Software Requirements Specification

for

Going Down

**Version 1.0 approved**

**Prepared by Bruce Conrad, Rebecca Aloisio, Seneca Koos, Joe Honeychuck, Michael Sullivan, and Nicholas DeRoss**

**Edinboro University**

**2/24/2020**

**Table of Contents**

**Table of Contents………………………………………………………………………………...[ii](#_1fob9te)**

**Revision History…………………………………………………………………………………[iii](#_3znysh7)**

**1. Introduction…………………………………………………………………………………..[1](#_2et92p0)**

1.1 Purpose……………………………………………………………………………………...……...[1](#_tyjcwt)

1.2 Document Conventions…………………………………………………………………………….[1](#_3dy6vkm)

1.3 Intended Audience and Reading Suggestions………………………………………………………[1](#_1t3h5sf)

1.4 Product Scope……………………………………………………………………………………....1

1.5 References………………………………………………………………………………………….2

**2. Overall Description…………………………………………………………………………..3**

2.1 Product Perspective………………………………………………………………………………...3

2.2 Product Functions…………………………………………………………………………………..3

2.3 User Classes and Characteristics …………………………………………………………………..3

2.4 Operating Environment…………………………………………………………………………….4

2.5 Design and Implementation Constraints……………………………………………………………4

2.6 User Documentation………………………………………………………………………………..4

2.7 Assumptions and Dependencies……………………………………………………………………4

**3. External Interface Requirements……………………………………………………………5**

3.1 User Interfaces……………………………………………………………………………………...5

3.2 Hardware Interfaces………………………………………………………………………………..5

3.3 Software Interfaces…………………………………………………………………………………5

3.4 Communications Interfaces………………………………………………………………………...6

**4. System Features………………………………………………………………………………7**

4.1 Player………………………………………………………………………………………………7

4.2 Elevator…………………………………………………………………………………………….7

4.3 Player Interactive Objects………………………………………………………………………….8

**5. Other Nonfunctional Requirements……………………………………………………….10**

5.1 Performance Requirements……………………………………………………………………….10

5.2 Safety Requirements……………………………………………………………………………...10

5.3 Security Requirements……………………………………………………………………………11

5.4 Software Quality Attributes………………………………………………………………………11

5.5 Business Rules…………………………………………………………………………………….11

**6. Other Requirements………………………………………………………………………...12**

**Appendix A: Glossary…………………………………………………………………………..12**

**Appendix B: Analysis Models…………………………………………………………………..13**

**Appendix C: To Be Determined List…………………………………………………………..16**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

## Introduction

## Purpose

The purpose of this document is to specify the requirements for “Going Down” (Version 1.0), a VR game being produced by our team for use in open house environments as a showcase experience with the Oculus Rift. This document describes the scope of the game and its functional and nonfunctional requirements. This document is limited to the game itself, as it is a self-contained system.

## Document Conventions

* References to other sections within the document and to external documents are **bolded**.
* References to game objects are underlined.
* The feature priorities are split into three categories which are defined as:
  + High - features that are required for the game to function and meet the client’s specifications.
  + Medium - features that are not required for the game to function but accomplish the client’s vision for the game.
  + Low - features that enhance the quality of the game but are not required for the game’s functionality nor the client’s specifications.

## Intended Audience and Reading Suggestions

This document is intended for all members of the development team to get a better understanding of the game requirements. It is broken into sections, such as functional and nonfunctional requirements for features present within the game. Each section should be reviewed by all members.

* Those working on the coding (testing & design) aspects should refer specifically to **Sections (2-4)**; it outlines the game as a whole.
* Those concerned with operating the game (specifically in the open house environment intended) should refer to **Section (5)** for safety requirements to ensure safe operation of VR equipment.
* Those interested in using the game outside of the open house environment should consult **Section (2)** for an overview of the game’s features, and **Section (3)** for external requirements.

