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 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

The Project 4

Background and Rationale 4

Budget 4

App Functionality 4

The Code 5

Scheduling 6

Responsibility Matrix 9

Risk Management 10

Problems & Solutions 11

Quality Assurance 12

Future Plans 13

Ghost Grab Updates 13

Similar Applications 13

# 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.

The idea for a game based on catching and releasing ghosts came from the movie franchise, Ghostbusters, which is getting a reboot this year. The ghost theme is generally well-received by consumers. As such, there is reason to believe that an application combining a virtual reality component with a ghost-based game will be successful in the marketplace.

## Budget

GhostGrab did not require any financial investment to create. If we were to launch it to the Android app store we would incur some costs but we have not reached that point. We would also have significant cost involved in handling a large volume of requests to our server. Our only investment was labor hours. We had four people working on the project and spent about 50 hours combined developing the product.

## App Functionality

This app is available for 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 1: 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 Amazon Web Services 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 for tasks like determining ghost locations. It also posts information to server as well. For example, the 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. 

Figure 2: 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

The responsibility matrix, assigning tasks to each person is shown below. As Project Manager, Dylan Dover had an ‘A’ for Approval on all tasks with the exception of the tasks which he was Primary or Secondary on.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step Name** | **Dylan Dover** | **Jason Mayer** | **Lawrence Waller** | **Selina Chen** | **Complete?** |
| Prepare Executive Summary | A |  | P |  | Y |
| Make Gantt Charts | A |  |  | P | Y |
| Make WBS & Responsibility Matrix | A |  | P |  | Y |
| UML Diagram | P |  |  |  | Y |
| Unity Web Requests--Client | S |  | I | P | Y |
| Unity Web Requests--Server | P |  | I | S | Y |
| Mobile Interactions | A | P | I | S | Y |
| Art Assets | S | P | S | S | Y |
| QA Testing | S | P | S | S | Y |
| Web Server | P |  | S |  | Y |
| App Integration with Web Server | A | S |  | P | Y |
| Location on App | A |  |  | P | Y |
| Map on App | A | P |  | S | Y |
| Collision and Encounters | A | S | P |  | Y |
| Client/Server Methods (e.g. Leaderboard Updating) | S |  | P |  | Y |
| Setting Traps | A |  | S | P | N |
| Trailer/Intro Video | A |  | P |  | N |
| Location Interface | S | P |  |  | Y |
| Risk | A |  | P | S | Y |
| Web Site/Forums for Fans | A |  | P |  | N |
| Prepare Final Bound Report | S |  |  | P | Y |
| Prepare Poster | A |  | P |  | Y |

# Risk Management

Some concerns that may pose risk to our project are shown in the table below.

The most critical risk is one of Copyright Infringement, as some of our ghosts (such as ‘Shrok’) are based on already existing franchises. Our game idea, in general was actually based on GhostBusters.

# Problems & Solutions

# 

