Software Requirements Specification

for

Obstacle Mania

Version 1.0 approved

Prepared by Adil Lalani, Shawn Reeves, Ruben Navarro

Team Adxatic

12/10/2018

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 2

1.5 References 2

2. Overall Description 2

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Classes and Characteristics 3

2.4 Operating Environment 4

2.5 Design and Implementation Constraints 4

2.6 User Documentation 5

2.7 Assumptions and Dependencies 5

3. External Interface Requirements 5

3.1 User Interfaces 5

3.2 Hardware Interfaces 6

3.3 Software Interfaces 6

3.4 Communications Interfaces 7

4. System Features 7

4.1 Gravity 7

4.2 Jump and Long Jump 7

4.3 Walk and Sprint 8

4.4 Health Pickups 9

4.5 Player Health Regeneration 9

4.6 Player Health Bar 10

4.7 Main Menu 10

4.8 Pause Menu 11

4.9 Damaging Traps 12

4.10 Trap Doors 12

4.11 Win Screen 13

4.12 Loss Screen 13

4.13 Axe Controls 14

5. Other Nonfunctional Requirements 15

5.1 Performance Requirements 15

5.2 Safety Requirements 15

5.3 Security Requirements 15

5.4 Software Quality Attributes 15

5.5 Business Rules 16

6. Other Requirements 16

7. Contributions 16

Appendix A: Glossary 19

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason for Changes** | **Version** |
| Revision 1 | 11/25/2018 | Added end section for Do Not Fall | 1b |
| Revision 2 | 11/26/2018 | Blades added to Gauntlet end section | 2b |
| Revision 3 | 12/01/2018 | Ruins improvements | 3b |
| Revision 4 | 12/05/2018 | Fixed misalignment of UI elements | 4b |
| Release 1 | 12/07/2018 | Presentation version finalized | 1.00 |

# Introduction

## Purpose

The purpose of this document is to present a detailed description of the Obstacle Mania video game version 1.0. It will explain the purpose and functions of the game, the interactive menus, the game’s overall purpose, the constraints under which it must operate. This document is intended for developer reference and for player use.

## Document Conventions

Highlighting and group editing tools were used to maintain organization of changes while making this document. In general Arial size 11 font was used.

## Intended Audience and Reading Suggestions

Readers should start with Overall Description (Section 2) which provides a general overview of the game. Then, readers can go to the System Features (Section 4) which talks in detail about the game and it’s various features. After reading System Features, it is recommended that readers move to the External Requirements (Section 3) to read about the hardware and software interfaces of the product. Lastly, the readers can finish by reading Other Nonfunctional Requirements (Section 5) and then Other Requirements (Section 6).

This Software Requirement Specification document is intended for:

* Developers who can review the project’s capabilities in order to improve the game by adding additional features.
* Project testers can use this document as a base for testing the product. This allows for testing to become more methodically organized.
* End users of this application who wish to read about what this project can do before experiencing gameplay firsthand.

## Product Scope

Obstacle Mania is an interactive computer game with a graphical interface, in which the user explores three levels containing obstacles, avoiding traps and contraptions, while picking up health pickups along the way. Each map caters to various difficulties ranging from easy, medium, and hard.