## Product Scope

The software of focus is the VR game “Going Down” – which will be available to everyone for 10-minute trials at Edinboro University’s open house. Going Down is being created with the goal of introducing new people to the Oculus Rift. Assuming the player is able to reach the climax of the short narrative, they will be forced to “jump” from a seemingly tall virtual building. If implemented properly, this highly immersive and exhilarating experience will leave a lasting impression on players, and they will walk away with a better understanding of the capabilities of VR technology. A secondary dividend is the promotion of Edinboro University's Computer Science program.

## References

This software was built using Unreal Engine, and thus follows both the **Unreal Engine End User License Agreement for Publishing** and the **Unreal Engine User License Agreement for Creators**, referred to later within this document.

Copies of these can be found at:

1. Unreal Engine End User License Agreement for Publishing:

https://www.unrealengine.com/en-US/eula/publishing

1. Unreal Engine User License Agreement for Creators:

https://www.unrealengine.com/en-US/eula/creators

1. Since the game requires the use of the Oculus Rift or the Oculus Rift S, users should refer to:

https://www.oculus.com/legal/health-and-safety-warnings/?locale=en\_US

for health and safety information associated with the devices.

1. For additional information about the game, refer to the **Going Down Game Manual**.

## Overall Description

## Product Perspective

Going Down is a self-contained VR game developed using Unreal Engine 4, and is consequently subject to the **documents 1 and 2** mentioned in **Section (1.5)**. The team’s client expects the game to function as an entertaining demonstration of VR technology at open houses and similar events. Once the game is compiled into an executable, it will run on Microsoft Windows operating systems and will not require Unreal Engine or other specialized software.

## Product Functions

* The virtual field of view must continuously adapt to the rotation of the player’s headset.
  + Example: The player turns their head to the left; their view within the game rotates in the same direction.
* The player must interact with the virtual environment via hand-held controllers.
* The player must also navigate the player character through the VR environment using the hand-held controllers.
* The player is restricted to bounded playable areas within the virtual landscape.
* The game must have an end condition for the player.
* The game must be compatible with the Oculus Rift and the Oculus Rift S.
* The game must operate on at least Windows 10.

## User Classes and Characteristics

Developers

* These users will be developing and testing the game.
* These users will be college students.
* These users will have knowledge of Unreal Engine 4.22.3, the Oculus Rift, and the client’s requirements for the game’s design.
* These users will have access to the Unreal Engine files – which is essentially the code – for the game and the GitHub repository that it is stored on.

Demonstrators

* These users will be presenting the game at open houses and other similar events.
* These users will be adults and students associated with Edinboro University.
* These users must have access to the executable files and any additional required information.
* These users will have knowledge of how to set up the demonstration, operate an Oculus Rift, execute the game, and operate the game.
* These users will understand the safety concerns associated with the game.

Players

* These users will be using and interacting with the game.
* These users can range from late elementary school students to adults.
* These users are not required to have knowledge of how to play the game, use an Oculus Rift, or operate computers in general.

## Operating Environment

* The game must be operable on Windows 10.
* The game must be compatible with the Oculus Rift and Oculus Rift S.
* During development and maintenance, the game will be compatible with Unreal Engine 4.22.3 or newer. If opened in a version of Unreal Engine newer than 4.22.3, the game must then be opened on the newer version of Unreal Engine from then on.
* <amount of RAM and storage space required to run the game will be determined at a later date>

## Design and Implementation Constraints

Tool constraints:

* Must use a Windows 10 computer.
* Must use Unreal Engine 4.22.3 or newer.
* Must use the Oculus Rift or the Oculus Rift S for VR.

Game design constraints presented by the client:

* Restrict graphic content (such as blood, gore, and/or sexual content) due to the wide age range of potential users.
* Users should be able to complete the game within 5 minutes and must take no more than 10 minutes overall.
* The game must involve the player traversing a tower-like structure.
* The player will travel to different floors within the tower via elevator.
* The elevator cannot go to any floor it has previously visited.
* There must be a minimum of three distinct floors – including a roof.
* An event must occur on the second floor that encourages the player to go to the third floor.
* The end of the game will involve the player leaping from a plank on the roof.

