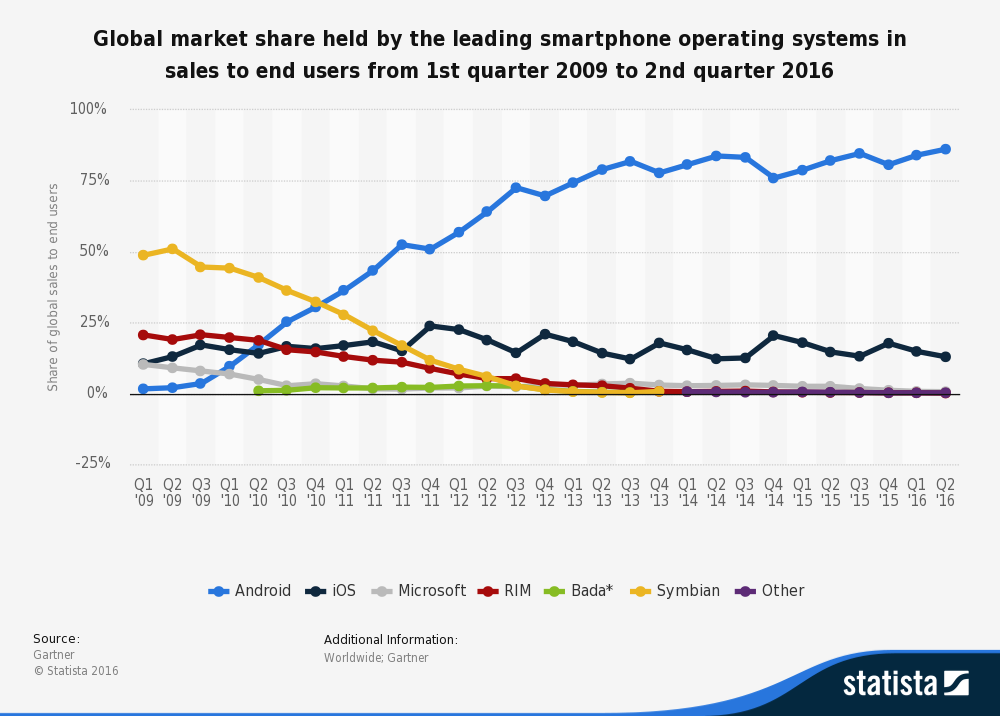


Figure 6: This image shows the current market share of mobile operating systems (Statista, 2016).

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.

## 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 7) (Figure 8). 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 9) (Figure 8).

