

CS408 Project Document (KAIST 2017 Spring)

Due: 04/05 23:59

Team Project Name: Tom and Jerry game (Transmitter Hunting, Virtual Foxhunt game)

Project Summary: Implementation of the virtual gears and the system for the sport called Foxhunt game, where 'hounds' hunt down 'a fox' by observing the virtual signal calculated from the signal simulation using physical GPS and Map data.

Game Reference: https://en.wikipedia.org/wiki/Transmitter_hunting

GitHub: <https://github.com/SuminHan/Tom-and-Jerry-Simulation>

Team Members:

20090484 Seung-hwan Song[SH], 20091238 Jung-woo Yang[JW], 20130690 Sumin Han[SM]

1. Project Overview

A. Motivation

Our team wanted to utilize various technologies that we learned throughout our courses or experienced personally. Some of these technological experiences include the Unity 3D game engine, TCP-IP bidirectional communication, and MPI parallel programming. Also, we discussed augmented reality (AR) games such as 'Pokemon GO!', which was one of the most popular games released last year (2016). Therefore, we agreed to make an AR game for mobile platforms which utilizes the various technologies that we have learnt.

One of our members, Seung-hwan, proposed the foxhunt game, which is a sport originated from England. This sport is similar to hide-and-seek. The biggest distinction is the usage of the signal generator and the detectors. Suppose Tom hunts down Jerry. Jerry carries the signal generator while running away. Tom observes the signal shown on the detector while hunting. However, it's very inefficient to carry all these equipment every time we play. It would be much more convenient if we can encapsulate the core features of the game into a mobile device instead.

Furthermore, the simulation of the signal generation and detection process does not look easy. If there are walls or forests, the signal can be disrupted or reflected. Also, the signal gets weakened by the distance. So we need to figure out a reasonable algorithm to calculate signal variation to achieve this simulation. We also need to calculate as many signals as possible to make the simulation more realistic. Plus, we have to reduce the delay for this whole calculation process since we want to make this game run on real-time. So we thought that this project could be challenging enough to satisfy the conditions for our needs.

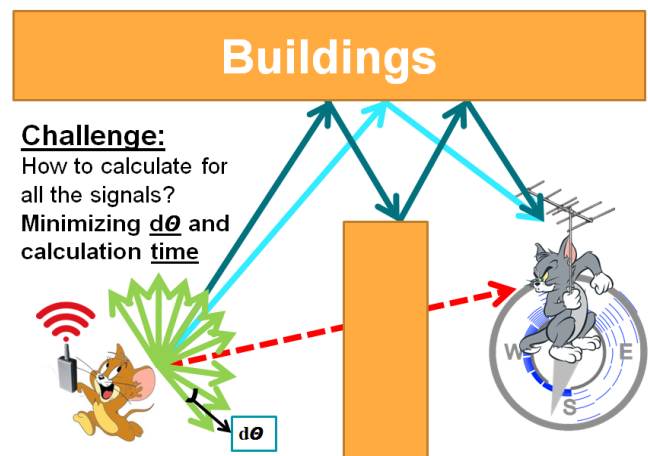


Figure 1. Calculating signal reflection.

B. Problem Statement

This game is based on a real sport, so the communication between the client and the server should be on real time, which limits the calculation time for the signal process. As the signal can be affected by surrounding buildings or forests, the calculation should be done based on existing map data. The Open Street Map data (open map database) has a representation of buildings and forests as polygons created with nodes. We will calculate the reflection using these edge data of the polygons. Our plan is to use the queue data structure to implement this algorithm. By using a queue, we can use the Map-and-Reduce technique for calculation, which enables to run multiple cores (CPUs or GPUs) to greatly reduce the time consumption.

In addition, we will operate the server to connect with the clients using bidirectional communication like TCP as TCP is more stable than UDP and more appropriate for the real-time communication. The GPS positions of the players will be sent to the server, and the server will send back the calculated signal information to the players' smartphones. In this process, there could be exceptional cases such as disconnection or unstable connection. We should take care of those possible faults by various testing methods.

Finally, we will provide graphical user interface (GUI) for each player. If the player takes the role of a fox (Jerry), we will show his state (running, caught) and the state of signal generation on the map. If the player takes the role of a hound (Tom), we will show the signal detection graph from different points of view (on the compass, on the map, in a first-person view). To start us off, we will use the Unity 3D map demo code created by a GitHub user, 'Reinterpretcat'.

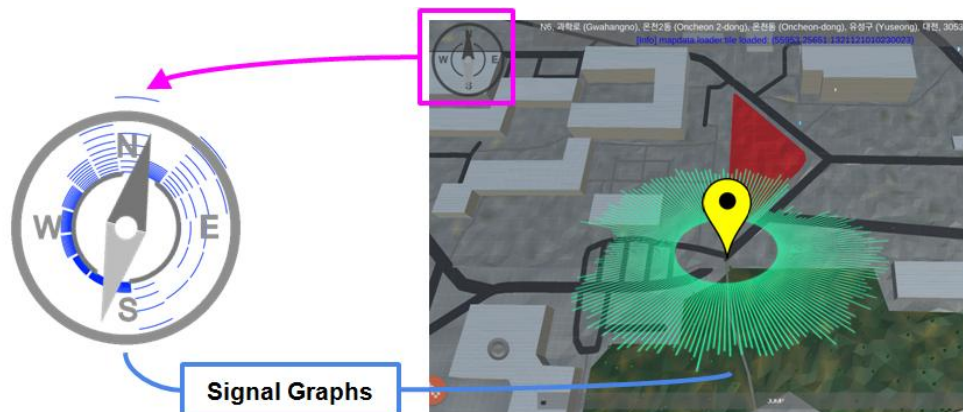


Figure 2. Possible screen for signal detection.

C. Goal Statement

These are the list of what we are going to achieve in this project.

1. Signal simulation calculation using parallel computing.
2. TCP real-time connection between the clients (Android) and the server.
3. Graphical User Interface of 3D map based on Unity 3D game engine.
4. Screen for the observer which pinpoints the positions of both fox and hound.

If we finish, we will record a video that we actually play this game outside on the campus as a final achievement.