## User Documentation

Within the game, there will be a Controls Menu that explains how to control the player character. There will also be written documentation explaining how to set up the game, how to use the controls, and listing safety information in the **Going Down Game Manual**.

## Assumptions and Dependencies

The developers are assuming that they will have access to models and assets that accurately simulate the desired visuals for the game or have the ability to create what is needed themselves (i.e. a model for a desk or a fire effect). The developers are also assuming that there will be adequate documentation for Unreal Engine features that they are unfamiliar with.

## External Interface Requirements

## User Interfaces

When the user starts the game, they will be presented with the Main Menu. The Main Menu will have four options to select from: starting the game, transitioning to a screen explaining the controls, exiting the game, and crediting those involved with the game’s creation.

After starting the game, the user will be able to bring up the Pause Menu. The Pause Menu will have three options to choose from: returning to the Main Menu, bringing up a reminder for the game controls, and exiting the game.

Both of the above menus will have standard game controls; the player will either point the hand-held controller at the button they intend to click, or they will use the joystick to navigate to the button they wish to select.

Throughout the game, the player will have a HUD that displays their current task, as well as a reminder of the Pause Menu and the button to access it.

An optional interface for the user will be a Reading Display. This would pop up when the player interacts with specific objects (i.e. a report, a terminal) within the game and presents the player with text and/or graphics. The player would navigate the text by either swiping with the hand-held controllers or using the joystick to scroll. To exit and return to the game, the player would simply click a button.

All of these interfaces must be centered in the player’s field of vision, that way they can clearly see and read the display regardless of which direction they are facing.

## Hardware Interfaces

The game needs to react to inputs from the Oculus Rift and the Oculus Rift S systems. The movement of the headset of the Oculus system moves the camera within the game, affecting what the player is able to view in the world. The inputs from the Oculus hand controllers move the player character and enable the player to interact with the virtual world. Specifics about the controls are listed in the **Going Down Game Manual**.

## Software Interfaces

The Windows 10 operating system is needed in order to run the game. Other than that, the game is a self-contained system and will not be interacting or communicating with other software components.

## Communications Interfaces

The game does not require any sort of communication with external entities.

## System Features

## Player Character

* + 1. Description and Priority

The player character is the character whose actions are controlled directly by the user of the game. In Going Down, there will only be one player character in the game at a time.

The player character is an integral part of the Unreal Engine; it’s built-in. The game would not be playable without a virtual puppet to explore the building and look around.

Overall:

* Benefit: High
* Penalty: High
  + 1. Stimulus/Response Sequences

To maneuver the player character, the user will employ the hand-held controllers and move the joystick. To look around the virtual landscape, the user can rotate their head in the direction that they desire to see, and the Oculus Rift headset will continuously update their perspective. More of the specific controls will be added once the game’s mechanics are official.

4.1.3 Functional Requirements

REQ-1: The player character must be controlled by the Oculus Rifts’ hand controllers. It is moved with the joysticks and the buttons enable the player character to interact with its environment.

REQ-2: The view of the player character must change with the rotation of the Oculus Rift headset.

**Refer to Section (6) Appendix B for the diagrams illustrating the interaction of the player**

## Elevator

4.2.1 Description and Priority

The elevator functions as a “waiting room” between levels. It is designed to conceal new levels as they are being loaded while previous levels are degenerating. This is crucial to maintain player immersion while minimizing the processing power required to run the game. Thus, the elevator is considered a high priority system feature.

Not all of the level’s assets in the project will need to be loaded simultaneously, increasing the efficiency of the game. With reduced processing power, the overall game could include significantly more levels and run more smoothly. In the case of “Going Down”, the virtual map is big enough that any efficiency gained from the implementation of the elevator will be highly beneficial.

