Software Requirements Specification

for

Unity Game

Prepared by: Tyler Sullivan, Eric Toy

September 26, 2016

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Initial version	9/26/2016	Initial draft	1.0

1. Introduction

1.1 Purpose

The purpose of this document is to detail the requirements of a game created with the Unity engine.

1.2 Document Conventions

Priorities for higher-level requirements are assumed to be inherited by detailed requirements.

1.3 Intended Audience and Reading Suggestions

The intended audience of the software, in the scope of a class project, would be hypothetically for casual gamers who can download, access, and maintain a network connection. Although the audience spectrum would be quite broad, the target audience would probably extend to Generation Y and older users who have history of playing similar retro games first released in the late 80’s, especially that of Konami’s classic title “Bomberman”.

1.4 Product Scope

The scope of this project is limited to a small-scale, player vs player game available to systems that run a Windows or Mac environment. AI will be implemented in the form of bots for an enhanced multiplayer experience.

1.5 References

Stylistic and implementation design is initially referenced to Konami’s “Bomberman” franchise, first released in 1983. System architecture is based on a hybrid of client-server and peer-to-peer model, see reference RS game clients.

2. Overall Description

2.1 User Classes and Characteristics

The user class expected to use this software is an entertainment-seeking individual with basic knowledge of computing.

2.2 Operating Environment

The intended environment in which the software will operate on the game client end will be in Windows and/or OSX. At the server level, the environment will operate in Linux and developed using GO.

2.3 Design and Implementation Constraints

Due to time constraints, the gameplay map will be limited to a square grid, with only one map available at launch.

2.4 User Documentation

Basic documentation detailing how to start up and play a game will be provided with the product.

2.5 Assumptions and Dependencies

The user of the software will be operating in a Windows or Mac environment.

3. External Interface Requirements

3.1 User Interfaces

On the game client end, the user interface will feature a GUI designed and engineered through the cross-platform game engine, Unity. The screen layout will consist of a single window constrained world where the user will interact with sprites in the game using simple keyboard controls and/or on-screen directional buttons. Minor GUI standards will also include a Menu display, achievements and an about screen. Details of this user interface design will be documented in a separate user interface specification.

3.2 Software Interfaces

The game will interface with a server hosted in a Linux environment. The server will manage the flow of the multiplayer gameplay, as well as store game data in a database (PostgreSQL).

3.3 Communications Interfaces

The client(s) will communicate with the server via TCP/IP protocols.

4. Functional Requirements

Initialization & Startup

REQ-1: Game will execute in Windows environment, and the software will launch a single window frame that will display a welcome screen followed by a login screen.

REQ-2: Login screen will allow the user to login with a name/password combination.
A checkbox for "new account" creates the name/password combination if selected.

REQ-3: Under the main menu the user will choose between three buttons, new game, achievements or about.

REQ-4: New game will generate a new screen that presents the "lobby" to the user.

REQ-5: The lobby shows all games available, provided any have been created. An option for "create game" or "join game" is presented.

REQ-6: The "create game" option allows the user to enter a number of AI bots, and upon "starting", will create a holding pattern while waiting for another player.

REQ-7: Once two players have connected to an available game, the game launches. A new window is rendered with the game world.

REQ-8: The game world consists of a square grid with a "maze" available to the players in which they will each control the movement of a sprite throughout the maze.

Gameplay

REQ-9: The world shall present destructible and indestructible objects for the player to interact with. For example the maze structure will be created using "blocks" and objects such as "bombs" will be used for user interaction.

REQ-10: Movement in the world shall be restricted to the four cardinal directions.

REQ-11: A player may place a bomb at intervals of no less than 5 seconds. Bombs detonate in 2 seconds.

REQ-12: The match shall end when a player comes into contact with the blast radius of any bomb. Scores and achievements are then calculated.

REQ-13: A game timer shall be included to prevent stalemates.

REQ-14: Powerups exist on the map that increase the radius of that player's bomb explosion.

Misc

REQ-15: AI will be implemented in the form of bots to provide an enhanced multiplayer experience.

REQ-16: A player account, designated by a provided name and password, will store player information.

REQ-17: Achievements can be acquired through various gameplay actions. An account's achievements are stored in a PostgreSQL database.

5. Nonfunctional Requirements

REQ-NF-1: The game will utilize the Client-Server model. The client will communicate with the server via TCP/IP sockets. The server will host a relational Postgres database.

REQ-NF-2: The client will run on the Windows OS, and the server on Linux.

REQ-NF-3: The game client shall be implemented in C#, and the server in Go.

5.1 Business Rules

The creators of this software acknowledge that the original concept of this game belongs to Konami.

Appendix A: Glossary

Sprites: Two-dimensional graphic entities that represent the players.
Bombs: Main "weapon" for each character that is used to defeat the opponent. Variable explosion radius.
Character: A player in the game.
Maze: The travelable route presented to players within the world.
World: The map in which the gameplay occurs.