**Capture the Campus!**

**Final Report**

Submitted for the BSc in

Computer Science

April 17

By

**Alexander C Whitehead**

Word Count:

Table of Contents

[1 Introduction 7](#_Toc481102932)

[2 Aim and Objectives 9](#_Toc481102933)

[Objective 1 – Create a Client Library and Test Wrapper 9](#_Toc481102934)

[Objective 2 – Create a TCP Server and Test Wrapper 9](#_Toc481102935)

[Objective 3 – Create a UDP Server and Test Wrapper 10](#_Toc481102936)

[Objective 4 – Add Multithreading to the TCP Server 10](#_Toc481102937)

[Objective 5 - Add Logging to the TCP Server 10](#_Toc481102938)

[Objective 6 - Create a Watchdog or Heat Beat Application for the TCP Server and a Test Wrapper 10](#_Toc481102939)

[Objective 7 - Create a mobile application with a main menu 10](#_Toc481102940)

[Objective 8 - Add a Simple Map Based GPS Locator to the Mobile Application 10](#_Toc481102941)

[Objective 9 - Create a Method to Set the Play Area 11](#_Toc481102942)

[Objective 10 - Create a Method to Redefine the Play Area by the Player’s Movements 11](#_Toc481102943)

[Objective 11 - Add Scoring and Game Ending Logic to the Game 11](#_Toc481102944)

[Objective 12 - Add a Game Over Screen 11](#_Toc481102945)

[Objective 13 - Add a Representation for the Enemy to the Game 11](#_Toc481102946)

[Objective 14 - Create a Method That Can Randomly Generate the Enemy a New Location 11](#_Toc481102947)

[Objective 15 - Create a Method That Allows the Enemy to Randomly Wander Within the Play Area 12](#_Toc481102948)

[Objective 16 - Create a Method That Deals With an Intersection between the Enemy and the Player 12](#_Toc481102949)

[Objective 17 - Add a Lobby for Local Multiplayer 12](#_Toc481102950)

[Objective 18 - Create a Method of Connection between Host and Players 12](#_Toc481102951)

[Objective 19 - Add Logic to Allow Local Multiplayer 12](#_Toc481102952)

[3 Background 13](#_Toc481102953)

[3.1 Problem Context 13](#_Toc481102954)

[3.1.1 Alternative Solutions 13](#_Toc481102955)

[3.1.2 Tracking 17](#_Toc481102956)

[3.2 Comparison of Technologies 20](#_Toc481102957)

[3.2.1 Platform 20](#_Toc481102958)

[3.2.2 Language 23](#_Toc481102959)

[3.2.3 Networking 26](#_Toc481102960)

[3.2.3.1 Architecture 26](#_Toc481102961)

[3.2.3.2 Protocol 28](#_Toc481102962)

[4 Technical Development 30](#_Toc481102963)

[4.1 System Design 30](#_Toc481102964)

[4.1.1 Networking Design 32](#_Toc481102965)

[4.1.1.1 Client Design 32](#_Toc481102966)

[4.1.1.2 Watchdog Design 34](#_Toc481102967)

[4.1.1.3 TCPServer Design 36](#_Toc481102968)

[4.1.1.4 UDPServer Design 38](#_Toc481102969)

[4.1.2 CaptureTheCampus Design 40](#_Toc481102970)

[4.1.2.1 Game Design 40](#_Toc481102971)

[4.1.2.1.1 GameActivity Design 41](#_Toc481102972)

[4.1.2.1.2 GameMap Design 42](#_Toc481102973)

[4.1.2.1.3 GamePosition Design 43](#_Toc481102974)

[4.1.2.1.4 Utilities Design 44](#_Toc481102975)

[4.1.2.2 Menu Design 45](#_Toc481102976)

[4.1.2.3 Score Design 46](#_Toc481102977)

[4.1.2.4 Search Design 47](#_Toc481102978)

[4.1.2.5 Set Design 48](#_Toc481102979)

[4.1.2.6 Static Design 49](#_Toc481102980)

[4.1.2.7 MainActivity Design 50](#_Toc481102981)

[4.2 User Interface Design 51](#_Toc481102982)

[4.3 System Implementation 56](#_Toc481102983)

[4.3.1 Processes and Methodology 56](#_Toc481102984)

[4.3.2 Version Control 57](#_Toc481102985)

[4.3.3 Polygon Clipping Algorithm 58](#_Toc481102986)

[4.3.4 Polygon Tessellation Algorithm 59](#_Toc481102987)

[4.3.5 Wandering Algorithm 60](#_Toc481102988)

[4.4 Test Design 61](#_Toc481102989)

[5 Evaluation 63](#_Toc481102990)

[5.1 Project Achievements 63](#_Toc481102991)

[5.1.1 Final Task List 64](#_Toc481102992)

[5.1.2 Final Time Plan 69](#_Toc481102993)

[5.2 Further Work 74](#_Toc481102994)

[6 Conclusion 75](#_Toc481102995)

[Appendix A: Capture the Campus! 76](#_Toc481102996)

[Appendix B: Code Map Legend 77](#_Toc481102997)

[Appendix C: Code Map 78](#_Toc481102998)

[Appendix D: Networking Code Map 79](#_Toc481102999)

[Appendix E: CaptureTheCampus Code Map 80](#_Toc481103000)

[Appendix F: Polygon Clipping Code 81](#_Toc481103001)

[Appendix G: Polygon Tessellation Code 89](#_Toc481103002)

[Appendix H: Wandering Code 93](#_Toc481103003)

[Appendix I: Initial Task List 98](#_Toc481103004)

[Appendix J: Initial Time Plan 100](#_Toc481103005)

[Appendix K: Interim Task List 102](#_Toc481103006)

[Appendix L: Interim Time Plan 105](#_Toc481103007)

[Appendix M: Risk Analysis 107](#_Toc481103008)

[Acknowledgements 110](#_Toc481103009)

[References 111](#_Toc481103010)

Table of Figures

[Figure 1: This is an image showing the gameplay of Qix (The International Arcade Museum, Museum of the Game, 2016). 8](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103011)

[Figure 2: This image shows a standard game screen for Pokémon GO (Wikipedia, 2017). 14](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103012)

[Figure 3: This image shows a standard game screen for Ingress (Google, 2017) 15](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103013)

[Figure 4: This image shows a standard game screen for Ingress (Wikipedia, 2017). 16](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103014)

[Figure 5: This image shows the orbits of some GPS satellites 17](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103015)

[Figure 6: This image shows a visualisation of the act of GPS trilateration (openclipart, 2014) 18](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103016)

[Figure 7: This image shows the current market share of mobile operating systems (Statista, 2016). 21](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103017)

[Figure 8: This image shows an example of a standard Android Studio integrated development environment window (Wikipedia, 2017). 23](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103018)

[Figure 9: This image shows an example of a standard Visual Studio integrated development environment window (Wikipedia, 2017). 24](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103019)

[Figure 10: This image shows an example of a standard Xcode integrated development environment window (Apple, 2017). 25](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103020)

[Figure 11: 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). 26](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103021)

[Figure 12: This diagram shows the latency involved in sending a message using the different transport layer protocols (Taing, 2011). 28](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103022)

[Figure 13: This diagram shows the throughput in Mbps of the different transport layer protocols (Taing, 2011). 29](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103023)

[Figure 14: This image shows a graphical representation of the workflow and a stepwise activities and actions of the system. 31](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103024)

[Figure 15: This image shows a diagram that describes the structure of the Client subsystem by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016). 32](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103025)

[Figure 16: This image shows a diagram that describes the structure of the Watchdog subsystem by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016). 34](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103026)

[Figure 17: This image shows a diagram that describes the structure of the TCPServer subsystem by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016). 36](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103027)

[Figure 18: This image shows a diagram that describes the structure of the UDPServer subsystem by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016). 38](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103028)

[Figure 19: This image shows a diagram that describes the structure of the Game subsystem by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016). 40](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103029)

[Figure 20: This image shows a diagram that describes the structure of the GameActivity subsystem by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016). 41](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103030)

[Figure 21: This image shows a diagram that describes the structure of the GameMap subsystem by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016). 42](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103031)

[Figure 22: This image shows a diagram that describes the structure of the GamePosition subsystem by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016). 43](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103032)

[Figure 23: This image shows a diagram that describes the structure of the Utilities subsystem by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016). 44](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103033)

[Figure 24: This image shows a diagram that describes the structure of the Area subsystem by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016). 44](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103034)

[Figure 25: This image shows a diagram that describes the structure of the Menu subsystem by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016). 45](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103035)

[Figure 26: This image shows a diagram that describes the structure of the Score subsystem by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016). 46](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103036)

[Figure 27: This image shows a diagram that describes the structure of the Search subsystem by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016). 47](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103037)

[Figure 28: This image shows a diagram that describes the structure of the Set subsystem by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016). 48](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103038)

[Figure 29: This image shows a diagram that describes the structure of the Static subsystem by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016). 49](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103039)

[Figure 30: This image shows a diagram that describes the structure of the MainActivity subsystem by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016). 50](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103040)

[Figure 31: This image shows a representation of the simplest interactions a user can have with the system. 51](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103041)

[Figure 32: This image shows two examples of menu styles used in mobile application (Vogel, 2016) 52](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103042)

[Figure 33: This image shows examples of the user interface for the Android launcher Arrow Launcher by Microsoft (Funk, 2015) 53](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103043)

[Figure 34: This image shows an example of a retro arcade style menu . (Park Productions, 2015) 53](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103044)

[Figure 35: This image shows a standard game screen for the Google Maps API based game War2Map (Dempsey, 2013) 54](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103045)

[Figure 36: This image shows a diagram that describes the structure of the Tests subsystem by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016). 62](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103046)

[Figure 37: This image shows the legend or key for the code map diagrams. 77](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103047)

[Figure 38: 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). 78](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103048)

[Figure 39: This image shows a diagram that describes the structure of the Networking subsystem by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016). 79](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103049)

[Figure 40: This image shows a diagram that describes the structure of the CaptureTheCampus subsystem by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016). 80](file:///Z:\Development%20Project\Final%20Report\Capture%20the%20Campus.docx#_Toc481103050)

# 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).

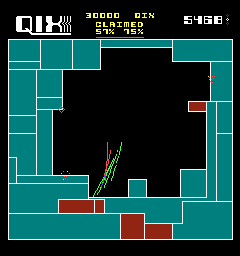


Figure : This is an image showing the gameplay of Qix (The International Arcade Museum, Museum of the Game, 2016).

The objective of Capture the Campus! will be like 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, 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 (‎3.2.3) 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 (‎3.2.3) 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 character’s 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. In Pokémon GO, the objective is to physically travel around the real world attempting to find and capture objects (Niantic, 2016).

Pokémon GO is an application developed for mobile devices, it operates using a mobile device’s GPS sensor to find the player’s location and synchronises all the player’s devices in the world using a centralised server (Niantic, 2016) (Lawson, 2016).

In Pokémon GO when an objective is captured its data is added to a database controlled by the player but it is not removed from the map and other players are free to capture it.

The user interface for Pokémon GO represents a player’s position using a human avatar that faces in the direction of travel and moves with a running motion from a player’s previous position to their current position when they move. Objectives are marked on the Pokémon GO map using a marker and are able to be interacted with when a player comes within a set distance or radius of them.

The gameplay of Pokémon GO borrows voraciously from capture the flag style games and could even be considered a semi pseudo geocaching game (Ordnance Survay, 2017) (Niantic, 2016).

An example of a standard Pokémon GO game screen can be seen below, including showing a player’s avatar and multiple objective markers (Figure 2).



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

Ingress is also a game developed by Niantic. In Ingress, the objective is to physically travel around the real world attempting to capture objects and control areas.

Ingress is an application developed for mobile devices, it operates using a mobile device’s GPS sensor to find the player’s location and synchronises the player’s devices in the world using a centralised server (Niantic, 2016).

In Ingress, the objective relies on controlling large swathes of the map by capturing nodes on the map. Players directly compete to hold the largest area; this is in contrast to Pokémon GO where players are not in direct competition with each other (Whelan, 2014).

The user interface for Ingress represents a player’s position using a marker that points in the direction of travel. Objectives are marked on the Ingress map using a glowing marker and are able to be interacted with when a player comes within a set distance or radius of them.

An example of a standard Ingress game screen can be seen below, including showing the areas controlled by two teams in one area of the map (Figure 3) (Figure 4).



Figure : This image shows a standard game screen for Ingress (Google, 2017)

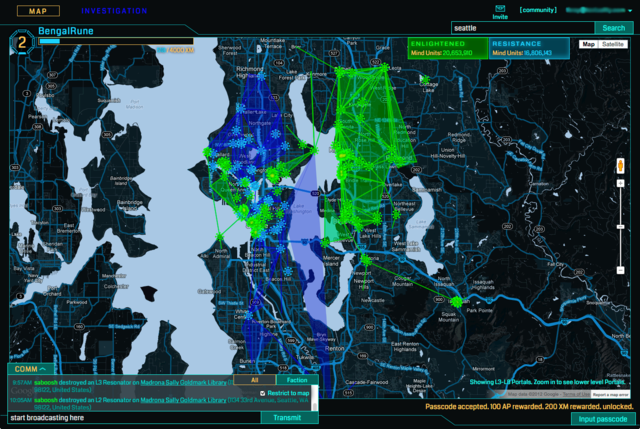


Figure : This image shows a standard game screen for Ingress (Wikipedia, 2017).

One useful critique of Pokémon GO and Ingress 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.

Another would be that Pokémon GO is a reskinning or asset swap of Ingress without any improvements if not actually a step backwards in technical advancements (Madhavan, 2016).

Ideas and themes that could be adapted from Pokémon GO and Ingress include; the server and database methods used in Pokémon GO and Ingress to synchronise captured objects between players of the game and the area capturing mechanics and user interface used in Ingress.

## Tracking

Imperative to the implementation of this project is for the mobile device, on which Capture the Campus! will be running, to be aware of its own location. In this case, the Global Positioning System (Brown & Sturza, 1995) will be used; this is because a large percentage of modern mobile devices contain some form of Global Positioning System sensor (Zhao, 2002).

