"Ethoreal Shift": A Role Playing Game for the PC Platform

BY

Swarna Kundu

ID: 142-40-128

AND

Fahim Ahmed

ID: 142-40-138

This Report Presented in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science in Multimedia and Creative Technology

Supervised By

Dr. Shaikh Muhammad Allayear

Associate Professor and Head

Department of MCT

Daffodil International University

DAFFODIL INTERNATIONAL UNIVERSITY

DHAKA, BANGLADESH

JULY 26, 2018

APPROVAL

This Project titled **Ethoreal Shift: A Role Playing Game for the PC Platform**, submitted by Swarna Kundu & Fahim Ahmed to the Department of Multimedia and Creative Technology, Daffodil International University, has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in Multimedia and Creative Technology and approved as to its style and contents.

DECLARATION

We hereby declare that, this project has been done by us under the supervision of **Dr. Shaikh Muhammad Allayear**, **Associate Professor and Head, Department of MCT,** Daffodil International University. We also declare that neither this project nor any part of this project has been submitted elsewhere for award of any degree or diploma.

Supervised by:

[illegible]

Dr. Shaikh Muhammad Allayear
Associate Professor and Head
Department of MCT
Daffodil International University

Submitted by:

Swarna kundu

Swarna Kundu
ID: 142-40-128
Department of MCT
Daffodil International University

Fahim

Fahim Ahmed
ID: 142-40-138
Department of MCT
Daffodil International University

ACKNOWLEDGEMENT

First, we express our heartiest thanks and gratefulness to almighty God for His divine blessing makes us possible to complete the final year project successfully.

We are really grateful and wish our profound our indebtedness to our supervisor, **Dr. Shaikh Muhammad Allayear**, **Associate Professor and Head,** Department of MCT Daffodil International University, Dhaka. Deep Knowledge & keen interest of our supervisor in the field of "*Game Design & Development* to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior draft and correcting them at all stage have made it possible to complete this project.

We would like to express our heartiest gratitude to Mr. Mirza Mohtashim Alam, Lecturer, Department of MCT, for his kind help to finish our project and to other faculty member and the staff of MCT department of Daffodil International University.

We would like to thank our entire course mate in Daffodil International University, who took part in this discuss while completing the course work.

Finally, we must acknowledge with due respect the constant support and patients of our parents.

ABSTRACT

For this project, we wanted to design a computer game in Unity. So we decided to design a Role-Playing game (RPG) where the objective is to collect three elemental stones, save the planet using the stones and get out of the game by finishing it. The player, who is a magician, has to collect three stones by defeating enemies in different levels and save the planet from being destroyed. The game is designed & developed in Microsoft Windows and written in C# using Unity Game Engine, Monodevelop and Microsoft Visual Studio. We have implemented many advanced programming functions in this project as taught in our Multimedia Authoring & Game Development course. As a result, we have created a 3D RPG game that we hope would be fun to play and enjoyable.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1: Introduction

In recent times, PC video games have become very popular and wide spread all around the world. Playing video games has become a way to know more about the world as they offer different opportunities for gamers to explore, practice, and re-enforce cultural and social identities as well as their own capabilities in terms of speed, reflex, calculation and strategy. And due to the potentials of video games, the gaming industry has grown from focused markets to mainstream that is currently worth billions of dollars and growing very fast.

1.1 Present State of Gaming Industry

In the rapidly growing sector of game design, the future of this sector has become hard to predict. Mid-level and independent developers are rising in the industry. Thanks to crowdfunding, technological advancements and marketing automation, game development has become cost efficient in 2015, resulting in the development and release of lots of high quality AAA game titles like Assassins Creed Origin, Battlefield 1, Halo Guardians, Call of Duty Black Ops III, Ghost Recon Future Soldier, GTA V etc.

Development of easy-to-use game engines like Unity 3D and professional game engines like Unreal Engine and Cry Engine becoming free for everyone to use and other development tools helped indie studios and smaller budget productions rise into the industry. Crowd funding initiatives like Kickstarter, Indiegogo and Fig pushed small studios to acquire enough funds to develop new games and digital distribution platforms like Steam, Origins, GOG, UPlay gave them online platform to publish their games for the gamers all over the world to download and play. This helped studios to reach potential clients without publishers and build a dedicated gamer base even before the game's release. The industry has gone through a remarkable growth that was worth of around $82 billion in 2017. [1]

In recent years, professional eSports competitions held by organizations like Electronic Sports League (ESL) and Major League Gaming (MLG) has brought gaming competition to a whole new level. ESL holds gaming competitions like League of Legends World Championship, DOTA 2 Major League, Call of Duty

World League and 9 other official Pro Leagues for Overwatch, Counter-Strike: Global Offensive, Rocket League, Gears of War, Guild Wars 2, Halo 5: Guardians, Hearthstone, Mortal Kombat X, and Tom Clancy's Rainbow Six Siege throughout the year. Top game publishers like Activision, Blizzard Entertainment, Electronic Arts and Epic Games also started their own gaming tournament on games developed by them to encourage gamers all over the world. Teams from all over the world competes with other teams in these tournaments to win the game and the prize. These competitions have become so popular in last few years that the Olympic Games are considering to include several eSports titles in the upcoming Olympics. [2]

1.2 Video Game Types

There are lots of different genres or categories of video games. Most famous genres of games are First Person Shooter (FPS), Third Person Shooter (TPS), Massively Multiplayer Online (MMO), Role Playing Game (RPG), Real Time Strategy (RTS), action, adventure, sports, simulation, horror, survival, etc. Each have their distinctive characteristics and fair share of games and loyal players base. Some games can be of two or three genres. The game we planned is a fantasy Role Playing Game (RPG) as the story takes place in a fantasy world and the player plays the game following the storyline. The concept of role playing games have been described in next section.

1.2.1 Role Playing Game

Role-playing games (RPG) are games where gamers assume the roles of game characters and collaboratively create stories. Most of the RPG games have their own world with different story and mechanics. RPG video games use similar settings, terminology and game mechanics of tabletop RPGs like Dungeons & Dragons. Players takes control of a game character, or sometimes several game avatars, called a party or guild, and achieves victory by completing objectives, quests or finishing the main storyline. Players roam around and explore the game worlds, while engaging in combat with other players or enemies, solving puzzles and completing objectives and quests. Characters grow in power and abilities, and characters customization done by the player is one of the key features of this genre. Player's strategic and tactical knowledge and reflex is rarely challenged in RPGs, except action role-playing games. The story of most RPGs puts the player in a mission of saving the world as the game world are often set in a fantasy or fictional universe. because of this, the players have powers and can do things that they cannot have or do in real life. Players can collect items and currencies and trade them for equipment, levels and ranks which is done by an inventory system. RPGs usually tries to offer dynamic and complex character interaction than other video game genres, which typically involves developers focus on scripted behavior and artificial intelligence of non-player characters (NPC). [3]

1.3 Report Overview

This section gives a complete overview of every chapters of this report to the reader.

Chapter 1: Introduction

This chapter discusses about PC gaming, video games and the industry that grew around gamers for all these years. The first section focuses on the present state of the gaming industry and how it has developed in last few years to the point that gaming has become national sports in some countries. Second section discusses about different types of video games, specifically Role-Playing Game (RPG), the category our game falls in to. The next chapter focuses on games similar to ours and some of their problems.

Chapter 2: Proposed Project and Software Requirements Specification of "Ethoreal Shift"

This chapter discusses about our game project, the story, requirements and our development tools and process. The first section focuses on our proposed project, project background and design choices we made for our game. Next couple of sections describes the scope of our game and work schedule of our game. The next sections focus on specific requirements of our game along with user story, details about game levels and our plans to deal with bugs and glitches of the game.

Chapter 3: Design and Implementation of “Ethoreal Shift”

In this chapter, we have discussed about game design, UX design and backend programming, game construction and platform integration. First and second section focuses on our game design elements like characters, enemies, items and level and environment design. In next sections, we have described UX design, backend programming, system features of this game and their details along with game construction, platform integration, key resources, required tools and the codes and scripts we have written for this game.

Chapter 4: Coding and Other Technical Aspects

This chapter gets down to the technical aspects of our game. First section of this chapter is about the scripts and the analysis of one of the scripts we have written for our game. The next section describes about the functional and beta tests we performed on our game and how we fixed the bugs and problems we faced.

Chapter 5: User Manual of “Ethoreal Shift”

This chapter describes about playing procedures and final look of our game. First section is about how to play the game and what to do to complete objectives, quests and the game. The next chapter consists of screenshots of our game to show the final look of the game.

Chapter 6: Conclusion

This chapter concludes this report of our game “Ethoreal Shift”. In this chapter, we have discussed about the obstacles we have faced while working on this project, our achievements and our future development plan about this game. The future development plan section contains our original plan of this game, that we couldn’t complete because of the limited time we had.

At the end of this document, we have listed the references and codes we used in this report and in our game respectively.

1.4 Related Works

Role playing games (RPG) aren’t new and have a good history in gaming. Earliest examples of popular RPG video games are Dragon Stomper (1982), Bokosuka Wars (1983), Dragon Warrior (1986), Legend of Zelda, Dragon Slayer, Final Fantasy, Phantasy Star, Diablo. Dungeon Master etc. Legends of Zelda, Final Fantasy, Pokémon, Kingdom Hearts, Dragon Age, Fallout, Mass Effect, Elders Scrolls, World of Warcraft and Warhammer are among the most famous and longest running RPG franchises, each of them having their own loyal player base and get new game titles frequently. [3] Kritika Online (2017), Sword Art Online Fatal Bullet (2018), Destiny (2015), Fallout 4 (2015), Monster Hunter: World (2018), Nier: Automata (2017), Pillars of Eternity (2017) are some of the best RPG titles that came out in recent years. Most of them focuses on their unique open worlds and storylines. Some of the recent RPGs are Massively Multiplayer Online (MMO) and have online battle arenas that can be played with other players over local network or the internet.

These games are well-developed and fun to play. But most of them have paid items, weapon, costumes and downloadable contents (DLC), which is actually not fair considering the fact that the players already bought the game. There are some RPG games like Kritika Online and Smite that are free-to-play. But they also have exclusive paid contents, which is reasonable for free games like this, but not on other games that needs to be bought in order to play. These game also have massive open worlds to keep the players engaged in battles. But because of this, the size of game files become huge. Size of high quality open world RPGs ranges from 6-7 GB to over 40-50 GB. For example, just the core files of Phantasy Star Online 2 is well over 50 GB, downloading this game from a digital distribution platform will take from 20-30 hours to over a week, depending on region and internet speed of the player. This kind of problems discourages lots of gamers and stops them from playing those games.

RPG video games also tend to have pretty complex combat system, where the player has to learn lots of new things to master the game completely. It sometimes takes away the fun of the game. So we decided make our game simple in combat and light weight in terms of file size. Details of our proposed project is described in the next section.

Chapter 2: Proposed Project and Software Requirements Specification of “Ethoreal Shift”

This chapter covers in-depth details about our proposed project and the initial requirements specification of our game “Ethoreal Shift”. The latter incorporates the definition of this report with basic descriptions, specified requirements, and evaluation of models. It also contains modification management of this requirement specification in case of any modification is done.

2.1 Our Proposed Project: Ethoreal Shift

For our Final Project Phase-II course, we were instructed to choose a project like animation, short film or game and complete them in this semester. Our team decided to develop a game project to work better with graphics, scripting, 3-D modeling, texturing, animation and game design. Because, in a game project we have the opportunity to implement everything we learned in last four years from graphics design, story and script writing, storyboarding, 3-D modeling, animation, texturing, lighting and many other things in the environment of the game, the story, characters, game play, the artworks and so on.

Our proposed project is a first person Role Playing game (RPG) named “Ethoreal Shift” with different quests and levels to play. When we developed a concept and finalized the story based on the concept, we realized that using elements of Role-playing Game (RPG) genre would make the game as we want. So we developed the game as a RPG that is played from first person (Player) perspective. The main character of our game is a boy who gets teleported into the game he was playing on his computer. He becomes the avatar of the main character in the game who is a magician. He cannot return to his own world unless he finishes the game. The main mission of the gamer is to use his powers to save the world. The game is set in a fantasy world. There are several levels and in each level, the gamer must complete the quests or objectives and collect the stones that he needs to use at the last level.