## References

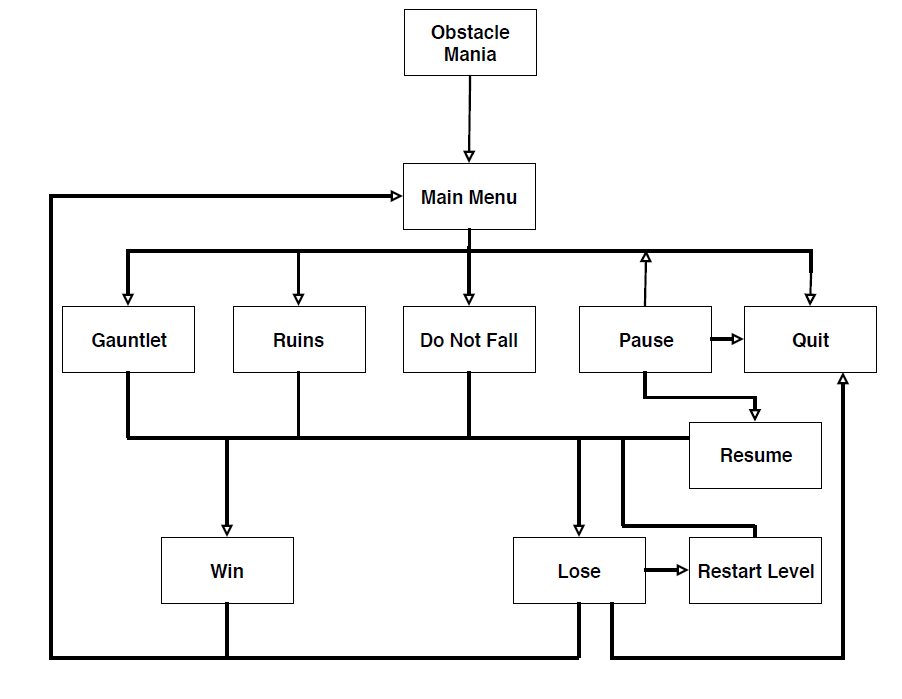
|  |  |
| --- | --- |
| [1] "Software Requirements Specification," 2018. [Online]. Available: <http://www.chambers.com.au/glossary/software_requirements_specification.php>.  [2] B. Stroustrup, "The C++ Programming Language," [Online]. Available: <http://www.stroustrup.com/C++.html>.  [3] "Unreal Engine 4," 2018. [Online]. Available: <https://www.unrealengine.com/en-US/what-is-unreal-engine-4>.   |  | | --- | |  | |

# Overall Description

## Product Perspective

This product is an entirely new game designed from scratch using the Unreal Engine 4 engine. It was made for a project for the Software Engineering (CEN 4020) class at Florida State University by a group of three developer students (Adil Lalani, Shawn Reeves, and Ruben Navarro). It does not follow a product line nor is it a replacement for an existing system. It was designed by utilizing C++ based Blueprints and each level of the game was designed by a different member of the development team.

## Product Functions

The following diagram demonstrates the functions of the game. It includes the various UI elements, the level names, and the various paths presented to the player from that position.

## User Classes and Characteristics

Developers

The game must use proper development practices to ensure improvements can be made to the product moving forward. Aspects important here would include thorough documentation of the product, and proper use of the Unreal Editor through blueprint comments, actor grouping, efficient scaling, and useful naming conventions.

Testers

Testers are important to please, as they should be selected to ensure enjoyment of the finished product by end users. Aspects important for achieving this goal are essentially identical to those for pleasing end users.

End Users

End Users are the primary focus of this product, as it is intended to be directly used. This would be composed of anyone wishing to experience the game for enjoyment. The most important aspects for this subset of users would be audio, visuals, and game mechanics. Game mechanics would include the ability to complete the levels, challenges for the user to overcome, and available features in-game.

## Operating Environment

SYSTEM Requirements – **Hardware**

CPU: Quad-Core Intel i5/i7 or AMD Ryzen R5/R7 processor, 2.5 GHz or faster

Space: 1 GB

Ram: 8 GB Ram

Video: NVIDIA GeForce GTX 970 or AMD Radeon RX 470 series card or higher

System Requirements – **Software**

OS: Windows 7 32-bit or higher

DirectX Runtime: DirectX End-User Runtimes (June 2010)

## Design and Implementation Constraints

* The software package should be designed to meet minimal specifications for hardware and software requirements as mentioned in section 2.4.
* Using Unreal Engine 4, changes to core aspects such as physics calculations must be heeded with care due to having unforeseen consequences on reliant fields.
* Developers should rely on C++ for blueprint construction and associated modifications.
* The assets used in the game will need to be compatible with Unreal Engine 4 and allow for the implementation of collision physics with the player and the axe.

## User Documentation

A readme is supplied with game files, with appropriate information and a link to a Team Adxatic support email.

## Assumptions and Dependencies

Upon acquiring the game files, the game is self-contained. Although, if the user’s computer is missing any external dependencies like DirectX or .NET, the game will ask permission from the user to be able to install those pieces of software.

