SOFTWARE DESIGN SPECIFICATION (SDS)

LABYRINTH

CREATED BY:

DANIEL MATHIEU

JOHN FOUAD

PATRICK BARRON

CHRISTOPHER WONG

December 14, 2016 ,REV B

1. Introduction

The goal of this project is to develop a 3D multi-player networked game, Labyrinth. There will be a maximum of 4 players per game, each attempting to reach the same final destination before the other players. There will be obstacles in the way, spread out randomly throughout the environment that will attempt to prevent the users from reaching the destination. User will have the ability to run away from obstacles. User controls will be limited to moving around using arrow keys.

Different conditions have to be satisfied by the final schedule and which have been specified in the SRS.

This document specifies the final system design for the system. It also gives some explanations on how the design evolved and why some design decisions were taken.

2. Design Considerations

There are several issues that will affect the design of this project. They are modularity, reliability, robustness, usability, performance, and scalability. Modularity is the ability of the code to be broken up into different sections, which allows for better debugging and optimization of the different parts. This is shown in the architecture where the different parts of the user interface, client, and server interact. Reliability is the ability of the project to perform for the given specifications of internal and environmental conditions over time periods. The robustness of the project is the system's ability to perform despite unpredictable or invalid input. The security is the ability to deter attacks against the designer's will, such as gaining access to the entire map layout. Usability is the ability for the users to navigate the project with ease. The performance is the project's ability to complete tasks within acceptable memory and time constraints. Scalability is the ability of the project to accommodate a growing number of users or data.

2. 2.1 Assumptions

Software

- The user interface is dependent on the unity libraries and gaming engine.
    - The user interface interacts with the unity gaming engine through C# scripts that interact with different built in objects.

Usage

- It is dependent and assumed that the user will use the arrow keys to move their player.
- It is assumed that the gaming environment is larger enough to host all the different users without a server failure.

Operational Environment

- It is assumed to be running as a download desktop executable.
- It is assumed that the hardware has a minimum of 1GB of RAM and a 1GB of virtual memory.
- It is assumed that the system latency is equal to or less than 100ms.

2.2 Constraints

- The data transmit rate between the server and user will increase the performance time.
- The memory limitation of the server and host machines will limit functionality and latency of the users.
- The internet connection limitations will impact the performance time of the system, as the automated bots and player updates will have a delay.

3. Architectural Design

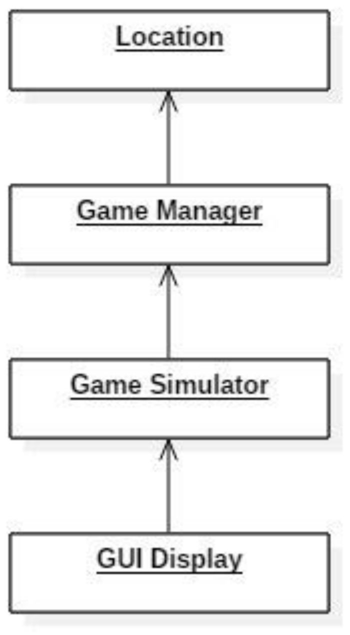

Figure 1: Aggregation Structure

The core of the architectural design is the use of a client that interfaces with a backend server over Photon Unity network connections to get the locations of the different players at any given time. The backend server will use a built-in unity database to store the location of all the players and bots at any given time. The client will then interface with the GUI by following a similar protocol of sending the players and bots locations. The GUI will take in this data and convert it into the movement of the different players and bots. The GUI will also send the client the updated location of the player to send to the backend server for storage.

3.1 Overview

Backend Server: Manages the location data for the different players and bots. The server's main role include sending the clients the updated location data for the others players in the game, while storing the updated location of the player in a Unity database, and connecting the players to the correct game rooms.