What sets our game apart from other role-playing games is our unique storyline, combat system, game mechanics and the game world itself. Our game emphasizes on storytelling, action and adventures. Unlike other RPGs where the players fight with enemies, fight with other players and complete linear quests, our game encourages the player to explore the world of the game. In the first two levels, the player has to explore the village and the city to complete his objectives. Even in the quests where the player has to defeat enemies, he will have to explore the whole level to find all of the enemies. We have designed a simple yet intuitive combat system for this game that is easy to learn, but hard to master.

2.1.1 Level and Character Design Choices

As the game is set in a fantasy world, we designed the environments based on old house and buildings to complement the story. Every aspect of the environment is designed logically either to convey the world setting, or to serve a function of the environment itself. We did reuse some 3D models from our previous work as they fitted our needs, but we designed everything else ourselves. The characters of the game were designed to match the environment and story of the game. Main player character designed to look as a hero, so that when the gamers see the character, they can connect to the main character easily and can feel that this character can save the world. Character design of the enemies were based on the roles they served in the story, but the design of the non-player characters was based on the environment and the purpose they served in the background. Most of the NPCs in first two levels (Sylvista and Daryas City) are adventurers, who came into the village and the city to search for work or to meet others. We didn’t put any NPCs in the quest levels, because those areas were exclusive to the role of the enemies, and the adventurer NPCs didn’t have any reason to be in those areas. We have described more about game design in the third chapter of this document further below.

2.2 Scope of Our Game

This document discusses all necessary requirements of this project in details. The basis of this groundwork is to provide an implicit image with both structured and unstructured information of our project "Ethoreal Shift". In this game, the player will advance through levels, which require precise and strategic manipulation of the player's creative and logical thinking. The episodic structure of the game simplifies the momentum of the game's story. We exhibit the strategic implementation of script, inputs and display (output) as we are focusing mainly on concept of the story, game levels, objects in the game, character animation, graphics, scripts and game engine features.

2.3 Project Scheduling

Start Date	End Date	Project States and Objectives
January 5	January 25	Project Proposal, meeting with supervisor about our idea
January 26	February 10	Planning, elaborating game story, levels planning and completing necessary preparations
February 16	February 28	Choose tools, environment, finished 3-D modeling and texturing of game objects
March 1	March 20	Start level designing and model implementation, rethinking game plot
March 21	April 8	Start programming, developing game logic, testing and enhancement
April 9	April 14	Project report writing and conducting tests on the constructed levels
April 16	Displaying project progress at our Predefense	
April 17	May 26	Perform fine-tuning on the game logics and

		fix bugs
May 28	June 19	Conducting final tests on the game, fixing bugs and compile executable files
July 5	Reviewing project report from our project supervisor	
July 6	July 20	Perform corrections on the report
July 26	Project final showcase and submission	

2.4 Software Requirement Specification of “Ethoreal Shift”

In this section, the specification of this documentation is described. It describes the rules of the document, document purview and also presents a suggestion for the readers of the document. [4]

2.4.1 Purpose of this Chapter

This Software Requirements Specification (SRS) part is meant to provide a comprehensive overview of our Project “Ethoreal Shift” incorporating the action flow, the story and game user interface. The SRS document describes all aspects of the project that shows the references, manners and relevance of their execution.

2.4.2 Intended Audience and Reading Suggestions

The SRS document provides us a method to establish the game’s commitment to our original vision. For a summery of the report and the project, please refer to Overall Description. For a detailed documentation of the game-play elements and their interaction with the player, refer to System Features. Readers curious about the game-play user interface and navigation between the front-end options should read External Interface Requirements. Specialized guidelines to which the developer team will work on the project are described in details in Other Nonfunctional Requirements. The design and development timeline will be preserved in the Key Milestones. [5]

2.4.3 Scope of this Document

This Software Requirements Specification (SRS) explains the functional and nonfunctional prerequisites for the project. The intension of this section is to deliver a close enough image for both structured and unstructured information of this game project. The concept of “Ethoreal Shift” was perceived by both of the members of our team, as having a predicted development time lengthy than the duration of this semester. Our team Desires to work on the project until it’s complete. We will continue to develop the game until we find it more than competent for open-source distribution.

2.5 General Description

This segment contains the viewpoint of our project and the structural framework it demands. It defines the QFD (Quality Function Deployment) of this game and also the story of it.

2.5.1 Product and Business Perspective of the Game

Design and development of a game like this from scratch is an outstanding challenge for any game studio. It demands substantial amount of investments in terms of cost and effort and also requires the client’s participation, knowledge and understanding about the expected marketplace (example: Steam, Origin). This report describes overall product perspective.

2.5.2 System Environment

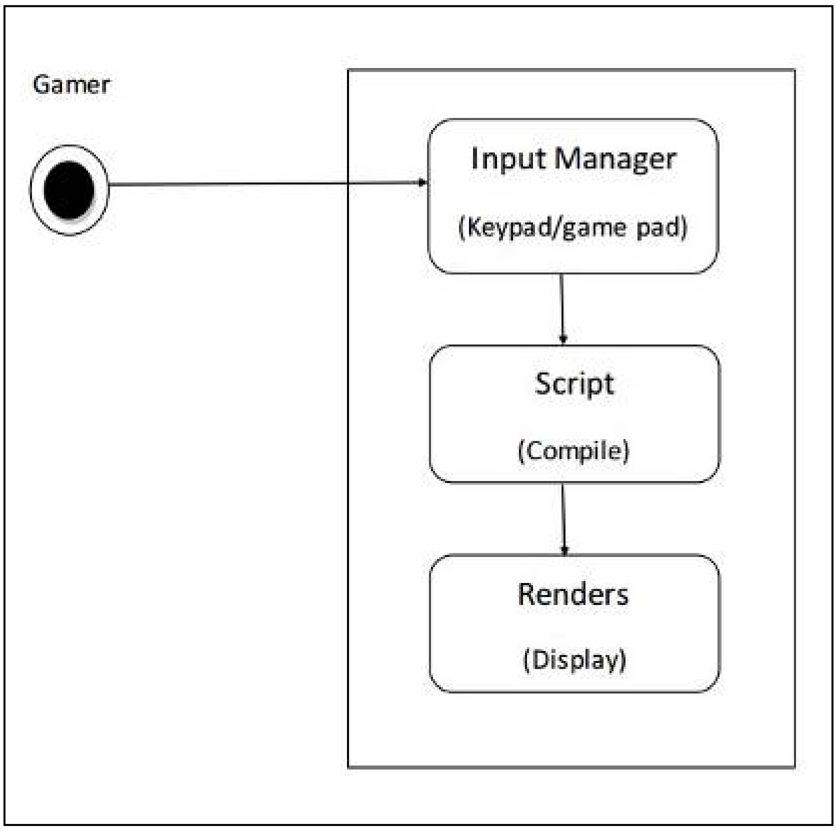


Fig 2.5.1: Input Graph

Player can interact and communicate with the system by giving input to the system. System commands those inputs to the related script, if any change happens like the value of certain variable is changed, the script sends required instructions to render the objects to display (Player changing his position).

2.5.3 Quality Function Deployment of "Ethoreal Shift"

Quality Function Deployment is a procedure that expresses the requirements of the client into structured requirements for software or game. It focuses on maximizing client satisfaction throughout the game development process. The following requirements concerning our project have been identified by a QFD.

- General Requirements.
- Expected Requirements.

General Requirements

General requirements include goals and objectives which have been specified in the meeting with the relevant peoples concerning the project. general requirements of our project are: -

1. Efficient, user friendly and productive system.

2. Least amount of effort in development and future support.

3. Achievability of desired requirements within targeted PC configuration.

4. Easy to play.

5. Developed with professional approach.

6. Well-structured coding, creative and professional thinking.

Expected Requirements

These requirements are inherent to the system and can be so rudimentary that the gamer and relevant people does not specify them comprehensively. Unable to implement them might become a cause of disappointment.

1. Develop the whole game as planned in the scheduled time.
2. Minimum hardware prerequisites, which is suitable for the game.
3. Design and develop the whole game in a well-organized style.

2.5.4 User Story

As we have specified above, “Ethoreal Shift” is a first person Role Playing game (RPG). We desire to develop it as a cross-platform game which would be supported by PC, console, android phone, IOS and also other available platforms. But as we have very limited time to complete the game, it has only been developed for PC (Microsoft Windows). So the gamer can use any windows pc to run the game. After running the game, the UX view of the game will appear on the screen. Then the gamer can select “Start Game” from the “Main Menu” and start playing the game. After starting the game, a cut scene would be shown to the player to give him an idea about the story. When the cut scene ends, player will be given a briefing about the backstory and his primary mission on the screen. He may also interact with “Pause Menu” by pressing “Escape”. If he loses he can replay the level as the level will automatically restart after the player’s death. Or he can exit game by pressing “Quit” in the “Pause Menu”.

The story behind the game is about a boy who gets teleported into a game he was playing in his pc. He becomes the avatar of the main character of the game, a magician who is the descendent of another great magician who saved the world from destruction long time ago. The objective of the gamer is to collect three elemental stones by completing three different quests, which he will use at the last level to save the planet from being destroyed by the rays of Super Red Moon. As mentioned earlier, the gamer will find tips in different steps which will help him to go to the finishing stage. In first level, the player has to search for information about his mission in Sylvista village. He finds a merchant of magic items in the village. The merchant tells him to find scrolls which are spread around the village. There is an Inventory System that stores everything the player collects. After collecting all the scrolls in the village, the player has to use those scrolls to teleport to Daryas city.

In that city, player needs to find more scrolls hidden around the city to find the location of the stones. Then he goes to the jungle to collect the Dryad stone. The player has to defeat every wood cutter in the forest to get the stone. The player uses fire to defeat his enemies. He has two attacks, first one is fireball that player throws at his enemies. Second one is a charged attack, a burst of flame, where he keeps throwing flames at his enemies. But he can use that charged attack after an interval as he needs to regain his magical energy. Then, he returns to the city with the stone. He has to go to Sylvista village again, to the merchant in order to obtain a magic artifact that will allow him to use explosion spell. Then he goes to the machine site with the artifact to get the Zephyrus stone, where a machine is polluting the air. The area is guarded by Paladin knights equipped with sword and shield. The player has to defeat every knight in the area and cast an explosion spell at the building that is producing smoke using the machine.

After collecting the Zephyrus stone, he goes to his last quest to get the last stone, Darya. Last level is a factory site with a forest at the front and is guarded by both woodcutters and Paladin knights. After defeating all the enemies, the player casts the explosion spell on the factory and the Darya stone appears before him. After collecting all 3 stones, he goes to a cave, a sacred waterfall where he can use the power of the stones to save the world. He puts the three stones on a magic table with ancient glyphs, the table unleashes the power of the stones creating a barrier around the planet to protect it from the Super Blood Moon and the player is teleported back to his own world in his room.

2.5.5 Game Levels

Levels	Description
Level 1 - Sylvista Village	This is the first level of the game. The player wakes up in the middle of a small forest in a village named Sylvista. He explores the village to find a magic shop, whose owner tells the player to find scrolls hidden throughout the village.
Level 2 - Daryas City	The player gets teleported to Daryas city. He again has to explore the city and collect scrolls hidden in the city.
Level 3 - The Forest	He goes to the forest to defeat the woodcutters in order to save the forest. After defeating all the enemies, he gets the Dryad stone.
Level 4 - Sylvista Village	He returns to Sylvista to get a magic artifact, which will allow him to use explosion spell.
Level 5 - Machine Site	The player fights Paladin Knights who are guarding a machine that is polluting the air. After defeating all knights, he casts explosion spell on the building the machine is in and destroys it. Then he gets the Zephyrus stone.
Level 6 - Factory Site	The player fights the woodcutters and Paladin Knights in the area to destroy a factory that is polluting water. After defeating all enemies, he casts explosion spell on the factory building and destroys it. Then he gets the Daryas stone.
Level 7 - Sacred Waterfall	The player goes at the Sacred Waterfall, puts the three stones on an ancient magical table that unleashes the powers of the stone and saves the world. Then he returns home.