# External Interface Requirements

## User Interfaces

The first interface is the main menu screen which gives the end-user the option to select between three maps and the ability to quit the game. If a level is selected two additional interfaces can be accessed by either completing the course or not completing the course. If the end-user fails to complete the course and falls victim to the presented obstacles an interface appears which allows the end-user to restart the level, return to the main menu, or quit the game. If the end-user completes the map an interface is presented which allows the end-user to return to the main menu or quit the game. The end-user also has the ability to bring up a pause screen interface by pressing P which allows for quitting or navigating to the main menu. A Health bar user interface is always present when the player playing through a level. The following table demonstrates the controls available to the user when playing a level.

|  |  |  |
| --- | --- | --- |
| **GAME CONTROLS** | | |
| KEY | FUNCTION | DESCRIPTION |
| W | Move Forward | Allows the player to move in the direction the player head is facing. |
| A | Move Left | Allows the player to move left while looking at the direction the player head is facing. |
| S | Move Backward | Allows the player to move back while looking at the direction the player head is facing. |
| D | Move Right | Allows the player to move right while looking at the direction the player head is facing. |
| P | Pause | Allows the player to pause the game to either resume, go to the main menu or quit the game. |
| Shift (Left or Right) | Sprint (with the W key) | Pressing and holding the key allows the player to sprint while also pressing and holding the W key. |
| Space Bar | Jump | Quick tap allows the player to make a short jump, while an elongated press and hold allows the player to make a longer and higher jump. |

## Hardware Interfaces

Obstacle Mania requires a computer system with an Intel or AMD 32-bit processor with a minimum of 1 GB of disk space and 8 GB RAM. The product also requires the use of other hardware devices such as a mouse and keyboard for input features. For output features a monitor is required to display the product contents and speakers or headset for sound distribution.

## Software Interfaces

The product will operate on Microsoft Window environments with a working version of Windows 7 or higher, DirectX End-User Runtime (June 2010), and .NET. Not having any of these pieces of software will present two options to the user. One being that the user would give the game permission to auto download and install the needed software, the second being the user not be able to play the game.

## Communications Interfaces

No communications are required by the product itself, as the product is able to be played entirely offline.

# System Features

Obstacle Mania consists of a basic set of features which include gravity, in air rotation, on-ground character controls like turn (left and right), jump, long jump, jog, and sprint. The game also incorporates axe controls consisting of aiming, throwing, and retrieving the axe. The environmental features consist of health pickups, a player health bar, and traps. UI features include a pause menu, win screen, loss screen, and a main menu.

## Gravity

4.1.1 Description and Priority

The gravity implementation allows for the ease of character control when jogging, jumping. It takes high priority as an incorrect implementation could ruin the gameplay has a whole.

4.1.2 Stimulus/Response Sequences

Jumping by pressing the space bar, walking by pressing W, or sprinting with Shift + W demonstrates the gravity in the game.

4.1.3 Functional Requirements

REQ-1: The gravitational force should be consistent at all times.

REQ-2: The three levels should make correct use of gravity.

## Jump and Long Jump

4.2.1 Description and Priority

The game implements two types of jumps. One being a quick jump, and the other is a long jump. It takes high priority since all three levels of the require the jump ability to be able to not take damage and thus pass the level.

4.2.2 Stimulus/Response Sequences

* By tapping the space bar, the player performs a quick jump.
* By pressing and holding the space bar for one to two seconds, the player can perform a long jump. Releasing the space bar earlier than one second cuts the jump length short. jogging or sprinting while holding the space bar can help the player jump over larger gaps.

4.2.3 Functional Requirements

REQ-1: Levels should be well adapted to use both jump and long jump.

REQ-2: Pressing space for one to two seconds should entail a correct jump duration at all times.

REQ-3: Gravity should be correctly utilized by jump and long jump.

REQ-4: Jumping forces should be realistic and should account for the player’s vertical and horizontal velocities as well as the surface inclination.

## Walk and Sprint

4.3.1 Description and Priority

The game implements two types of forward movements. One being a jog, and the other is a sprint. It takes high priority since all three levels of the require the walk/spring ability to be able progress through and pass the level.