GUI: Gives users a simplistic approach to setting up and playing the game. Consists of code to make a simple user interface that can be easily navigated. This GUI will interface with the client simulator by sending updated location data and updating the image based on location data from the client. (C#)

Client: Poles the server over the Photon Unity network connections for updated location data of the other players and sends the updated location data of the player to be stored. The client then interfaces with the GUI by sending updated location data received from the server to be converted into a graphical image. (C#)

3.2 Rationale

We are using the architecture that we have chosen because it is feasible and will allow for a simplistic user interface from the user's perspective, as in it will be easy for them to navigate through the menu and setting up games. Also we must start off with having the backend server be the backbone of our architecture because it is critical in getting our game to function on a multiplayer level. Lastly, Labyrinth is very much a user interface game which is why GUI is our second most important function behind the server.

3.3 Conceptual (or Logical) View

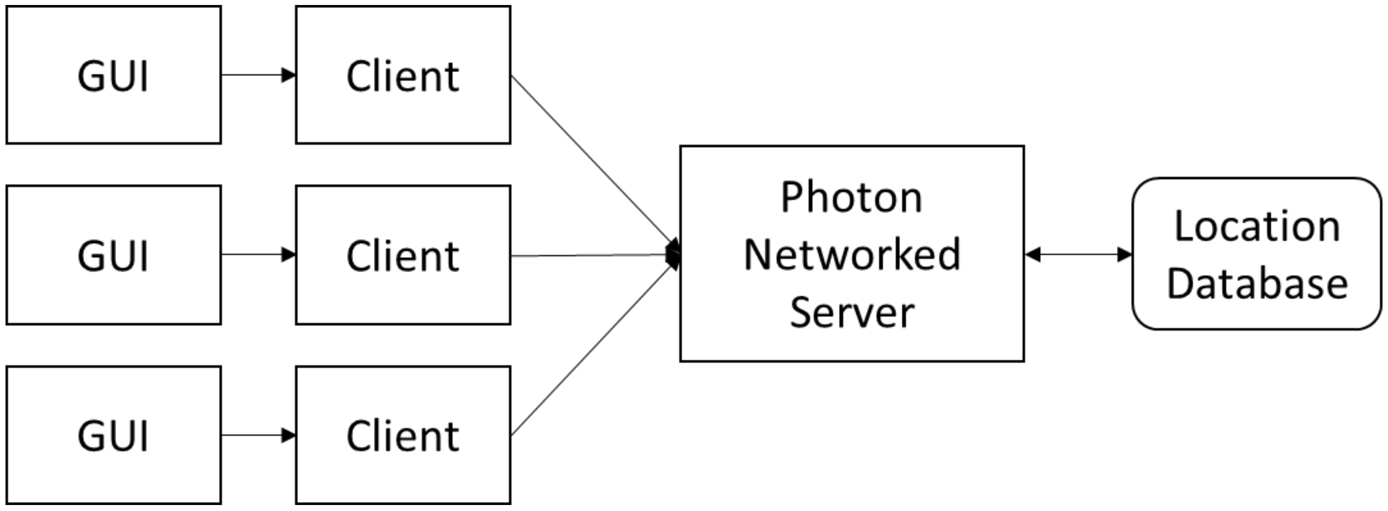

Figure 2: Conceptual Architure

The backend server manages the location data for all the players in the game, along with the automated bots. The server interfaces with a database to store the locations of all of the players at any given time. The server is routinely polled by the client for the location of the other players and the bots. The client also sends the updated location of the player it is managing to the server in order to be stored and used by other users. The client finally interfaces with the GUI by sending the updated location of all of the other players, while receiving the updated location of the player. The GUI converts these new data points into the movement of the other players and bots.