D. Overall Plan

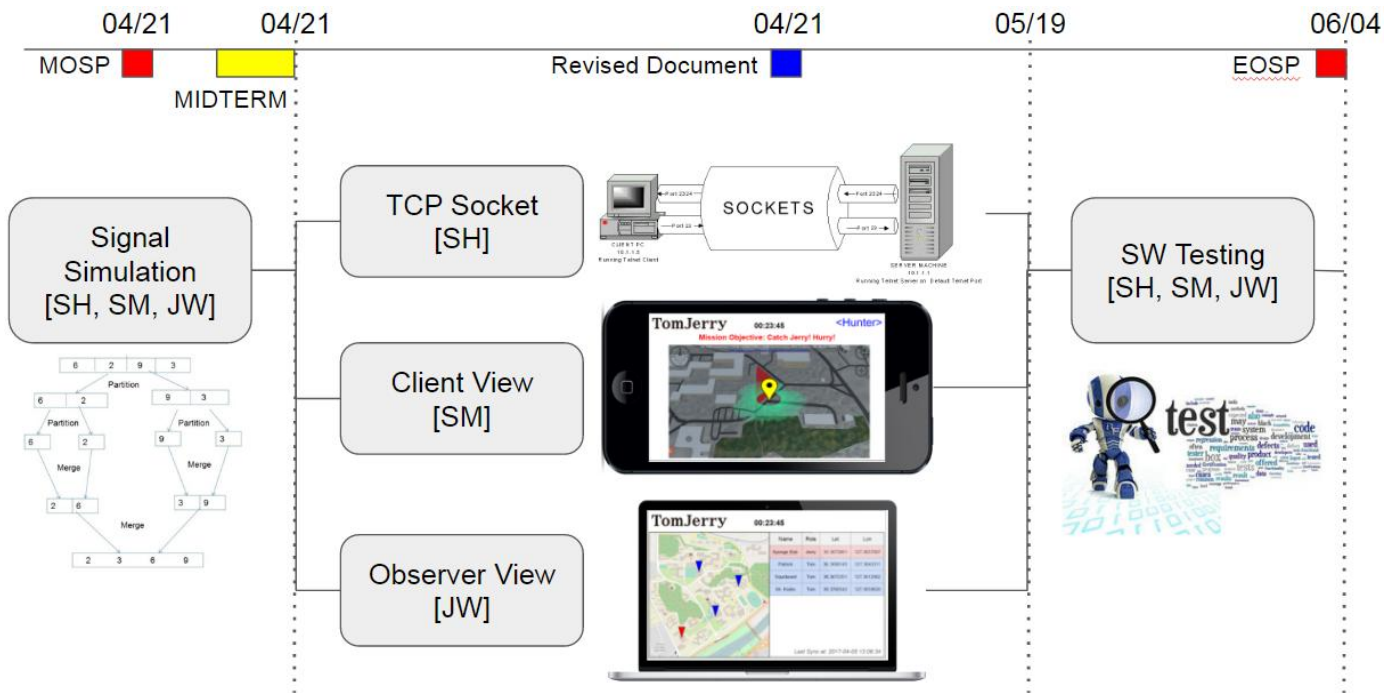


Figure 3. Overall plan with milestones.

We will apply the **incremental model** for our project.

[Basic milestones: **MOSP: 17.04.10 / Revised Documentation: 17.05.10 / EOSP: 17.06.05**]

First, We will spend time to implement the signal simulation until the midterm period. Since the signal simulation is the core function for our project, we need to check its feasibility as soon as possible. (**Milestone: 17.04.21**) Next, we will separate our job to implement TCP socket[SH], Android client view[SM], and Observer View[JW]. Each process may apply the waterfall model. (Milestone: **Milestone: 17.05.19**) Then we will procure enough time to test and actually play with it. We will go outside on a sunny day and play the actual Foxhunt with our own device. We will also take a video to show others about our game. (**Milestone: 17.06.04**)

Task Name	Assigned To	Duration	Start Date	End Date
Tom and Jerry Game		74d	17. 03. 24	17. 06. 05
MOSP	Seung-hwan Song	1d	17. 04. 10	17. 04. 10
Midterm Period		5d	17. 04. 17	17. 04. 21
Revised Documentation		1d	17. 05. 10	17. 05. 10
EOSP	Jung-woo Yang	1d	17. 06. 05	17. 06. 05
+ Signal Simulation		29d	17. 03. 24	17. 04. 21
+ TCP Real-time Communication	Seung-hwan Song	28d	17. 04. 22	17. 05. 19
+ Unity 3D Graphical User Interface	Sumin Han	28d	17. 04. 22	17. 05. 19
+ JavaScript Web View for Administrator	Jung-woo Yang	28d	17. 04. 22	17. 05. 19
+ Testing		16d	17. 05. 20	17. 06. 04

Figure 4. Overall plan with duration (start and end date).

E. Schedule and Task Management Plan

We specified actual tasks that are required to implement each function. (Also, you can see our full Gantt chart in the following URL: <http://goo.gl/81I3pE>)

a. Signal Simulation (17.03.24 ~ 17.04.21)

1. MPI and OpenCL installation
2. Learn & running example codes
3. OSM XML data crawling and refinement
4. Reflection algorithm implementation
5. Map-reduce implementation

b. TCP Real-time Communication (17.04.22 ~ 17.05.19)

1. Choose language and setup the environment
2. TCP socket implementation
3. Android real-time connection testing
4. Real-time calculated signal information communication based on Android GPS location

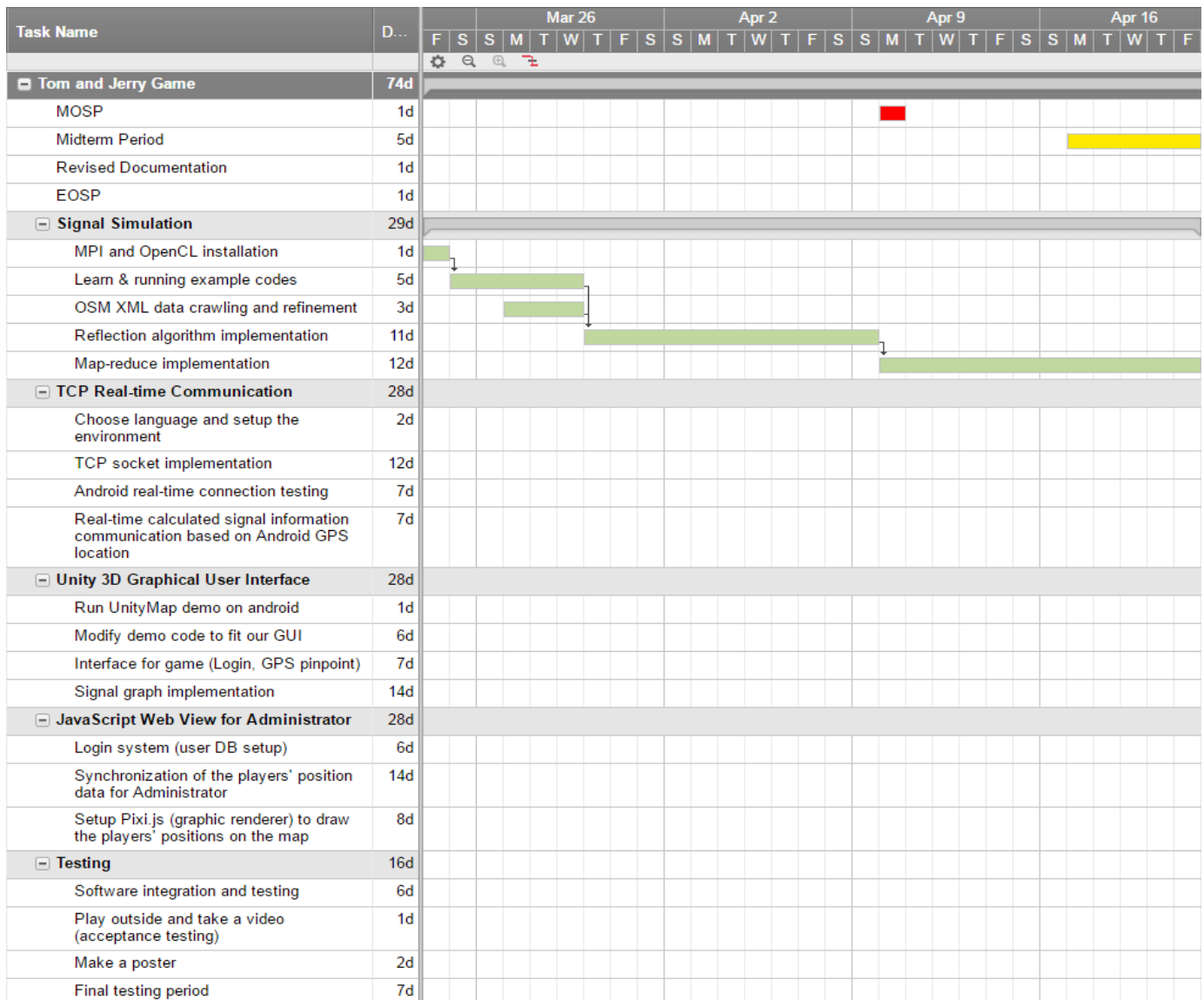


Figure 5. Gantt chart (1/2).

