SOFTWARE REQUIREMENT SPECIFICATION (SRS)

for

LABYRINTH

CREATED BY:

JOHN FOUAD

DANIEL MATHIEU

PATRICK BARRON

CHRISTOPHER WONG

Rev B, December 14, 2016

1: Introduction

1.1 Purpose

Labyrinth is designed to be a 3D, multiplayer game where players will have to avoid obstacles and enemies on a terrain and also users will be able to play a game where they can navigate a military base or a mountainous terrain.

This document is meant to delineate the features of Labyrinth so as to serve as a guide to the developers on one hand and a software validation document for the prospective client on the other.

1.2 Scope

In Scope:

a. Competitive multiplayer desktop platform that is accessed through an executable
b. Select from three different game modes to play
c. Automated enemies moving around in Labyrinth
d. Graphical User Interface that will display a 3D rendering of the playing field

Out of Scope:

a. Saving player's state and progress after disconnection
b. No mobile platform or application development
c. Health bar for the players that keeps track of a player's remaining life

1.3 References

a. NCURSES library c++ for first iteration
http://hughm.cs.ukzn.ac.za/~murrellh/os/notes/ncurses.html#using
b. Unity user manual for version 1.0
https://docs.unity3d.com/Manual/index.html

1.4 Overview

The rest of this SRS is organized as the following: Section 2 gives an overall description of the software. This description will include the level of proficiency that is expected of the user, general constraints that go into the development of the software, along with any assumptions and dependencies that go along with the software. Section 3 gives functional requirements, listed by use cases.

2: Overall Description

2.1 Product Perspective

Labyrinth is aimed towards an audience, which ranges from kids as young as 10 years old to adults in the online gaming community, including anyone who enjoys playing online games in their free time. It will allow for an enjoyable experience for users to take part in when they are seeking entertainment. Labyrinth will be user friendly, easy to play, and shall be reliable as long as the network connection holds.

Labyrinth intends to be a stand-alone product that should not depends on Unity. Our game will run on Unix and Windows. There are no memory constraints because we are not storing data. Labyrinth will run by an executable.

2.1.1 Software Interfaces

Unity
Photon Unity Networking Asset
Cloud Server
Version 1.79

This sends and receives information regarding the room that the client is in and its position on the map between the clients and the server.

2.1.2 User Interfaces

- Arrow keys for movement:
  Our code uses Unity's physics library to move our characters based on which key the user presses and will move the character in the respective direction on the map.
- Moving the mouse and clicking on the game mode:
  In order to play a game mode the user shall move their mouse onto a box that has a game mode option and then the user shall click the option they want to play and the game will load once they click their choice.

2.1.3 Communications Interfaces

We use the Photon Unity Networking Asset which uses RPC to communicate between our user interfaces and our system interfaces which allows the game to function properly and display the character movement properly and to select the correct game mode.

2.1.4 Memory Constraints

- 1 GB of Ram to host the game and to improve gameplay by allowing better graphics and reducing latency.
- 1GB of Virtual memory to host the data and executables.

2.2 Product Functions

Class of use cases	Use cases	Description of use case
Use case related game play	Create game	Creates a new game
	Simulate Game Play	Simulate game play
	Add Player	Adds player to the game session
	End Game	Ends the game
Use case related to user controls	Join Game	Adds the user to the current game
	Move Avatar	Moves the avatar in a direction
	Exit Game	Removes the user from the game
Use case related to visual display	Display Maze	Display the maze as a 3D image
	Display Start Menu	Display Start Menu

Use case related to automated bot	Create bot	Create an automated bot
	Delete bot	Delete automated bot
	Move Bot	Bot automatically navigates the maze

2.3 User Characteristics

a. The user should be familiar with controlling an avatar in the third person point view

b. The user should be familiar with the basic concept of navigating a maze.

c. The user should be familiar with opening and launching an executable

2.4 Principal Actors

The two principal actions in Labyrinth are the "player" or "user" and the "system"

2.5 General Constraints

a. Requires a constant internet connection

b. Internet bandwidth delay will be significantly noticeable

2.6 Assumptions and Dependencies

a. Labyrinth is dependent on the availability of internet connection.

b. Labyrinth would be available only from an executable file.

2.7 Apportioning of Requirements

Future requirements

- The user shall be able to exit the game without killing the window
- The game shall be available in a web browser
- The user shall be able to preform more function with their character such as jumping, shooting for tanks, and animated running.

3: Specific Requirements

3.1 Functional Requirements

We describe the functional requirements by giving various use cases.