2.6 Specific Requirements

This segment discusses the project external requirements of the game and also signifies the user attributes for this project.

2.6.1 External Interface Requirements of the Game

2.6.1.1 User Interfaces

Every game must have a main menu, so the gamers can easily start the game. The menu is an important element for creating the SRS document. In this SRS document section, we have used snapshots of the menu in the user manual subsection. Those snapshots are established on the menu of this game.

2.6.1.2 Hardware Interfaces

“Ethoreal Shift” is a pc game designed specifically for the Microsoft Windows and can be played on all mid to high-end desktops and laptops. Game data is stored locally on the game engine elements. Windows is graphically and universally adaptable with both 2D and 3D graphics library based on OpenGL ES 2.0, 3.0 and different DirectX specifications as well as hardware configuration, pixel format conversion, accelerated 3D graphics and scaling.

2.6.1.3 Software Interface

“Ethoreal Shift” has been designed and developed using a series of 3D modelling, game design and development software.

Software and tools used in this project:

- ❖ Unity3D 5.6 and 2018.1 (Game Engine)
- ❖ Autodesk Maya 2016 (3D modeling and animation)
- ❖ Adobe Fuse CC *Beta* (Character Generation)
- ❖ Adobe Photoshop CC 2014 (Creating Texture)
- ❖ Substance Painter 2 (Model Texturing)
- ❖ Substance Bitmap2Material (Image to Material Map Conversion)
- ❖ Microsoft Visual Studio 2017 Community Edition (Programming)
- ❖ Microsoft Visual Studio Code (Programming)
- ❖ Adobe After Effects (Editing Cut scene)

2.6.2 User Characteristics for the Game

The game can be played by only one player at a time and the user can interact with the game in different ways. The player is the one who can communicate with the system by playing the game. And the player can be any individual. The initial requirement is, that the player should read the playing mechanism provided by the developers (us).

2.7 Requirement Change Management of Our System

We hope to deliver a complete and fully working game that works according to the guidelines stated in the SRS document. But, as the game will be released for multiple Windows versions (i.e. the different desktops and laptops running different Windows version), updates will likely be necessary to address issues like fixing bugs that made into final version of the game, compatibility patches and expansions of the content. If the gamers experience any issues, bugs or want to give feedback, they can communicate with the developers through our official support email address which will be specified later. For managing the changes, we are releasing versions of this document. This one is version 1.1.

2.8 Bugs and Glitches

The players would be able to communicate with us, the developers through the support email. This is where they would notify us of any bugs or glitches they have detected and if they have any feeling that the game is not working properly. Issues of gamers or comments would also be submitted by support email. We will monitor this email continuously in order to answer all questions and fix any problems that have surfaced.

2.9 Patches

As the recent Windows 10 is frequently updated and the vast amount of combination of pc configuration exists, the game would also need to be updated frequently. We would constantly be making changes in order to fix any compatibility issues that may arise. These modifications, bug or glitches fixes will be updated by these patches.

Chapter 3: Design and Implementation of "Ethoreal Shift"

This chapter discusses about the design phases of this project, the system characteristics and also the implementation of system features.

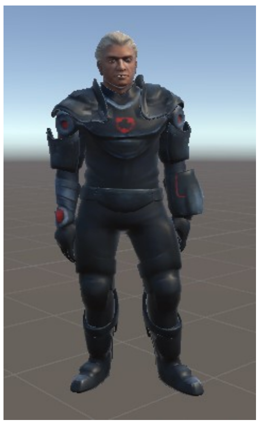
3.1 Game Design

For this game, we had to design lots of characters for the Player, enemies and town-folks. We also design several levels and environments where the game's story will take place. [6]

3.1.1 Characters

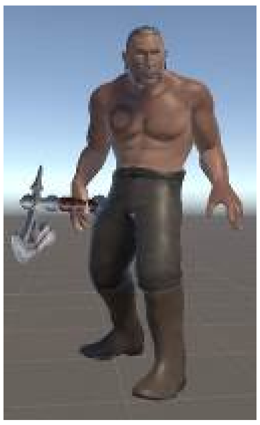
3.1.1.1 The Player

In first person view, this capsule works as the player body. Since the player won't be visible, using this capsule as player helps to take some load off the computer.

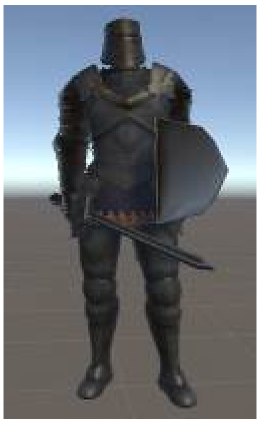


This is the Player character model, only used in cut-scenes of the game.

3.1.1.2 Enemies



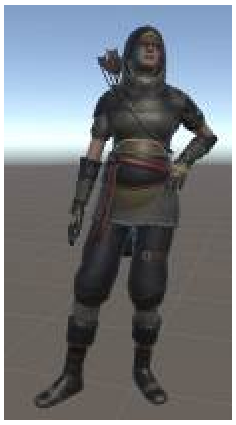
This is the wood cutter, first enemy in the game. They are seen in every level working or talking, but the player only fights with them in Quest 1 and 3. They stays in idle when player is not around. But they keep moving toward the player when the player is in range and when they get close to the player, they start attacking with the axe.



This is the Paladin Knight, second enemy in the game and more powerful than the woodcutter. They are also seen in every level talking or guarding the Citadel gate, but player only fights with them in Quest 2 and 3. As they have armor and shield, they have more health than the woodcutter.

3.1.1.3 Non-Player Characters (NPC)

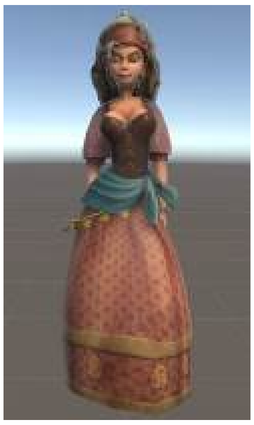
These are non-player characters or local peoples who are either talking to others or walking around. They are created to liven up the environment.



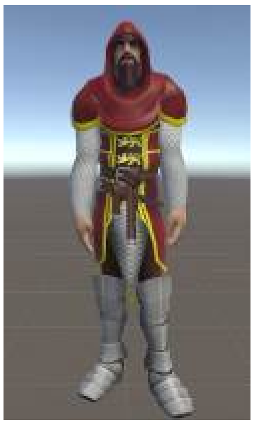
Archer and adventurer who is seen talking with other peoples or walking around in Sylvista and Daryas city.

Another archer and adventurer who is seen talking with other peoples or walking around in Sylvista and Daryas city.

Local girl of Asian ancestry seen walking around or talking with other peoples in Sylvista and Daryas city.



Local women and also a merchant who is seen talking with other peoples, walking around or running magic shop in Sylvista and Daryas city.



Guardian knight of the great king who is seen talking with other peoples or walking around in Sylvista and Daryas city. (We wanted to use him as an enemy, but he seemed more like a noble knight. So we kept him as a local)

Just an old man, seen talking with other peoples or walking around in Sylvista and Daryas city.

3.1.1.4 Items

Magic health potion to heal the player when injured.

Magic scroll that the player collects in Sylvista and Daryas city.

3.1.1.5 Elemental Stones

The Dryad stone representing forest goddess in greek mythology. It is given to player in Quest 1 after the player defeats every woodcutter in the forest.

The Zephyrus stone representing the god of west wind in greek mythology. It is given to player in Quest 2 when he defeats all paladin knights and destroys the machine room.

The Darya stone representing water in greek. It is given to the player in Quest 3 when he defeats all wood cutters and paladin knights and destroys the building polluting the water.