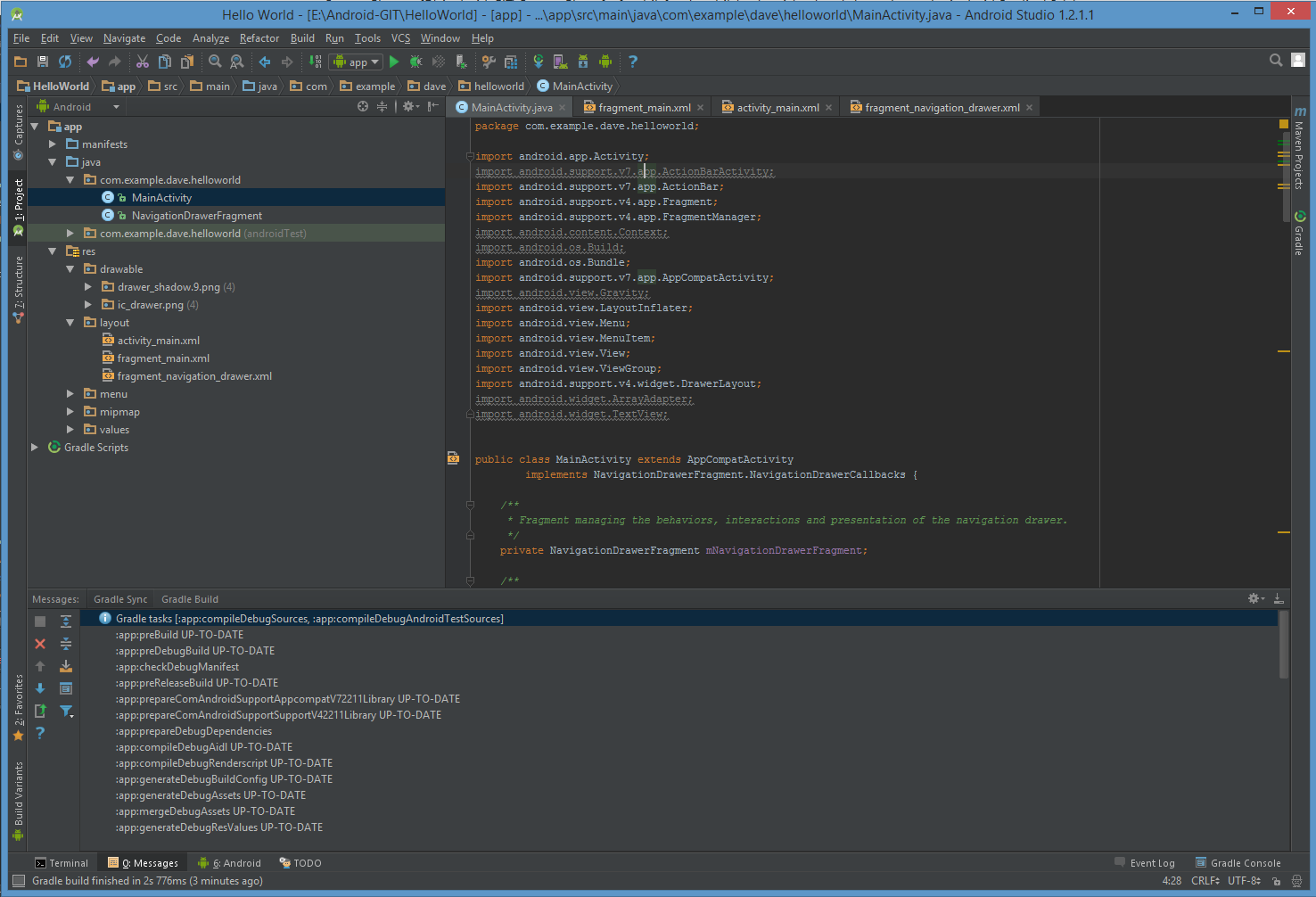


Figure 7: This image shows an example of a standard Android Studio integrated development environment window (Wikipedia, 2017).