4.3.2 Stimulus/Response Sequences

* By pressing and holding the W key, the player performs a steady jog.
* By pressing and holding the Shift plus the W key, the player can sprint. Releasing the Shift key while holding the W key converts the spring back into a jog.
* Releasing the W key ends the jog or sprint.

4.3.3 Functional Requirements

REQ-1: Levels should be well adapted to use both jog and sprint.

REQ-2: Pressing W should ensure a steady jog speed with accurate acceleration.

REQ-3: Holding the Shift key should simulate accurate acceleration and then maintain a steady sprint speed.

REQ-4: Releasing W should ensure a steady stop from either sprint or jog with accurate deceleration.

REQ-5: Releasing Shift while W pressed should ensure a smooth transition to a steady jog speed with accurate deceleration.

## Health Pickups

4.4.1 Description and Priority

The game offers a power up in the form of health pickups. When acquired, the health of the player will be set to 100% from whichever stage it was at before. The health pickup takes medium priority as it is an important aspect of the game and ensures quick recovery after taking damage from a trap.

4.4.2 Stimulus/Response Sequences

* Making contact with the health pickup activates the health replenishment.

4.4.3 Functional Requirements

REQ-1: Health regain amount should always be consistent.

REQ-2: When the player makes contact with the pickup, then the health regain should activate within one second.

REQ-3: The health pickup should always restore the player health to 100%.

## Player Health Regeneration

4.5.1 Description and Priority

After taking damage, the player will start regenerating its health slowly until it is at 100% health.

4.5.2 Stimulus/Response Sequences

* Taking damage from traps will start the heal process.
* At 100% health, the increase in health will stop.

4.5.3 Functional Requirements

REQ-1: Taking damage from a trap should always start the healing process.

REQ-2: When the player health has reached 100%, the healing process should stop.

REQ-5: The health bar should be able to access the increase in player health every second so that it is able to display the increase.

## Player Health Bar

4.6.1 Description and Priority

The player health bar represents the current health percentage of the player. Taking damage from traps will cause the bar to display the percentage lost. After taking damage, the health bar displays the player’s ability to heal itself by showing the steady increase in health per second.

4.6.2 Stimulus/Response Sequences

* Starting any level will cause the display of player health to activate.
* Taking damage from traps will show the damage taken.
* When at less than 100% health, a steady increase in health can be seen.

4.6.3 Functional Requirements

REQ-1: The health bar should be easy to understand.

REQ-2: Taking damage from a trap should accurately display the amount of damage taken for that trap.

REQ-3: Health pickup effect should be clearly displayed whenever utilized.

REQ-4: The health bar should be clearly visible in every level.

REQ-5: The health bar should clearly display the increase in player health every second.

## Main Menu

4.7.1 Description and Priority

The first UI the player sees is the Main Menu. It provides the option of player one of the three levels or quitting the game. It also utilizes a background to give a demo of how the game looks like.

4.7.2 Stimulus/Response Sequences

* By launching the game’s executable, the player can get to the main menu.
* By pressing the first button named Gauntlet, the player is spawned into the Gauntlet map.
* By pressing the second button named Ruins, the player is spawned into the Ruins map.
* By pressing the third button named Do Not Fall, the player is spawned into the Do Not Fall map.
* Pressing the Exit button exits the game.

4.7.3 Functional Requirements

REQ-1: The buttons launching the levels should spawn the player into the correct level and enable game input.

REQ-2: The button for exit should exit the game cleanly.

REQ-3: The buttons should be clearly visible and easily clickable.

REQ-4: The text for the levels and exit action should be clearly visible and easily understandable.

REQ-5: The Main Menu background should accurately display the game graphics.

## Pause Menu

4.8.1 Description and Priority

The pause menu in the game is activated when the player wants to pause the game and either take a break and/or utilize one of the three features provided by the menu. The features consist of

* Resume – Resuming the game
* Main Menu – Go to the Main Menu
* Exit – Exit the game

4.8.2 Stimulus/Response Sequences

* By pressing the P key, the pause menu is activated.
* After the menu is activated, simply clicking one of the three buttons should perform the appropriate action.
* Clicking the Resume button will exit the pause menu.