3.2 Level and Environment Design

3.2.1 Sylvista Village

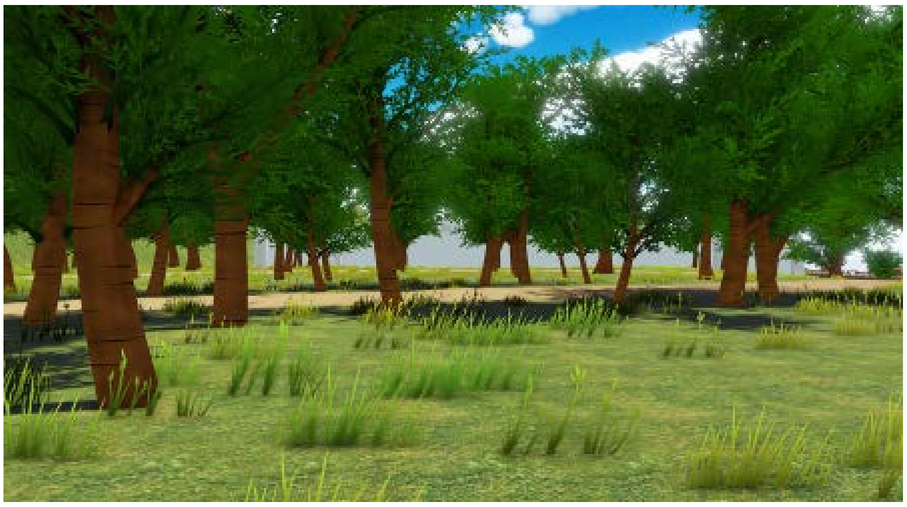


Fig 3.2.1: Starting Point

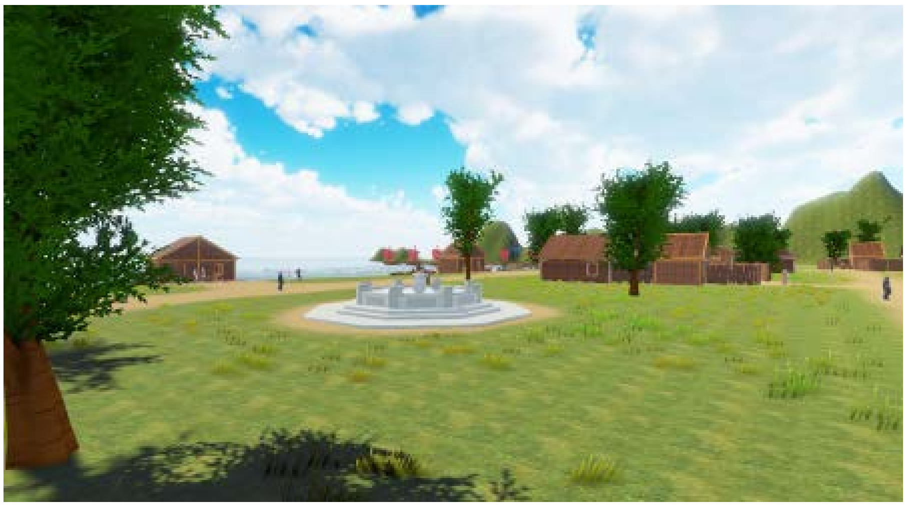


Fig 3.2.2: Teleportation Point

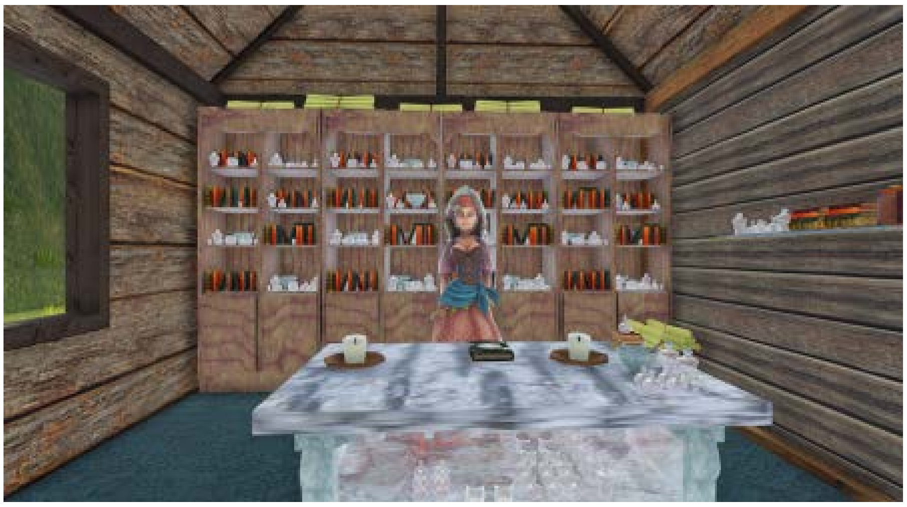


Fig 3.2.3: Magic Item Shop

3.2.2 Daryas City

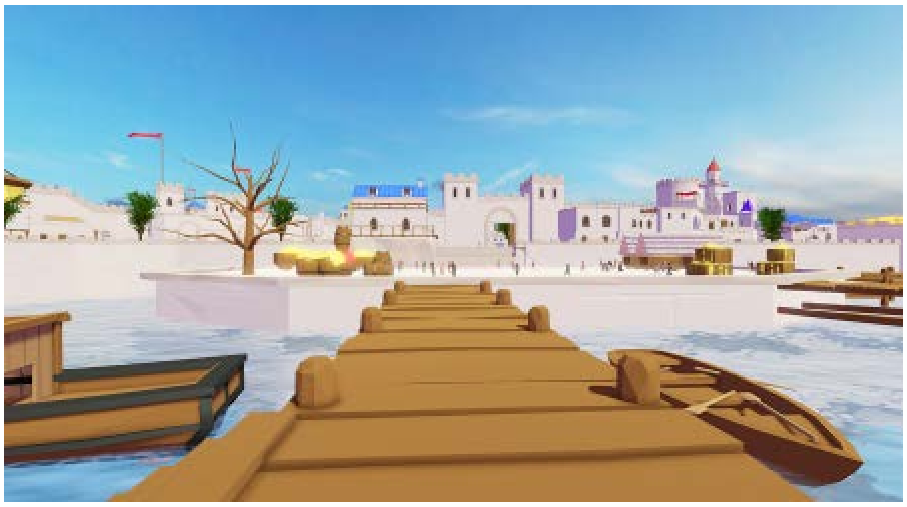


Fig 3.2.4: Daryas City Port

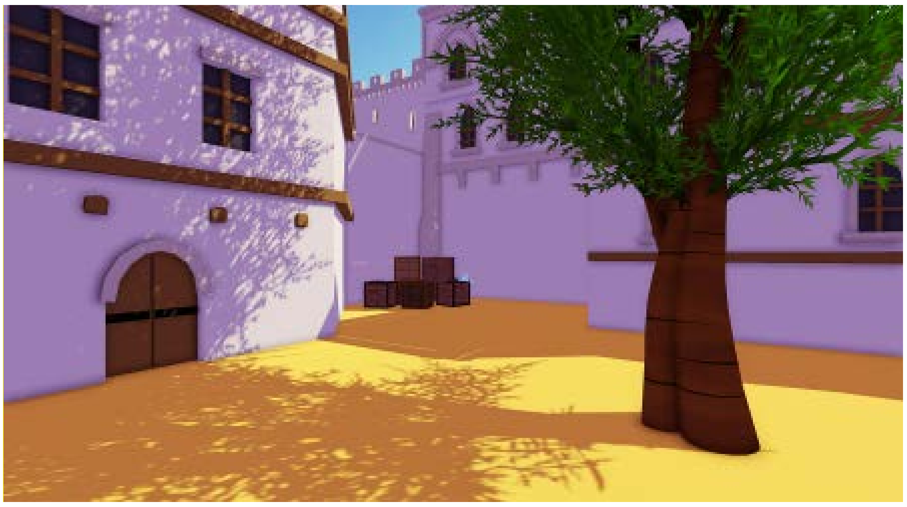


Fig 3.2.5: Scrolls

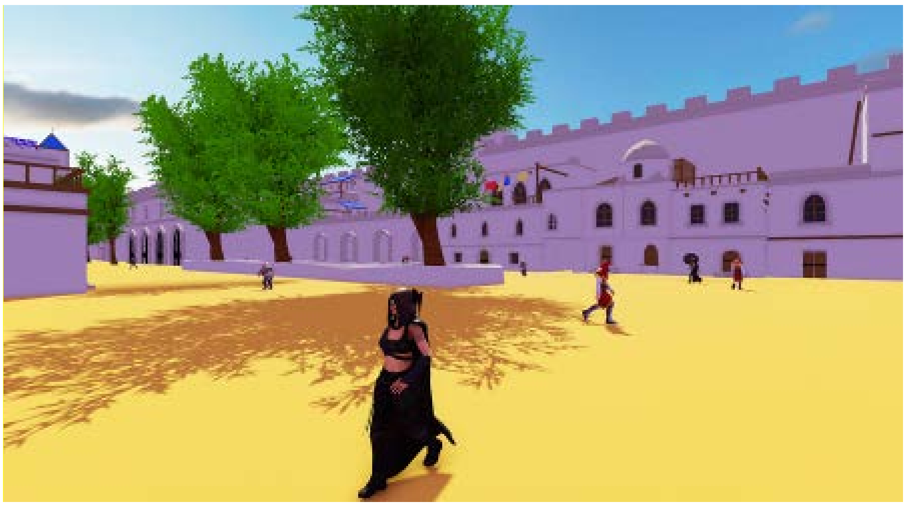


Fig 3.2.6: Adventurers Wandering in the City

3.2.3 Quest 1 – The Jungle

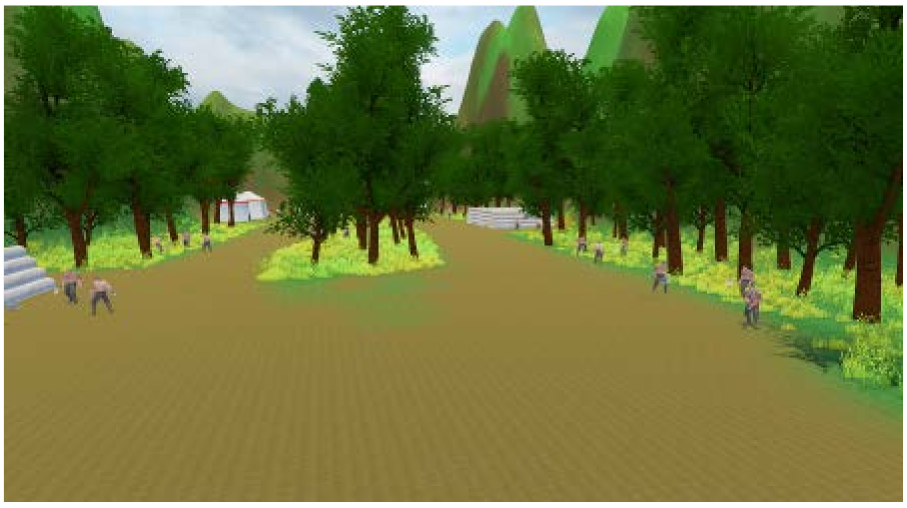


Fig 3.2.7: The Jungle

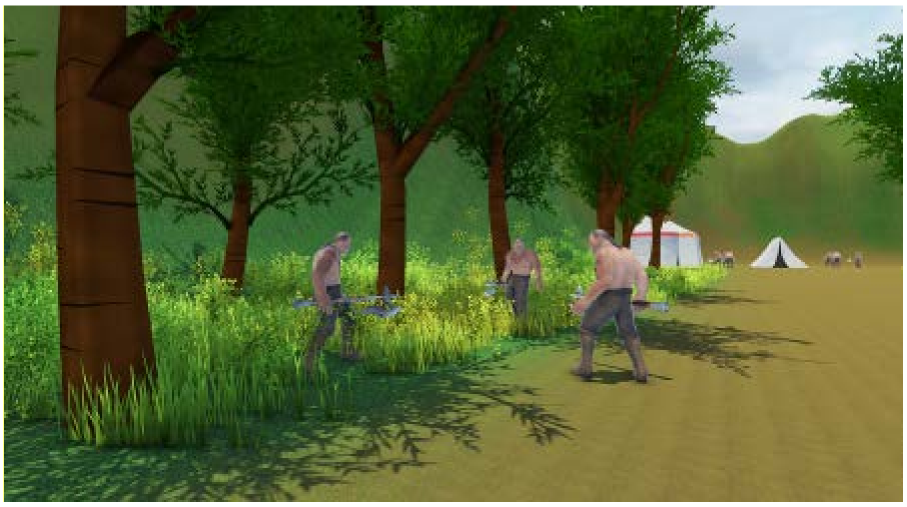


Fig 3.2.8: Woodcutters Standing Idle

Fig 3.2.9: Enemies in the Jungle

3.2.4 Quest 1 – The Forest

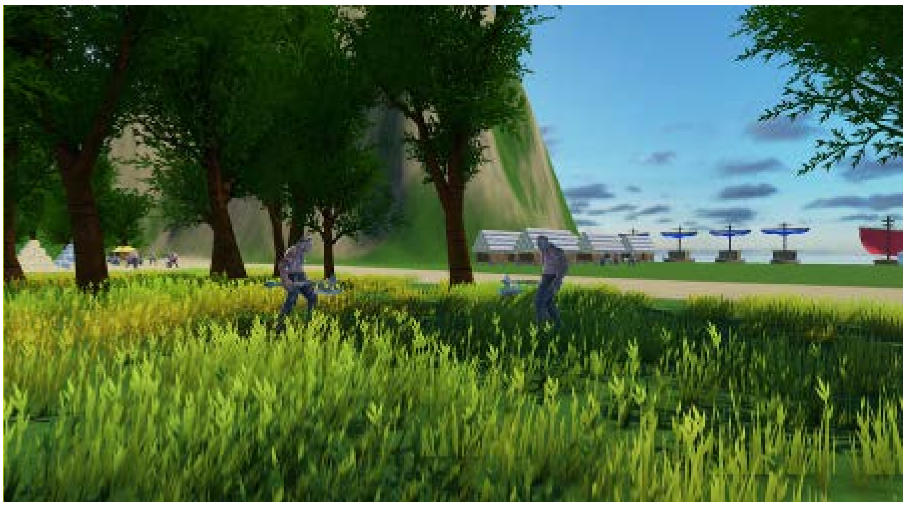


Fig 3.2.10: The Forest Port

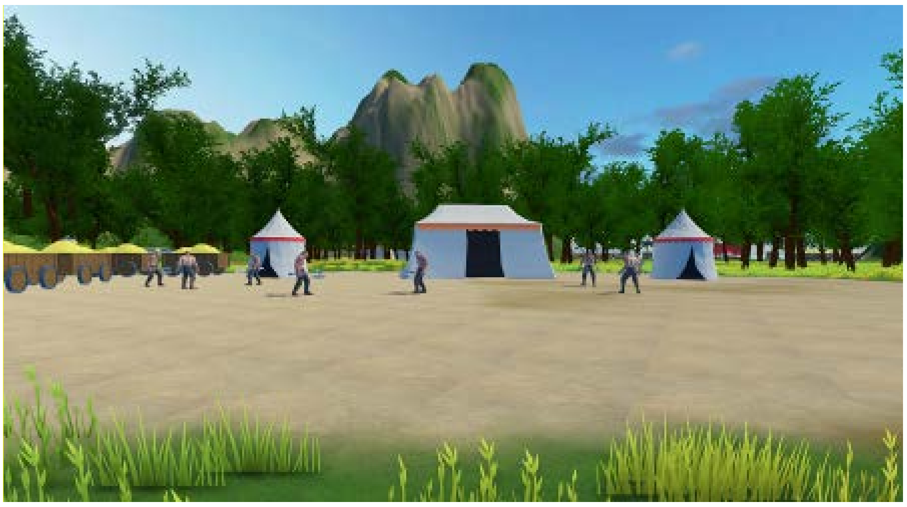


Fig 3.2.11: Woodcutter’s Tents

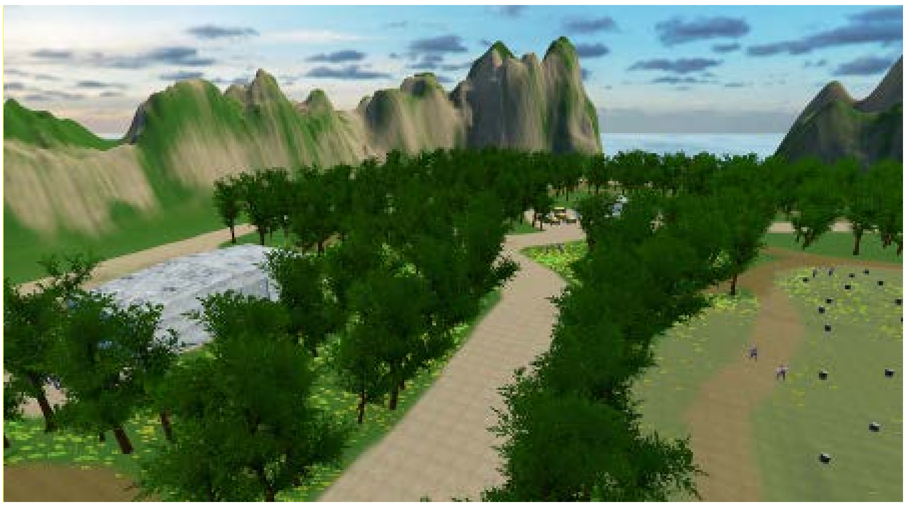


Fig 3.2.12: The Forest

3.2.5 Quest 2 - The Machine Site

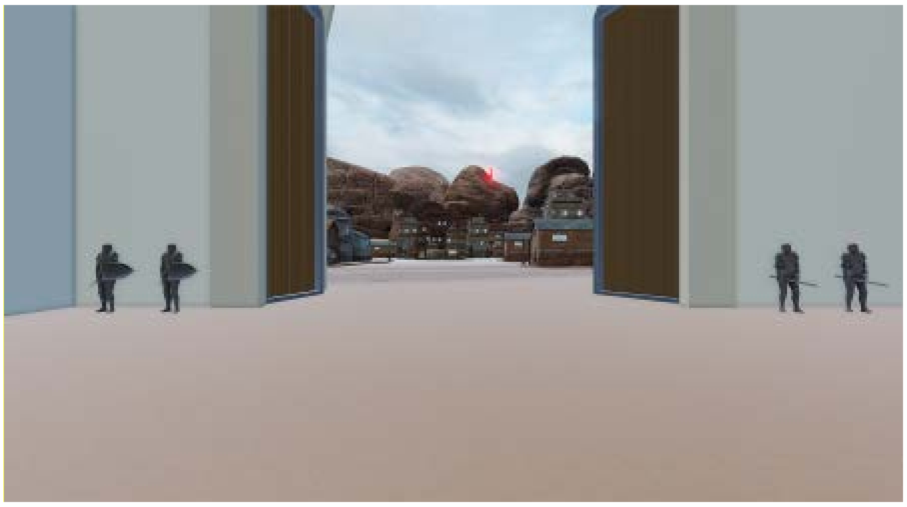


Fig 3.2.13: Gate of the Machine Site

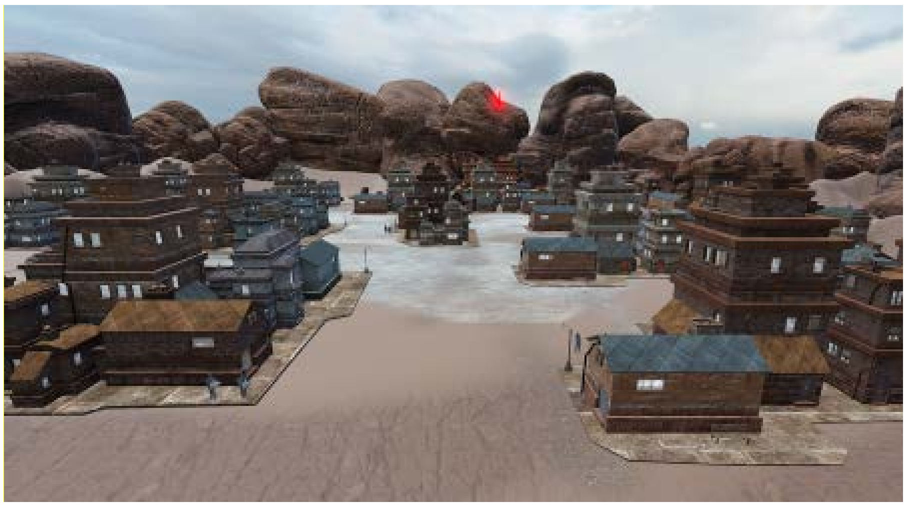


Fig 3.2.14: The Machine Site

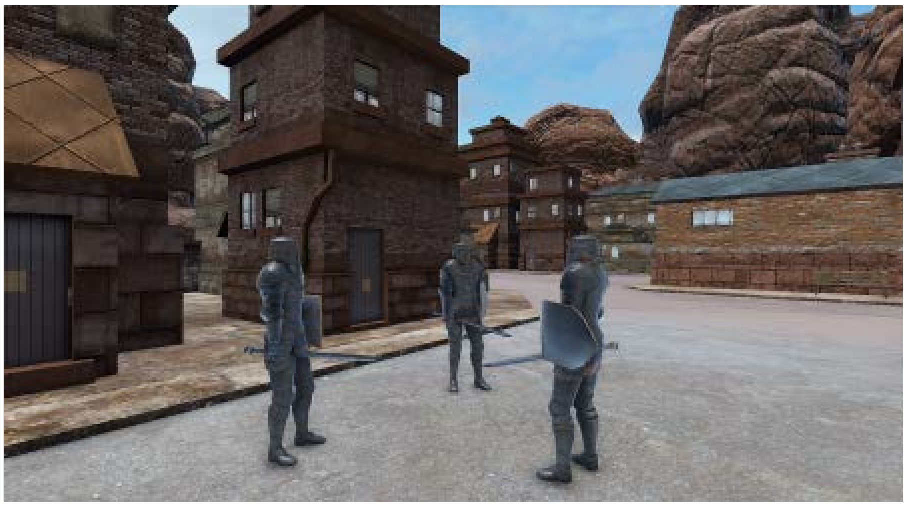


Fig 3.2.15: Paladin Knights in the area

3.2.6 Quest 3 – The Factory Site

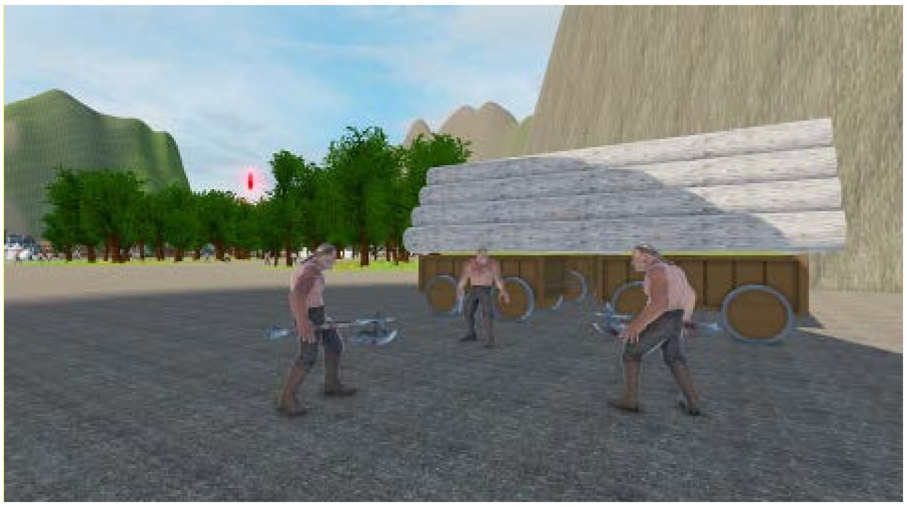


Fig 3.2.16: Woodcutters Standing around

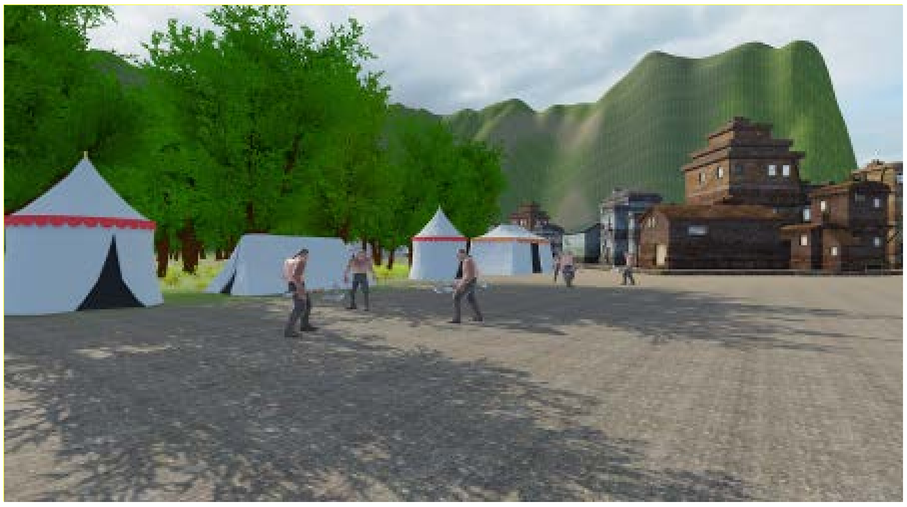


Fig 3.2.17: Woodcutter's Tents

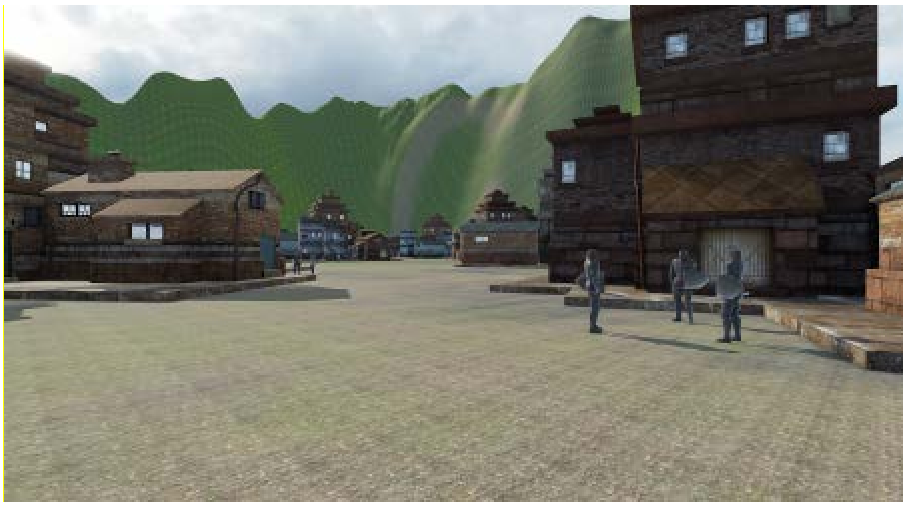


Fig 3.2.18: Paladin Knights Guarding the Factory

3.2.7 The Sacred Waterfall

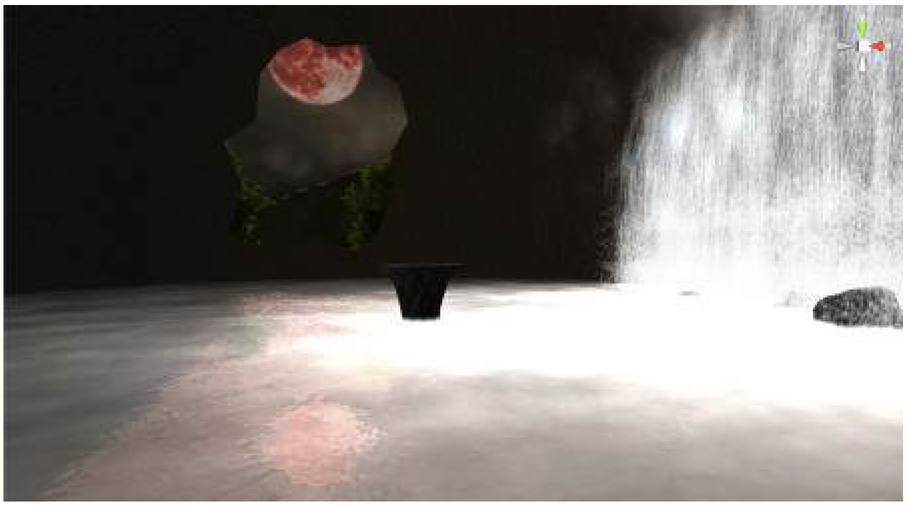


Fig 3.2.19: The Cave of Sacred Waterfall

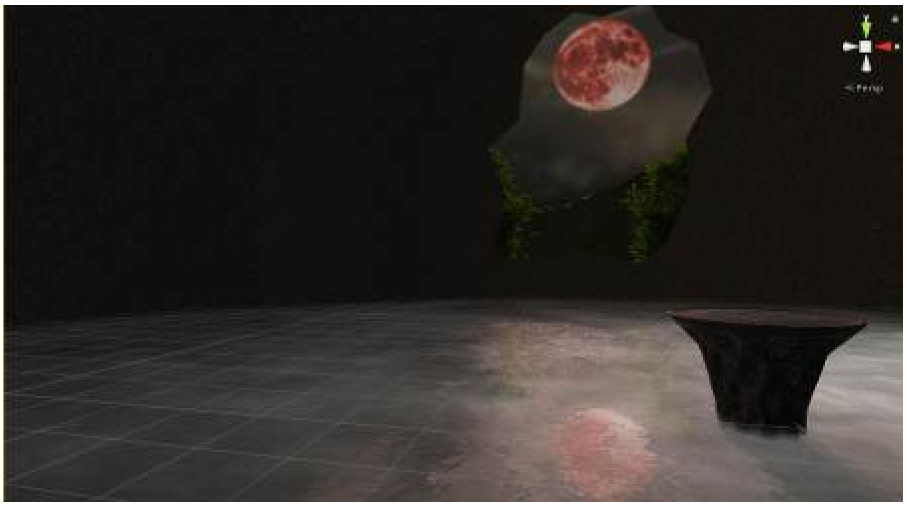


Fig 3.2.20: Super Red Moon

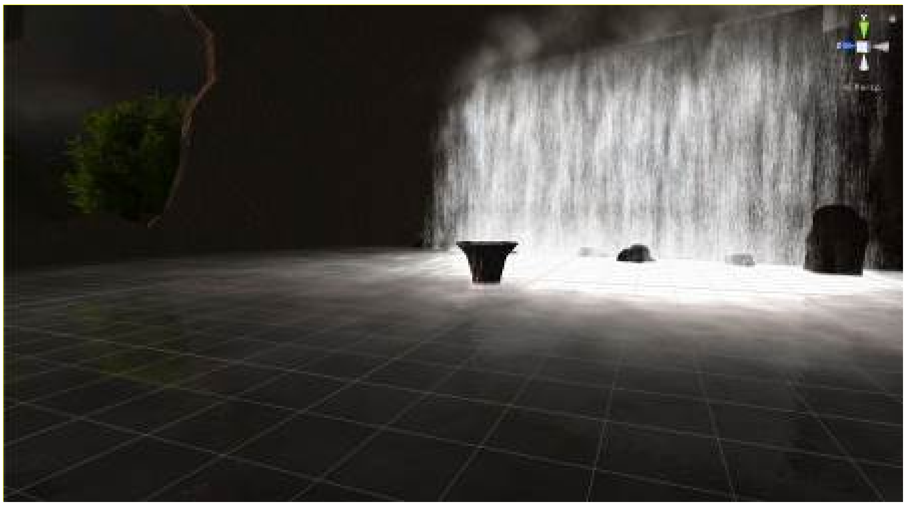


Fig 3.2.21: The Sacred Waterfall

3.3 Project Design Terms

For every project, two primary terms of design are very important. They are:

- User Experience (UX)
- Backend Programming

3.3.1 User Experience (UX)

In order to avoid needless features, make the design report simple and to make the gamer’s interaction with the game easy and efficient, UX design is must. User experience design (UXD or UED) is any impression of an individual's experience with a specified system, along with the interface, design, graphics and manual interaction. User Experience Design completely incorporates conventional Human-Computer Interaction (HCI) concepts and design, and spreads it through evaluating all attributes of a product or service as comprehended by users. UX represents primarily suitable approach to feasibility, functionality and HCI. UX describes user experience as “an individual’s perceptions and acknowledges that outcome from the use or predicted use of a system, product or service”.

3.3.2 Backend Programming

The "back end" is the code or script sustaining that front end. In short, front end of a software or game is what we see (i.e. the user interface) and back end is the engine that we do not see. For effective execution and to expand user acceptance, both of them are very important in game industry.

3.4 System Features of Our Game

- Title Screen
- Main Menu
- Pause Menu
- Health Meter
- Magical Power Meter
- Attack Interval Timer
- Inventory System
- Easy Control
- Immersive Action
- Quests
- Dialogue System
- Exit Point

3.4.1 Title Screen

3.4.1.1 Description and Priority

The title screen is the screen the player will see every time upon starting the game. Through this interface, the player can choose to start the game or exit the game. Since the title screen is the “hub” for all activities in the project, it must be included.

3.4.1.2 Response Sequences

Step 1: The player launches the game from their device.

Step 2: The start screen loads and appears, prompting the player with two buttons: “Play Game” and “Exit”.

Step 3: The player presses one of the buttons, triggering its respective function.

3.4.1.3 Functional Requirements

REQ-1: The title screen must load and appear every time the game is launched.

REQ-2: If the player quits the game during any stage of a level, they must be returned to the title screen.

REQ-3: If the player presses the exit button, the game will end and return the player to the computer’s regular interface.

REQ-4: If the player completes the game, the game will end and return the player to the title screen.