The Global Positioning System is usually accurate regardless of weather conditions (Ian A.R. Hulbert, 2001). The Global Positioning System will work wherever there is an unobstructed line of sight to at least four Global Positioning System satellites.

Unfortunately, access to at least four satellites is required as each satellite is needed to pinpoint the user in one dimension of space or time (Crato, 2010).

The Global Positioning System can operate with only line of sight rather than through a mobile or internet connection because it works via a direct connection to the Global Positioning System satellites, which each contain their own atomic clock (Kaplan & Hegarty, 2006).

The position of the Global Positioning System satellites in orbit around the earth can be seen in the image below. (Figure 5)

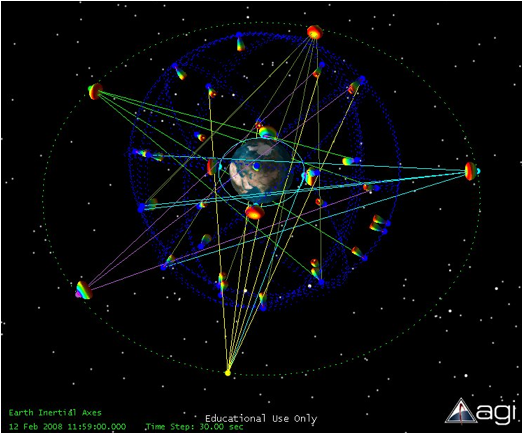


Figure : This image shows the orbits of some GPS satellites

The Global Positioning System works using modulated electromagnetic signals. These signals are first sent from the sensor in the host device to the Global Positioning System satellites orbiting the earth; this initial signal carries a pseudorandom code that verifies the identity of the sender.

After an initial connection is established an additional signal is sent which requests the time as stated by the atomic clocks contained within the orbiting Global Positioning System satellites, these satellites then respond with a similarly modulated signal containing the exact time that they believes it to be plus their position in orbit around the earth (Mooney, 1985).

The sender, after having waited to receive a response from a suitable number of Global Positioning System satellites in range, then uses the time as stated in each response to calculate the flight time of each message.

Knowing the flight time, the speed of the communication and the location in three-dimensional space of the Global Positioning System satellite that sent the response, the device can use basic trigonometry to determine a spherical area around each Global Positioning System satellite in which the device may be located. The point where the spheres of all the Global Positioning System satellites intersects is the point where the device is located (Ackermann, 1994).

This method is known as trilateration and a graphical representation of the Global Positioning System satellites and their possible location spheres can be seen in the image below (Tarasenko, 2009) (Figure 6).

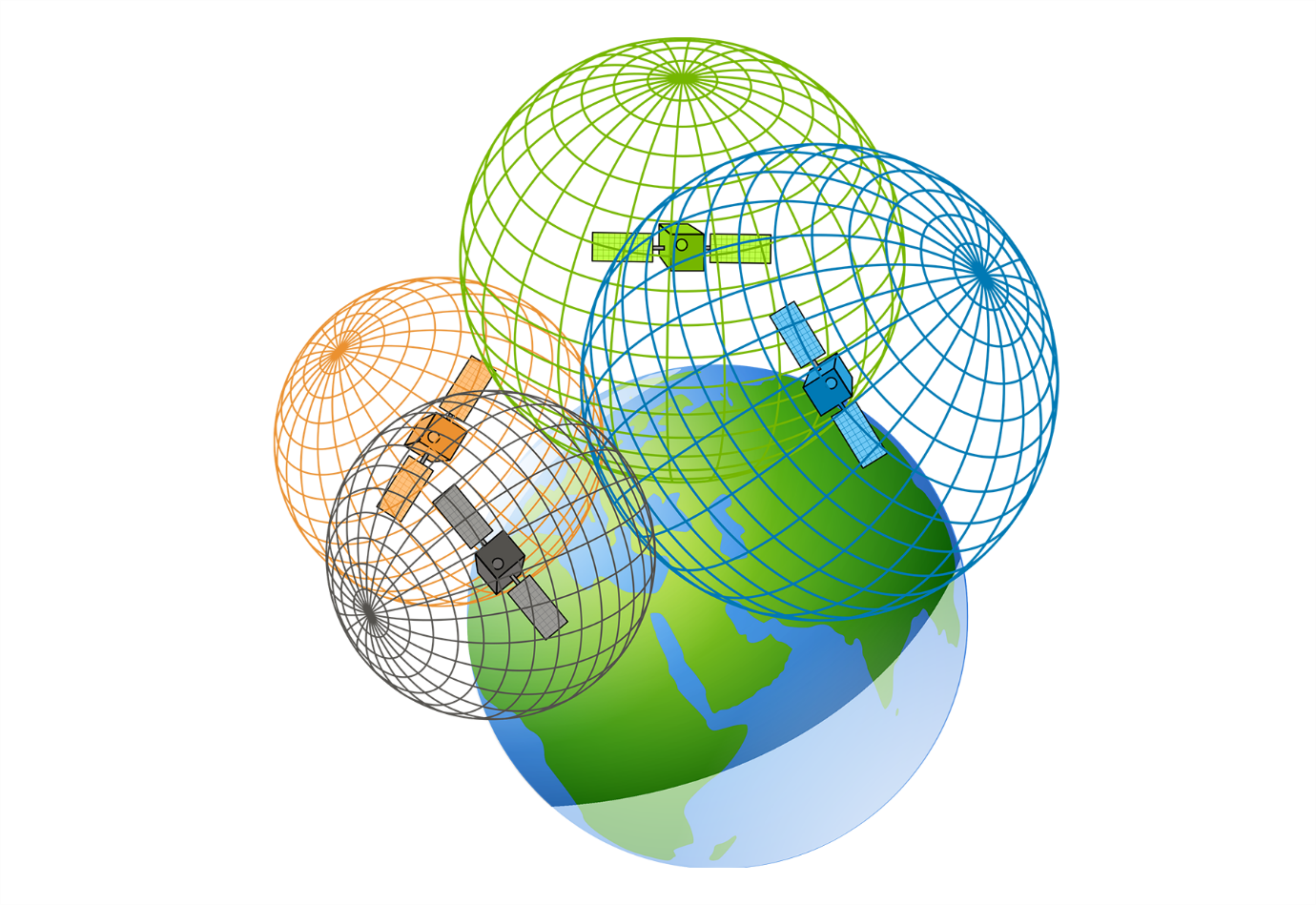


Figure : This image shows a visualisation of the act of GPS trilateration (openclipart, 2014)

Using modern programming languages and APIs, like C# or Java, it is a rather trivial affair to implement accurate Global Positioning System tracking (Anna, 2016) (Xamarin, 2017).

For instance, the Google Play Services SDK available for Android development allows the use of the devices Global Positioning System sensor through the Google Location Services API and the Fused Location Provider API.

The Google Location Services API provides access to the phones Global Positioning System sensor, which is highly accurate but drains the devices battery and will not work without line of sight (Google, 2017).

The Fused Location Provider API augments the Google Location Services API with device location information attained using the devices mobile or internet connections, if applicable. This allows the device to attain highly accurate location information without draining the mobile devices battery and even when there is not direct line of sight to the Global Positioning System satellites (Google, 2016).

## Comparison of Technologies

## Platform

Before development begins with aplomb, a target platform must be chosen for the application to be developed for. One of the requirements for this project is that the target platform must be a mobile platform. Thus, a desktop or laptop computer platform can almost immediately be ruled out, as it would be either impossible or laborious to physically transport something with such size, weight and power requirements around to play. In addition, these kinds of platforms also do not tend to contain the required GPS sensors.

Therefore, the choice of target platform can be reduced even further with these requirements in mind to only include; Android phones, iOS phones, Windows phones, and GPS handhelds.

From these choices, the GPS handhelds can be safely disregarded as it can be assumed that the project should ideally be easily distributed. For the project to be run on a GPS handheld a specific handheld would have to selected and the project would need to be developed using the manufacturers APIs and SDKs to run solely on that device which does not meet the requirements for the specification (Garmin, 2008).

This leaves the choice of target platform as a tossup between the three-main mobile phone operating systems; Android, iOS, and Windows phone.

Unfortunately, because Windows phone has such a low market share it can also be eliminated as documentation and support will be lacking thus increasing the difficulty and length of development and drastically impacting the distribution of the project (Statista, 2016). A graph showing the market share of the main mobile operating systems can be seen below (Figure 7).

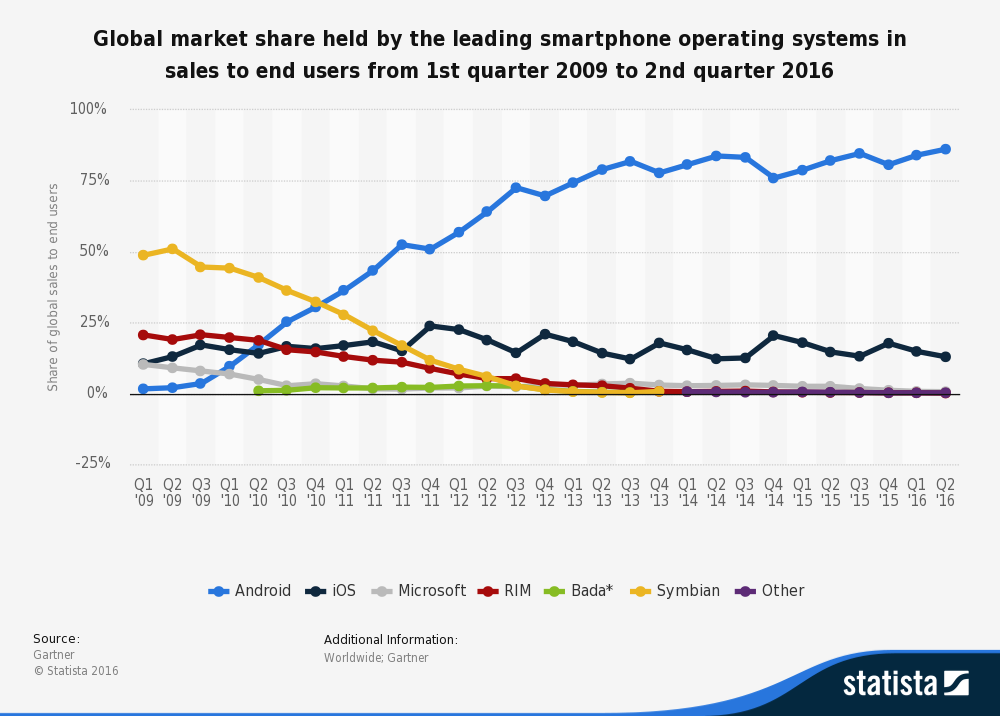


Figure : This image shows the current market share of mobile operating systems (Statista, 2016).

After the elimination of Windows phone, only Android and iOS remain as contenders for the target platform for the project.

On the surface it may appear that Android would be the obvious choice over iOS as it has a far greater market share. However, it has been noted in numerous reports that on average iOS users tend to spend substantially more money on quality content more than making up for their poor market share (Stenovec, 2015) (Sinicki, 2016).

Therefore, it matters more what the exact goal of this project is in regards to distribution and profitability. If the goal of this project were mass distribution, then Android would by far be the best choice. However, if the goal were to be profitable, then iOS would be a better target platform (Sunny, 2016).

Rather than attempting to separate the candidates on business ventures alone it would be more fruitful to acknowledge the experience of having to developing in and for their respective environments.

Designing the user interface for an Android application is far easier than designing the user interface for an iOS application as the design specification for Android applications are much more thoroughly documented than the design specifications for iOS applications (O’Sullivan, 2015).

The process to publish applications to the Google Play Store and to receive approval from the Google Play store is far simpler and takes less time on average than it does to publish applications to the Apple App Store. This is because anyone can upload their Android application to the Google Play store and receive approval in minutes by an autonomous system put into place by Google, whereas Apple insists on having each application tested by a real-life human employee before issuing approval for the application to enter the Apple App Store. This process takes a very long time and has a high rejection rate (Sinicki, 2016).

Obviously both platforms have their merits, given the current features of both operating systems it is too close to call a superior platform.

## Language

For development to begin finally on the project a programming language must be chosen for the project to be written in.

As discussed in the Platform section (‎3.2.1) previously the platform to develop the project for has now been narrowed down to either Android or iOS.

If the decision is made to develop for Android the languages available to write the project in are Java using the Android Studio integrated development environment or in C# via the Xamarin for Visual Studio integrated development environment.

Examples of the environment used to write in these languages can be seen in the images below (Figure 8) (Figure 9).