4. Low Level Design

4.1 Class Diagram

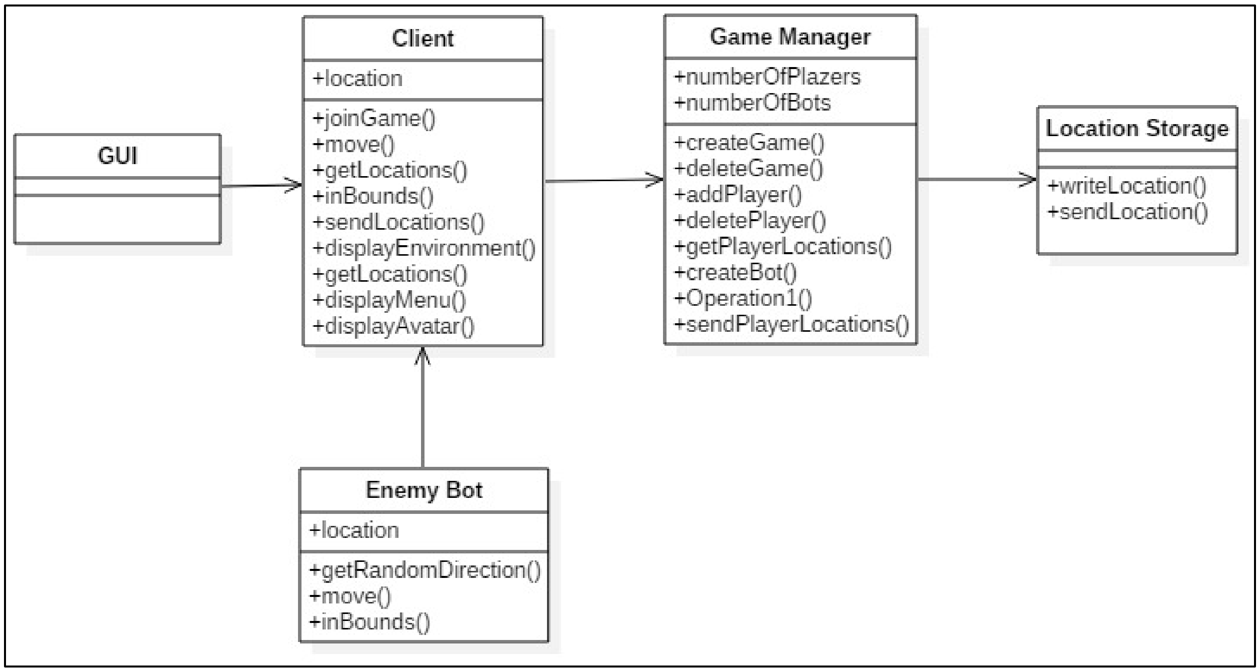

Figure 3: Class Diagram showing all classes and association in the system

4.2 Sequence Diagrams

4.2.1 Principle Action: Join/Leave Game

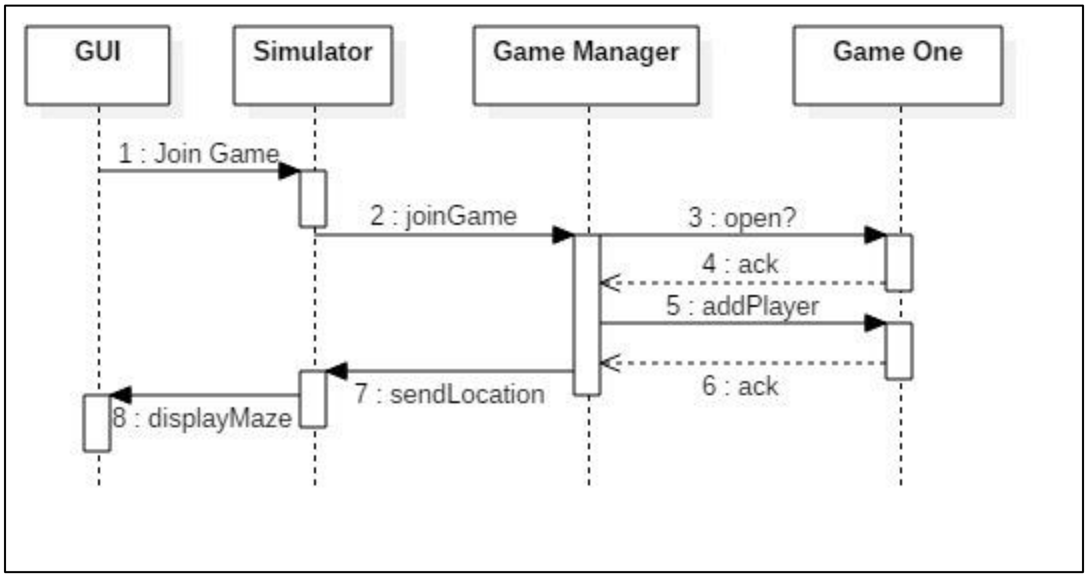

Figure 4: Sequence Diagram for Principle Action Join Game

4.2.2 Principle Action: Create/ Delete Game

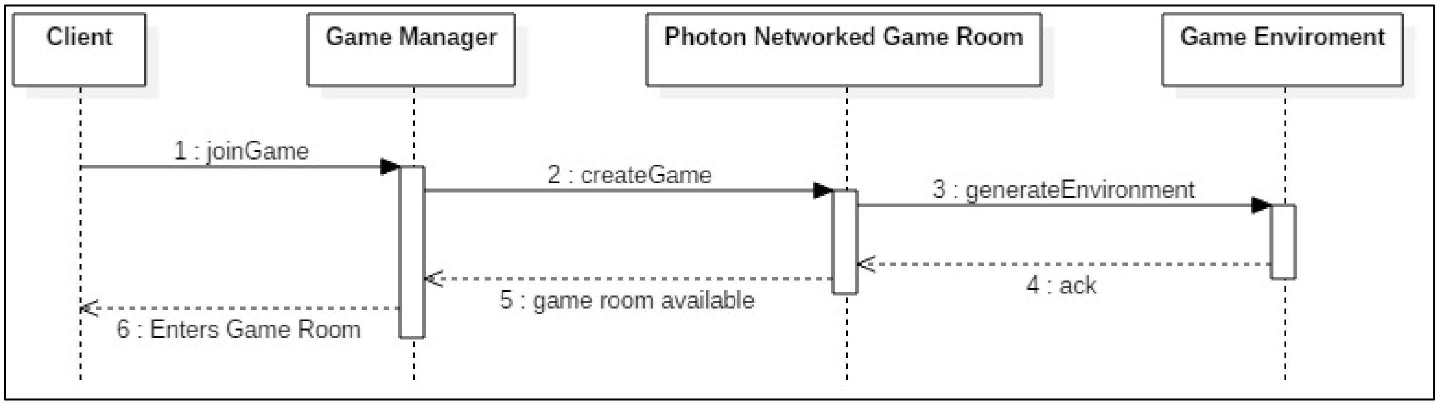

Figure 5: Sequence Diagram for Principle Action Create Game

4.2.3 Principle Action: Display 3D Environment

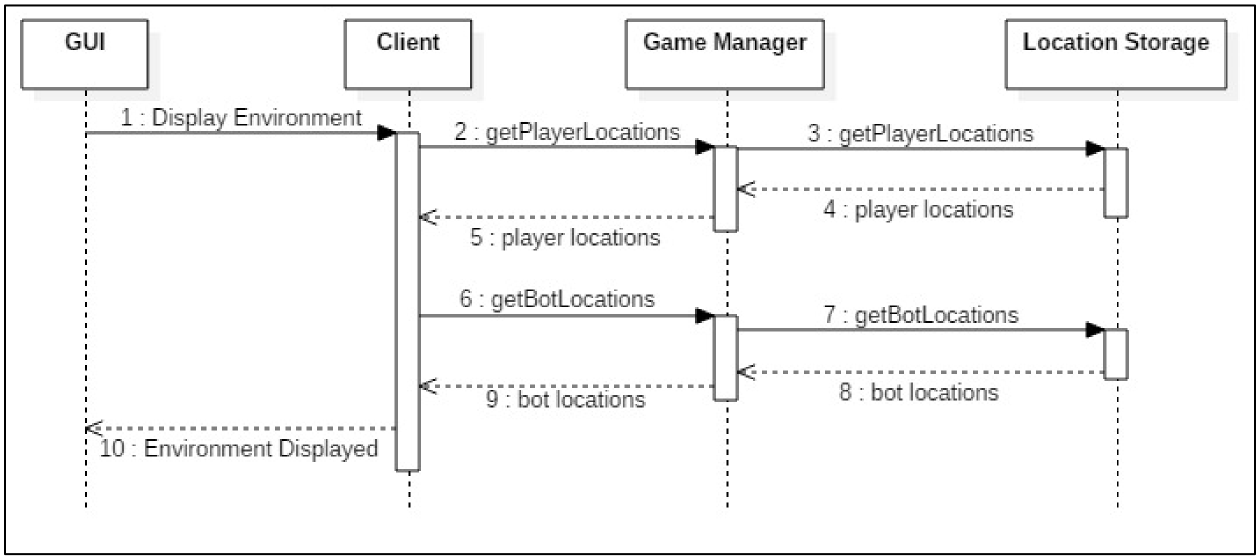