3.4.2 Pause Menu

3.4.2.1 Description and Priority

The player should be able to pause anytime during gameplay, and this screen fulfills that requirement. The pause menu also allows the player to navigate between gameplay and title screens. Player can also change level of music and sound effects in this menu.

3.4.2.2 Response Sequences

Step 1: The player presses the Escape button on the keyboard.

Step 2: The level pauses, drawing up the pause menu which prompts the player with two options: “Resume Game” and “Exit Game.”

Step 3: The player presses one of the buttons, triggering its respective function.

3.4.2.3 Functional Requirements

REQ-1: The “Resume Game” option must continue the game without any change to the character’s vector or the state of the level from the moment of the pause action.

REQ-2: The “Quit Game” option must close the game and return to the desktop screen.

3.4.3 Dialogue

3.4.3.1 Descriptions and Priority

Dialogue is a method by which the player will be briefed about his next missions and objectives. The player is guided to his next objectives using dialogue with the silent protagonist, providing context and narrative. While this feature is secondary in importance to the primary game mechanics, it is an important aspect of the game's atmosphere and it helps to heighten the player's connection to the experience.

3.4.3.2 Response Sequences

Step 1: The player goes to the Merchants office and presses the specified button.

Step 2: Dialogue is triggered and a text box pops up.

Step 3: To dismiss text boxes or continue reading multiple-page text boxes, the player clicks on Continue button.

3.4.3.3 Functional Requirements

REQ-1: Dialogue should not pause the game to prevent player disorientation.

REQ-2: Text boxes should be brief and placed away from UI components so as not to interfere with game-play.

REQ-3: The text must be readable from any screen.

3.4.4 Exit Point

3.4.4.1 Descriptions and Priority

Exit point is the finishing place of the levels. The player needs to complete the objectives and mission given to him to proceed to the next level or quest.

3.4.4.2 Response Sequences

Step 1: The player completes the objectives or mission.

Step 2: Player goes to the exit point.

Step 3: The Player is teleported to the next level.

3.4.4.3 Functional Requirements

REQ-1: Objectives must be completed to finish the level.

REQ-2: Exit point must be in reach of the player.

3.5 Assumptions and Dependencies

The target platform of our game project will be the Microsoft Windows devices. However, Unity will be responsible for both the construction of the game and its integration within the Windows framework.

3.5.1 Construction of the Game

Unity includes many built-in components which will expedite the process of game development immensely. These include:

- Physics Engine
- Collision Detection and Handling
- Input Recognition
- Object Creation and Transform Manipulation (position and rotation of game objects)
- Scene Integration (transition of one level to the next)
- Model Attachment (representing game objects with 3D models from external programs)

3.5.2 Integration with Windows

Unity's build settings simplify the process of compiling our game for the Windows platform. After completing the project, or during any intermediary step for testing, we can select Windows from the list of options, build the project, and play it on the same pc.

3.6 Key Resource Requirements of the Project

Major Project activities	**Skill/Expertise Required**	**Works by Teammates**	**External Resources**	**Issues/Constraints**
Level Design	Ability to translate aspects of the story into playable levels	Both members made the decision about game levels together	Ideas from our existing games (Ex. Miami's Adventure)	No Issues
Physics Engine	Knowledge of functions available in Unity and the ability to change them as needed	Both members worked on Unity game engine	Unity game engine	Ability to angle interactive portions of levels
3D Modeling	Knowledge of 3D modeling software	Both members created necessary 3D models	3D model design using Autodesk Maya	Difficulties in UV mapping because of complex 3D models