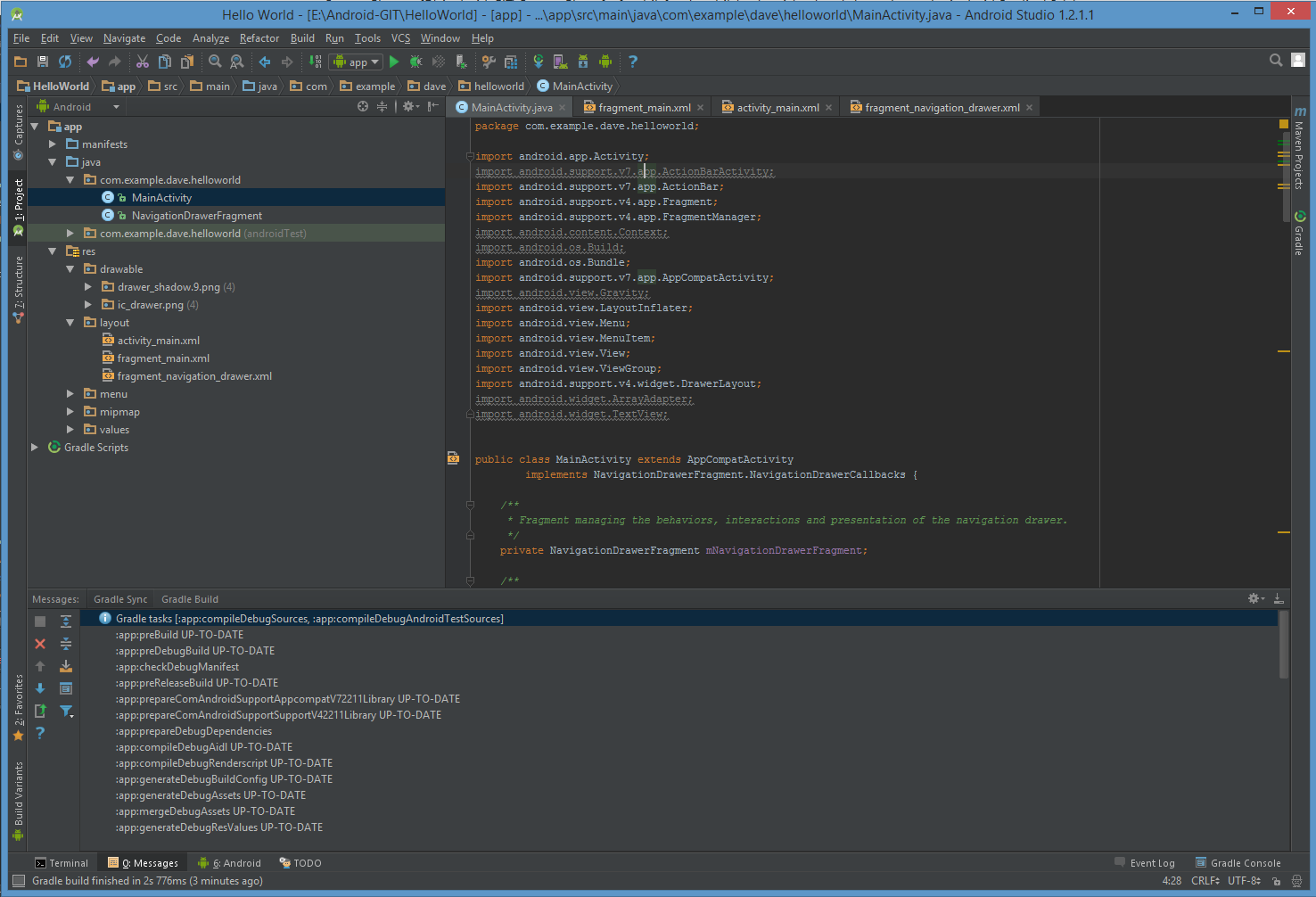


Figure : This image shows an example of a standard Android Studio integrated development environment window (Wikipedia, 2017).

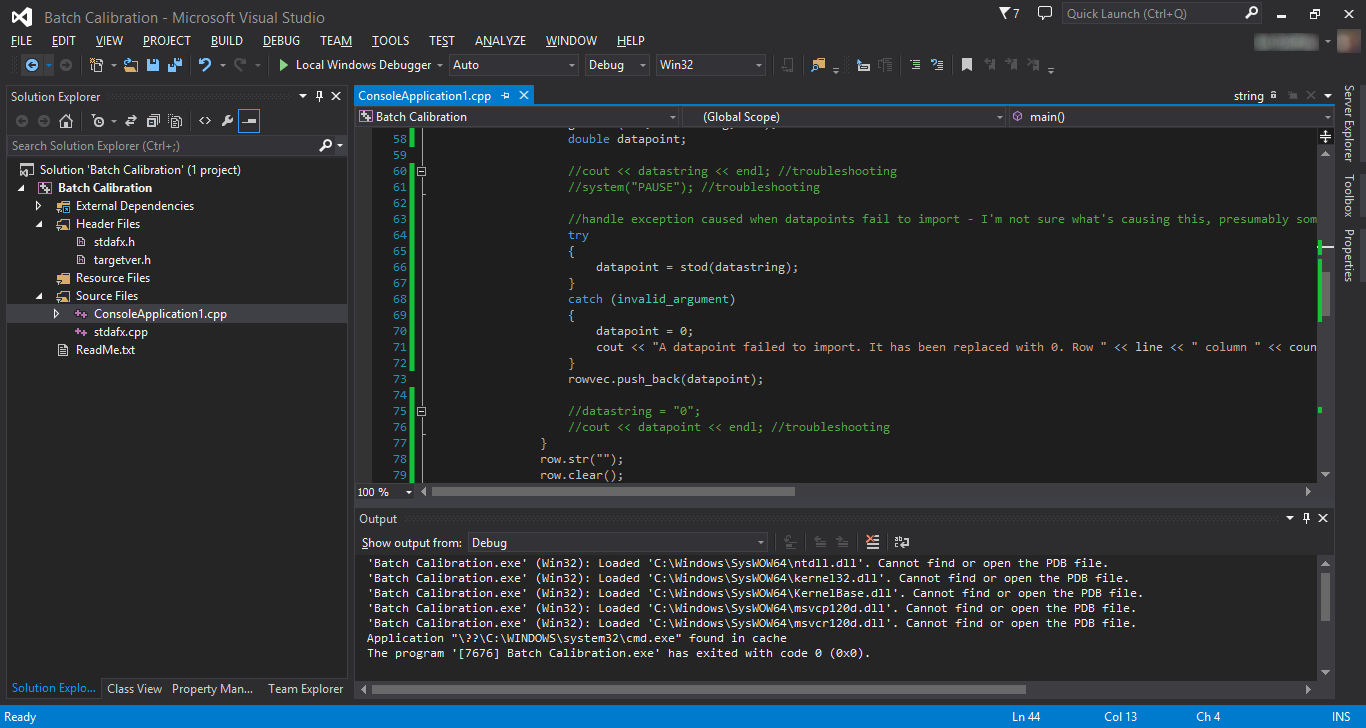


Figure : This image shows an example of a standard Visual Studio integrated development environment window (Wikipedia, 2017).

The Android compilers for both the Java and Xamarin programming languages produce native code solutions for Android devices, these native code solutions run directly on the target devices processor without an emulation layer (Google, n.d.) (Xamarin Inc., 2016).

An emulation layer would slow down the execution of the application as it sits between the compiled code and the processor translating the compiled code in real time into something the processor can understand (Rouse, 2006).

Because the Java and Xamarin programming languages both produce native Android solutions, they should both produce applications which run at similar speeds. However, Xamarin also allows for the use of the standard Microsoft .NET libraries, which Java does not (Montemagno, 2016). Thus, due to the project’s developer already being familiar with the standard Microsoft .NET libraries a solution should be able to be created in Xamarin at a greater speed and to a higher quality standard than in Java because of the lack of a bottleneck associated with the project’s developer having to learn a new programming language (Scholtz, 1992).

However, if the decision is made to develop for iOS instead, the languages available to write the project in are Swift using the Xcode integrated development environment or in C# via the Xamarin for Visual Studio integrated development environment.

Examples of the environment used to write in these languages can be seen in the following images (Figure 10) (Figure 9).

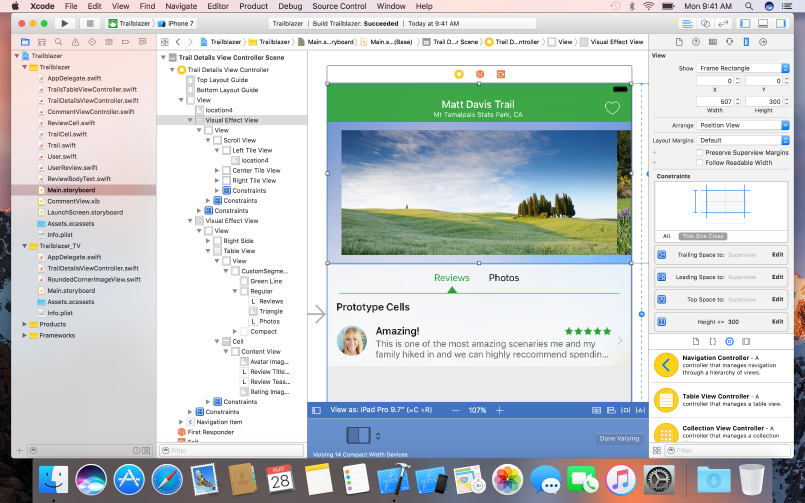


Figure : This image shows an example of a standard Xcode integrated development environment window (Apple, 2017).

The iOS compilers for both the Swift and Xamarin programming languages produce native code solutions for Android devices (Apple, 2016) (Xamarin Inc., 2016).

Because the Swift and Xamarin programming languages both produce native iOS solutions, they should both produce applications which run at similar speeds. However, Xamarin also allows for the use of the standard Microsoft .NET libraries, which Swift does not (Montemagno, 2016). Thus, due to the project’s developer already being familiar with the standard Microsoft .NET libraries a solution should be able to be created in Xamarin at a greater speed and to a higher quality standard than in Swift.

Another advantageous feature of Xamarin is that it can be used to write programs for both Android and iOS using a shared common code base, because of this it is unnecessary to choose between both Android and iOS as the target platform.

This means that the theoretical user base for the project raises from 86.2% or 12.9% alone targeting either Android or iOS alone respectively to a combined total user base of 99.1% (Xamarin Inc., 2016) (Statista, 2016).

Therefore, C# via the Xamarin for Visual Studio integrated development environment can be safely chosen as the programming language of choice to write this project in.

## Networking

## Architecture

To implement a multiplayer version of the project some form of networking component must be created to synchronise all current players.

The networking component, as discussed in the Language section (‎3.2.2), will be written in the Xamarin programming language as it allows for the development of a high quality, cross platform solution (Xamarin Inc., 2016).

Xamarin will also allow the network component to be run from either a desktop or a mobile device using a wrapper program that allows for easy debugging and rapid prototyping (Xamarin Inc., 2016).

The ability for the networking component to be able to be run from either a desktop or a mobile device is desirable as it allows the option for the project’s developer to define either a predefined, offsite, central server for all users to use or multiple local servers that can be run directly from the player’s mobile device.

There are several models and schools of thought on how networking components should be implemented and the manner in which they go about synchronising data between themselves.

One network model would be peer-to-peer architecture. In a peer-to-peer network each device on a network synchronises directly with every other device on the network equally; this form of network structure generally has no central storage or user authentication (Posey, 2000).

Another network model would be client/server architecture where there are separate dedicated clients and servers; this form of network structure works solely through a central storage server and usually has some form of user authentication (Posey, 2000).

These two different architectures can be visualised in the image below (Figure 11).

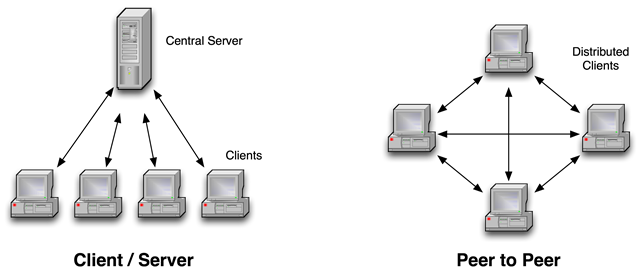


Figure : This image shows a visual model of the difference between a client/server model on the left and a peer-to-peer model on the right (Philips, 2014).

Peer-to-peer networks operate at a reasonable speed with few clients over short distance, similar speed network connections (Baccelli, et al., 2013).

However, peer-to-peer networks struggle when scaled up to a larger number of clients, this is due to the whole network being bottlenecked by the speed of the slowest network connection in the network (Baccelli, et al., 2013).

Conversely, client/server networks can handle a large number of users simultaneously with ease. This is because each client in a client/server network has their own private connection to the server; the performance of each client’s connection is only affected by their own connection to the server (Sparrow, 2017).

However, client/server networks usually require an offsite server or servers be dedicated to the application at all times to operate, if these server become unavailable the whole network is compromised (Sparrow, 2017).

For this project, a client/server model would be most effective, this is because the player’s mobile device on which the application is running will probably have a unreliable network connection (Zhang & Soong, 2006). Unreliable mobile network connections are partly due to frequent network handovers where a device moves between the boundaries of areas covered by different networks (Ali, et al., n.d.).

Therefore, if a peer-to-peer architecture were selected the poor mobile network connection between the players would cause the application to constantly halt while waiting for a response from a player who is no longer in fact connected to the network.

In addition, because the application is designed for mobile devices it can be assumed that the players are on a rated connection, where every piece of data could cost the players money to send, it would be unnecessary to cause the players avoidable cost.

Thus again, a client/server model would be the most effective implementation as it requires the players to send and receive only one message to synchronise data across all devices, this is because all players communicate with one central server where all data is stored.

## Protocol

To create a network connection between client and server a transport layer protocol is required to carry each message. There are a number of protocols that the client and server could be written to send and receive, each with its own uses, advantages and disadvantages.

The protocols that the client and server could be written to send and receive include:

The Remote Method Invocation protocol, a high latency, low reliability, low throughput protocol (Taing, 2011).

The User Datagram Protocol, a low latency, lower reliability protocol, high throughput protocol with the ability to broadcast a message to all IP addresses on the local network (TechTarget, 2015) (Seguin, 2014).

The Transmission Control Protocol, a higher latency, high reliability connection-oriented protocol that as long as there is a network connection between two devices any message sent using it is guaranteed to arrive (TechTarget, 2014) (Diffen, n.d.).

The relative performance of these three transport layer protocols can be seen in the two diagrams below (Figure 12) (Figure 13).

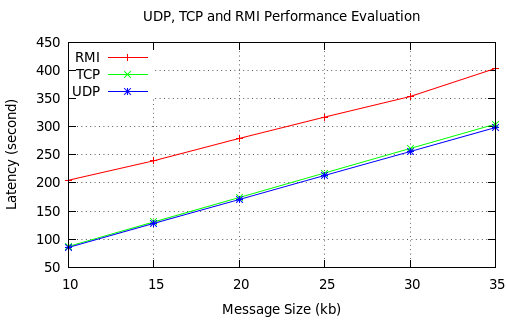


Figure : This diagram shows the latency involved in sending a message using the different transport layer protocols (Taing, 2011).

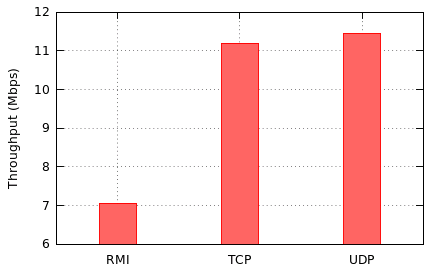


Figure : This diagram shows the throughput in Mbps of the different transport layer protocols (Taing, 2011).

