**IEEE 1016: Software Design Specification**

**CS4398 Online Multiplayer Game**

# Purpose of Game

An online multiplayer platformer game that may be played on desktop browsers and mobile browsers. Up to two players may participate in a game session.

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason for Changes** | **Version** |
| Raul Zuniga | 3/2/2015 | Initial version. | 1.0 |
| Raul Zuniga | 3/2/2015 | Added Section 5.2 Class Diagram | 1.1 |
| Raul Zuniga | 3/2/2015 | Added Section 3 Project Scope | 1.2 |
| Raul Zuniga | 3/2/2015 | Add Section 4 References | 1.3 |

# Project Scope

The system will provide a fun, free, online platformer shooter game experience. Only one person and two person games will be allowed. Multiple game sessions shall be allowed via a real-time connection mechanism. The user will only need to navigate to the URL and configure some settings and start a game from any personal computer, tablet or smartphone.

The game will be provided by a host server which will contain all the project files, a game engine called Impact.js, All maps and characters in the game will be provided by a level editor called Weltmeister that comes packaged with Impact.js. A Node.js controller running on the server, and a real-time connection using a web socket library called socket.io that is available with Node.js.

Players will start at one end of map and progress to the other end of the map where there is a portal they can enter to finish the level. There will be players in the map that can kill the player. The player will have to shoot and kill the enemies to allow him to progress towards the portal. Entering the portal, the player will advance to the next level. As the player progresses through the levels of the game the difficulty of the game will get progressively harder. The system will not save the state of the game when the player quits the game. When the user plays the game again, the player will start at the beginning of the first level.

# References

# Davy Cielen, Arno Meysman. *HTML5 Game Development with ImpactJS*. Packt Publishing, 2013. <http://www.packtpub.com>.

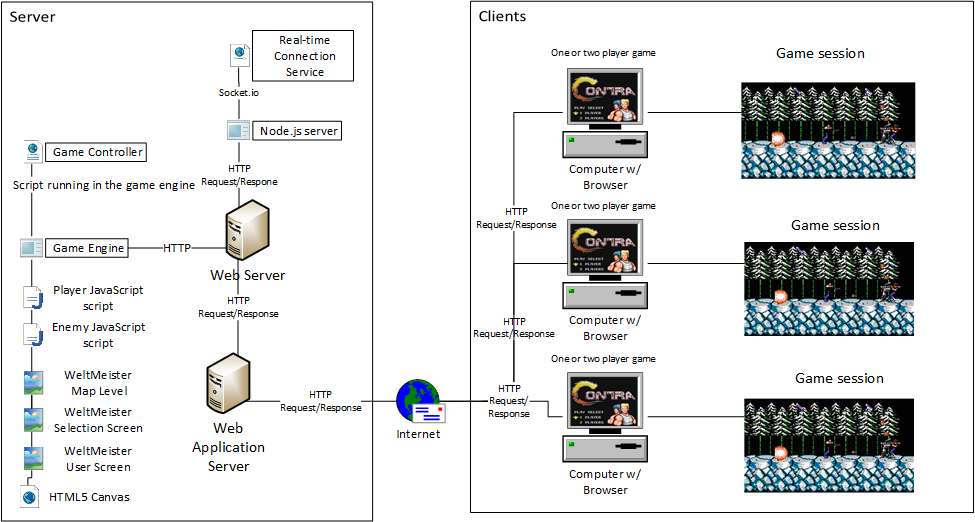
Foundation, Node.js. *https://nodejs.org/en/*. 2016. March 2016.

Freeman, Jesse. *HTML5 Game Development*. Sebastopol: O'Reilly, 2012.

Szablewski, Dominic. *Impact - HTML5 Canvas & JavaScript Game Engine*. n.d. March 2016. <http://www.impactjs.com>.

