Ghost Grab



Dylan Dover, Selina Chen, Jason Mayer, Lawrence Waller

ENGM 3700.02

04/21/2016

# Executive Summary

Our company will design an innovative Android game called GhostGrab that will utilize GPS coordinates to create an immersive environment. We will utilize a number of tools and frameworks to accomplish this goal. Among them are Unity on the client-side and node.js on the server-side.

GhostGrab will be a location-based game that will allow users to use their phones to view and capture “ghosts” across a chosen location radius and release them or set them on others for points. Users will have to play mini-games or solve riddles to capture ghosts and there may be random team competitions, allowing users to collect bonus points. There will be a leaderboard, which will update in real-time to show who is leading in points and allow users to track their rank and the ranks of their friends. Created in Unity, it will be easily portable to a variety of platforms, though for a minimum viable product, we will be releasing an Android-only version.

To keep the game interesting over time, updates will be available in the form of new ghosts and new mini-games. This model makes it possible to easily update the game with minimal time investment, meaning that our company will have time to pursue other projects simultaneously.

Contents

[Executive Summary 2](#_Toc448850915)

[Scheduling 4](#_Toc448850916)

[Responsibility Matrix 5](#_Toc448850917)

[Risk Management 6](#_Toc448850918)

[Problems & Solutions 7](#_Toc448850919)

[Quality Assurance 8](#_Toc448850920)

[Future Plans 9](#_Toc448850921)

# The Project

## Background and Rationale

GPS has become widely embedded in mobile devices, and games that interact with users in the real world have become more and more popular. With the recent announcement of the VR (Virtual Reality) game, Pokémon GO™, and already existing augmented reality mobile apps such as Ingress, there is a lot of widespread interest in games that users can bring into the real world.

## About the App/Functionality?

This app is available to Android phones. A basic walkthrough of how the app is used is demonstrated through the pictures below:

First login by creating a username and password? [INSERT PIC]

Next

Happy ghost hunting!

Some other features include a leaderboard, where you can see who is leading in points.

[INSERT PIC]

## The Code

The code for this app as well as other documentation is publicly available and can be found at <https://github.com/wallerl2/ghostgrab>. A basic layout of the way the code is set up is demonstrated in the UML below:

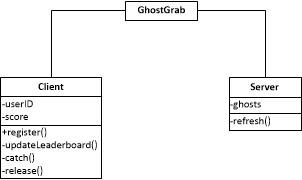


Figure : UML Diagram for App

The client, which is the code written in Unity, holds the graphics for the game as well as the most of the game logistics. The server, which is being hosted on an AWS instance holds information such as ghost types, location of all the ghosts in the game, and the scores of the players. The client requests information from the server whenever it needs to (ex: when a user wishes to see the leaderboard, it sends a request) and it also posts information to server as well (ex: updateLeaderboard() function is called when a user’s score changes, and the updated score is posted to the server).

# Scheduling

This project took place over a period of about 3 weeks, starting on Tuesday, March 29th, 2016 and ending on April 21, 2016. An overview of the tasks performed for the project are listed below in Figure 2.(MIGHT NEED TO CHANGE #)



Figure : Gantt Chart Documenting the Project Schedule

The project was split into four phases: Initial Design, Bare Bones, Art and Advertising, and Final Product. Though called ‘phases’, these periods do not necessarily go in order or depend on each other’s completion. For example, the tasks in Phase III: Art and Advertising necessarily succeed the Phase II subtask ‘Set up running Project in Unity’, but do not need to succeed the other subtasks in Phase II. The Phase III subtasks of designing ghost types and making the app pretty can easily be performed concurrently with the Phase II subtasks of setting up the server and client.

Our project has a single milestone on 4/20/16, which is the printing and binding of the final report, which is done on 4/21/16, the day before the report will be handed in.

For a better look at the entire project schedule and which tasks must precede others, one can view the Network Diagram form of the schedule below.



Taking a closer look at the beginning of the schedule we see that the Initial Design Phase had four subtasks: Prepare Tools, Writing Documention, Take Pictures, and Research. The 2nd Phase, which involved getting the bare bones of the app set up followed after the Research activity.



Phase II: Bare Bones had Phase IV: Final Product as a successor. However, Phase III has only ‘Set up running project in Unity’ as a predecessor and no successors. The disconnect from the other Phases is due to the fact that the subtasks in Phase III involve activities that are really only about adding extra features to the app, which can be added to or stopped at any time (there is no cap to how beautiful we can make the app or how many ghost types we can add).



The server must be set up before we can connect the server to the app and obviously, the final product must be finished before we can present and hand in the project.

# Responsibility Matrix

# Risk Management

# Problems & Solutions

Much of the initial problems that arose came from client-server interaction. There was some confusion at first about how to connect the server, which is written using jQuery and JSON, with the client code in Unity, which was written in C#.

This issue was solved using a public library called UnityHTTP provided by <https://github.com/andyburke/UnityHTTP>, which is itself based on Simon Wittber’s UnityWeb code, and therefore is licensed under GPL. GPL stands for General Public License, which allows end users, whether they be companies or individuals, to run, study, modify, and share the software. This eliminated much of the work and allowed us to write web requests without needing the full understanding of how Unity interacts with JSON.

Other issues that appeared were?

# Quality Assurance

# Future Plans

The current model of our product allows users to capture ghosts for points, which while acceptable for a minimum viable product is rather boring.

As mentioned in the executive summary, we intend to release updates, which will contain more features for our product to make the game more interesting. One such update we plan to release is one that will allow users to set their captured ghosts on other users, which hit points.

Currently, ghosts are differentiated by their appearance and the points they carry. In the future, we would differentiate more between the ghosts by adding mini-games and movements specific to each ghost type.