This project requires a method by which to establish an initial connection between the clients and server, then once a connection has been established this project requires a method by which to synchronise data between the clients and server.

The User Datagram Protocol is perfect for establishing an initial connection between the clients and server because of its low latency and ability to broadcast a message across all IP addresses on the local network (Mey, n.d.).

The Transmission Control Protocol is then useful once the initial connection has been established to synchronise data between the clients and server, this is because the Transmission Control Protocol that guarantees that if there is a network connection between two devices any message sent using it is will arrive (Snader, n.d.).

# Technical Development

## System Design

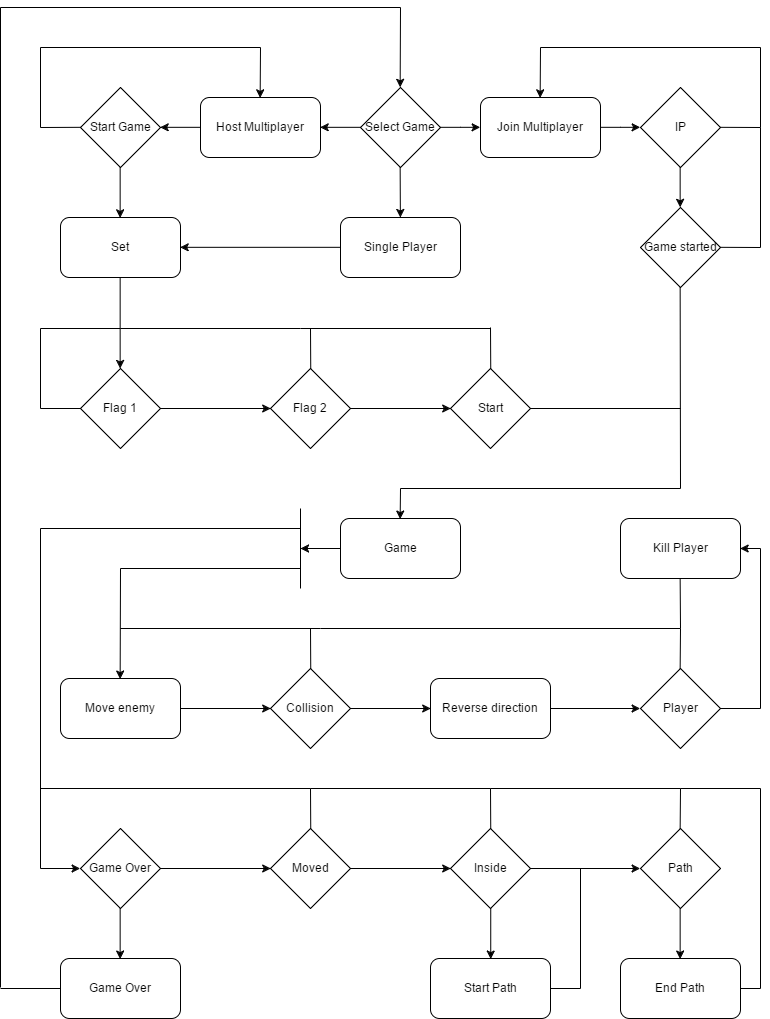


Figure : This image shows a graphical representation of the workflow and a stepwise activities and actions of the system.

## Networking Design

## Client Design

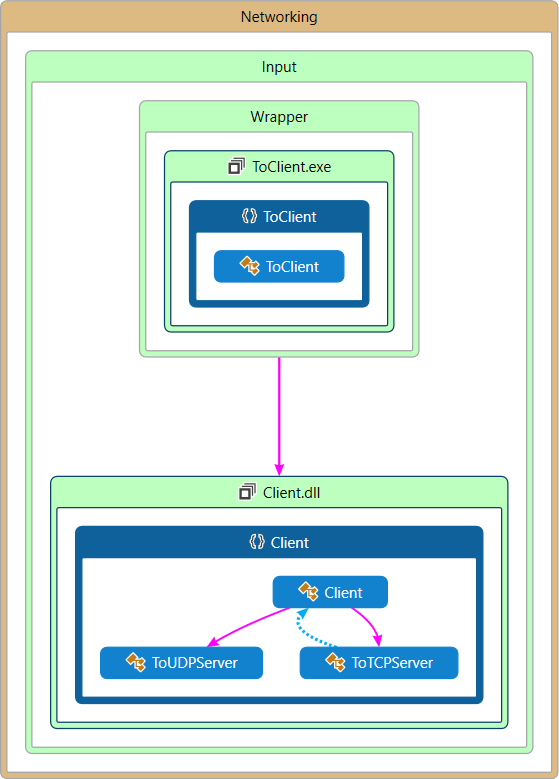


Figure : This image shows a diagram that describes the structure of the Client subsystem by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016).

The diagram above (Figure 15) shows a graphical representation of the solution for the Client, this diagram is called a code map.

This code map shows both the projects of the solution and classes contained within these projects which when compiled create the client.

Included in the code map are arrows that depict inheritance between classes, objects being called, fields that are read from and written to and references from one class to another. In this instance, green arrows represent inheritance, purple arrows represent object calls, blue arrows represent field reads or writes and grey arrows represent references.

The full legend for the code map can be seen below (‎Appendix B:).

As can be seen in the code map above (Figure 15) in the solution for the Client there are two projects; an application called ToClient contained within a folder named Wrapper and a library called Client, both projects are themselves contained within a folder called Input which itself is contained within a folder named Networking.

The Client library has been designed in a modular fashion so that it can be adapted to work in any solution to send either UDP or TCP messages.

The Client library contains three classes:

The first class takes an input from the user, parses it and then passes the relevant information onto one of the other two classes in the project, this class is called the Client class.

Within the Client class is a public method named Input, the Input method is the method that should be called whenever a network communication is to be sent. To call this method an array of strings must be passed to it as an argument, the array of strings must at least contain a flag denoting the protocol to be used and the identifier of the information to be received or the identifier of the information to be sent and the piece of information.

The order of the strings in the string array does not necessarily matter; however, a piece of information must always follow its flag.

The flags that can be used to control the behaviour of the method include; “-t” which denotes that the message should be transmitted using the TCP protocol, “-u” which denotes that the message should be transmitted or received using the UDP protocol, “-i” which denotes that the following string in the string array is the IP address that the message should be sent to or received from and “-p” which denotes that the following string in the string array is the port that the message should be sent to or received from.

The second class takes the output from the Input class, sends it via the TCP protocol to a socket over the network defined by the IP address and the port specified by the input string and then waits a suitable amount of time for a relevant response from the recipient, this class is called the ToTCPServer class.

If the argument string from the Input class contains both an identifier and a piece of information the ToTCPServer class will immediately return after sending its message. However, if the argument string from the Input class contains only an identifier the ToTCPServer class will wait for a response and then return that, if the ToTCPServer class does not receive a response after three seconds an appropriate error message is returned instead.

The third class either takes the output from the Input class and broadcasts it via the UDP protocol to any port specified by the input string on the local network or the third class receives a message broadcasted via the UDP protocol on the port specified by the input string on the local network. Regardless of which mode the third class is set to it waits a suitable amount of time for a relevant message, if the third class does not receive a response after three seconds an appropriate error message is returned. This class is called the ToUDPServer class.

The ToClient application is a test wrapper for the Client library. The ToClient application has been designed to allow arguments for the Client library to be built from the command prompt for rapid testing and prototyping.

The ToClient application contains one class:

The class takes an input from the user, constructs a string array from it and then passes this string array as an argument to the public Input method of the Client library, this class is called the ToClient class.

The command line argument to this application is a series of strings separated by spaces, the string must at least contain a flag denoting the protocol to be used and the identifier of the information to be received or the identifier of the information to be sent and the piece of information.

The order of the strings in the command line argument does not necessarily matter; however, a piece of information must always follow its flag.

The flags that can be used to control the behaviour of the Client library include; “-t” which denotes that the message should be transmitted using the TCP protocol, “-u” which denotes that the message should be transmitted or received using the UDP protocol, “-i” which denotes that the following string in the command line argument is the IP address that the message should be sent to or received from and “-p” which denotes that the following string in the string array is the port that the message should be sent to or received from.

A useful advantage of the ToClient application test wrapper is that it allows for the autonomous testing of the Client libraries functionality through the piping of inputs from a text file to the post build arguments section of the application (Jeremy, 2014).

Most of the classes contained within the projects of this solution and their contents are private.

This means that the contents of the solution’s classes are only accessible from within the classes of the solution, this is because the contents of these classes have no need to be edited from anywhere but from within themselves.

Being private allows for the application of encapsulation meaning that the value or state of structured data within the class is hidden from the user when they are outside of the class.

This is useful as it blocks unauthorised access to sensitive information held within the class, information such as a user’s location (Noble, 2002).

Encapsulation also stops fellow developers, who may not fully comprehend an implementation, from breaking fundamental aspects of the underlying system (Berard, 2015).

## Watchdog Design

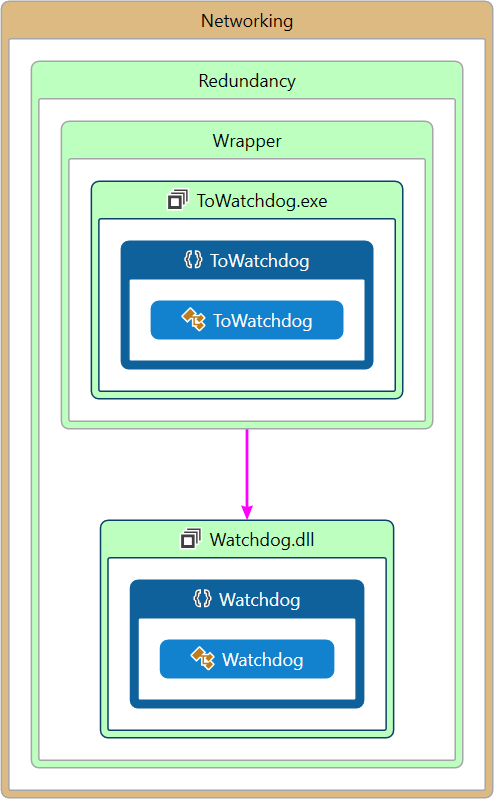


Figure : This image shows a diagram that describes the structure of the Watchdog subsystem by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016).

The diagram above (Figure 16) shows a graphical representation of the solution for the Watchdog, this diagram is called a code map.

This code map shows both the projects of the solution and classes contained within these projects which when compiled create the client.

Included in the code map are arrows that depict inheritance between classes, objects being called, fields that are read from and written to and references from one class to another. In this instance, green arrows represent inheritance, purple arrows represent object calls, blue arrows represent field reads or writes and grey arrows represent references.

The full legend for the code map can be seen below (‎Appendix B:).

As can be seen in the code map above (Figure 16) in the solution for the Watchdog there are two projects; an application called ToWatchdog contained within a folder named Wrapper and a library called Watchdog, both projects are themselves contained within a folder called Redundancy which itself is contained within a folder named Networking.

The Watchdog library has been designed in a modular fashion so that it can be adapted to work in any solution as a watchdog or heartbeat to a server.

The Watchdog library contains one class:

The first class takes an input from the user, parses it and then passes the relevant information onto one of the other two classes in the project, this class is called the Client class.

Within the Client class is a public method named Input, the Input method is the method that should be called whenever a network communication is to be sent. To call this method an array of strings must be passed to it as an argument, the array of strings must at least contain a flag denoting the protocol to be used and the identifier of the information to be received or the identifier of the information to be sent and the piece of information.

The order of the strings in the string array does not necessarily matter; however, a piece of information must always follow its flag.

The flags that can be used to control the behaviour of the method include; “-t” which denotes that the message should be transmitted using the TCP protocol, “-u” which denotes that the message should be transmitted or received using the UDP protocol, “-i” which denotes that the following string in the string array is the IP address that the message should be sent to or received from and “-p” which denotes that the following string in the string array is the port that the message should be sent to or received from.

The ToWatchdog application is a test wrapper for the Watchdog library. The ToWatchdog application has been designed to allow the Watchdog library to be built from the command prompt for rapid testing and prototyping.

The ToWatchdog application contains one class:

The class takes an input from the user, constructs a string array from it and then passes this string array as an argument to the public Input method of the Client library, this class is called the ToClient class.

The command line argument to this application is a series of strings separated by spaces, the string must at least contain a flag denoting the protocol to be used and the identifier of the information to be received or the identifier of the information to be sent and the piece of information.

The order of the strings in the command line argument does not necessarily matter; however, a piece of information must always follow its flag.

The flags that can be used to control the behaviour of the Client library include; “-t” which denotes that the message should be transmitted using the TCP protocol, “-u” which denotes that the message should be transmitted or received using the UDP protocol, “-i” which denotes that the following string in the command line argument is the IP address that the message should be sent to or received from and “-p” which denotes that the following string in the string array is the port that the message should be sent to or received from.

A useful advantage of the ToClient application test wrapper is that it allows for the autonomous testing of the Client libraries functionality through the piping of inputs from a text file to the post build arguments section of the application (Jeremy, 2014).

Most of the classes contained within the projects of this solution and their contents are private.

This means that the contents of the solution’s classes are only accessible from within the classes of the solution, this is because the contents of these classes have no need to be edited from anywhere but from within themselves.

Being private allows for the application of encapsulation meaning that the value or state of structured data within the class is hidden from the user when they are outside of the class.

This is useful as it blocks unauthorised access to sensitive information held within the class, information such as a user’s location (Noble, 2002).

Encapsulation also stops fellow developers, who may not fully comprehend an implementation, from breaking fundamental aspects of the underlying system (Berard, 2015).