# System Design description

## Architectural Design



## Class Diagram

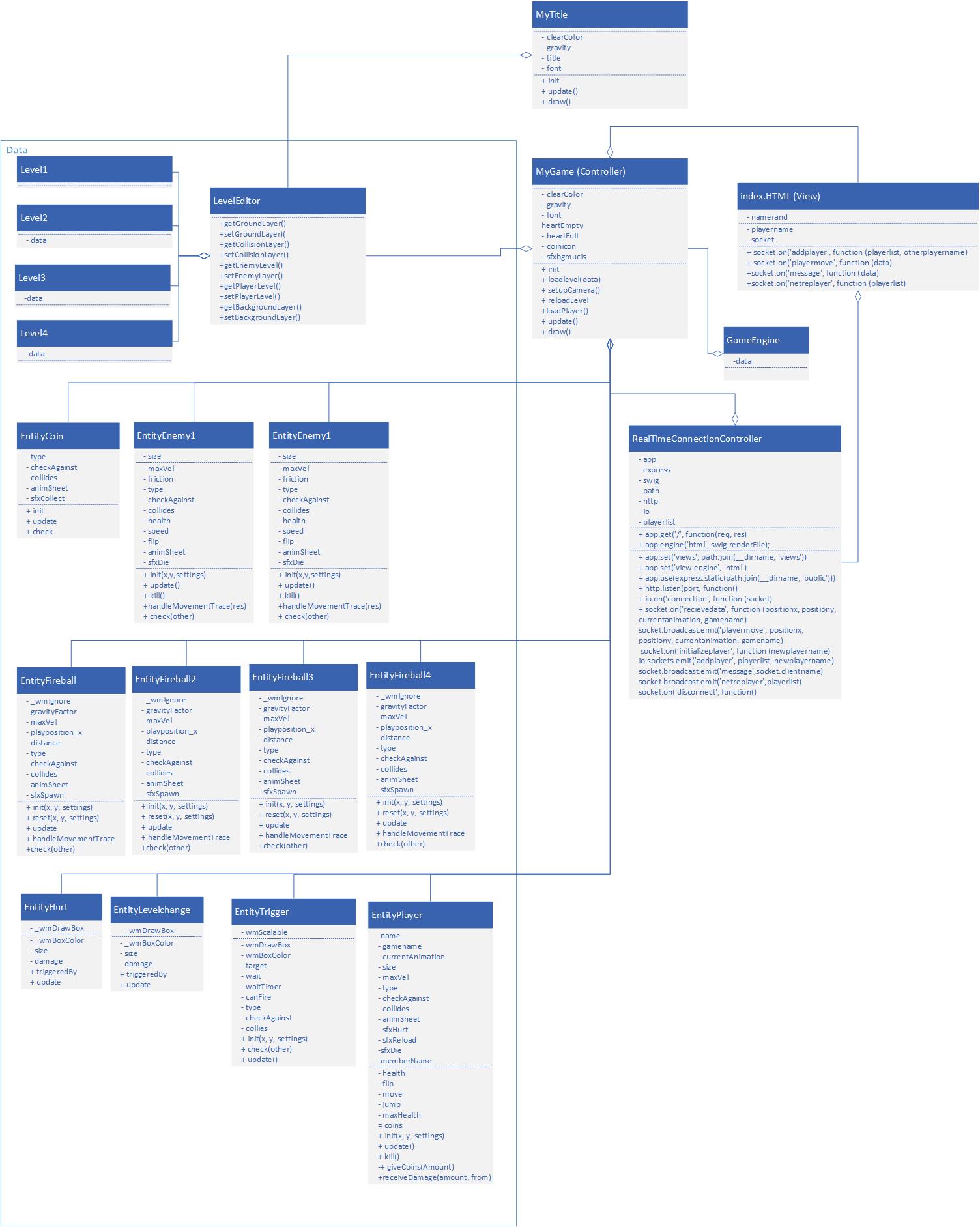


Figure 1 Class Diagram

## State Diagram Game Controller

## 

## Functional Requirements

### User Screen

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Technical Requirement ID** | **Test Case ID** | **Functional Requirement Description** | **Priority** | **Type** |
| 3.1.1 | 1 | In the User screen the input box for username will have system created username by default. |  | Interface |
| 3.1.2 | 2 | A User screen shall be the first screen displayed in the browser when the user types in the game URL and presses enter. |  | Functional |
| 3.1.3 | 3 | In the User screen the input box for username will have system created username by default. |  | Functional |
| 3.1.4 | 4 | The User screen shall have a selector to create a room for a one player game. |  | Functional |
| 3.1.5 | 5 | The User screen shall have a selector to join a room for a two-player game. |  | Functional |
| 3.1.6 | 6 | Creating a room takes the player to the Selection screen. |  | Functional |

We did not implement a User Screen.