The penalty of excluding the elevator is far greater than the cost of creating it. There would be no game without the elevator; it is integral to the design itself. While the creation of the elevator may provide some challenges, the benefits outweigh the risks. The primary obstacle will be smoothly syncing the elevator model with the level-loading function.

Overall:

* Benefit: High
* Cost: Low
* Penalty: High

4.2.2 Stimulus/Response Sequences

The user will interact with a small button that is adjacent to the elevator model. Similar to a real-life elevator, the doors will open, and the user is expected to enter. The interior will also consist of multiple buttons spaced uniformly along the wall. After pressing a button on the inside, the elevator doors will close and the elevator will travel to the floor that corresponds to the pressed button. By the time the elevator arrives at the new floor, a new level will already be generated, and the user must exit and complete a task before returning to the elevator for reuse.

**Refer to Section (6) Appendix B for the diagrams illustrating the interaction of the elevator**

4.2.3 Functional Requirements

REQ-1: The elevator must load levels and unload other levels as it moves.

REQ-2: The elevator must have a model that is detailed enough that it appears to function as a real elevator to the user.

## Player Interactive Objects

4.3.1 Description and Priority

The player will be able to interact with several objects throughout the virtual world. Each object class will have a unique function, and the priority of each object varies.

Terminals:

The terminals will be visual displays that contain the informational details necessary to fully enjoy the game’s story. While the game may benefit from enhancing player immersion, the terminals are optional, and there will be little to no penalty for excluding them.

Overall:

* Benefit: Moderate
* Penalty: Low

Failsafe Switch:

The switch allows the player to revitalize the virtual building’s power once it is knocked out. Because the switch is necessary to properly traverse the building and complete the game, the switch is a high priority feature. The game also benefits from including another interactive element, heightening its entertainment value.

While it is not difficult to make a switch model, proper implementation of its function will take some careful analysis and consideration. The team is less familiar with this type of feature, and it will prove difficult to estimate the time it will require to create the switch.

Regardless, the switch is a necessity, so the existence of the switch presents a high penalty for its exclusion. It’s imperative that the team prioritizes this feature.

Conversely, the cost of the switch is low. Once the switch has been created and debugged, the feature may be reused as many times as necessary. Furthermore, the switch will be linked directly to the elevator, allowing for streamlined functionality and reactivity among the objects involved.

Overall:

* Benefit: High
* Penalty: High
* Cost: Low

4.3.2 Stimulus/Response Sequences

The player will interact with each object, activating a specific sequence depending on the object type. Some sequences will progress the game, and other sequences may simply enhance player experience.

Terminals:

The user will interact with terminals by clicking on them. The terminal will then bring up the reading display that occupies the majority of the player’s field of vision (these details may change later). The player can scroll through the information and may exit the reading display by clicking again.

Failsafe Switch:

The switch will also be activated by a click. It cannot be interacted with after the initial click.

**Refer to Section (6 Appendix B) for the diagrams illustrating the use of the objects**

4.3.3 Functional Requirements

REQ-1: The switch must allow for interaction between the elevator and the switch.

REQ-2: The terminal must allow for the display of items on the player character’s view port.

## Other Nonfunctional Requirements

## Performance Requirements

* The game should load each new level as the player enters the elevator to lessen the loading time between levels.
* The user should not have to wait longer than 5 seconds for a new level to load while in the elevator to ensure that the gameplay is quick and thus letting many people play through the game.
* The game should not drop below 30 FPS when the player is interacting with any objects to ensure a smooth, quality playing experience for the user.
* The game should transition seamlessly into cutscenes to ensure that the player does not experience confusion or lose immersion.
* Restarting the game should not take longer than 5 seconds to ensure that more people can play through the game in a smaller timeframe.