Texturing of 3D Models	Knowledge of texturing software	We both did the texturing of 3D models	Adobe Photoshop Substance Painter Substance Bitmap2Material	Frequent software crash because of low-end computers
Music Implementation	Ability to incorporate sound clips smoothly into the game	-	Sound clips from the Internet	-

Level Implementation	Familiarity with scripting language of game engine	Both members have some knowledge about scripting language	Scripts from our previous games	Level size dependent on hardware configuration
Documentation	Knowledge about Formal Report Writing	Both members worked on the Report	Idea from Existing Reports (Ex. University Report Guideline)	Game Reports are different from conventional ones

3.7 Implementation Tools Required

Software Name	Developer Company	Usage	Work Area
Unity3D 5.6 & 2018.1	Unity Technologies	Game Engine	Game Design
Autodesk Maya 2016	Autodesk	3D Modeling and Animation	Creating 3D models and game objects
Adobe Fuse CC	Adobe Incorporated	Character Generation	Creating characters for the game
Adobe Photoshop CC 2014		Image Editing	Creating textures for 3D models
Adobe After Effects CC 2014		Creating & Editing Videos	Editing and applying necessary effects to cut scenes
Substance Painter 2	Substance	Texturing & Shading	Texturing of the 3D models
Substance Bitmap2Material		Bitmap to Material Conversion	Converting images to textures for 3D models
Microsoft Visual Studio 2017 Community Edition	Microsoft	Programming	Coding scripts for the game
Microsoft Visual Studio Code			

3.8 Implemented Codes & Scripts

We used C# as our programming language for the game, as it is easy to create, modify and implement. Unlike other games, we haven't implemented any scoring system as score doesn't have any significance in this game. We heavily emphasized on item collection and mission objectives. We reused codes from our previous games and projects as we saw fit.

We did face lots of problems while writing codes for our games, most of the time the codes didn't work as we expected and lots of errors showed up in the console window. We fixed those problems by either developing workarounds and using different code structures. We took help from several game developments related forums and websites to solve some problems that we weren't able to fix by ourselves. Because of this, programming took much more time than we expected.

The codes and scripts we wrote for this game have been described in the next chapter.

Chapter 4: Coding and Other Technical Aspects

This chapter discusses about the coding and scripting we have done for this game and refers to some of the test cases we conducted on the game to check if the game works as intended in various stages and situations of development.

As we have stated before, we used C# as our programming language for this project because it is easy to create, modify and implement. As Unity uses Mono engine for scripting and execution, writing codes for Unity in C# has become more efficient. We wrote all of the codes for this project ourselves and used internet to search for solutions of the problems we faced. We also used relevant codes from our own previous projects when we felt necessary.

4.1 Scripts Written for This Project

Name of the scripts we wrote and their purpose are described below:

- ❖ **First Person Controller:** This script handles the movement, health, damage and other necessary parameters for the player. It is a component of the FPS Player prefab.
- ❖ **Mouse Look:** This script is used to handle the movement of the first person camera with the mouse. This script is implemented in the First Person Controller script.
- ❖ **Player Magic:** This script is used to control player attack types, attack animations, damage and range. This script is a component of the FPS Player prefab.
- ❖ **Target (Enemy):** This script handles enemy movements, animation, attacks and damage. This script is applied to both of the enemy types.
- ❖ **Items:** This script is the base class script for all of the items the player can interact with and their inventory behavior. It isn’t applied to any objects, as it is intended to be accessed by other scripts.
- ❖ **Items Pickup:** This script handles the player’s interaction with the in-game items and their behaviors. It uses Items script as a base class for the items and is applied to every in-game items.