Figure 6: Sequence Diagram for Principle Action Display 3D Environment

4.2.4 Principle Action: Move Player

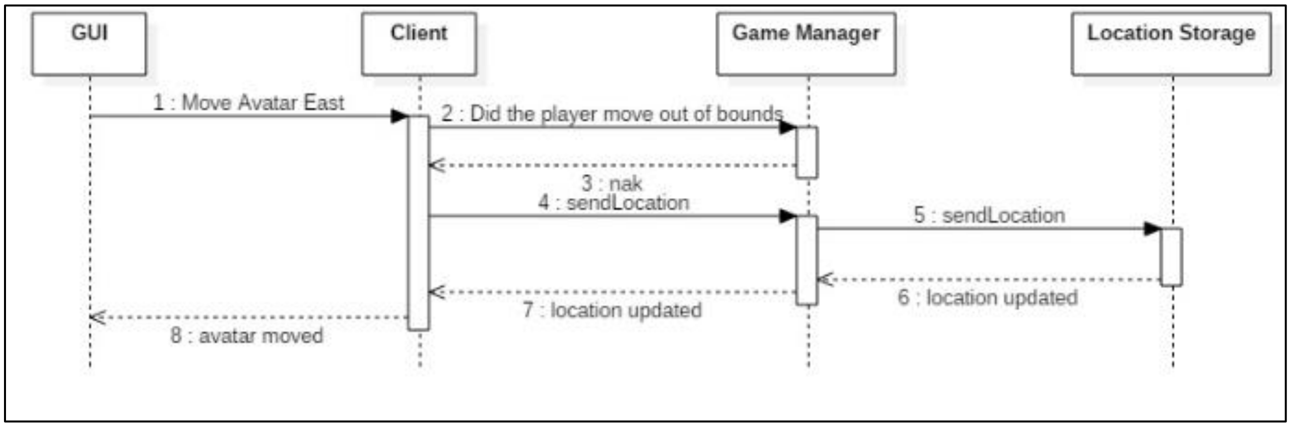

Figure 7: Sequence Diagram for Principle Action Move Player

4.3 State Diagram

4.3.1 Game Manager

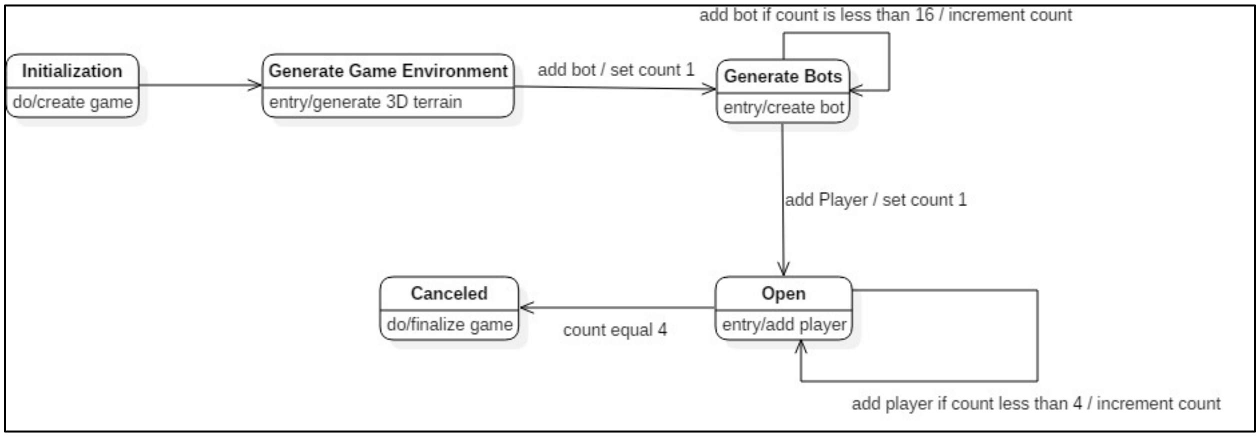

Figure 8: State Diagram for Game Manager

4.3.1 Game Simulator

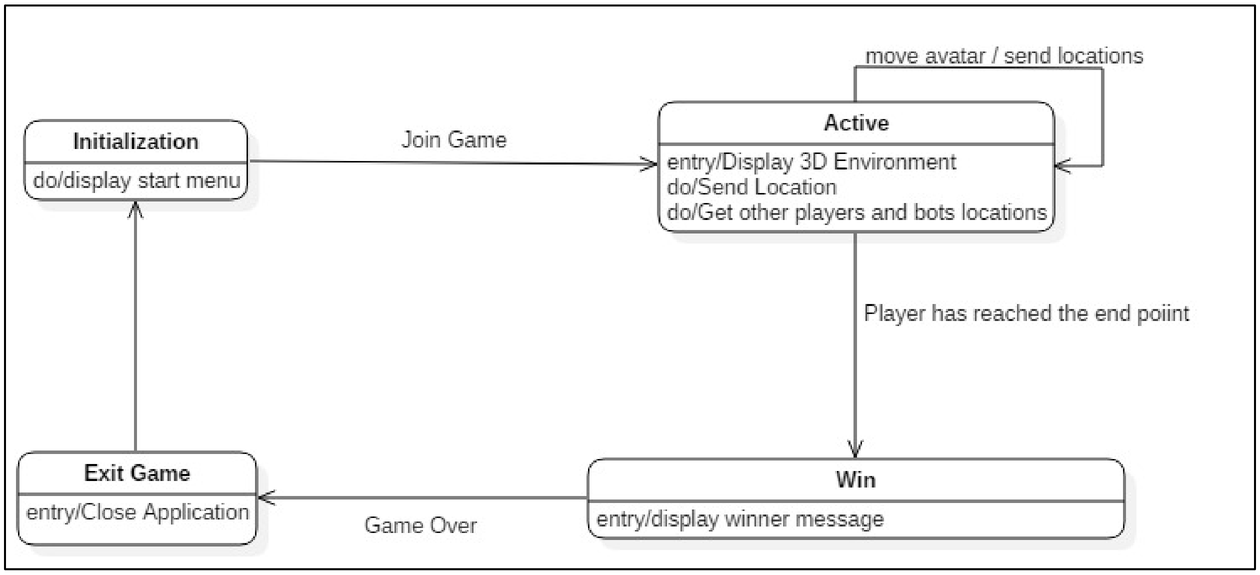
move avatar / send locations
Initialization
do/display start menu
Join Game
Active
entry/Display 3D Environment
do/Send Location
do/Get other players and bots locations
Player has reached the end poiint
Exit Game
entry/Close Application
Game Over
Win
entry/display winner message

Figure 9: State Diagram for Game Simulator

5. User Interface Design

Upon downloading the executable, one will see a menu with the following options:

- Join Game
    - User selects join game from the start menu
    - User assigned a 3D avatar and randomly placed on the maze.
- How to Play
    - The objective of the game, like in a traditional maze, is to reach the end location.
    - Users shall be given control of the avatar's motion using the arrow keys

Upon successfully joining a game:

- User will see the back of the player, as the game is played in third person point of view.
- To exit the game while it is still running, the player will need to quit the application using alt-F4 for windows users.
- All of this will be created with Unity Personal, free software available for download that helps create 3D games
    - Coding will be done using C# scripts.
    - Unity offers a vast number of publicly available asset packages that will be used to create game environments and characters.

    - Photon network services and database will be used for networking with the different clients.