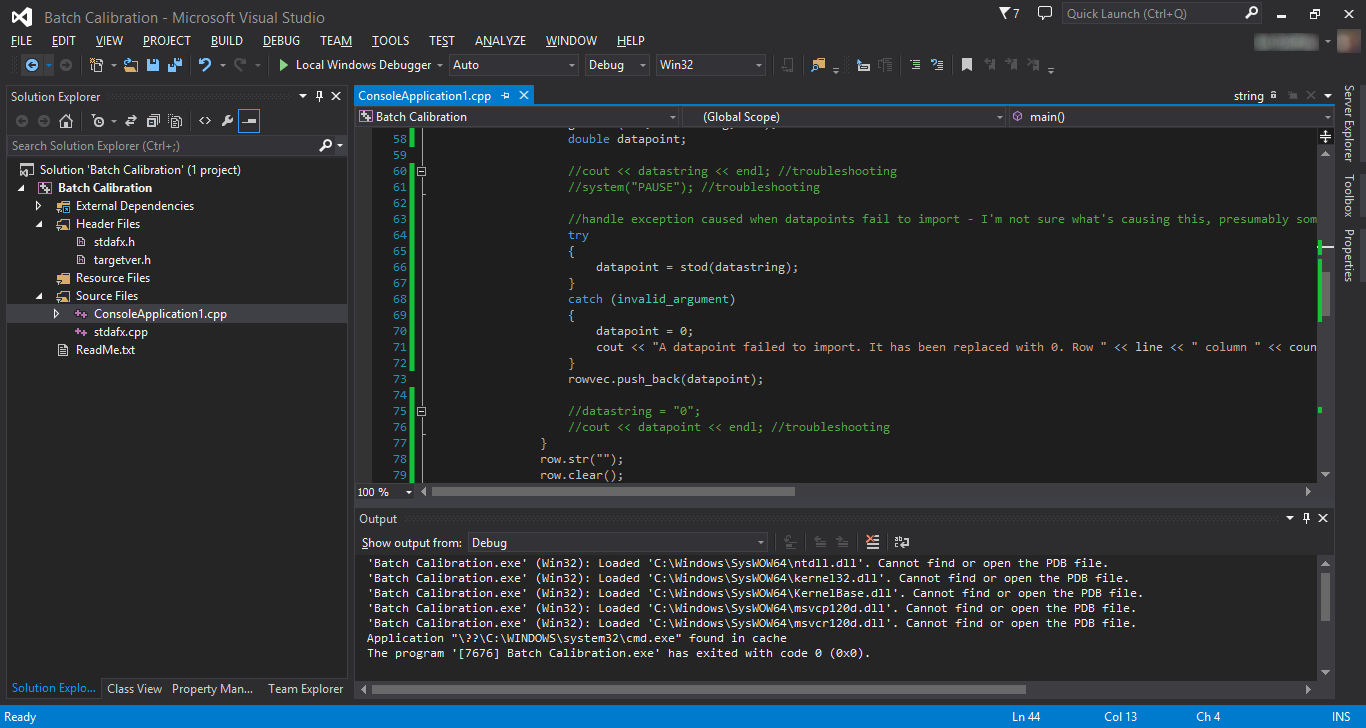


Figure 8: This image shows an example of a standard Visual Studio integrated development environment window (Wikipedia, 2017).

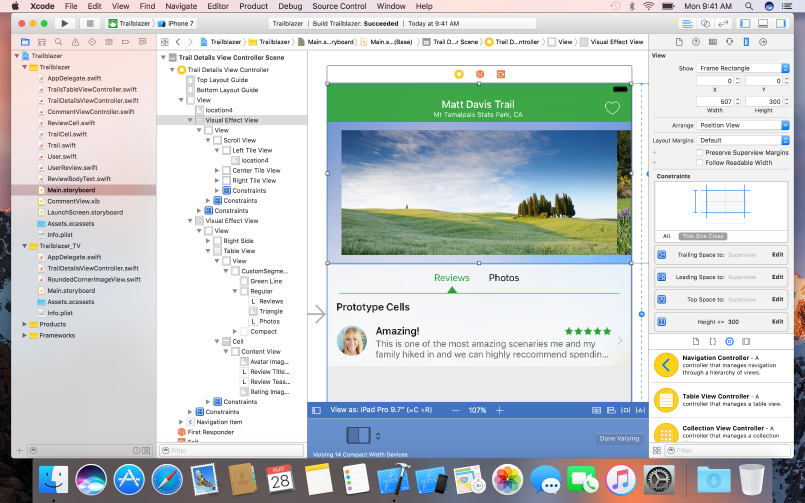


Figure 9: This image shows an example of a standard Xcode integrated development environment window (Apple, 2017).

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.

## 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 10).

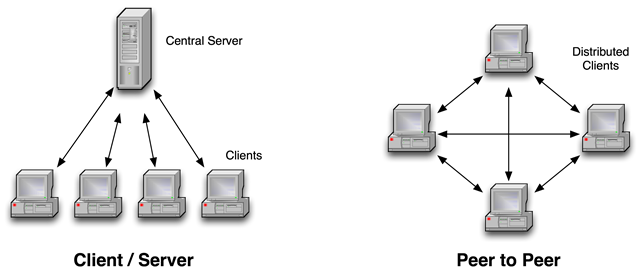


Figure 10: This image shows a visual model of the difference between a client/server model on the left and a peer-to-peer model on the right (Philips, 2014).