### Starting a 1-player game

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Technical Requirement ID** | **Test Case ID** | **Functional Requirement Description** | **Priority** | **Type** |
| 3.2.1 | 7 | In the User screen the player starts a one-player game by selecting the selector for creating a room. |  | Functional |
| 3.2.2 | 8 | After the selector for creating a room is selected the Selection screen appears. |  | Functional |

### The game was started by going to the following URL: <https://impact-game.herokuapp.com/>. The game starts automatically after the keyboard (desktop browser) or on-screen button (mobile browser) is pressed in the splash screen. The splash screen was created in Weltmeister Level Editor that came with the game engine ImpactJS. The main page is an index.html file that includes the main javascript file which includes all entity classes used in the game. The game engine running on the server loads all the javascript classes and starts the game in a socket session which has a unique identifier.

### Starting a 2-player game

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Technical Requirement ID** | **Test Case ID** | **Functional Requirement Description** | **Priority** | **Type** |
| 3.3.1 | 9 | In the User screen two players can start a two-player game by selecting the selector to join a room. |  | Functional |
| 3.3.2 | 10 | After the selector for creating a room is selected the Selection screen appears. |  | Interface |

### Since the game is multiplayer a second player could join the game by going to the URL: <https://impact-game.herokuapp.com/>. The game starts automatically after the keyboard (desktop browser) or on-screen button (mobile browser) is pressed in the splash screen. The players are rendered after being spawned on the client by the game engine. The game engine running on the server loads all the javascript classes and starts the game in a another socket session which has a unique identifier

### Selection Screen

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Technical Requirement ID** | **Test Case ID** | **Functional Requirement Description** | **Priority** | **Type** |
| 3.4.1 | 11 | The Selection screen shall have a character select screen for each player. |  | Functional |
| 3.4.2 | 12 | The character select screen shall have an image of each character that can be selected by using an arrow button. |  | Functional |
| 3.4.3 | 13 | In the character select screen the image of the character will appear in color if there exists another player. |  | Interface |
| 3.4.4 | 14 | In the character select screen the image of the character will appear in black and white if there doesn’t exist another player. |  | Functional |
| 3.4.5 | 15 | In the Selection screen there shall be a button called “ready” to start the game. |  |  |
| 3.4.6 | 16 | In the Selection screen there shall be a “ready” button that will light up green after it’s clicked. |  | Functional |

### A selection screen was not implemented.



Mockup of Selection Screen

### Starting a game

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Technical Requirement ID** | **Test Case ID** | **Functional Requirement Description** | **Priority** | **Type** |
| 3.5.1 | 17 | In the Selection screen the user starts a one-player game by pressing the “ready” button. |  | Functional |
| 3.6.1 | 18 | In the Selection screen the user starts a two-player game by pressing the “ready” button for a two-player game. |  | Interface |

### The game was started by going to the following URL: <https://impact-game.herokuapp.com/>. The game starts automatically after the keyboard (desktop browser) or on-screen button (mobile browser) is pressed in the splash screen.

### Creating a game session

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Technical Requirement ID** | **Test Case ID** | **Functional Requirement Description** | **Priority** | **Type** |
| 3.7.1 | 19 | Once the game is started by pressing the “ready” button for either a one-player or two-player game, the game engine loads the level map. |  | Interface |
| 3.7.2 | 20 | After the game session has been initiated and the level map loaded, the dependent game libraries are loaded. |  | Interface |
| 3.7.3 | 21 | The main game screen shall briefly display “Level 1” which indicates the current level of the game. |  | Interface |
| 3.7.4 | 22 | The game shall display the health indicator of the player during game play. |  | Interface |
| 3.7.5 | 23 | The player character(s) shall appear at the start of the map with default weapon loaded |  | Functional |
| 3.7.6 | 24 | The game engine will load enemy sprites. |  | Interface |
| 3.7.7 | 25 | Interaction events between enemy sprites and players will wait to be triggered by player movement. |  | Interface |

After the user presses a button in the splash screen, this triggers the game engine to immediately load the first map level. Positioned at the beginning of the map in the level editor, the game engine loads the player character. The default weapon is set in the player.js class.

Enemy sprites are positioned in each map within the level editor and are then loaded by the game engine when a game session is initiated. Sprites wait to be triggered by player movement. These triggers are integrated within the enemy class.

Certain elements of the static background environment may also kill players (such as spikes). These are set within the level editor. Triggers that span the length of the hazard are linked to damage indicators. The game engine loads these with the level.