4.8.3 Functional Requirements

REQ-1: Pressing the P key while in a level should always result in the activation of the pause menu.

REQ-2: The button for exit should exit the game cleanly.

REQ-3: All buttons should be clearly visible and easily clickable.

REQ-4: The text for each action should be clearly visible and easily understandable.

REQ-5: Clicking the resume button should exit the pause menu.

REQ-5: Clicking the Main Menu button should bring the player to the main menu.

## Damaging Traps

4.9.1 Description and Priority

Assorted traps are employed by the game to present obstacles for players to overcome. This subsection of traps is composed of those with parts that collide with the player and cause damage to them. Effective traps are essential to creating challenge for the player and creating an engaging feel in the environment.

4.9.2 Stimulus/Response Sequences

When the user collides with damage-causing components of these traps, damage is caused by reducing the player’s health by the magnitude of the trap’s damage capability.

4.9.3 Functional Requirements

REQ-1:Traps should be used in an engaging manner to enhance the experience. All moving parts should fit into the level, if visible

REQ-2:The health system must be used to cause damage to the player.

## Trap Doors

4.10.1 Description and Priority

Trap doors are used in the game as the only spatial trap available. They utilize the gravity system by opening an empty space below the player when the players walks over them. They provide a valuable alternative to damaging traps when creating obstacles.

4.10.2 Stimulus/Response Sequences

When the user puts pressure on the base of the trap door, the door must swing down and drop the player if they remain in the area.

4.10.3 Functional Requirements

REQ-1:Trap doors should be able to swing on their hinges to properly open into empty space.

REQ-2:Trap door should only be used to control the player’s flow of movement or to drop them into precarious areas.

## Win Screen

4.11.1 Description and Priority

A user interface must be displayed when victory conditions for the level are met. Options are given to the player to return to the main menu or quit.

4.11.2 Stimulus/Response Sequences

When the user selects the different interface buttons, they must be directed to the correct location.

4.11.3 Functional Requirements

REQ-1:Main Menu must return the user to the main menu

REQ-2:Quit game must exit the application

REQ-3:The screen must always and only be displayed upon reaching a level’s victory conditions.

## Loss Screen

4.12.1 Description and Priority

A user interface must be displayed when the player dies. Options are given to the player to Restart the level, return to the main menu, or quit.

4.12.2 Stimulus/Response Sequences

When the player’s health is reduced to zero, they are considered dead and this interface must be presented

4.12.3 Functional Requirements

REQ-1:Main Menu must return the user to the main menu

REQ-2:Quit game must exit the application

REQ-3:Restart Game must return to the beginning of the current active level, with the character’s original state.

REQ-4:The screen must always and only be displayed upon the player dying

## Axe Controls

4.13.1 Description and Priority

The user must have responsive controls over the use of the axe to aim, throw, and retrieve the axe while playing appropriate animations.

4.13.2 Stimulus/Response Sequences

* By pressing the right mouse button, the aiming functions is activated, and its animation displayed
* By pressing the left mouse button while aiming, the axe is thrown, playing the throw animation and throwing the axe on a proper course in accordance with the user’s facing direction and gravity
* The ‘R’ key must activate the automagical retrieve ability of the axe. The axe flies back to the player and lands perfectly in the right hand.
* Throwing the axe at a destructible object like a vase shatters that object and allows the player to pass through.

4.13.3 Functional Requirements

REQ-1: The axe must obey gravity.

REQ-2: The axe must display animations for all functions available

REQ-3: The axe must trigger collisions when overlapping other actors and destructibles

REQ-4: All functions must be triggered by their respective input buttons

REQ-5: The right mouse button should present the zoom along with the hit marker.

REQ-6: Axe throwing should use realistic acceleration and consistent velocity.

# Other Nonfunctional Requirements

## Performance Requirements

Obstacle Mania is a relatively small application that requires minimal system resources. It has been designed to provide no delay to the system from other key processes. Although, if the user has one or more memory and/or graphics intensive applications open while playing, a significant drop in frames per second may be experienced with levels that have high poly assets like Gauntlet (fire assets). Therefore, it’s recommended to have any resource intensive applications closed when playing Obstacle Mania.