## Safety Requirements

Because the game utilizes VR, the game should only be operated in a room:

* With enough physical space to move around in
* Free of obstructions that may cause damage to the player or their surroundings

In the very last stage of the game, a physical board will be incorporated without the player’s knowledge to further enrich the player’s experience. When the board is introduced, sufficient procedures should be carried out to ensure player safety. This includes but is not limited to:

* Sufficient padding to ensure prevention of injury in the event of a fall
* Marking off the area where the player is permitted to operate the VR headset using tape
* Supervision when the player is in-game

Protection of the equipment is also a priority, and while the player is being supervised in-game, any potential damage to the VR headset should be noted and prevented. This includes:

* Monitoring of wiring and connections to the headset
* Monitoring of the player to prevent movement that could potentially cause damage to the headset (i.e. falling off)

Because the game was developed for use in an open house showcase environment, the headset should also be cleaned and checked after each individual use.

As the game was developed with the Oculus Rift in mind, additional safety information can be found at the link in **Section (1.5), document 3**.

## Security Requirements

Since this is a small single player game that is only used in open house environments, there is no real need to protect any data within the game. The demonstrator of the game should secure the user’s belongings while they play, since their vision will be obscured with the headset on. After use, the Oculus Rift must be stored away in a secure room. The Oculus Rift must not be left unattended while it is not in the secure room.

## Software Quality Attributes

Since the game is being designed for use within an open house environment, it will only be available to those in attendance. The game should be sufficiently tested and free of bugs that could hinder the player experience and/or delay the demonstration. Also, because of the open house environment, the gameplay should be around 5 minutes on average and should take no longer than 10 minutes maximum. This is to ensure that as many people as possible get an opportunity to try VR. The game should be easy for the users to pick up and play; any “learning curve” should only involve initial movement.

## Business Rules

Any modification of the game’s source code should only be performed by members of the development team. The player should not have access to the game’s code or the ability to change the code while playing.

## Other Requirements

**Appendix A: Glossary**

* VR - Virtual Reality
* Oculus Rift - a VR headset and hand controllers that connect to a computer and allows the user to explore a virtual world that simulates a 3D environment
* HUD - a Heads-Up Display is a visual presentation of the player’s game attributes such as health, ammunition, tasks, etc.
* Level Loading and Unloading - a process in Unreal Engine 4 that allows a specific level within an entire project to be used at one time while storing the other levels within the project away for later use
* Player Character - a game object; the model that the game creates for the user to manipulate and play the game
* Main Menu - a game object; the menu utilized for the user to begin the game, view the controls, etc.
* Pause Menu - a game object; the menu used in-game to stop gameplay, review controls, quit game, etc.
* Reading Display - a game object; an optional display that is activated when the user interacts with any in-game object that has readable text for the user

**Appendix B: Analysis Models**

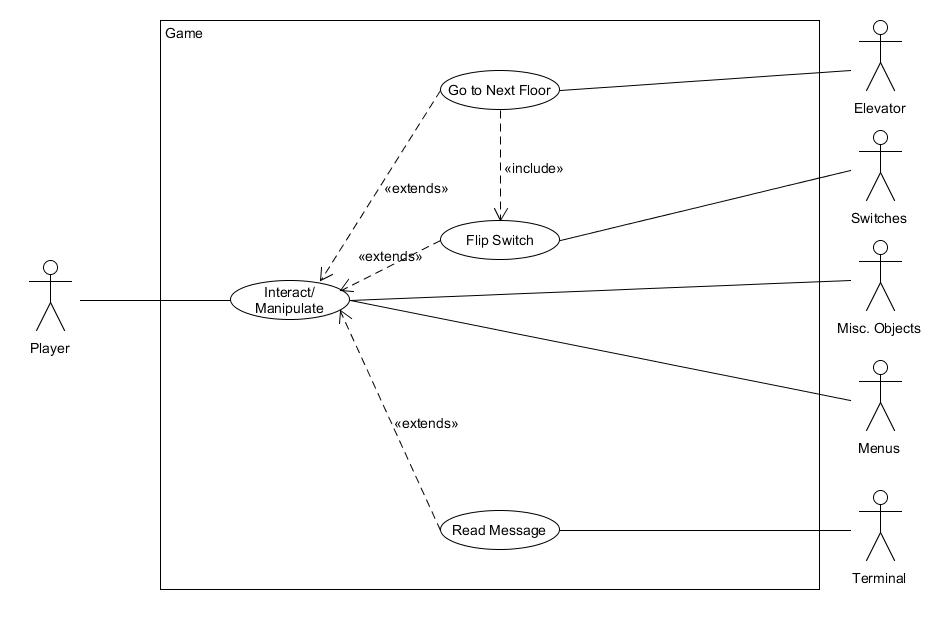


Figure 2.1 - Player Use Case Diagram (Refer to **Section (4.1 - 4.2)** for details)