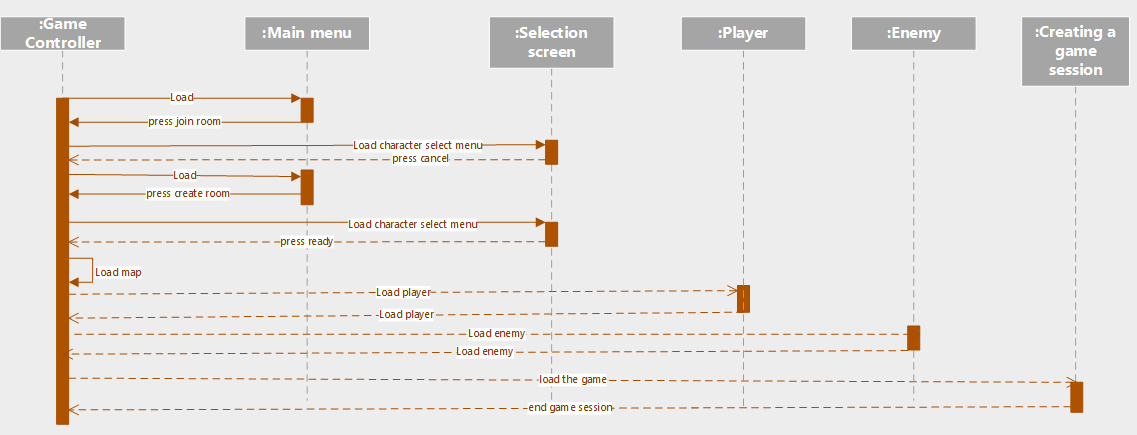


Diagram of a game session sequence

### Enemy Movement

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Technical Requirement ID** | **Test Case ID** | **Functional Requirement Description** | **Priority** | **Type** |
| 3.8.1 | 26 | Enemy classes shall move enemy sprites within the game. |  | Interface |
| 3.8.2 | 27 | Enemies will have autonomous movement. |  | Interface |
| 3.8.3 | 28 | The game shall allow enemy movement dependent on enemy type. |  | Interface |
| 3.8.4 | 29 | Riflemen-type enemies shall not be allowed to move. |  | Functional |
| 3.8.5 | 30 | Soldier-type enemies shall be allowed to move (run and jump) toward players. |  | Functional |
| 3.8.6 | 31 | Static-type enemies shall be allowed to change direction of their weapon toward a player but otherwise may not move. |  | Functional |

Implemented two enemy classes have differing enemy sprites that have autonomous movement. The first was a riflemen-type sprites that shoot at the player and may turn in their direction but otherwise are static. The second type of enemy class was a roadrunner that could kill the player by touch and was running in the map and needed to be killed from a distance. Everything the enemy sprite could do is specified in the Impact.js enemy classes.

### Enemy Combat

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Technical Requirement ID** | **Test Case ID** | **Functional Requirement Description** | **Priority** | **Type** |
| 3.9.1 | 31 | The game shall allow enemies to use various weapons but only one at a time. |  | Interface |
| 3.9.2 | 32 | The game shall allow the weapon type: pistol. |  | Functional |
| 3.9.3 | 33 | The game shall allow the weapon type: shotgun |  | Interface |
| 3.9.4 | 34 | The game shall allow the weapon type: assault rifle |  | Functional |
| 3.9.5 | 35 | The game shall allow the weapon type: laser. |  | Interface |
| 3.9.6 | 36 | Each weapon will have different capabilities (bullets per shot, distance bullets travel, firing rate). |  | Functional |
| 3.9.7 | 37 | The enemy weapon shall be dropped upon enemy death. |  | Interface |
| 3.9.8 | 38 | The game shall allow enemies to kill a player. |  | Functional |

Weapons for enemy combat are part of the enemy classes. A default weapon is loaded at the start of a map. When enough coins are collected, reaching this threshold triggers the game engine to upgrade the player’s weapon. The max damage and range of the bullets increase and the change is reflected visually in a different colored shot. Currently only one upgrade is available. When a player advances to the next level, the player is not able to keep an upgraded weapon and instead the default weapon is loaded.

### Player Movement

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Technical Requirement ID** | **Test Case ID** | **Functional Requirement Description** | **Priority** | **Type** |
| 3.10.1 | 39 | An individual Player class shall move a player within the game. |  | Functional |
| 3.10.2 | 40 | Players will have autonomous movement within the game. |  | Functional |
| 3.10.3 | 41 | The player shall have the ability to run. |  | Functional |
| 3.10.4 | 42 | Each player shall have the ability to jump. |  | Functional |
| 3.10.5 | 43 | Each player shall have the ability to crouch. |  | Functional |
| 3.10.6 | 44 | The player shall die if player falls into a hazard such as a pit. |  | Functional |

Implemented one player class with one type of sprite whose movement types were specified in the class. The user could run, jump, and shoot. The user could also collect coins which would increase the range of the rifle. We used collision maps using the Weltmeister Level Editor to specify how a player could die. These collision maps were one layer of the player map.