c. Unity 3D Graphical User Interface (17.04.22 ~ 17.05.19)

1. Run UnityMap demo on android
2. Modify demo code to fit our GUI
3. Interface for game (Login, GPS pinpoint)
4. Signal graph implementation

d. JavaScript Web View for Observer (17.04.22 ~ 17.05.19)

1. Login system (user DB setup)
2. Synchronization of the players' position data for the observer's view.
3. Setup Pixi.js (graphic renderer) to draw the players' positions on the map

e. Testing (17.05.20 ~ 17.06.04)

1. Software integration and testing
2. Play outside and take a video (acceptance testing)
3. Make a poster
4. Final testing period

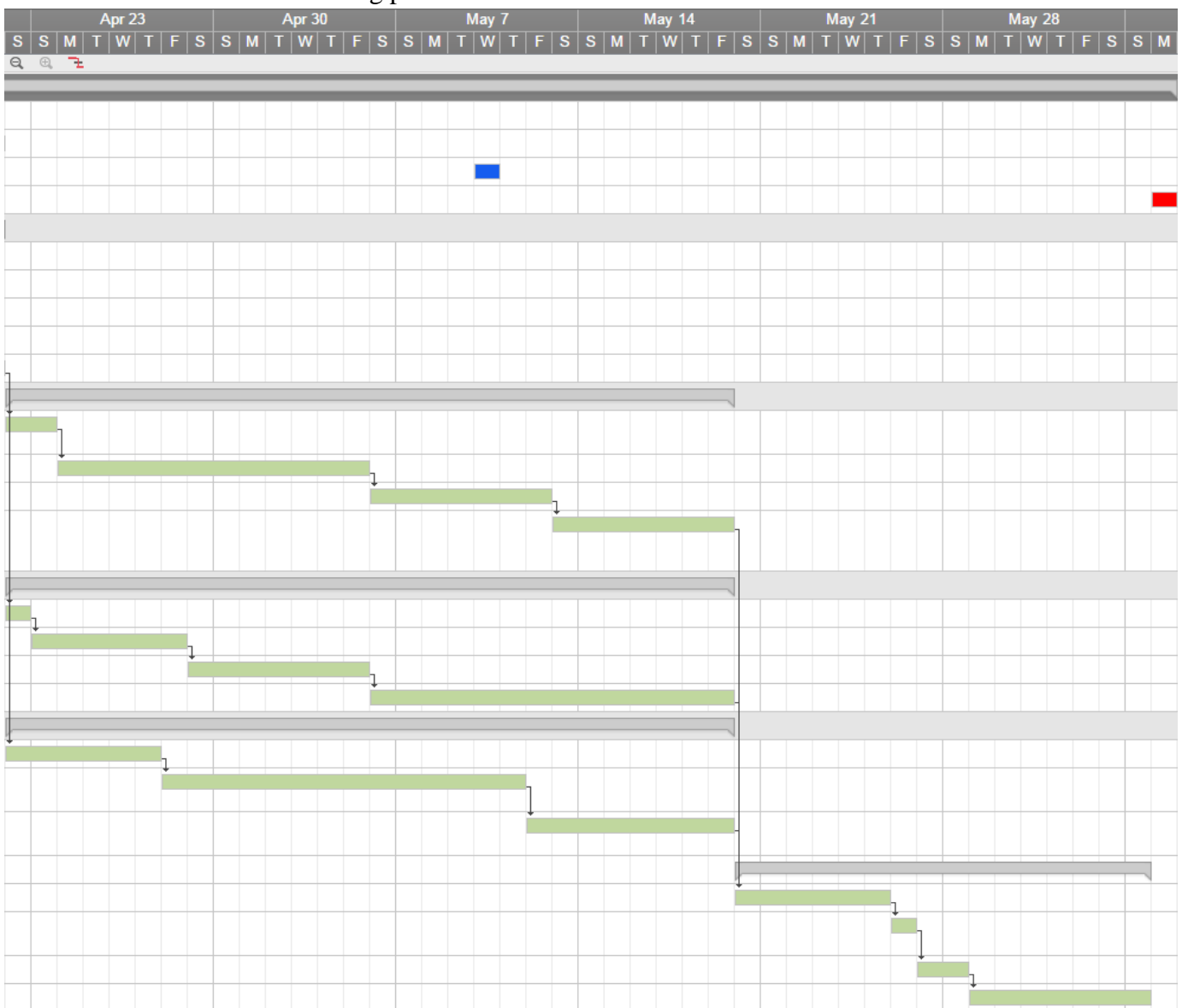


Figure 6. Gantt chart (2/2).

F. Final Deliverables

The final product of this project will obviously include this documentation, MOSP and EOSP presentations. There will also be a user client, a server program and a demonstration video of our program. The user client is run on a smartphone. It will consist of a login screen and a play interface. The server client (observer) is run on a PC, with a web-based environment. The schedule to produce our final deliverables is specified above, in section E, Schedule and Task Management Plan. Every deliverables including source code and documentation will be published well organized on our GitHub.



Figure 7. Expected screenshots for runner and observer during the game.

G. Internal Roles

First, SH, JW, SM will work with the signal simulation algorithm together. However, as SM has experience of working with a parallel algorithm, he may bring related materials to learn. We all have to learn about related mathematical theories to find the most effective way to detect signal reflection and detection. In this process, we expect to deal with matrix calculation of vectors (the signal vector and the edges of the buildings which are polygons).

After we successfully solve this, we will separate our workload equally. SH will take care of TCP socket implementation since he took Computer Networks course. SM will implement Unity 3D GUI since he had made a 3D game using Unity game engine in another class. JW will work on JavaScript web viewer for observer to manage and control the users.

This section should provide a table that shows what roles are required for the project and which member would take the role.

Name	E-mail	Role
Seung-hwan Song	sik2603@kaist.ac.kr	Coder (Java Server, TCP Socket)
Jeong-woo Yang	jwy1991@kaist.ac.kr	Coder (HTML, JS)
Su-min Han	hsm6911@kaist.ac.kr	Coder (Unity, Parallel), Project Manager

Chart 1. Internal roles.