Name: Player Use-Case Diagram

Actor(s): Player, Elevator, Switches, Misc. Objects, Menus, Terminal

Preconditions:

* The game as a whole has been loaded for play

Scenario:

1. Player interacts/manipulates other actors in the game
2. Player does one of the following depending on the type of object interacted with
   1. Goes to the next floor via the elevator
   2. Flips the switch to reactivate the power
   3. Opens one of the menus of the game
   4. Reads a message on a terminal

Exceptions:

* TBD

Open Issues:

* TBD

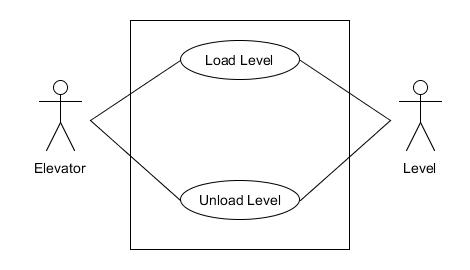


Figure 2.2 - Elevator Use Case Diagram (Refer to **Section (4.1)** for details)

Name: Elevator Use-Case Diagram

Actor(s): Elevator, Level

Preconditions:

* The player must be within range to activate the elevator
* The switch must be flipped to ON position if the power was knocked out

Scenario:

1. The elevator loads the next level
2. The previous level is simultaneously unloaded

Exceptions:

* During the initial stages of game design, levels will not be connected via this system

Open Issues:

* How to connect the levels together for this process (Note: we are considering using an Unreal Engine feature called level streaming to accomplish this task)

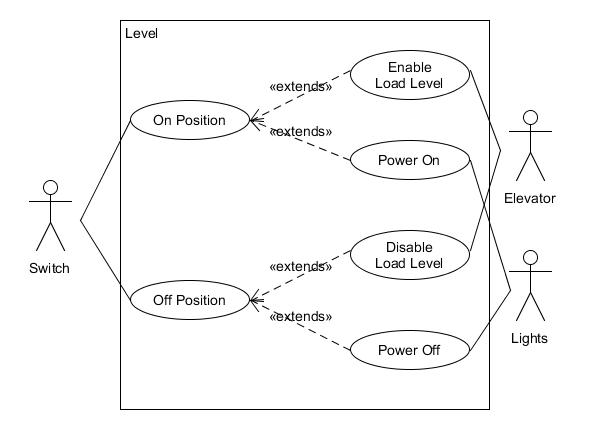


Figure 2.3 - Switch Use Case Diagram (Refer to **Section (4.2)** for more details)

Name: Switch Use-Case Diagram

Actor(s): Switch, Elevator, Lights

Preconditions:

* The player has reached the point in the game where the elevator is rendered inactive
* The level containing the switch has been loaded without issue

Scenario:

1. The switch begins in the OFF position
2. The OFF position will disable the level loading and the power of the building
3. Upon player activation (refer to **Figure 2.1**) the switch is in the ON position
4. The ON position reenables level loading and the power of the building

Exceptions:

* During testing, these may be defaulted to one specific setting to check if everything is functioning properly

Open Issues:

* How to connect the switch blueprint to the operations described above

**Appendix C: To Be Determined List**

* What to include in the **Going Down Game Manual**
* The client requested that the player character can be controlled via the physical motions of the player if possible. Research will need to be conducted to see if this is possible with just the Oculus Rift and Unreal Engine.