## TCPServer Design

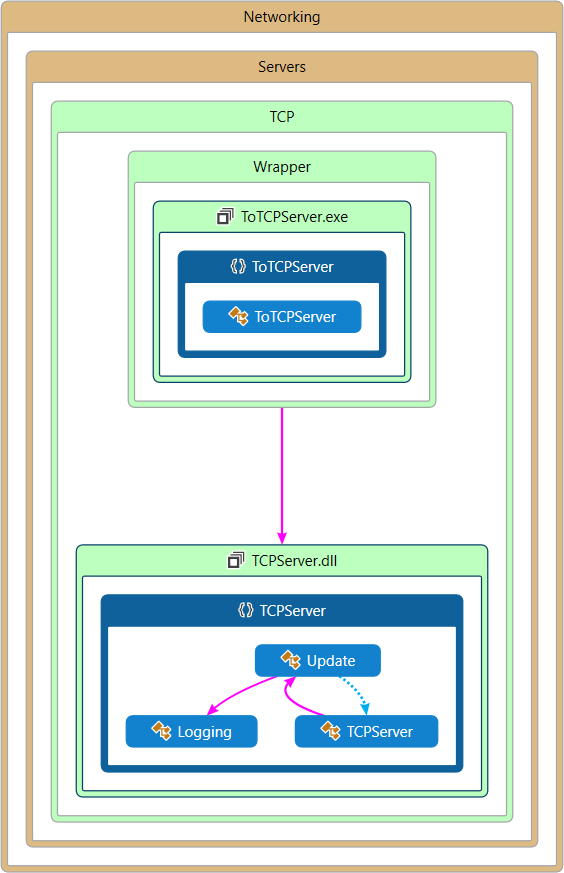


Figure : This image shows a diagram that describes the structure of the TCPServer subsystem by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016).

The diagram above (Figure 15) shows a graphical representation of the solution for the client, this diagram is called a code map.

This code map shows both the projects of the solution and classes contained within these projects which when compiled create the client.

Included in the code map are arrows that depict inheritance between classes, objects being called, fields that are read from and written to and references from one class to another. In this instance, green arrows represent inheritance, purple arrows represent object calls, blue arrows represent field reads or writes and grey arrows represent references.

The full legend for the code map can be seen below (‎Appendix B:).

As can be seen in the code map above (Figure 15) in the solution for the client there are two projects; an application called ToClient contained within a folder named Wrapper and a library called Client, both projects are themselves contained within a folder called Input which itself is contained within a folder named Networking.

The Client library has been designed in a modular fassion so that it can be adapted to work in any solution to send either UDP or TCP messages.

The Client library contains three classes:

The first class takes an input from the user, parses it and then passes the relevant information onto one of the other two classes in the project, this class is called the Client class.

Within the Client class is a public method named Input, the Input method is the method that should be called whenever a network communication is to be sent. To call this method an array of strings must be passed to it as an argument, the array of strings must at least contain a flag denoting the protocol to be used and the identifier of the information to be received or the identifier of the information to be sent and the piece of information.

The order of the strings in the string array does not necessarily matter; however, a piece of information must always follow its flag.

The flags that can be used to control the behaviour of the method include; “-t” which denotes that the message should be transmitted using the TCP protocol, “-u” which denotes that the message should be transmitted or received using the UDP protocol, “-i” which denotes that the following string in the string array is the IP address that the message should be sent to or received from and “-p” which denotes that the following string in the string array is the port that the message should be sent to or received from.

The second class takes the output from the Input class, sends it via the TCP protocol to a socket over the network defined by the IP address and the port specified by the input string and then waits a suitable amount of time for a relevant response from the recipient, this class is called the ToTCPServer class.

If the argument string from the Input class contains both an identifier and a piece of information the ToTCPServer class will immediately return after sending its message. However, if the argument string from the Input class contains only an identifier the ToTCPServer class will wait for a response and then return that, if the ToTCPServer class does not receive a response after three seconds an appropriate error message is returned instead.

The third class either takes the output from the Input class and broadcasts it via the UDP protocol to any port specified by the input string on the local network or the third class receives a message broadcasted via the UDP protocol on the port specified by the input string on the local network. Regardless of which mode the third class is set to it waits a suitable amount of time for a relevant message, if the third class does not receive a response after three seconds an appropriate error message is returned. This class is called the ToUDPServer class.

The ToClient application is a test wrapper for the Client library. The ToClient application has been designed to allow arguments for the Client library to be built from the command prompt for rapid testing and prototyping.

The ToClient application contains one class:

The class takes an input from the user, constructs a string array from it and then passes this string array as an argument to the public Input method of the Client library, this class is called the ToClient class.

The command line argument to this application is a series of strings separated by spaces, the string must at least contain a flag denoting the protocol to be used and the identifier of the information to be received or the identifier of the information to be sent and the piece of information.

The order of the strings in the command line argument does not necessarily matter; however, a piece of information must always follow its flag.

The flags that can be used to control the behaviour of the Client library include; “-t” which denotes that the message should be transmitted using the TCP protocol, “-u” which denotes that the message should be transmitted or received using the UDP protocol, “-i” which denotes that the following string in the command line argument is the IP address that the message should be sent to or received from and “-p” which denotes that the following string in the string array is the port that the message should be sent to or received from.

A useful advantage of the ToClient application test wrapper is that it allows for the autonomous testing of the Client libraries functionality through the piping of inputs from a text file to the post build arguments section of the application (Jeremy, 2014).

Most of the classes contained within the projects of this solution and their contents are private.

This means that the contents of the solution’s classes are only accessible from within the classes of the solution, this is because the contents of these classes have no need to be edited from anywhere but from within themselves.

Being private allows for the application of encapsulation meaning that the value or state of structured data within the class is hidden from the user when they are outside of the class.

This is useful as it blocks unauthorised access to sensitive information held within the class, information such as a user’s location (Noble, 2002).

Encapsulation also stops fellow developers, who may not fully comprehend an implementation, from breaking fundamental aspects of the underlying system (Berard, 2015).

## UDPServer Design

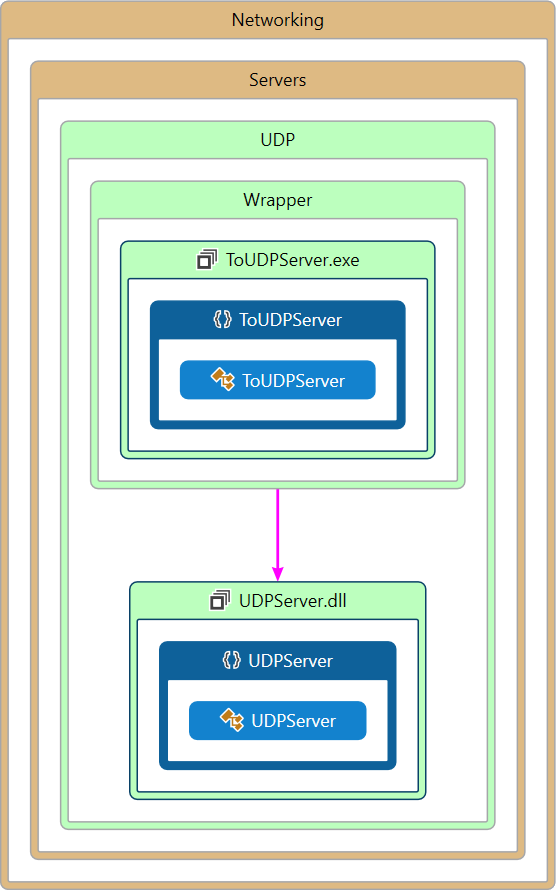


Figure : This image shows a diagram that describes the structure of the UDPServer subsystem by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016).

The diagram above (Figure 15) shows a graphical representation of the solution for the client, this diagram is called a code map.

This code map shows both the projects of the solution and classes contained within these projects which when compiled create the client.

Included in the code map are arrows that depict inheritance between classes, objects being called, fields that are read from and written to and references from one class to another. In this instance, green arrows represent inheritance, purple arrows represent object calls, blue arrows represent field reads or writes and grey arrows represent references.

The full legend for the code map can be seen below (‎Appendix B:).

As can be seen in the code map above (Figure 15) in the solution for the client there are two projects; an application called ToClient contained within a folder named Wrapper and a library called Client, both projects are themselves contained within a folder called Input which itself is contained within a folder named Networking.

The Client library has been designed in a modular fassion so that it can be adapted to work in any solution to send either UDP or TCP messages.

The Client library contains three classes:

The first class takes an input from the user, parses it and then passes the relevant information onto one of the other two classes in the project, this class is called the Client class.

Within the Client class is a public method named Input, the Input method is the method that should be called whenever a network communication is to be sent. To call this method an array of strings must be passed to it as an argument, the array of strings must at least contain a flag denoting the protocol to be used and the identifier of the information to be received or the identifier of the information to be sent and the piece of information.

The order of the strings in the string array does not necessarily matter; however, a piece of information must always follow its flag.

The flags that can be used to control the behaviour of the method include; “-t” which denotes that the message should be transmitted using the TCP protocol, “-u” which denotes that the message should be transmitted or received using the UDP protocol, “-i” which denotes that the following string in the string array is the IP address that the message should be sent to or received from and “-p” which denotes that the following string in the string array is the port that the message should be sent to or received from.

The second class takes the output from the Input class, sends it via the TCP protocol to a socket over the network defined by the IP address and the port specified by the input string and then waits a suitable amount of time for a relevant response from the recipient, this class is called the ToTCPServer class.

If the argument string from the Input class contains both an identifier and a piece of information the ToTCPServer class will immediately return after sending its message. However, if the argument string from the Input class contains only an identifier the ToTCPServer class will wait for a response and then return that, if the ToTCPServer class does not receive a response after three seconds an appropriate error message is returned instead.

The third class either takes the output from the Input class and broadcasts it via the UDP protocol to any port specified by the input string on the local network or the third class receives a message broadcasted via the UDP protocol on the port specified by the input string on the local network. Regardless of which mode the third class is set to it waits a suitable amount of time for a relevant message, if the third class does not receive a response after three seconds an appropriate error message is returned. This class is called the ToUDPServer class.

The ToClient application is a test wrapper for the Client library. The ToClient application has been designed to allow arguments for the Client library to be built from the command prompt for rapid testing and prototyping.

The ToClient application contains one class:

The class takes an input from the user, constructs a string array from it and then passes this string array as an argument to the public Input method of the Client library, this class is called the ToClient class.

The command line argument to this application is a series of strings separated by spaces, the string must at least contain a flag denoting the protocol to be used and the identifier of the information to be received or the identifier of the information to be sent and the piece of information.

The order of the strings in the command line argument does not necessarily matter; however, a piece of information must always follow its flag.

The flags that can be used to control the behaviour of the Client library include; “-t” which denotes that the message should be transmitted using the TCP protocol, “-u” which denotes that the message should be transmitted or received using the UDP protocol, “-i” which denotes that the following string in the command line argument is the IP address that the message should be sent to or received from and “-p” which denotes that the following string in the string array is the port that the message should be sent to or received from.

A useful advantage of the ToClient application test wrapper is that it allows for the autonomous testing of the Client libraries functionality through the piping of inputs from a text file to the post build arguments section of the application (Jeremy, 2014).

Most of the classes contained within the projects of this solution and their contents are private.

This means that the contents of the solution’s classes are only accessible from within the classes of the solution, this is because the contents of these classes have no need to be edited from anywhere but from within themselves.

Being private allows for the application of encapsulation meaning that the value or state of structured data within the class is hidden from the user when they are outside of the class.

This is useful as it blocks unauthorised access to sensitive information held within the class, information such as a user’s location (Noble, 2002).

Encapsulation also stops fellow developers, who may not fully comprehend an implementation, from breaking fundamental aspects of the underlying system (Berard, 2015).

## CaptureTheCampus Design

## Game Design

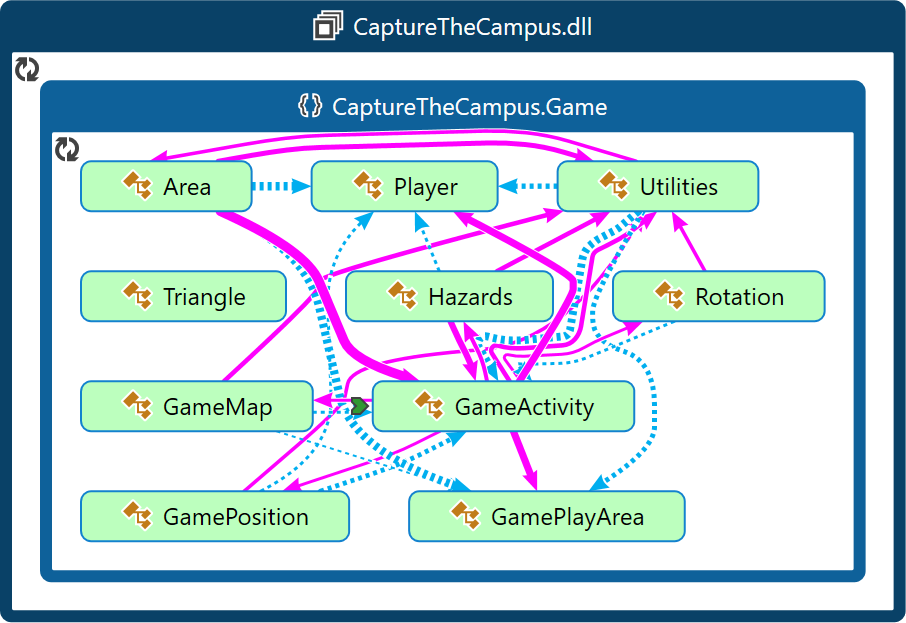


Figure : This image shows a diagram that describes the structure of the Game subsystem by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016).