2. Software Requirements Specification

A. Use Case Diagram

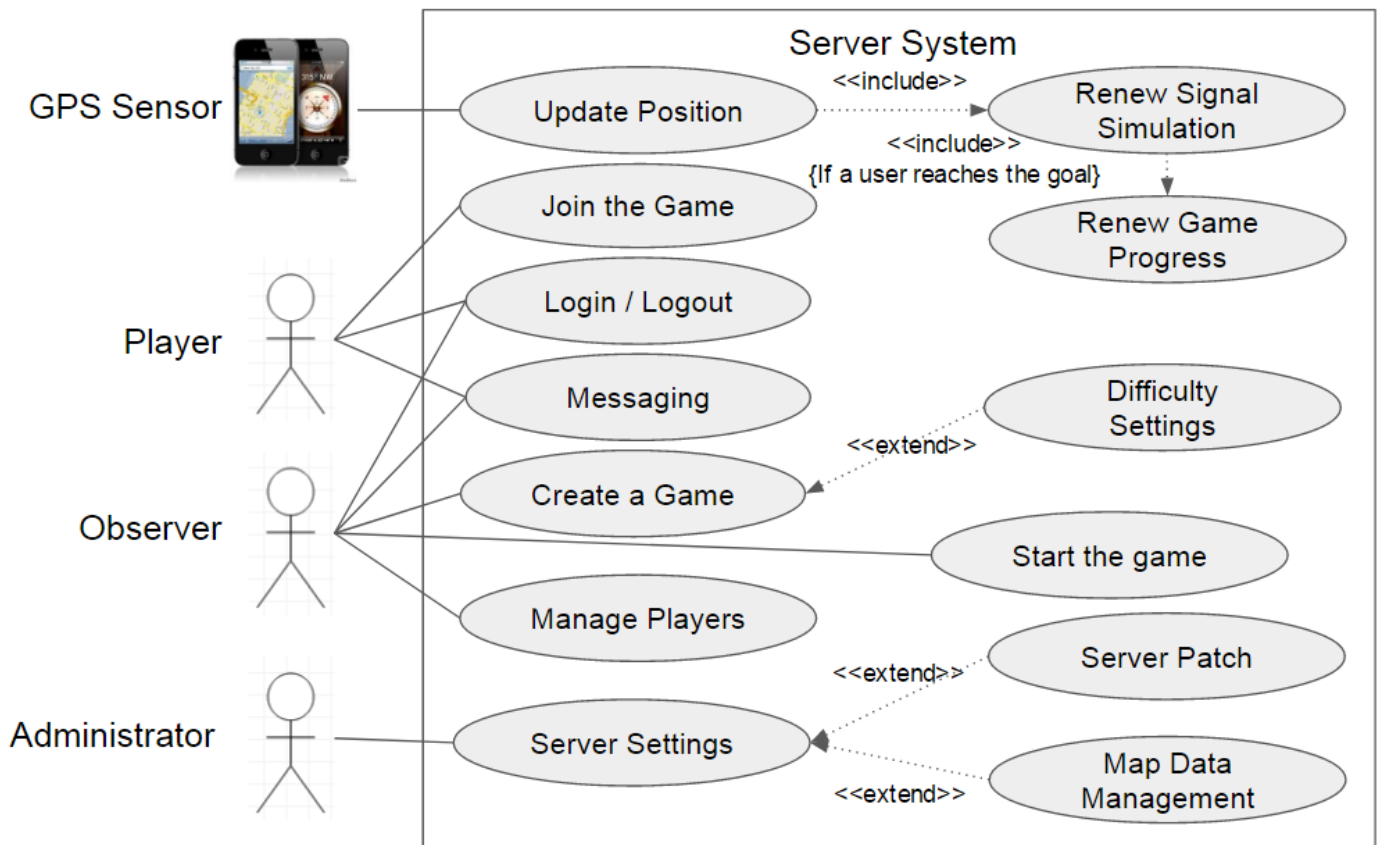


Figure 8. Use Case Diagram.

B. Use Case Description

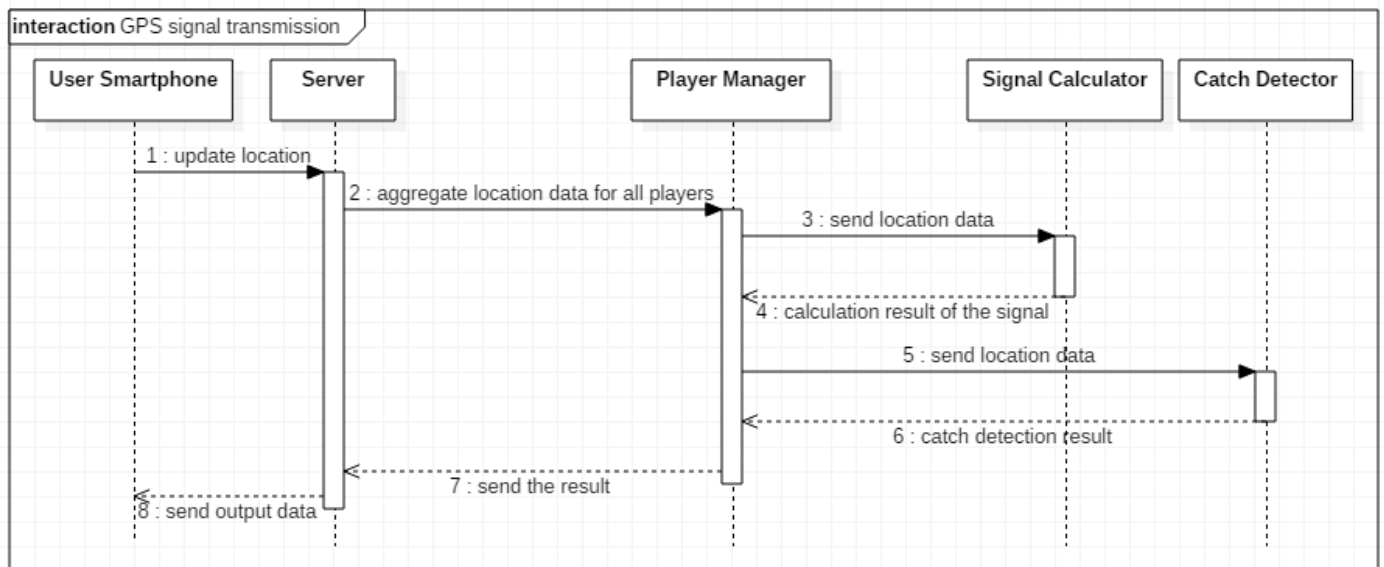


Figure 9. Sequence diagram of GPS signal transmission.

This is the sequence diagram for our key use case, which is the process of sending the information to the server and receive the calculated data. You may refer the terminologies such as “Player Manager”, “Signal Calculator”, “Catch Detector” from SW architecture figure (Figure 12).

Use case	B.1. Update Position
Primary actor	GPS sensor of each smartphone having player client
Goal in context	Send the location of a player to continue radio signal simulation.
Preconditions	Wireless internet connection and GPS sensor must be available. A game must be on process.
Trigger	The GPS sensor installed in a smartphone periodically measures the location. If the sensor works without problem, it triggers this use case. This usecase also includes <u>the catching action of the Hunter</u> .
Scenario	<ol style="list-style-type: none"> 1. A game client checks the map and renews the player's current position. 2. The client transmits the current location to the server. 3. The server confirms whether the transmission is valid 4. The server calculates the simulated strength of radio signal, available for the client that sent a location. 5. The server responds to the client. The response includes the simulated radio signals. 6. If the player reaches around the beacon, the server sends about the event to all players. The game ends if the player clears all the objectives. 7. The client displays the overall changes on the screen. 8. If Tom catches Jerry, it shows on the screen. (by calculating GPS)

Use case	B.2. Login
Refer UI	E.a.1, E.b.1
Primary actor	Game player, Observer
Goal in context	Establish an Internet connection with the server to login.
Preconditions	Wireless internet connection and GPS sensor must be available. The client application must be installed before. Player must be logged off
Trigger	A player executes the client application.
Scenario	<ol style="list-style-type: none"> 1. The game client is turned on. 2. The user enters his ID and password. 3. The game server receives the player's information and allows further operations if valid. 4. Server responds to the client. If logged in successfully, the user is ready to use other functionalities.