### Player Combat

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Technical Requirement ID** | **Test Case ID** | **Functional Requirement Description** | **Priority** | **Type** |
| 3.11.1 | 45 | The default weapon for a player will be a pistol. |  | Functional |
| 3.11.2 | 46 | A player shall not possess more than one weapon at a time. |  | Interface |
| 3.11.3 | 47 | The PC user shall be able to shoot a weapon by pressing a key on the keyboard. |  | Interface |
| 3.11.4 | 48 | The mobile device user shall be able to shoot a weapon by touching a spot on the screen. |  | Interface |
| 3.11.5 | 49 | The player shall be able to kill an enemy by shooting it. |  | Interface |
| 3.11.6 | 50 | The player shall die if touched by enemy. |  | Functional |
| 3.11.7 | 51 | The player shall die if shot by an enemy with one bullet. |  | Functional |

Implemented one player class with the ability to shoot. We configured the player class with certain keys on the keyboard to move and shoot. Collision maps in Weltmeister were used to configure how a player could die if touched. Player class configuration included health so if the player gets shot he would die.

### Player Completes a mission

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Technical Requirement ID** | **Test Case ID** | **Functional Requirement Description** | **Priority** | **Type** |
| 3.13.1 | 57 | The end of a map level shall be designated by a door or portal. |  | Functional |
| 3.13.2 | 58 | A player shall finish a map level by finding the exit portal and entering it. |  | Functional |
| 3.13.4 | 59 | A new map level will be loaded when a player finishes a map level. |  | Functional |
| 3.13.5 | 60 | A player will start a new level with only the default weapon (a pistol). |  | Functional |

Visual portals (doorways) are located at the end of each level. Triggers linked to loading maps are set in the level editor to trigger loading the next level. Once the player reaches the portal within the game, the game engine loads the next level. Players start each level with the default weapon.

### Game input devices

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Technical Requirement ID** | **Test Case ID** | **Functional Requirement Description** | **Priority** | **Type** |
| 3.14.1 | 61 | On a personal computer the user shall be able to move the player using a keyboard. |  | Functional |
| 3.14.2 | 62 | On a personal computer the user shall be able to move the player using a joystick. |  | Functional |
| 3.14.3 | 63 | On a tablet the user shall be able to move the player using the touchscreen. |  | Functional |
| 3.14.4 | 64 | On a mobile phone the user shall be able to move the player using the touchscreen. |  | Functional |
| 3.14.5 | 65 | On a mobile phone the user shall be able to move the player using the keyboard. |  | Functional |

For desktop, users may use their keyboard to control player movement. For mobile browsers, users control player movement through on-screen buttons. The game was not tested on mobile phones with physical keyboards.

### Pausing the game

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Technical Requirement ID** | **Test Case ID** | **Functional Requirement Description** | **Priority** | **Type** |
| 3.15.1 | 66 | The game will have a pause system to temporarily halt game progress. |  | Functional |
| 3.15.2 | 67 | For one-player games, the pause system shall have no timeout duration. |  | Functional |
| 3.15.3 | 68 | For two-player games, players shall not have the ability to pause. |  | Functional |

In a one player game, the game does not have a pause capability. In a two player scenario, the game does not pause upon the death of one player. The player’s screen follows the movements of the other player.

## Nonfunctional Requirements

### Response Time

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Technical Requirement ID** | **Test Case ID** | **Functional Requirement Description** | **Priority** | **Type** |
| 4.2.1 | 69 | The response time from the server shall be under 100 milliseconds. |  | Functional |
| 4.2.2 | 70 | The server shall be able to handle a maximum of 100 real-time connections. |  | Functional |
| 4.2.3 | 71 | The game should allow a minimum of 1 one-player and 1 two-player game to be played simultaneously. |  | Functional |
| 4.2.4 | 72 | The game shall support a minimum of two users at the same time. |  | Functional |

The player movement was configured with a timer so communication to the server was only done every 1/30 of a second. Communication between the clients and Node JS server was done through a Node JS API called socket.io. Socket io functions were developed for the client in JavaScript using socketio calls and for the server also using socket.io calls. Socket.io library is a very robust Node JS library and communicates using websockets. Multiple client sockets could be opened using a browser on a desktop or mobile device.