## GameActivity Design

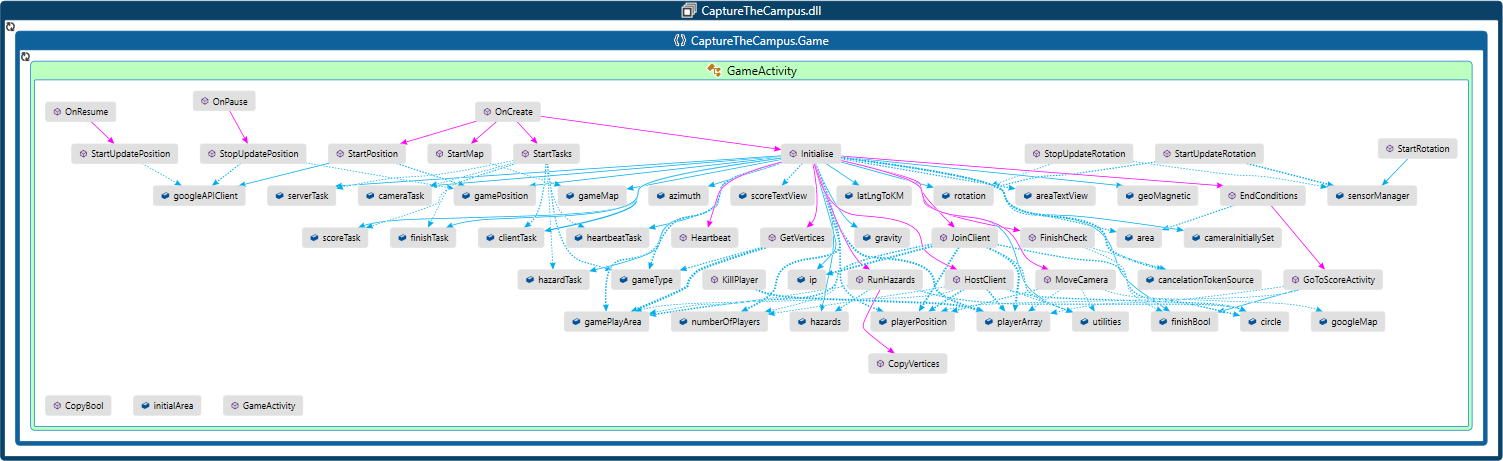


Figure : This image shows a diagram that describes the structure of the GameActivity subsystem by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016).

## GameMap Design

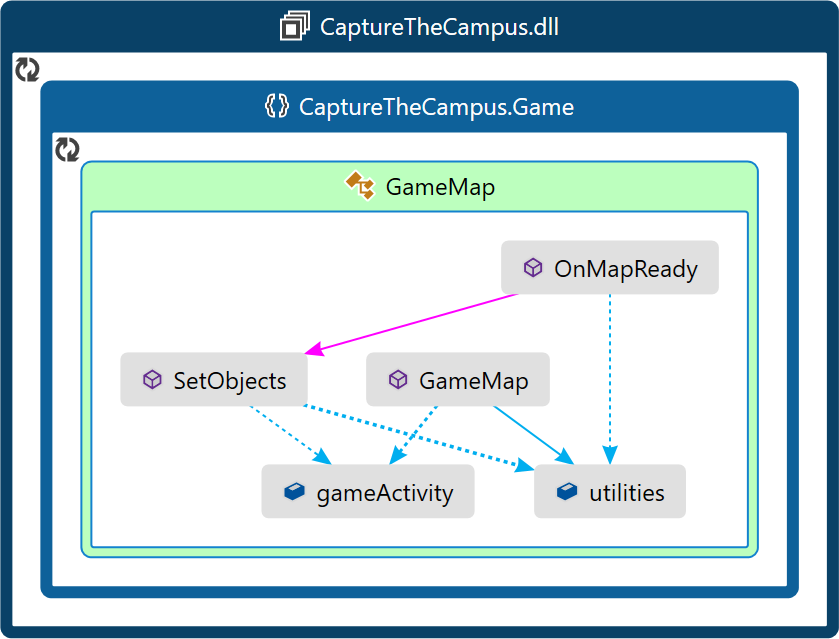


Figure : This image shows a diagram that describes the structure of the GameMap subsystem by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016).

## GamePosition Design

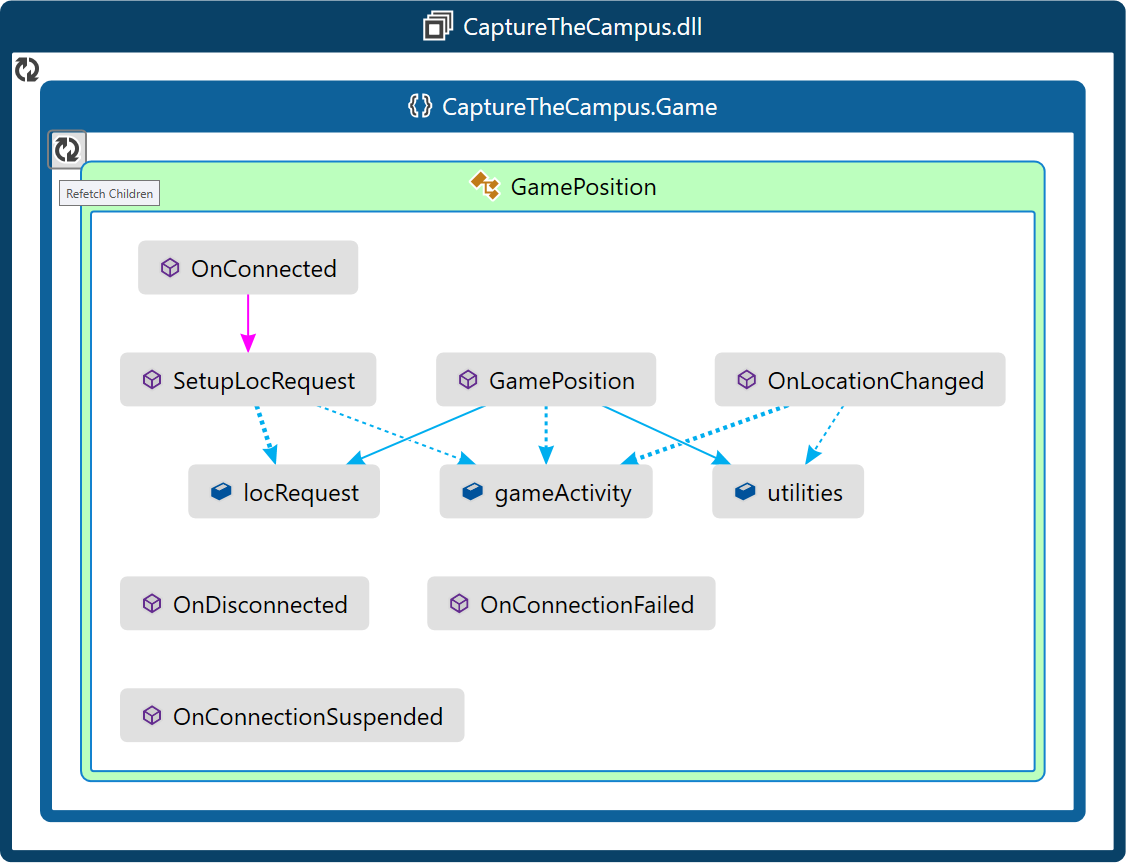


Figure : This image shows a diagram that describes the structure of the GamePosition subsystem by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016).

## Utilities Design

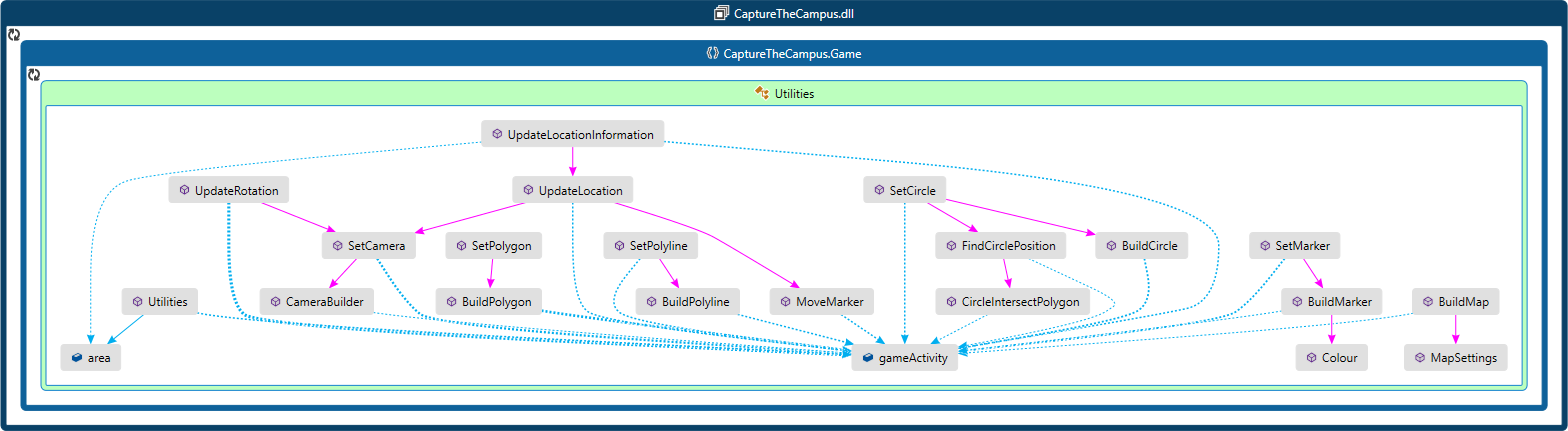


Figure : This image shows a diagram that describes the structure of the Utilities subsystem by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016).

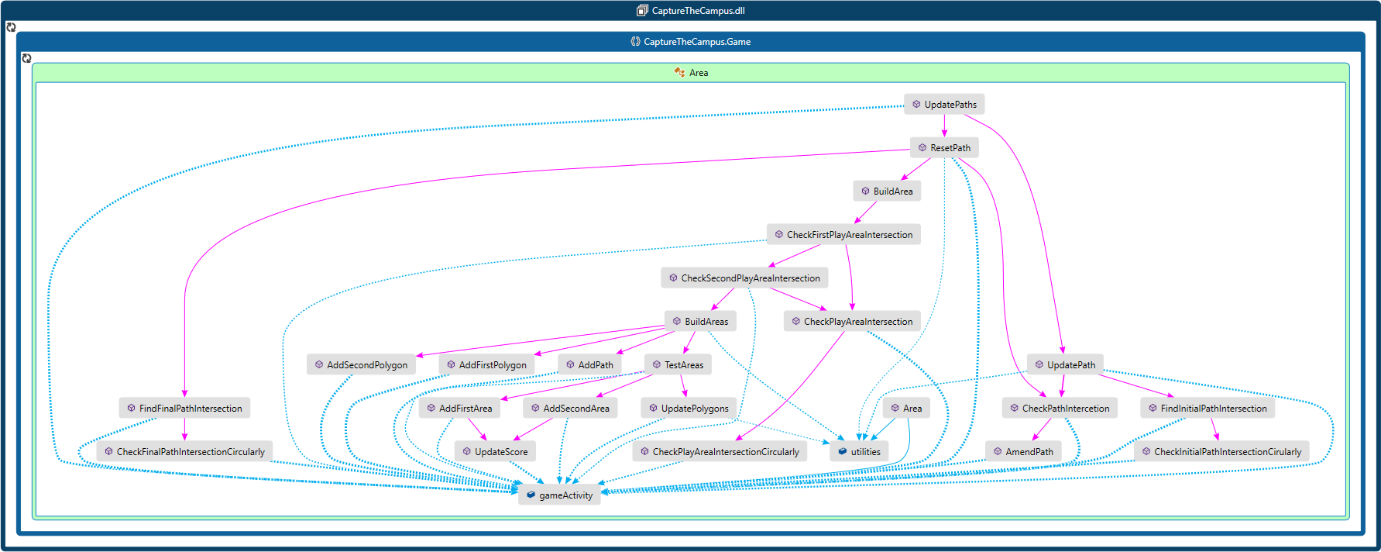


Figure : This image shows a diagram that describes the structure of the Area subsystem by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016).

## Menu Design



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

## Score Design



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

## Search Design

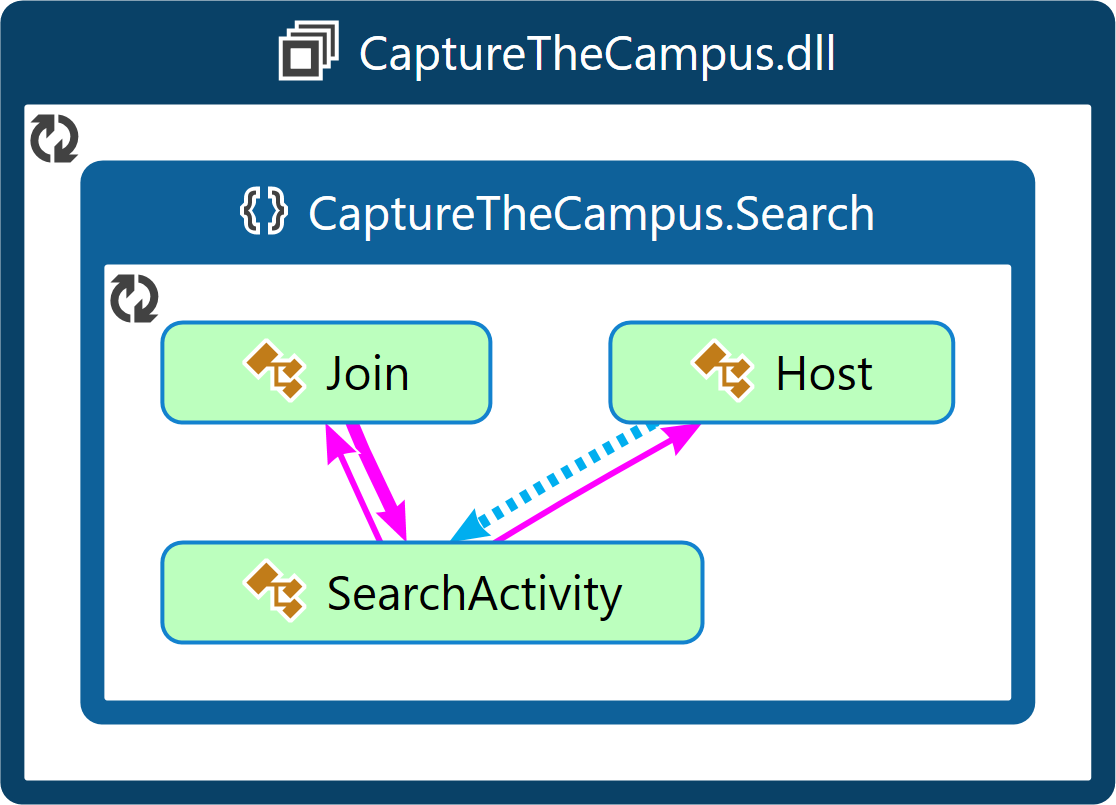


Figure : This image shows a diagram that describes the structure of the Search subsystem by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016).