Use case	B.3. Logout
Primary actor	Game player, Observer
Goal in context	Finish connection with the server.
Preconditions	The client application must be turned on.
Trigger	The player selects the logout menu.
Scenario	<ol style="list-style-type: none"> 1. The client sends signal to server for logout. 2. The server rejects the player from any game he/she joined. 3. Deactivate player status on game he/she joined.

Use case	B.4. Messaging
Primary actor	Game player, Observer
Goal in context	Communicate with other players
Preconditions	Wireless internet connection must be available. The user must be logged on
Trigger	A player selects the chat menu.
Scenario	<ol style="list-style-type: none"> 1. A player enters a message, and sends it to server. 2. The server checks the receiver options for the message 3. Transmit the message to allowed players.

Use case	B.5. Join the Game
Refer UI	E.b.2
Primary actor	Player
Secondary actor	Other game players
Goal in context	Join the game
Preconditions	Wireless internet connection and GPS sensor must be available. The user must be logged on
Trigger	Player clicks to join the game.
Scenario	<ol style="list-style-type: none"> 1. The player chooses the game he wants to join. 2. If there's a password, he should type it correctly to enter(or there will be error message shown). 3. The player enters the room and wait until the observer starts the

	game.
--	-------

Use case	B.6. Create a Game
Refer UI	E.a.2
Primary actor	Observer
Goal in context	Prepare a game to play with other players.
Preconditions	Wireless internet connection and GPS sensor must be available. The user must be logged on
Trigger	A player selects the 'Create a game' menu.
Scenario	<ol style="list-style-type: none"> 1. The observer creates a group for a new game. 1.a. He can set a password if he want. 2. Selects the location where game is held, and options. Those information is sent to the server. 3. The observer waits until the server matches suitable players. 4. The observer starts the game if all players are ready. Send the initialization signal to the host. 5. The server decides the goal(location of beacons), and starts the simulation. 6. The server starts to send game status to the players.

Use case	B.7. Starts the Game
Refer UI	E.b.3
Primary actor	Observer
Goal in context	Starts the game by clicking on the button.
Preconditions	All players are ready.
Trigger	The observer starts the button.
Scenario	1. Observer clicks on the start button when evereybody is ready.

Use case	B.8. Server Settings
Primary actor	Administrator
Goal in context	Manage the overall server settings, modify map data.
Preconditions	The user must be logged on.

Trigger	Observer executes the management program
Scenario	1. Administrator updates the system or source code.

Use case	B.9. Manage the players
Refer UI	E.b.4
Primary actor	Observer
Goal in context	Modify the player status. (May be useful for debugging)
Preconditions	The user must be logged on.
Trigger	Observer executes the management program
Scenario	1. Watch for the player's action and manage them.

C. Functional Requirements

As our product is a game that involves simulation of radio signals, our functional requirements are as follows, in the order of importance:

1. Radio signal simulation

Our software(game) revolves around trying to catch a simulated radio signal. Therefore, it is the essential functional requirement of our product. As we planned to operate this program with our campus as the setting, we would have to consider geographical obstacles such as trees and buildings. This will be done through an algorithm involving mathematical and physical calculations. Furthermore, to prevent players from getting used to the game and tiring out early, we also plan to add a randomization algorithm as well so that we can provide a different gaming experience for repetitive plays.

2. Server-client TCP communication

Our game involves using mobile devices (smartphones) as the signal detectors. Therefore, we would need consistent intercommunication between the game server (which has the information about the source) and the users' devices which are used to track down the signal source.

3. Graphical User Interface (GUI)

Our game needs to show the current user's location because we are making a game based on the real physical space. Since Tom(Hunter) has to track Jerry(Runner)'s signal, Tom's smartphone should show the signal graph on the map. A simple and effective GUI is required for better user experience.

4. Login system

The original signal hunt game is a social game, and we would like to facilitate this aspect of the original game through registration of users. With dedicated IDs, users would be able

to find their friends or meet new people to play together. Also, this is done to prevent confusion caused by users using multiple devices. This also makes the game more enjoyable, as registering the users will prevent them from cheating, as cheaters could get banned or have other measures taken against them.

5. Game Room System

In order to start the game, the observer should create a game room. Then each player will join the room using their smartphone. There can be game password if the observer wants to make a private game. The observer can also ban player before start from game room. Refer to the section E: User Interface Requirements.

D. Software Quality Attributes

The quality attributes of our program are as follows, in order of importance:

1. Reactivity

Our software has to keep on updating the users about the location of the signal source. As the users move around to find the source, the software has to consistently update the users about the location of the source to make sure that the game proceeds. The standard for reactivity that we have set is a response time of 1~3 seconds per update.

2. Precision

This application needs to provide the users the accurate location of the target while still maintaining the fun. If we keep the error tolerance too high, then there will be no fun in tracking down the target, as the hunt will become too easy to accomplish. If we keep the tolerance too low, then it would be too stressful for the players as they would have to be unfeasible for us as the error rate of the GPS is around 5 meters. We plan to keep our error radius as 5 meters, as it is the precision of the GPS signals, and it is a reasonable range that the players can “see” the target.

3. Optimization

Our program is designed to run on a mobile environment. So, battery drain is an important issue for our app. The original game could go for more than 2 hours. Our players could be using their smartphones for other uses such as messaging or phone calls during the game session. Therefore, we must ensure that the program does not consume too much battery. The standard for optimization is around 10% battery drain per hour.

4. Portability

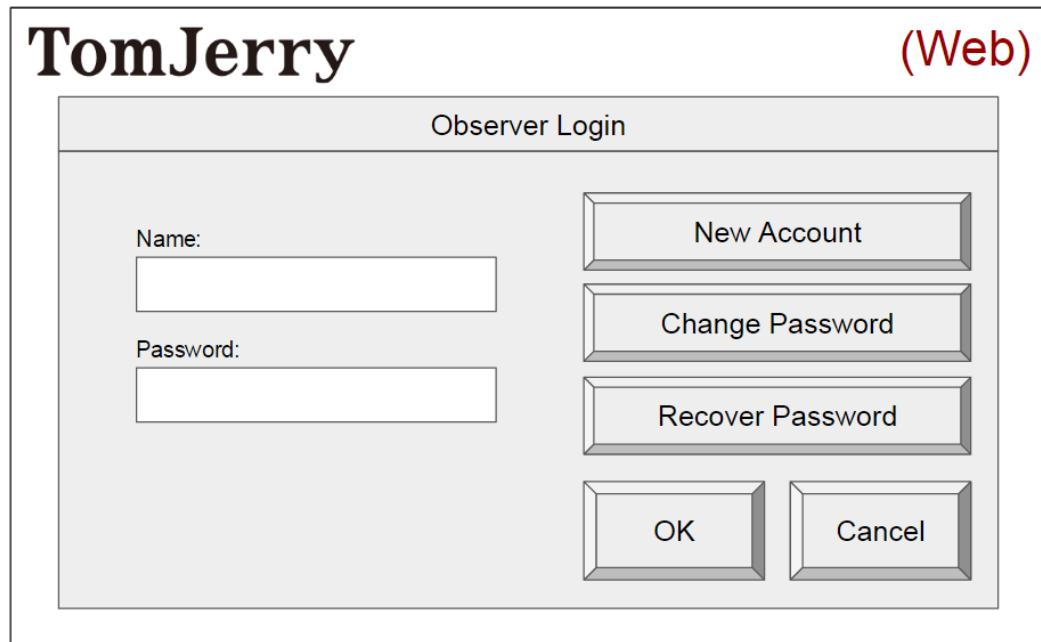
As stated above, the program runs on a mobile environment. Around three-quarters of smartphone users in South Korea are android users. As this program is set for domestic purposes, we plan to make it run on an Android operating system for mobile devices.