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.

## Comparison of Algorithms

## Polygon Clipping

## Polygon Tessellation

## Wandering

# Technical Development

## System Design

## CaptureTheCampus Design

## Client Design

The diagram above () 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).

## TCPServer Design

The diagram above () 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).

## Tests Design

## ToClient Design

## ToTCPServer Design

## ToUDPServer Design

## ToWatchdog Design

## UDPServer Design

The diagram above () 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).

## Watchdog Design

The diagram above () 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).

## User Interface Design

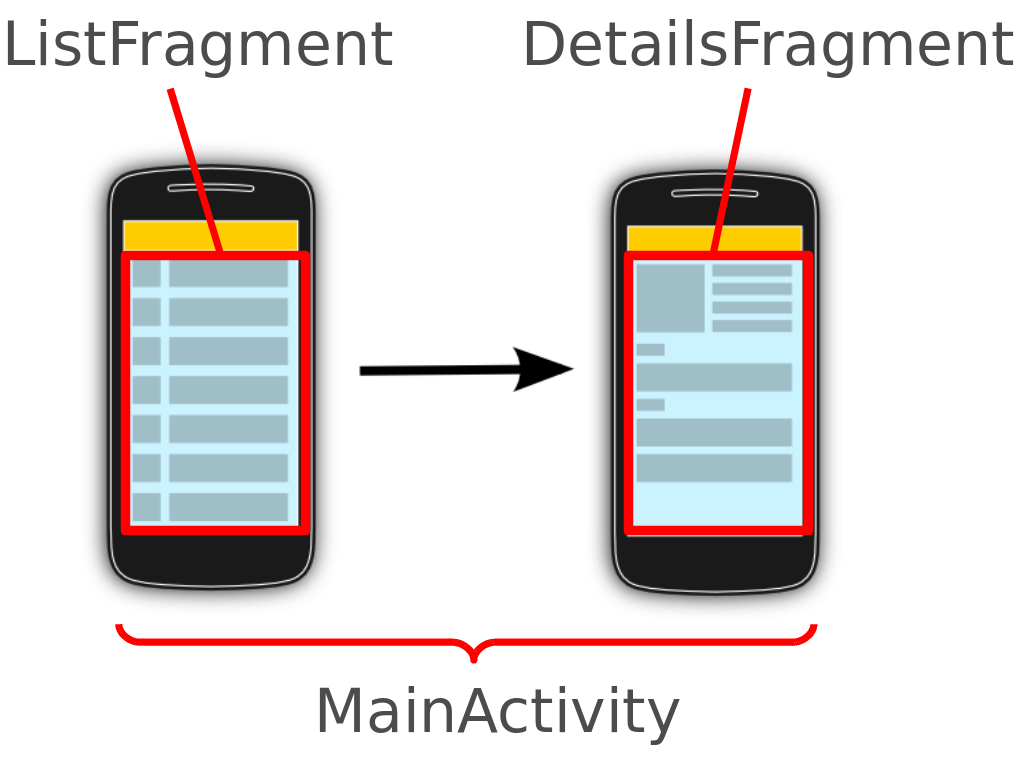


Figure 11: This image shows two examples of menu styles used in mobile application (Vogel, 2016)

The image above (Figure 11) shows two representations of menu styles generally 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 12: This image shows examples of the user interface for the Android launcher Arrow Launcher by Microsoft (Funk, 2015)

The image above (Figure 12) 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.