Use case related game play:

Use Case 1: Create Game

Primary Actor: System

Precondition: User trying to join game

Main Scenario:

1. The application server shall check if there are any open game sessions currently running that are not at the full capacity of 4 players. Each map can hold a max of 4 players
2. If there are no available games currently running on the server in each map, user has to wait until someone exits one of them
3. The server shall add the player to the game

Alternate Scenario:

1. An active game session is not at capacity of four players. The create game function is canceled.

Use Case 2: Add Player

Primary Actor: System

Precondition: A game session is available
Main Scenario:

1. The server shall randomly position the user in the maze/mountains/desert
2. The server shall render the maze and player into a 3D representation.

Use Case 3: Simulate Game Play
Primary Actor: System
Precondition: Player is in a game session
Main Scenario:

1. The server shall restrict the user's movement to be within the confines of the maze and boundaries.
2. The server shall create and control automated bots.

.**Use Case 4:** Delete Game
Primary Actor: System
Precondition: A player has won
Main Scenario:

1. To exit the game, one must kill the program completely

Use related to User Controls:
Use Case 5: Join Game
Primary Actor: User
Precondition: User is at Start Menu
Main Scenario:

1. From the start menu displayed, the user shall be given the option to join a game in one of three environments
2. The server shall add the player to the game session

Use Case 6: Move Avatar
Primary Actor: User
Precondition: User is in the game
Main Scenario:

1. User shall be given control of the avatar's motion through the use of the "w", "a", "s", "d", or arrow keys.

Use Case 7: Exit Game
Primary Actor: User
Main Scenario:

1. Any point in the game, the user shall have the ability to exit the game through killing the process; "ALT+F4" for PC's and "COMMAND+Q"

Use case related to visual display:
Use Case 8: Create Maze
Primary Actor: System
Precondition: A new game has been created
Main Scenario:

1. Maze is hard-coded on the Labyrinth Terrain.

Use Case 9: Display Start Menu
Primary Actor: System
Precondition: Internet Connection
Main Scenario:

1. The display menu shall pop up when user runs the executable, gives user option which game to enter

Use case related to automated bot :
Use Case 10: Create Bot
Primary Actor: System

Pre Condition: Created game

Main Scenario:

1. The server shall generate enemy automated bots (cyclopes) in the maze.

Use Case 11: Move Bot

Primary Actor: System

Pre Condition: Active gameplay

Main Scenario:

1. The bots shall move randomly throughout the maze.

Use Case 12: Delete Bot

Primary Actor: System

Pre Condition: End of game

Main Scenario:

1. The servers shall delete all the bots at the end of the game.

3.2 Non-functional Requirements

a. The server shall have the capacity to host four players per game without system failure.

b. The server shall not have a latency of more than 100ms between the exchange of player locations to avoid the appearance of delay in the GUI.

c. The player locations shall be stored in a backend database.

d. The application shall be run on windows system only.

e. The usability of the system shall be limited to simple player movement controlled through the keys.

d. The portability of the system shall be limited to the transfer of the executable file and supporting data files required to access the unity gaming engine tools.

3.3 Performance Requirements

a. Internet bandwidth speed should be fast enough to avoid the issue of delay in the GUI.

b. The program shall be run by an executable.

c. 99% of the response should be within 100ms, except for loading the map.

3.4 Logical Data Requirements

a. Location data that is being transferred must be valid location on the map.

b. In the menu age the user must click the box of one of the option in order to join a game.

3.5 Design Constraints

a. Security: The map layout and location of players should be secured from being viewed or manipulated by hackers.

b. Fault Tolerance: The system should be able to restart in case of a system crash or power failure.

3.6 External Interface Requirements

-Photon Unity Networking Asset

-Allows for communication between the client and the server which connects the UI to the server and makes the gameplay function properly

- Input is the client's current player position and the output the updated player positions

-Range of place on the map

-Measured in frames

-10ms

-Synchronizes all inputs and outputs

- RPC data formats

4. Change Management Process

Changes in the requirements document will be updated along the way and will depend on our progress and based on what requirements we meet. Our final document will be released upon the release of version 1.0 which is our final product. The customer could contact us via email if problems occur and we will do our best to fix those problems and provide a refund if need be. Our team will make changes to the requirements and we will have to reach a consensus for changes.

5. Possible Future Extensions

a. Potential functionality to attack obstacles in your way
b. Potential functionality to attack obstacles in your way.
c. Potential change in maze where the walls of the maze rotate to alter the direction the player is running
d. Beam of light down from the sky pointing at the destination for where the players should be headed
e. Ability to run the program in a web browser.