## Set Design

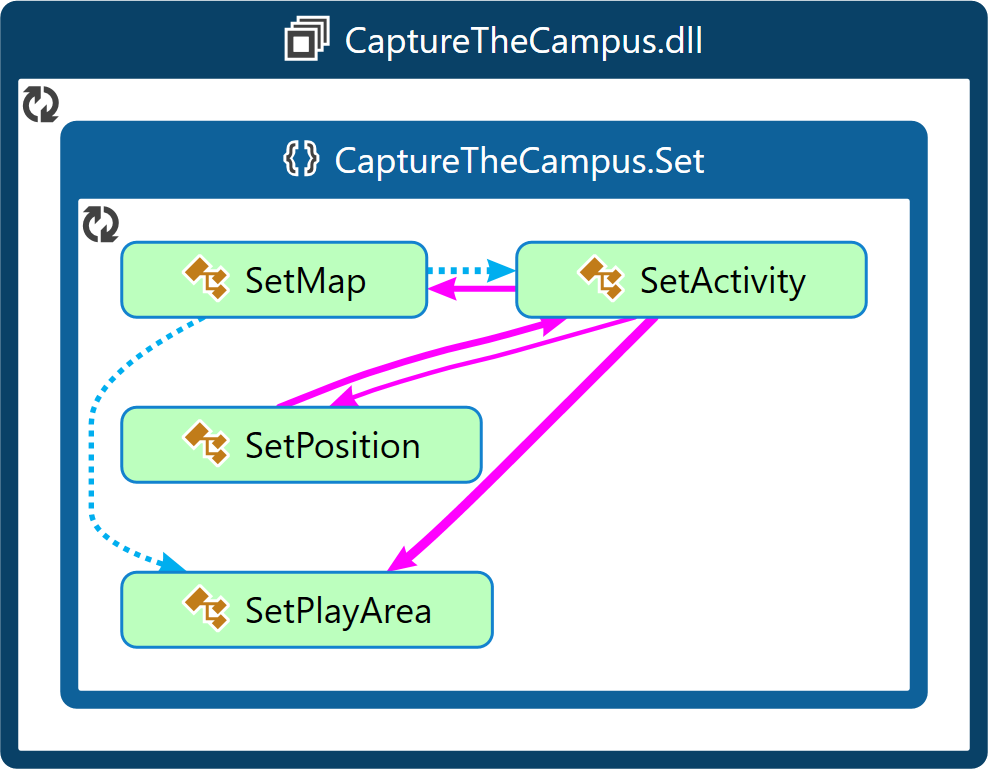


Figure : This image shows a diagram that describes the structure of the Set subsystem by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016).

## Static Design

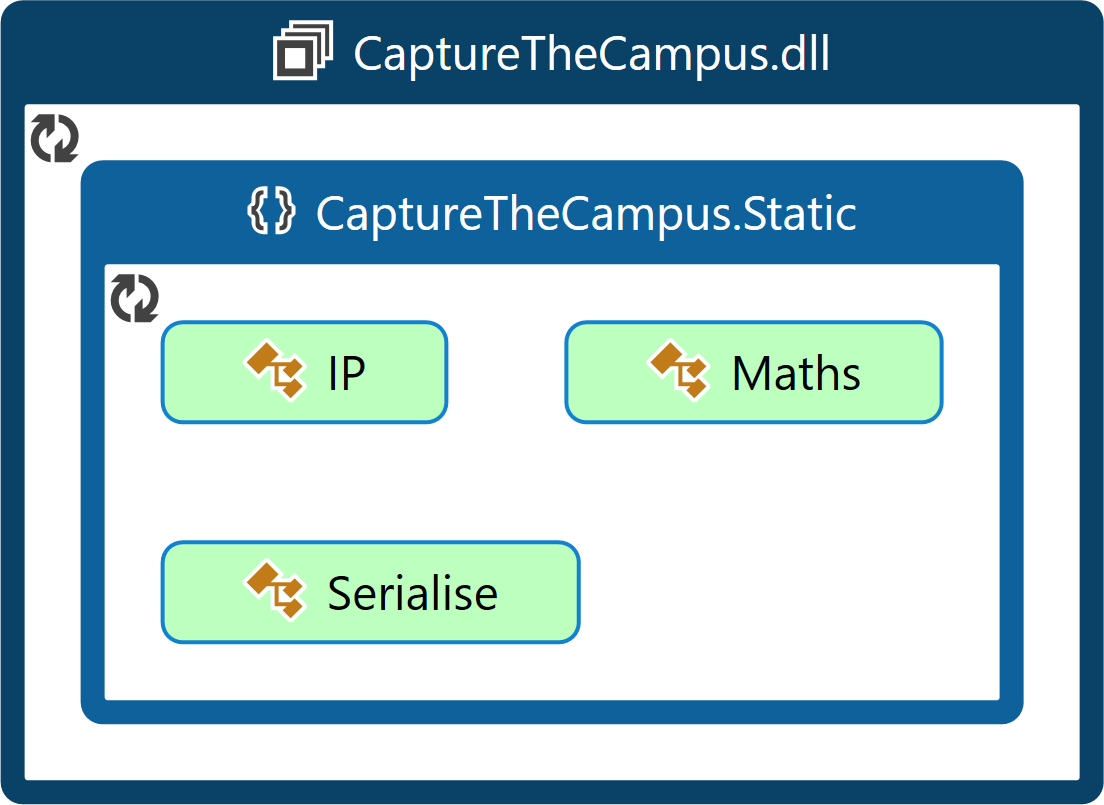


Figure : This image shows a diagram that describes the structure of the Static subsystem by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016).

## MainActivity Design



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

## User Interface Design

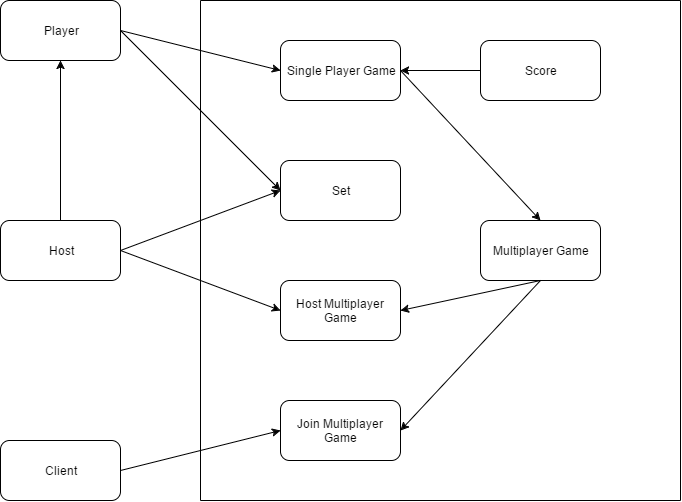


Figure : This image shows a representation of the simplest interactions a user can have with the system.

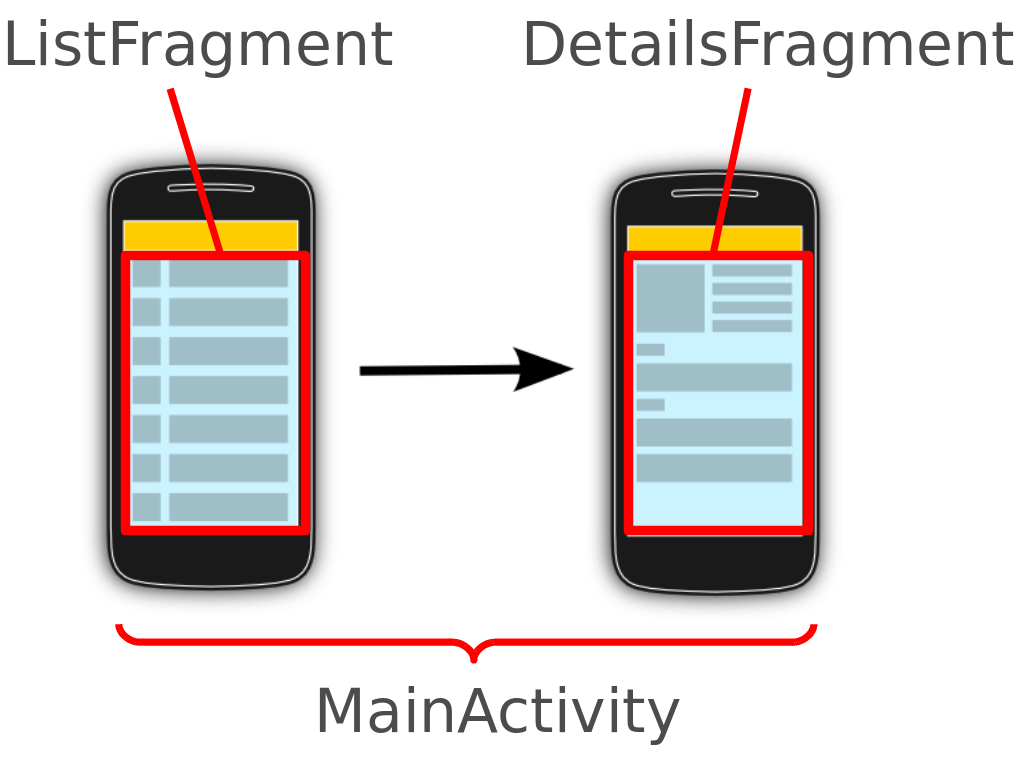


Figure : This image shows two examples of menu styles used in mobile application (Vogel, 2016)

The image above (Figure 32) 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 : This image shows examples of the user interface for the Android launcher Arrow Launcher by Microsoft (Funk, 2015)

The image above (Figure 33) 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.

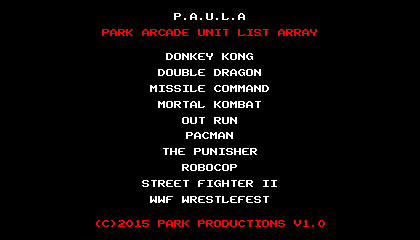


Figure : This image shows an example of a retro arcade style menu . (Park Productions, 2015)

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 34).

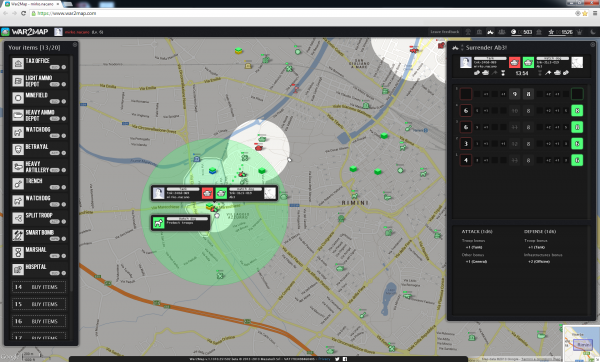


Figure : This image shows a standard game screen for the Google Maps API based game War2Map (Dempsey, 2013)

The image above (Figure 35) 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.

## System Implementation

## Processes and Methodology

## Version Control

## Polygon Clipping Algorithm

## Polygon Tessellation Algorithm

Do While (# vertices > 3)

Find an ear.

Add the ear to the triangle list.

Remove the ear's middle vertex from the polygon.

Loop

Make a triangle from the 3 remaining vertices.

## Wandering Algorithm

## Test Design

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Test** | **Expected Result** | **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 |  |

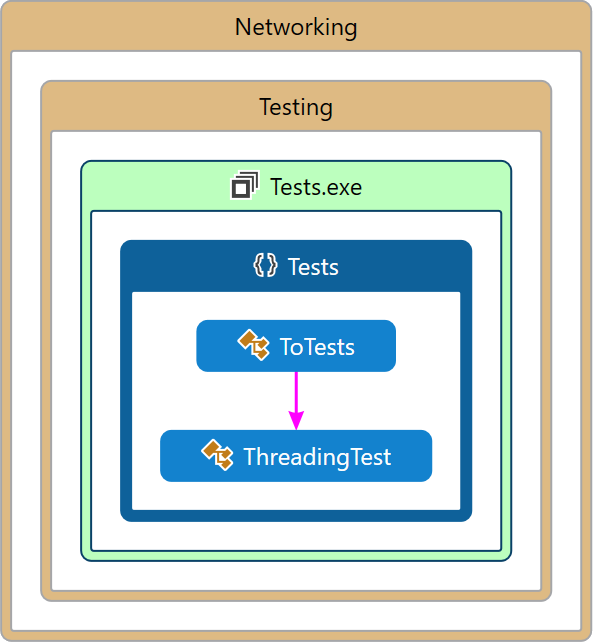


Figure : This image shows a diagram that describes the structure of the Tests subsystem by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016).