- **Consumable:** This script controls the behavior and parameter of items that can be consumed by the player to gain health or magic. It hasn't been implemented yet.
- **Interactable:** This is a base script for interactable objects. It controls the interaction behaviors and interaction range of the items. It isn't implemented to any object, as it is intended to be accessed by other scripts.
- **Inventory:** This is the base script for the inventory system. This script controls the inventory system, its capacity, behavior, item storage and item removal parameters.
- **Inventory Slot:** This script handles the behaviors and parameters of each inventory slot and is applied to Inventory Slot prefab.
- **Inventory UI:** This script handles the user interface and player interaction with the inventory. It is implemented in Game Manager prefab.
- **Dialogue:** This script is the base class for dialogue system. It controls the dialogue sentences, text behaviors and the number of sentences to be shown.
- **Dialogue Manager:** This script uses Dialogue script as base script and handles the dialogue box animation, text animation and text display behaviors.
- **Dialogue Trigger:** This script triggers the dialogue box to open when the player is in range and certain button is pressed.
- **Quest Marker:** This script handles the behavior, movement and animation of quest marker prefabs.
- **Teleportation:** This script makes the player teleport to other levels by changing to specified levels when the objectives have been completed by the player. This script has been used in every level to advance to the next level.

4.1.1 Code Analysis

We have written lots of scripts and codes for this project. All of the codes of this game is listed at the end of this document. To make the codes easy to understand for the reader, we are analyzing and describing the codes of "**Player Magic"** script below:

Library Functions:

using UnityEngine;
using System.Collections;
using UnityEngine.UI;

This is the starting part of the script, the library functions and namespaces that will be used in this script have been defined here.

Functions:

void Start ()

Return Type: No Return Type

Data Type: Void

Description: Runs the codes in it at the start of the game.

void Update ()

Return Type: No Return Type

Data Type: Void

Description: Runs the codes in it at every frame.

void PrimaryAttack ()

Return Type: No Return Type

Data Type: Void

Description: Executes the codes in it when called.

void ChargedAttack ()

Return Type: No Return Type

Data Type: Void

Description: Executes the codes in it when called.

public class PlayerMagic : MonoBehaviour

Declares the name (In this script, Player Magic) and type (MonoBehaviour) of the public class.

<u>Variables:</u>

public float pDamage = 10f;

Public variable of float data type that contains the floating point value of the primary attack damage of the player.

public float cDamage = 30f;

Public variable of float data type that contains the floating point value of the charged attack damage of the player.

public float range = 50f;

Float variable containing the value of the player's attack range.

public ParticleSystem pAttack;

Variable defining the particle system used for the effect of primary attack.

public ParticleSystem cAttack;

Variable defining the particle system used for the effect of charged attack.

public float pAttackInterval = 1f;

Float variable containing the value of primary attack interval time.

public float pTimer = 3f;

Variable containing the value of primary attack time.

public float cAttackInterval = 10f;

Float variable that contains the value of charged attack interval time.

public float cTimer = 10f;

Variable containing the value of charged attack time.

float pEffectDisplayTime = 1.10f;

Variable that defines how long the primary attack effect will be displayed.

float cEffectDisplayTime = 2.80f;

Variable defining how long the charged attack effect will be displayed.

Animator anim;

Variable referring to animator component of player.

Target target;

Variable referring to the static variables and public functions of the Target script.

public Image manaBar;

Image variable referring to the image of mana bar.

Camera cam;

Variable that refers to the scene camera.

AudioSource aSource;

Variable that refers to the Audio Source component of the player.

public AudioClip PAttackSound;

Contains the primary attack audio clip.

public AudioClip CAttackSound;

Variable containing the charged attack audio clip.

<u>Codes:</u>

anim = GetComponent<Animator> ();

Referring to the Animator component of the player.

aSource = GetComponent<AudioSource> ();

Refers to the Audio Source component of the player.

cam = Camera.main;

Refers to the camera that is tagged as the Main Camera in the scene.

pTimer += Time.deltaTime

Updates the primary attack timer according to real world time.

cTimer += Time.deltaTime;

Updates the charged attack timer according to real world time.

manaBar.fillAmount = cTimer / 10f;

Fills the mana bar image based on the value of charged attack timer. As the fill amount has the floating point value of 1, but the charged attack timer has the value of 10, the charged attack timer is divided by 10 to match this value with the fill amount value.

if (Input.GetButtonDown("Fire1") && pTimer >= pAttackInterval && Time.timeScale != 0)

Conditional statement that checks if certain conditions are met in order to run the codes in the loop.

PrimaryAttack ();

Calls the Primary Attack function to run the codes in that function.

if (Input.GetButtonDown("Fire2") && cTimer >= cAttackInterval && Time.timeScale != 0)

Conditional statement checking if certain conditions are met in order to run the codes in the loop.

ChargedAttack ();

Calls the Primary Attack function to run the codes in that function.

pTimer = 0f;

Resets primary attack timer to 0.

pAttack.Simulate(1);

Starts the particle effect.

pAttack.Play();

Plays the flame effect of primary attack.

aSource.PlayOneShot (PAttackSound);

Plays the sound effect of primary attack

RaycastHit hit;

Creates a ray cast hit variable named hit to store Raycast hit point.

if (Physics.Raycast(cam.transform.position, cam.transform.forward, out hit, range))

Defines the ray cast start position, ray direction and range of the ray.

target = hit.transform.GetComponent<Target> ();

Refers to the Target script component of the enemy that got hit by ray.

if (target != null)

Checks if the ray hit an enemy or not.

target.TakeDamage(pDamage);

Applies damage to the enemy.

cTimer = 0f;

This code resets Charged Attack timer to 0.

cAttack.Simulate(1);

Starts the particle effect.

cAttack.Play();

Plays the flame effect of charged attack.

aSource.PlayOneShot(CAttackSound);

Plays the sound effect of charged attack

4.2 Functional Tests

We have done lots of tests to make sure that everything in the game is working as we wanted. Some details of tests for different stages and situations are given below. [5]

4.2.1 Test Case 1

Test Case	:	Player movement animation.
Procedure	:	Import the Player character model with all animation files, create animator, place all animations in it, connect all states, code all animations in C# script, put the character in a scene and play the scene.
Expected Outcome	:	Animations will work properly.
Actual Outcome	:	Animations not working in some movements.
Found Reason	:	Animations not assigned properly in the script. Assigned all animations properly.
Further Test	:	Run the scene again.
Expected Outcome	:	Animation should work properly now.
Actual Outcome	:	All animations are working properly.

4.2.2 Test Case 2

Test Case	:	Player movement and physics.
Procedure	:	Add codes for movement of the player in the script. Play the scene.
Expected Outcome	:	Player moving properly as expected.
Actual Outcome	:	Player moves like sliding on the ground in some movements.
Found Reason	:	Player speed is too high. Experimented with the values to find correct speed value.
Further Test	:	Run the scene again.
Expected Outcome	:	Player should move properly now.
Actual Outcome	:	Player movement seems smooth and proper.

4.2.3 Test Case 3

Test Case	:	Player interaction with surrounding environment.
Procedure	:	Add collider to the player and all other objects, add scripts of interaction in objects that the player can interact with. Play the scene.
Expected Outcome	:	Player interacting with everything properly.
Actual Outcome	:	Player goes through some objects.
Found Reason	:	No colliders added to those objects. Added proper colliders to objects.
Further Test	:	Run the scene again.
Expected Outcome	:	Player should interact with all objects now.
Actual Outcome	:	Player interaction with objects is working.

4.2.4 Test Case 4

Test Case	:	Player attacks
Procedure	:	Create two fire attacks using particles in Unity, add necessary scripts to Player, play the scene.
Expected Outcome	:	particle attacks playing properly.
Actual Outcome	:	Particles move faster than player's attack animations.
Found Reason	:	Particle delay time is faster than player animation speed. Matched the delay with player animation.
Further Test	:	Run the scene again.
Expected Outcome	:	Particle emission time should match player attack animation.
Actual Outcome	:	Particle emits according to attack animation.

4.2.5 Test Case 5

Test Case	:	Enemy movement and damage.
Procedure	:	Add animations in enemy models, add necessary scripts, assign player as the target and damage amount to the player, play the scene.
Expected Outcome	:	Enemy will move toward player when in range.
Actual Outcome	:	Enemy falling through the ground.
Found Reason	:	Enemy collider is not assigned. Assigned and modified proper collider to enemy models.
Further Test	:	Run the scene again.
Expected Outcome	:	Enemy should stay on the ground.
Actual Outcome	:	Enemy stays on the ground and moves toward the player.

4.2.6 Test Case 6

Test Case	:	Level switching after player triggers certain colliders.
Procedure	:	Create teleportation colliders in specified places, add necessary scripts and play the scene.
Expected Outcome	:	Player will teleport to assigned level.
Actual Outcome	:	Player is teleporting to the assigned level.

4.2.7 Test Case 7

Test Case	:	Quest completing mechanisms.
Procedure	:	Implement quest completing mechanics to the quest levels, play the scene.
Expected Outcome	:	Quest should be completed after fulfilling necessary requirements.
Actual Outcome	:	Quests are completing after fulfilling given requirements.

4.2.8 Test Case 8

Test Case	:	Dialogue box trigger.
Procedure	:	Add dialogue box in the scene, create trigger area for the box, add necessary scripts and play the scene.
Expected Outcome	:	Dialogue box appears correctly.
Actual Outcome	:	Dialogue box displaying correcting.

4.2.9 Test Case 9

Test Case	:	Player health and magic bar UI.
Procedure	:	Create player health and magic bar UI, add necessary scripts and play the scene.
Expected Outcome	:	Health and magic amount showing correctly.
Actual Outcome	:	Magic bar isn't working properly.
Found Reason	:	Errors in the UI script. Fixed the error.
Further Test	:	Run the scene again.
Expected Outcome	:	Magic bar should be displayed properly now.
Actual Outcome	:	Magic bar is displaying properly.

4.3 Post-Build Beta Tests

4.3.1 Beta Test Bug 1

Expected Outcome	:	Game will run properly without bugs.
Actual Outcome	:	Player movement doesn't match the animation.
Found Reason	:	Player speed is lower than animation speed. Re-examined the speed values.
Further Test	:	Re-build the game and run again.
Expected Outcome	:	Player should move properly now.
Actual Outcome	:	Player is moving properly.

4.3.2 Beta Test Bug 2

Expected Outcome	:	Game will run properly without bugs.
Actual Outcome	:	Dialogue box doesn’t close at magic shop.
Found Reason	:	Dialogue box closing animation isn’t assigned in the script. Assigned the animation.
Further Test	:	Re-build the game and run again.
Expected Outcome	:	Dialogue box should close now.
Actual Outcome	:	Dialogue box is closing properly.

4.3.3 Beta Test Bug 3

Expected Outcome	:	Game will run properly without bugs.
Actual Outcome	:	Enemies keep randomly bouncing when moving.
Found Reason	:	Enemy physics isn’t defined properly in the script. Re-scripted enemy physics.
Further Test	:	Re-build the game and run again.
Expected Outcome	:	Enemy should move properly now.
Actual Outcome	:	Enemy is moving properly.

4.3.4 Beta Test Bug 4

Expected Outcome	:	Game will run properly without bugs.
Actual Outcome	:	Enemies keep randomly bouncing when moving.
Found Reason	:	Enemy physics isn’t defined properly in the script. Re-scripted enemy physics.
Further Test	:	Re-build the game and run again.
Expected Outcome	:	Enemy should move properly now.
Actual Outcome	:	Enemy is moving properly.