E. User Interface Requirements (UI/UX)

We need some interface for user to easily create game and join the game. We benchmarked Starcraft I Battle.net user interface for our UI, because they shared some similar requirements.

a. Web Observer View

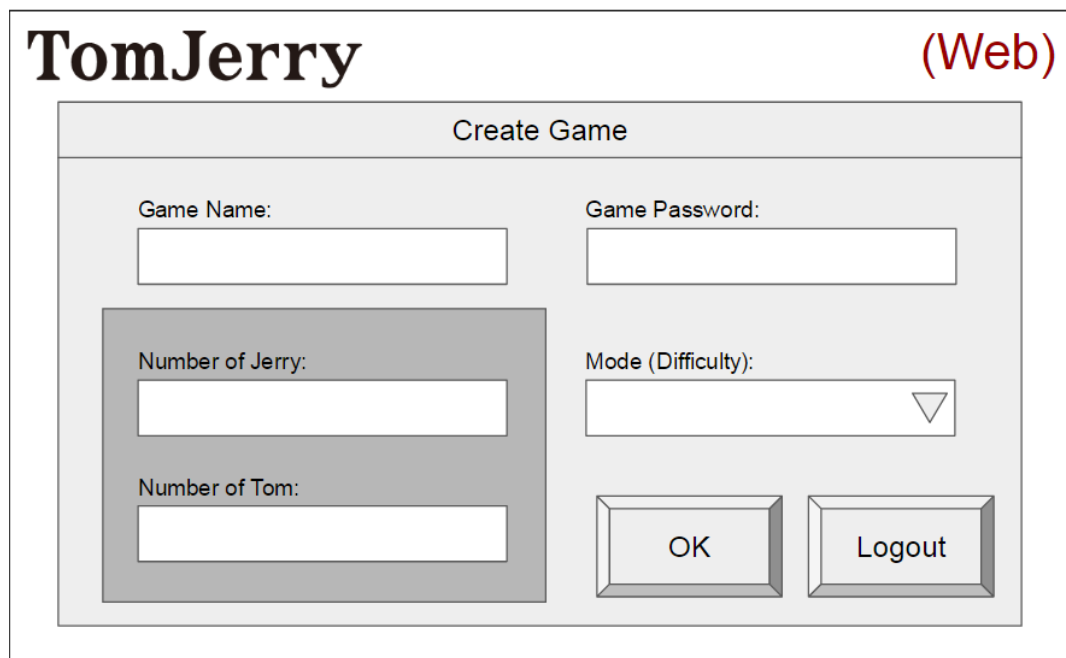
1) Login



The image shows a web browser window titled "TomJerry (Web)". Inside the window is a form titled "Observer Login". The form has two input fields on the left: "Name:" and "Password:". To the right of these fields are four buttons stacked vertically: "New Account", "Change Password", "Recover Password", and "OK" and "Cancel" at the bottom.

➤ This is a login page. You can create new account or change and recover password. Each button will link to typical process for each purpose.

2) Create the Game



The image shows a web browser window titled "TomJerry (Web)". Inside the window is a form titled "Create Game". The form has four input fields: "Game Name:", "Game Password:", "Number of Jerry:", and "Number of Tom:". There is also a "Mode (Difficulty):" dropdown menu. At the bottom right are two buttons: "OK" and "Logout".

➤ After log in, observer can directly create game, setting game name, game password, number of Jerry and Tom (foxes and hounds).

3) The Game Room

TomJerry
(Web)

Jerry

Tom

- Open
- Close
- Ban this player**

Game Name:
Welcome KAIST!

Password:
1234

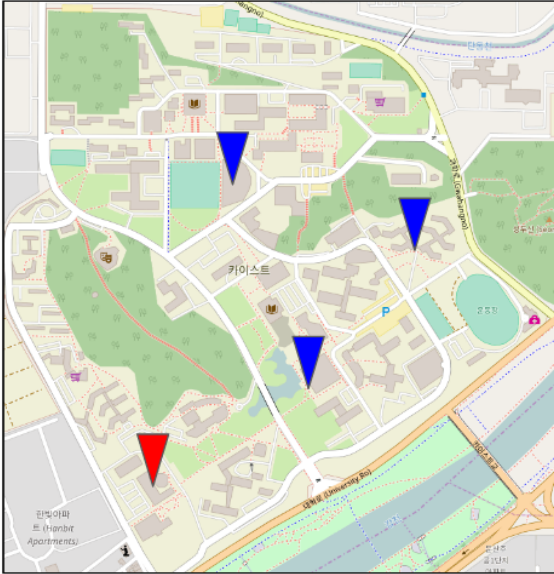
Observer (Referee)

Starting in 5 seconds...

- You can wait for other player. You can close or open or ban the player by click and drag down the triangle button on the right side of the nick name.

4) During the Game

TomJerry
00:23:45
(Web)



Name	Role	Lat	Lon
Sponge Bob	Jerry	36.3673981	127.3637097
Patrick	Tom	36.3696145	127.3643311
Squidward	Tom	36.3672351	127.3612982
Mr. Krabs	Tom	36.3745542	127.3639620

Last Sync at: 2017-04-05 13:06:34

- After game starts, you can see the position of each player on the map.

b. Client Smartphone View (Player)

1) Login

The screenshot shows the 'TomJerry' app interface for a smartphone. The title 'TomJerry' is in a large, bold, black serif font on the left, and '(Smartphone)' is in a blue sans-serif font on the right. Below the title is a light gray rectangular box with the title 'Player Login' centered at the top. Inside this box, on the left, are two text input fields: the first is preceded by the label 'Name:' and the second by 'Password:'. To the right of these fields are four buttons stacked vertically: 'New Account', 'Change Password', 'Recover Password', and 'OK' (which is positioned to the left of a 'Cancel' button). All buttons have a 3D effect with a gray border and a light gray fill.

➤ You can log in through smartphone. If you click “new account”, “change password”, “recover password” button, it will link to the same web page shared with web view buttons.