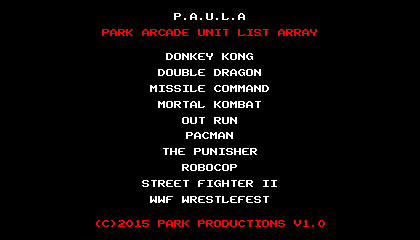


Figure 13: This image shows an example of a retro arcade style menu . (Park Productions, 2015)

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 13).

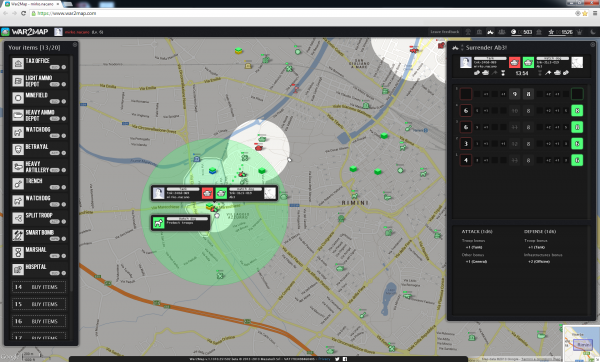


Figure 14: This image shows a standard game screen for the Google Maps API based game War2Map (Dempsey, 2013)

The image above (Figure 14) 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.

## Test Design

|  |  |  |
| --- | --- | --- |
| **#** | **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.

## System Implementation

## Polygon Clipping

## Polygon Tessellation

## Wandering

# Evaluation

## Project Achievements

Generally, progress on the project has been impeded 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 (‎Appendix C:) 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 (‎7.1.1) and time plan (‎7.1.2) 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 F:) 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 F:) and/or are beyond the scope of the project.

## Final 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 and test wrapper | Create a client library that is capable of interfacing with the server and a test wrapper which will allow the client to be used as both a library and an application | 5 |
| 4 | Create TCP server and test wrapper | Create a TCP server that can accept TCP packets from the client and a test wrapper which will allow the TCP server to be used as both a library and an application | 2 |
| 5 | Create UDP server and test wrapper | Create a UDP server that can accept UDP packets from the client and can be used to identify the IP of the TCP server and a test wrapper which will allow the UDP server to be used as both a library and an application | 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 map to game activity | Add a suitable map to the game screen | 2 |
| 13 | Add player character to game screen |  | 2 |
| 14 | Move player character and camera with player’s movements |  | 2 |
| 15 | Add method to define the play area |  | 2 |
| 16 | Add a method to track the player’s path and display it on the map |  | 2 |
| 17 | Add a method to determine if an object is inside a polygon |  | 2 |
| 18 | Add a method to clip a line segment by another line segment |  | 2 |
| 19 | Add a method which clips a polygon by a line segment |  |  |
| 20 | Add a method which finds the larger of two areas |  |  |
| 21 | Add a method which redefines the play area |  |  |
| 22 | Add a method which updates the score |  |  |
| 23 | Create a representation of the enemy |  |  |
|  | Add a method which finds a random point in a polygon |  |  |
|  | Add a method that finds a random unit vector in a range |  |  |
|  | Add a method that detects a collision between a sphere and a line segment |  |  |
|  | Add a method that calculates the resultant vector normal after a collision |  |  |
|  | Add a method that handles a collision between the enemy and a path |  |  |
|  | Create a lobby for local multiplayer |  |  |
|  |  |  |  |
|  |  |  |  |

## Final 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 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

## Further Work

# Conclusion

1. 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

1. Initial 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 movements to map | Make the player character move as the player moves, this should work via GPS | 2 |
| 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 |