## Safety Requirements

Obstacle Mania is 100% safe to use. To avoid any complications when using the product please ensure that the system requirements are available to avoid over heating issues and possible damage to hardware devices such as the CPU and Video card.

## Security Requirements

Obstacle Mania doesn’t require the user to provide any confidential information. All it requires is a minimum of 1 GB of storage on the user’s hard disk, no internet connection is required as well.

## Software Quality Attributes

In order to ensure reliability, Obstacle Mania will respond to player actions virtually instantly. When the player triggers the commands for movement the character will promptly proceed to navigate towards that direction given that no obstacles are stationed in its path. As the player holds and lets go of the axe the axe will navigate to the specific location being aimed at by the mouse hardware component. Once the R key is pressed the axe will promptly return to the user, timing varies depending on the distance the axe has traveled.

When it comes to usability and ease of translation, the graphical user interface is straight forward and easy to decipher. Providing options to the user relevant to what their next choice might entail. Adaptability is also demonstrated by the difference in level difficulty, if the end-user feels one map is specifically to easy or too difficult the other provided maps contain a slight difference in difficulty factor. Our game focuses ease of use and ease of learning, while provided a necessary challenge for all end-users.

## Business Rules

Players have full freedom with the game when the game is open. It is made sure that there are no portions of the software that could harm either the player’s hardware or software on which he/she’s playing on.

# Other Requirements

This project may be used by the developers at any time own time to base their next game off, or to expand this game further. Outside developers may also use the game’s source code to make their own iteration of an obstacle game as well. Outside users may download this game anytime this game is available online.

# Contributions

Listed below are the contributions made by each team member. This was very much a team effort: some aspects were assisted by other team members

**Adil Lalani**

* Do Not Fall Concept
* Do Not Fall Layout
* Do Not Fall Graphics
* Do Not Fall Soundtrack
* Do Not Fall Level Blueprint
* Main Menu
* Pause Menu
* Modified Loss Screen
* Sprint Functionality
* Modified Jump Functionality
* Gauntlet Soundtrack
* Audio/Video Editing

SRS Contributions

* Sections: 1.3, 2.5, 2.7, 3.1 (half), 3.4, 4, 4.1-4.8, 5.5, 6, appendix (1/4)

**Shawn Reeves**

* Gauntlet Concept
* Gauntlet Layout
* Gauntlet Graphics
* Gauntlet Level Blueprint
* HUD Health Bar
* Health Regeneration
* Health Pick up 1
* Exit Portal 1
* Victory Screen
* Initial Loss Screen
* Resource Collection

SRS Contributions

* Sections: 1.2, 2.3, 2.6, 3.4, 4.9 to 4.13

**Ruben Navarro**

* Ruins Concept
* Ruins Layout
* Ruins Graphics
* Ruins Soundtrack
* Ruins Level Blueprint
* Health Pickup 2
* Exit Portal 2
* Large Player Statue
* Lead Tester
* Audio Editing
* Documentation Management

SRS Contributions

* Sections: 1.1, 1.4, 1.5, 2.1, 2.2, 2.4, 3.1 (half), 3.2, 3.3, 5.1 to 5.4, appendix 3/4

Appendix A: Glossary

|  |  |  |
| --- | --- | --- |
| **Acronym** | **Term** | **Definition** |
| SRS | Software Requirement Specification | **SRS** stands for Software Requirements Specification, which is a **document** that fully describes the expected behavior of a software system. Functional requirements are documented in an **SRS**, as are non-functional requirements such as performance goals and descriptions of quality attributes [1]. |
| UI | User Interface | The Main Menu, Pause Menu, Win Menu, and Loss Menu constitute as an UI element. |
| C++ | C Plus Plus | C++ is a general-purpose programming language. It has imperative, object-oriented and generic programming features, while also providing facilities for [low-level](https://en.wikipedia.org/wiki/Low-level_programming_language) memory manipulation[2]. |
| UE4 | Unreal Engine 4 | **Unreal** is a pure C++ **engine** designed for high performance. Its advanced CPU/GPU profiling tools and flexible renderer equips developers to efficiently achieve quality VR experiences[3]. |