2) Join the Game

The screenshot shows the 'TomJerry' app interface for a smartphone. The title 'TomJerry' is in a large, bold, black serif font on the left, and '(Smartphone)' is in a blue sans-serif font on the right. Below the title is a light gray rectangular box with the title 'Join Game' centered at the top. Inside this box, on the left, is a list box containing six items: 'Room...1', 'Room...2', 'Welcome KAIST!', 'Room...3', 'Room...4', and 'Room...5'. To the right of the list box are three text input fields: the first is preceded by the label 'Game Name:' and contains the text 'Welcome KAIST!'; the second is preceded by the label 'Game Password:' and contains the text '1234'; and the third is preceded by the label 'Creator:' and contains the text 'CS408_Team16'. Below these fields are two buttons: 'OK' and 'Cancel'. All buttons have a 3D effect with a gray border and a light gray fill.

➤ You can join the game. You can also type the game password. If you fail, it will show an error message.

3) The Game Room

TomJerry

(Smartphone)

Jerry

SpongeBob

Tom

Patrick

Squidward

Mr. Krabs

Unknown Player

Game Name:
Welcome KAIST!

Password:
1234

Observer (Referee)



CS408_Team16

Starting in 5 seconds...

Cancel

- After you enter the game room, you can show other players waiting. However, you can't ban other players or slot state. Also the player can't start the game (there's no START button unlike Web view).

4) Players' view

<div style="border: 1px solid #ccc; padding: 5px;"> <div style="display: flex; justify-content: space-between;"> TomJerry 00:23:45 <Runner> </div> <p style="color: red; font-weight: bold;">Mission Objective: Don't be caught until the time count.</p>  </div> <p>Jerry's view has a map with pinpoint on it.</p>	<div style="border: 1px solid #ccc; padding: 5px;"> <div style="display: flex; justify-content: space-between;"> TomJerry 00:23:45 <Hunter> </div> <p style="color: red; font-weight: bold;">Mission Objective: Catch Jerry! Hurry!</p>  </div> <p>Tom's view is similar to Jerry's, but with signal visualization</p>
---	--

- The screens of Jerry and Tom are different as their roles.

F. Software Interface Requirements

1. Server
 - a. OS: Windows 7
 - b. Server: Java (TCP socket) → Eclipse IDE
 - c. Database: SQLite → JDBC (Java DataBase Connectivity) driver
 - d. CPU: 4 cores (Intel(R) Core(TM) i5-4590 CPU @ 3.30GHz)
→ Open MPI: Version 2.1.0 for parallel computing (on multi-cores)
 - e. GP-GPU: NVIDIA GeForce GTX 650 Ti
→ CUDA 8.0v (OpenCL) for parallel computing (on GPU)
2. Web Observer's GUI
 - a. HTML 5 : supports canvas object.
 - b. Pixi.js: Version 4.4.3
 - c. We need those browsers: since they support HTML5 required for Pixi.js.
 - i. Internet Explorer 11+
 - ii. FireFox 15+
 - iii. Chrome 11+
 - iv. Safari 5.1+
 - v. Opera 19+
3. Client's GUI
 - a. Development Tool: Unity 3D 5.5.2f1 (Unity Engine), Android Studio 2.3 (Android)
 - b. Android 6.0 Marshmallow + (supports Unity 3D)

G. Design and Implementation Constraints

1. Hardware Limitations.

- CPU: **4 cores** (Intel(R) Core(TM) i5-4590 CPU @ 3.30GHz)
- GP-GPU: NVIDIA GeForce GTX 650 Ti

If we choose the number of threads per block to 672, then each multi-processor can run 2 blocks, so 4 cores can run 8 blocks. $672 * 4 * 2 = 5376$ GPU threads can run simultaneously.

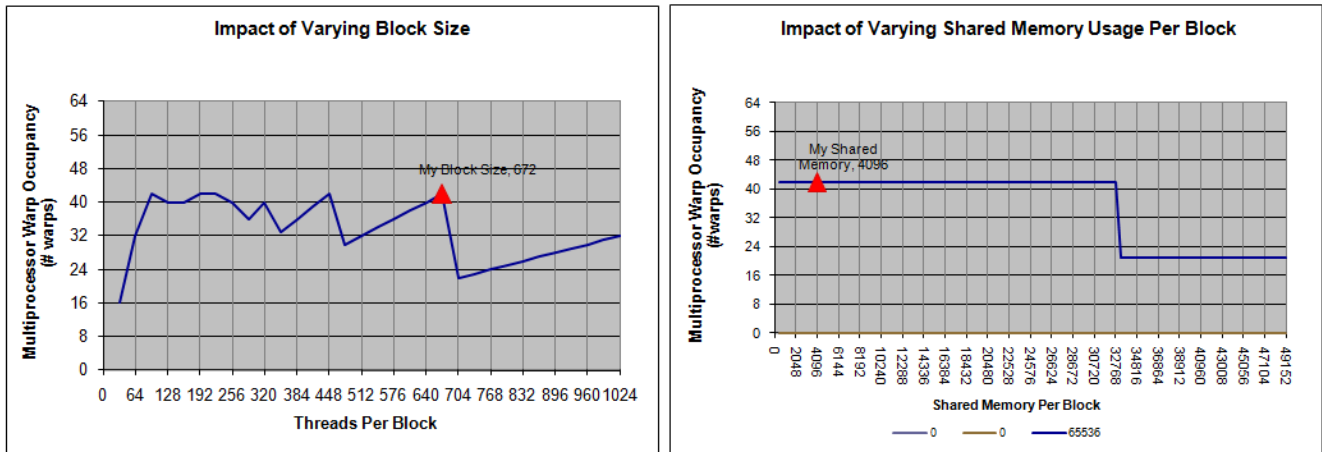


Figure 10. Test result of GPU performance on the PC. (672 threads and 4096 bytes of memory per block)

2. Application Memory and Storage Usage.

The property of the current Demo UnityMap application:

- Memory: 4.3 MB ← This is due to the dynamically downloaded map data while testing.
- Storage: 70 MB
- Since our application does not download map data dynamically, rather it will use pre-installed map data to draw. So we only need to consider the GPS and the Signal Data transmission. These data will be carried using JSON format to minimize the size.
- We have to renew signal data for each second, we may need to free the allocated memory using the Garbage Collector on Android (Java).
- Suppose we download 360 signal data from each 1 degree angle to create a circular signal graph. each signal is double float (8byte), so $[360 * 8 \text{ byte} + \text{JSON formatter} = 3 \text{ KB} / \text{second}]$. This is equivalent to 10.5 MB per hour. So our plan is:
 - MUST: No more than 20 MB
 - PLAN: No more than 10 MB
 - WISH: No more than 5 MB
- Also we don't think we will add much feature to current Demo. Our target Storage:
 - MUST: No more than 100 MB
 - PLAN: No more than 80 MB
 - WISH: No more than 70 MBwhere MB defined: MegaByte.