1. Initial 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 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

1. Interim 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 |

1. Interim 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 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

1. 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 |

1. Code Map

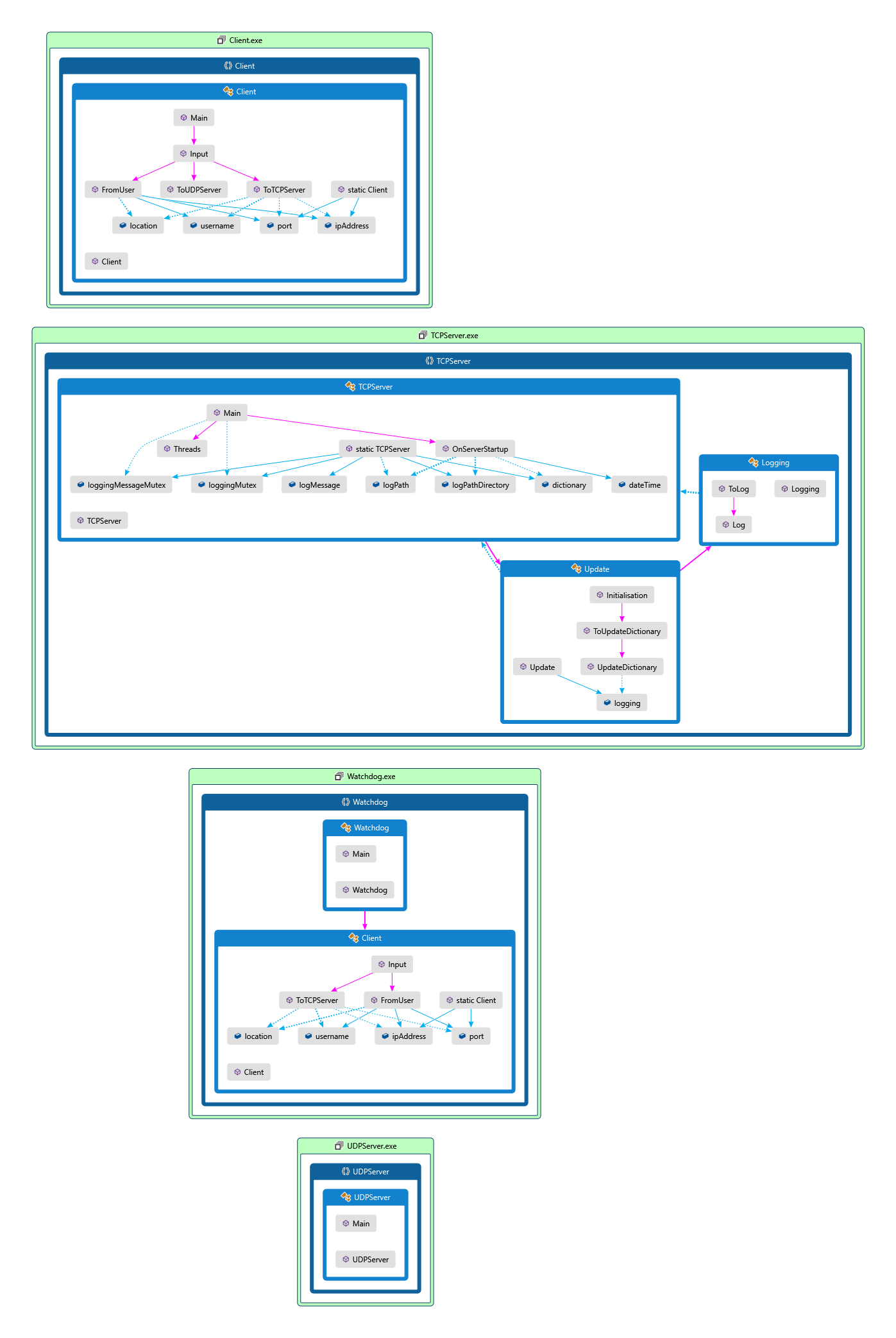


Figure 15: This image shows a diagram that describes the structure of the system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. (Wikipedia, 2016)

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].

Spikejumper2, 2017. *Are lives in videogames obsolete?.* [Online]   
Available at: http://gdforum.freeforums.net/thread/46514/lives-videogames-obsolete  
[Accessed 19 April 2017].

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.