# 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 J:) 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 (‎5.1.1) and time plan (‎5.1.2) 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 M:) 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 M:) 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 the report and design of the project | 10 |
| 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 send UDP packets to the client which will 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 TCP server | Add multithreading to the TCP server so that it is capable of accepting multiple client request simultaneously | 2 |
| 7 | Add logging capabilities to the TCP server | Add logging to the TCP server so that it is capable of recovering from a fatal error.  The log can be used to debug the TCP server. | 2 |
| 8 | Work on other projects | Time set aside to work on other projects | 9 |
| 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 |
| 12 | Add player character to game screen | Add a representation of the player’s position to the game screen | 2 |
| 13 | Move player character and camera with player’s movements | Move the player character to the latitude and longitude coordinate given by the device’s GPS sensor at a reasonable interval.  The camera should also be moved to the player’s location | 2 |
| 14 | Add method to define the play area | Add a method that allows the player to define the initial play area. This should work by allowing the placement of two markers representing two opposing sides of a rectangular play area. | 2 |
| 15 | Add a method to track the player’s path and display it on the map | Add a method which creates a polyline based on the latitude and longitude coordinates of the player’s movements | 2 |
| 16 | Add a method to determine if an object is inside a polygon | Add a method that returns if a point is within the play area.  This should work for any play area. | 2 |
| 17 | Add a method to detect an intersection between a sphere and a line segment | Add a method which finds the point where two line segments intersect if they intersect | 2 |
| 18 | Add a method to clip a line segment by another line segment | Add a method which removes the part of a line segment which is beyond the point where it intersects another line segment | 2 |
| 19 | Add a method to clip a polygon by a line segment | Add a method which creates two polygons from the points where a line segment bisects a larger polygon | 3 |
| 20 | Add a method to finds the larger of two areas | Add a method which calculates the area of and then determines the larger of two polygons | 1 |
| 21 | Add a method to redefine the play area | Add a method that redefines the play area as the larger of two polygons.  The smaller polygon should be added to the map filled in with a relevant colour | 1 |
| 22 | Add a method to update the score and area | Add a method that updates the score with the area of a polygon.  The area of the play area should also be updated with this area | 1 |
| 23 | Create a game over screen | Create a game over screen for the game that will be displayed when the game ends.  The game over screen should display the players score for the previous game and return to the menu screen | 1 |
| 24 | Add enemy to game screen | Add a representation of the enemy to the game screen | 1 |
| 25 | Add a method to find a random point in a polygon | Add a method which returns a uniformly random latitude and longitude coordinate located within the play area | 2 |
| 26 | Add a method to find a random unit vector in a given range | Add a method which when given a unit vector returns a new unit vector randomly within a reasonable range of the input unit vector | 2 |
| 27 | Add a method to detect an intersection between a sphere and a line segment | Add a method that calculates the distance between a circle and a line segment.  If the distance is found to be less than the radius of the circle then the method should return that they have intersected | 2 |
| 28 | Add a method to calculate a reflected unit vector from a line segment | Add a method which given a unit vector and a line segment, calculates the normal to the line segment and then multiplies the unit vector by this normal giving a unit vector reflected from the line segment | 2 |
| 29 | Add a method to remove a path from the play area | Add a method which removes a path from the map on the game screen | 2 |
| 30 | Create a lobby for local multiplayer | Create a lobby for the local multiplayer game that will be displayed when the game is starting a local multiplayer game. | 2 |
| 31 | Add a method to find the IP address of a device | Add a method which finds the IP address of the host device | 2 |
| 32 | Add a method to connect two devices via a UDP server | Add a method that broadcasts the host’s IP address between the host and clients devices using the UDP server | 2 |
| 33 | Add a method to synchronise two devices via a TCP server | Add a method that mediates communication between the host and clients devices using the TCP server.  Information that should be synchronised includes; the initial play area, the position of the enemy and the current location of each player | 2 |
| 34 | Add a method to allow the game to update multiple players simultaneously | Add a method that is used to augment the game activity to update the information of multiple players simultaneously from the TCP server.  This will be used to implement a local multiplayer game. | 2 |
| 35 | Create a video demonstration of the project | Film a short video demonstration of the project | 2 |
| 36 | Final report | Write the final report deliverable | 3 |

## 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 library and test wrapper |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 | Create TCP server and test wrapper |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5 | Create UDP server and test wrapper |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 6 | Add multithreading capabilities to TCP server |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 7 | Add logging capabilities to the TCP server |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 8 | Work on other projects |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 9 | Interim report |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | D |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 10 | Create main menu for game |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 11 | Add map to game activity |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 12 | Add player character to game screen |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 13 | Move player character and camera with player’s movements |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 14 | Add method to define the play area |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 15 | Add a method to track the player’s path and display it on the map |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 16 | Add a method to determine if an object is inside a polygon |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 17 | Add a method to detect an intersection between a sphere and a line segment |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 18 | Add a method to clip a line segment by another line segment |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 19 | Add a method to clip a polygon by a line segment |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 20 | Add a method to finds the larger of two areas |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 21 | Add a method to redefine the play area |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 22 | Add a method to update the score and area |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 23 | Create a game over screen |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 24 | Add enemy to game screen |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 25 | Add a method to find a random point in a polygon |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 26 | Add a method to find a random unit vector in a given range |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 27 | Add a method to detect an intersection between a sphere and a line segment |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 28 | Add a method to calculate a reflected unit vector from a line segment |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 29 | Add a method to remove a path from the play area |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 30 | Create a lobby for local multiplayer |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 31 | Add a method to find the IP address of a device |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 32 | Add a method to connect two devices via a UDP server |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 33 | Add a method to synchronise two devices via a TCP server |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 34 | Add a method to allow the game to update multiple players simultaneously |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 35 | Create a video demonstration of the project |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 36 | Final report |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

## 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 several flags (or capture points) are distributed in GPS locations around the campus. The players are required to visit the location for a specified 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. Code Map Legend



Figure : This image shows the legend or key for the code map diagrams.

1. Code Map

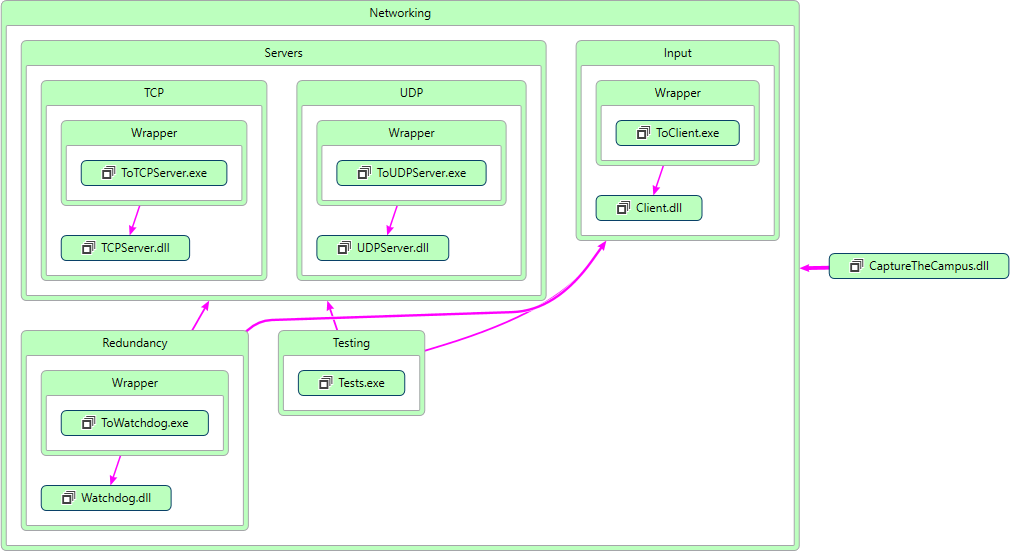


Figure : This image shows a diagram that describes the structure of the system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016).

1. Networking Code Map

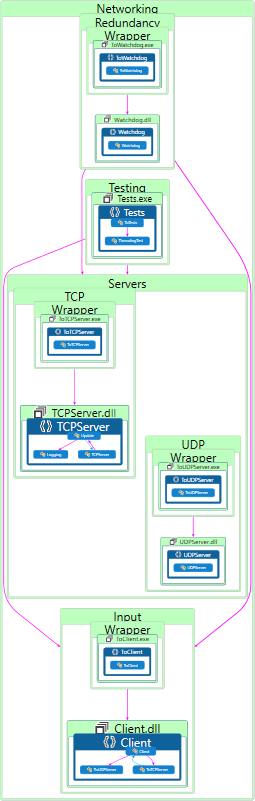


Figure : This image shows a diagram that describes the structure of the Networking subsystem by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016).

1. CaptureTheCampus Code Map

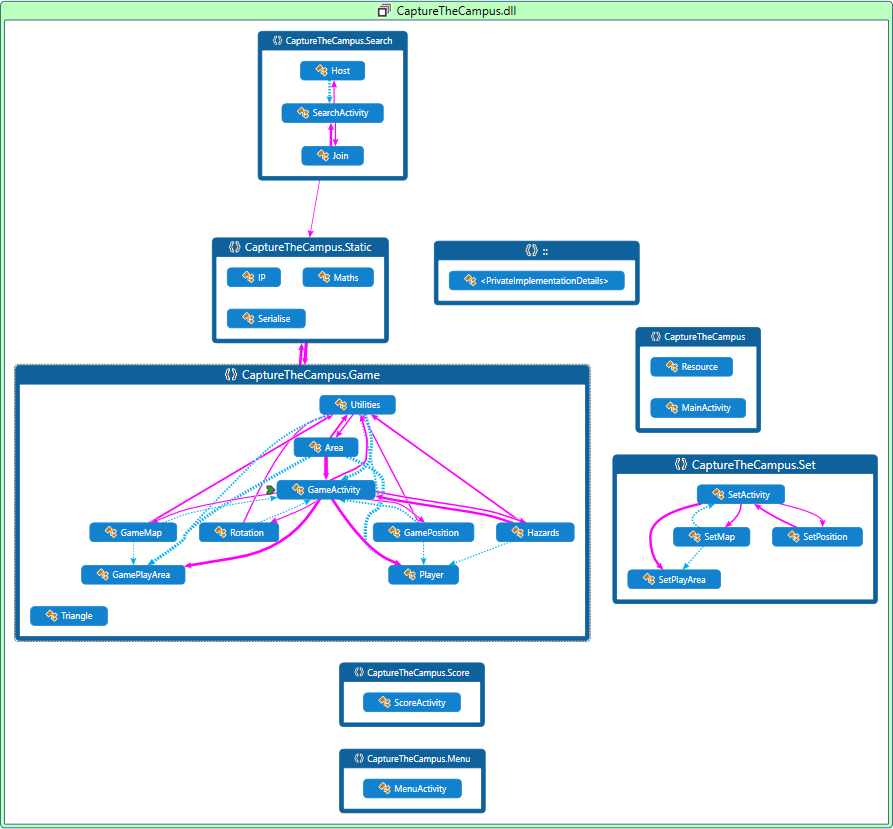


Figure : This image shows a diagram that describes the structure of the CaptureTheCampus subsystem by showing the system's classes, their attributes, operations (or methods), and the relationships among objects (Wikipedia, 2016).

1. Polygon Clipping Code





























1. Polygon Tessellation Code















1. Wandering Code













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 |

Acknowledgements

References

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

Ali, D. F. A., Mustafa, F. a. & Fdolelsid, M. O. M., n.d. *Automatic Handover Control for Distributed Load Balancing in Mobile Communication,* s.l.: Future University, Faculty of Engineering, Department of Computer Engineering.

Anna, 2016. *HOW TO BUILD A MOBILE APP WITH GEOLOCATION?.* [Online]   
Available at: https://theappsolutions.com/blog/development/develop-app-with-geolocation/  
[Accessed 20 April 2017].

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].

Garmin, 2008. *Garmin Proprietary.* [Online]   
Available at: http://www.garmin.com/support/pdf/NMEA\_0183.pdf  
[Accessed 20 April 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. *Ingress.* [Online]   
Available at: https://play.google.com/store/apps/details?id=com.nianticproject.ingress  
[Accessed 25 April 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].

Ian A.R. Hulbert, J. F., 2001. The accuracy of GPS for wildlife telemetry and habitat mapping. *Journal of Applied Ecology,* 38(4), pp. 869-878.

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

Lawson, V., 2016. *A forensic examination of Pokemon Go,* Ann Arbor: ProQuest Dissertations Publishing.

Madhavan, A., 2016. *Pokémon GO: A Lesson In Product Rebranding.* [Online]   
Available at: https://amplitude.com/blog/2016/07/12/pokemon-go-lesson-product-rebranding/  
[Accessed 20 April 2017].

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.

O’Sullivan, C., 2015. *A Tale of Two Platforms: Designing for Both Android and iOS.* [Online]   
Available at: https://webdesign.tutsplus.com/articles/a-tale-of-two-platforms-designing-for-both-android-and-ios--cms-23616  
[Accessed 20 April 2017].

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

Ordnance Survay, 2017. *GETTING OUTSIDE WITH POKEMON GO.* [Online]   
Available at: https://www.ordnancesurvey.co.uk/getoutside/guides/getting-outside-with-pokemon-go/  
[Accessed 20 April 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].

Rouse, M., 2006. *native code.* [Online]   
Available at: http://searchmicroservices.techtarget.com/definition/native-code  
[Accessed 25 April 2017].

Scholtz, J., 1992. The role of planning in learning a new programming language. *International Journal of Man-Machine Studies,* 37(2), pp. 191-214.

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].

Snader, J. C., n.d. *Effective TCP/IP Programming: 44 Tips to Improve Your Network Programs.* s.l.:s.n.

Sparrow, P., 2017. *Client Server Network : Advantages and Disadvantages.* [Online]   
Available at: http://www.ianswer4u.com/2011/05/client-server-network-advantages-and.html#axzz4fFqNt6qT  
[Accessed 25 April 2017].

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].

Stenovec, T., 2015. *Apple's iOS is beating Android where it matters.* [Online]   
Available at: http://uk.businessinsider.com/ios-users-spend-more-than-android-users-2015-12?r=US&IR=T  
[Accessed 20 April 2017].

Sunny, K., 2016. *Distribution vs Profitability – the future of hotel revenue management.* [Online]   
Available at: http://www.hospitalitynet.org/news/4077889.html  
[Accessed 20 April 2017].

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

Taing, N., 2011. *TCP UDP and RMI Performance Evaluation.* [Online]   
Available at: http://lycog.com/performance-evaluation/tcp-udp-rmi-performance-evaluation/  
[Accessed 25 April 2017].

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].

Whelan, F., 2014. *Review: Google's Ingress makes every day a glorious battle.* [Online]   
Available at: http://www.independent.ie/entertainment/games/mobile/review-googles-ingress-makes-every-day-a-glorious-battle-30459932.html  
[Accessed 20 April 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].

Zhang, Y. & Soong, B.-H., 2006. Performance of Mobile Networks with. *IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS,* 5(5), pp. 990-995.

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