3. Software Architecture Design

A. Context Diagram

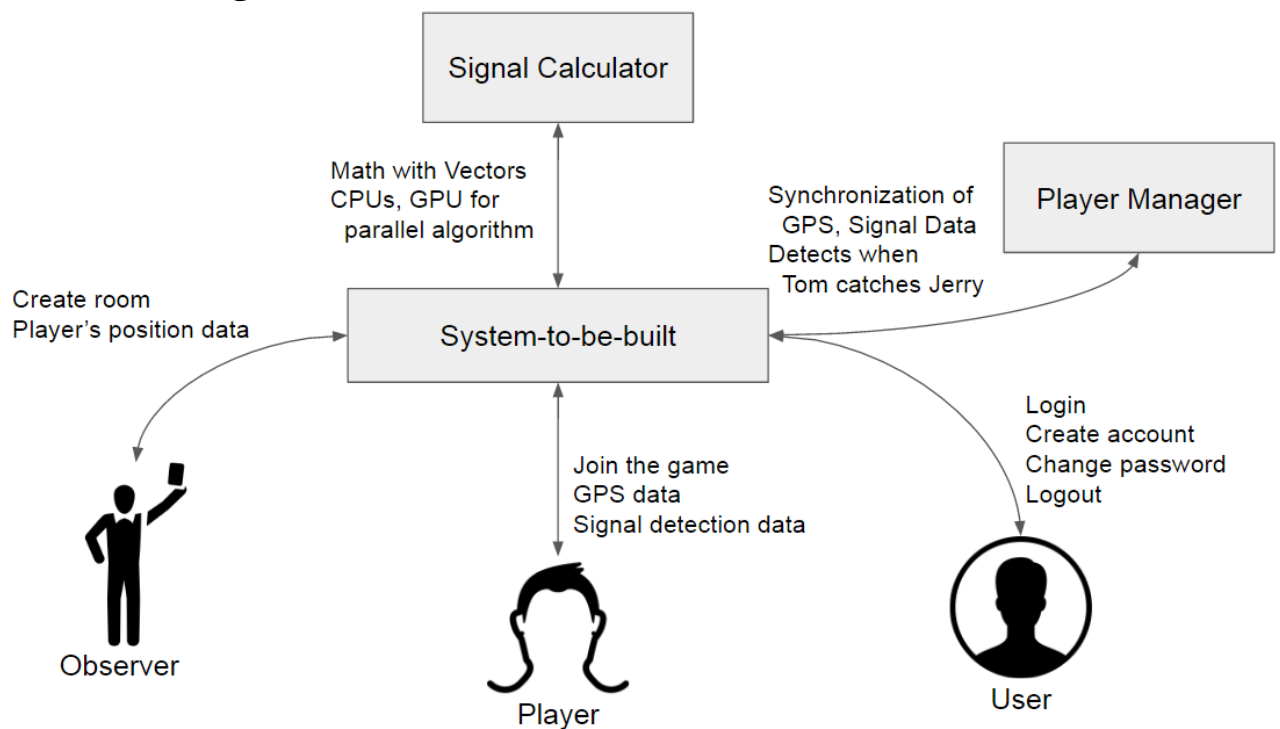


Figure 11. Context Diagram.

B. Architecture Diagram

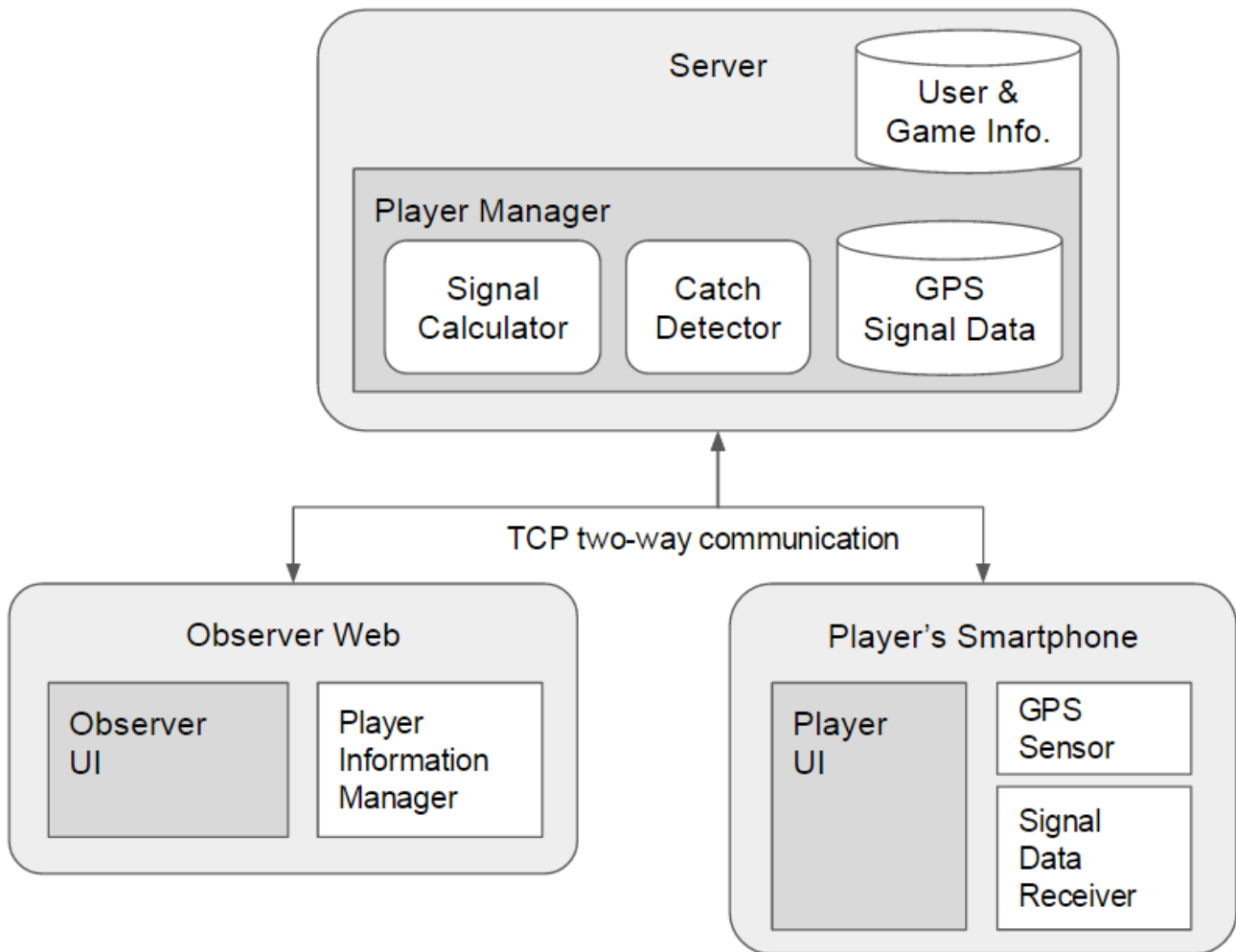


Figure 12. Architecture Diagram.

1) Server

- a) **Signal Calculator:** apply the parallel algorithm using MPI or OpenCL to utilize multiple cores(CPUs) or GP-GPU to shorten the calculation time.
- b) **Player Information Manager:** controls the players' position data and signal data that will be sent to each player. Also this unit detects catch condition for the victory.
- c) **User & Game Information:** this database contains login and game information (e.g. player's nickname, player's role, game channel, game room number, etc.)

2) Observer Web

- a) **Observer UI:** use graphic renderer Pixi.js(<http://www.pixijs.com/>), a javascript graphic renderer that can run without extra installation on the webpage.
- b) **Player Information Manager:** manages players' information received by TCP socket.

3) Client Smartphone

- a) **Player UI:** use Unity 3D game engine to display map and 3D signal graph.
- b) **GPS Sensor:** sends GPS data to the server through TCP socket.
- c) **Signal Data Receiver:** connects through TCP socket and receive signal Data.

*Each component is connected through TCP two-way communication socket.