4.3.5 Beta Test Bug 5

Expected Outcome	:	Game will run properly without bugs.
Actual Outcome	:	Camera isn’t moving as expected.
Found Reason	:	The assigned camera isn’t well-suited for third-person view. Used Unity FreeLookRig camera.
Further Test	:	Re-build the game and run again.
Expected Outcome	:	Camera should move properly now.
Actual Outcome	:	Camera is not moving properly.
Found Reason	:	Unity FreeLookRig camera has lots of bugs. Used Unity MultiPurposeCameraRig instead.
Further Test	:	Re-build the game and run again.
Expected Outcome	:	Camera should move properly now.
Actual Outcome	:	Camera goes through objects.
Found Reason	:	Unity FreeLookRig camera also has bugs in object collision. Used Unity First Person Controller.
Further Test	:	Re-build the game and run again.
Expected Outcome	:	Camera should move properly now.
Actual Outcome	:	Camera is moving properly.

4.3.6 Beta Test Bug 6

Expected Outcome	:	Game will run properly without bugs.
Actual Outcome	:	Level objectives aren’t clear enough for gamers.
Found Reason	:	An objective window is necessary. Added objectives window.
Further Test	:	Re-build the game and run again.
Expected Outcome	:	Level objectives should be easy to understand.
Actual Outcome	:	Level objectives are easy to understand.

Chapter 5: User Manual of “Ethoreal Shift”

This chapter contains instructions for the players. It describes the methods of playing the game and also includes some in-game screenshots to give the player some ideas about the game before they start playing it. [5]

5.1 Playing Procedures

Players first interact with the system UI to start playing. We provide tips to the gamer about game mechanics so that he/she can easily understand how to play the game. There are different levels in this game which follows a planned storyline. Gamers can play each level by completing the objectives of the previous one. Player have to use his/her logic to accomplish the game. He needs to explore the levels, find scrolls to advance in some levels, defeat lots of enemies to complete his objectives. If the gamer’s health decreases completely, he loses the game and the level restarts. Here at the first level, the gamer’s objective is to find the magic shop first, then find all the scrolls hidden across Sylvista village. Player have to explore every house in the village to collect the scrolls. In the second level, the player is in Daryas city and he have to find scrolls hidden inside the city again to get information about the location of the elemental stones. Like that different level has different complexity and different logics to finish. But the main objective is that gamer should find all 3 stones to save the world and finish the game.

5.2 In-Game Screenshots

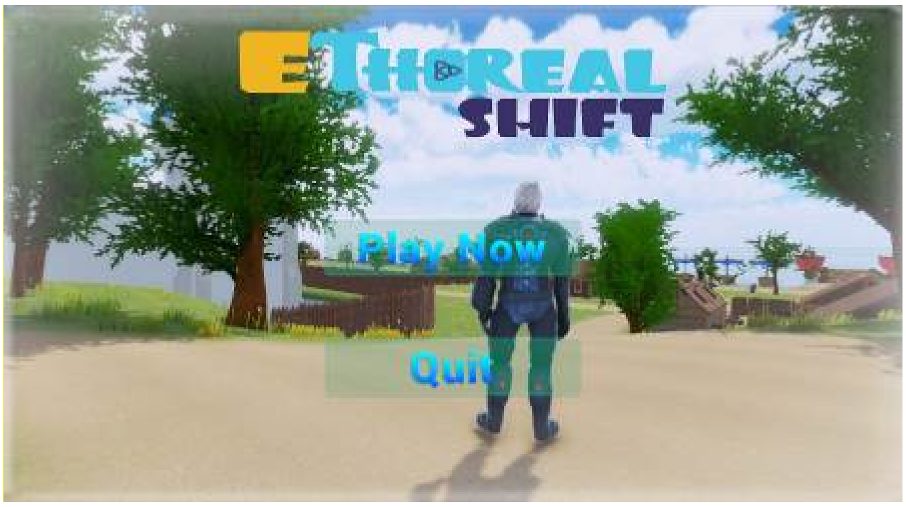


Fig 5.2.1: The Start Menu

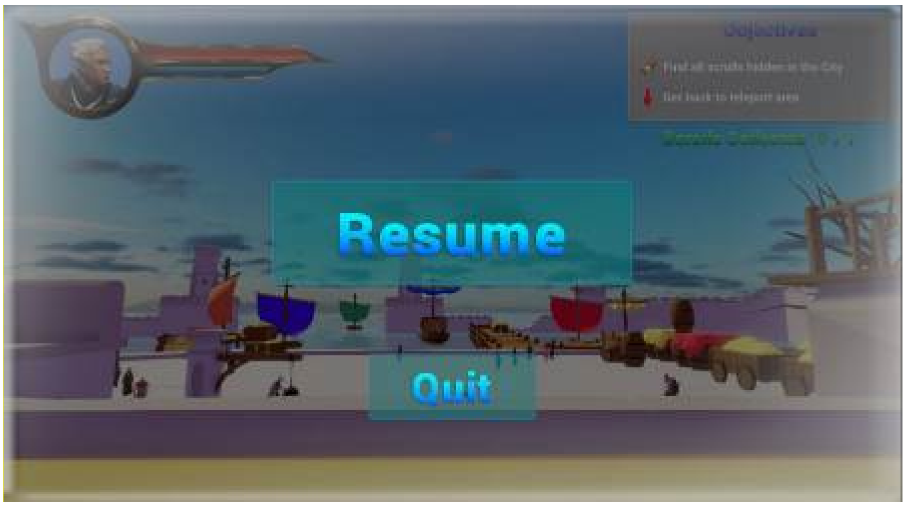


Fig 5.2.2: The Pause Menu

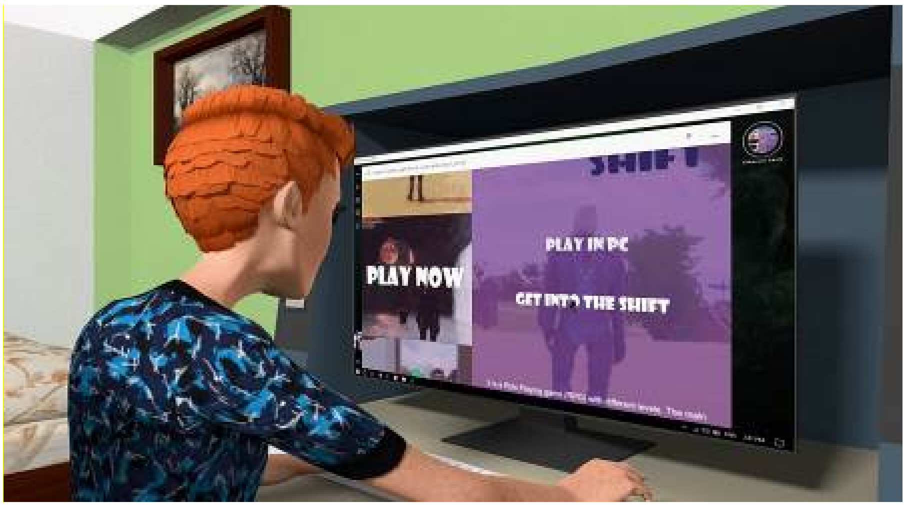


Fig 5.2.3: Game Start Cut scene

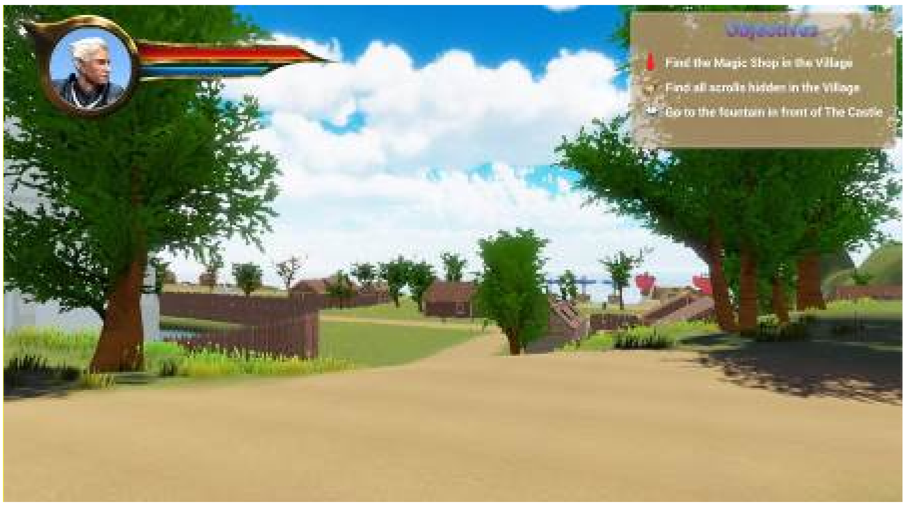

Objectives
Find the Magic Shop in the Village
Find all scrolls hidden in the Village
Go to the fountain in front of The Castle

Fig 5.2.4: Sylvista

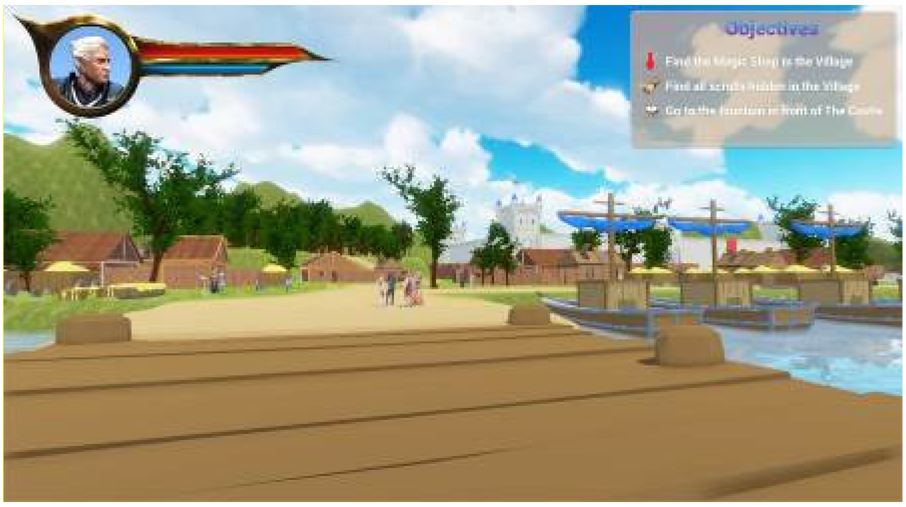

Objectives
Find the Magic Shop in the Village
Find all scrolls hidden in the Village
Go to the fountain in front of The Castle

Fig 5.2.5: Sylvista Port

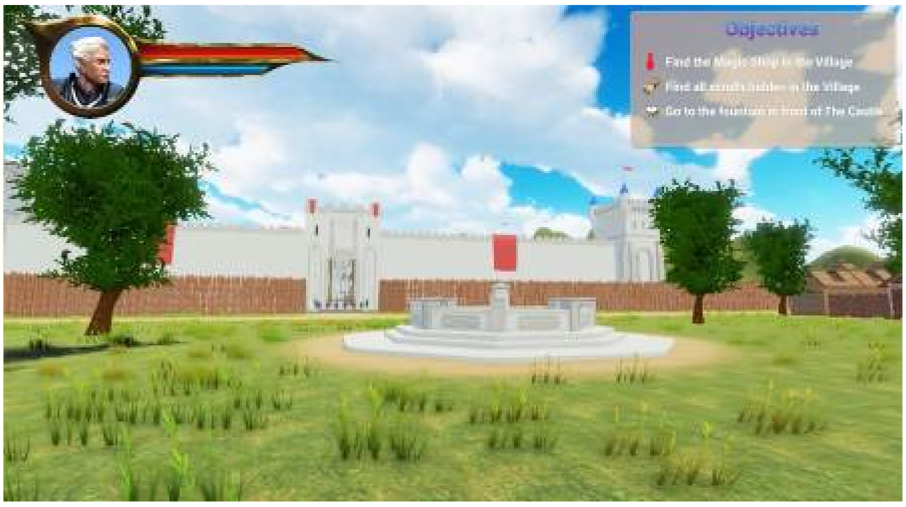

Objectives
Find the Magic Shop in the Village
Find all scrolls hidden in the Village
Go to the fountain in front of The Castle

Fig 5.2.6: Citadel Castle and the Fountain

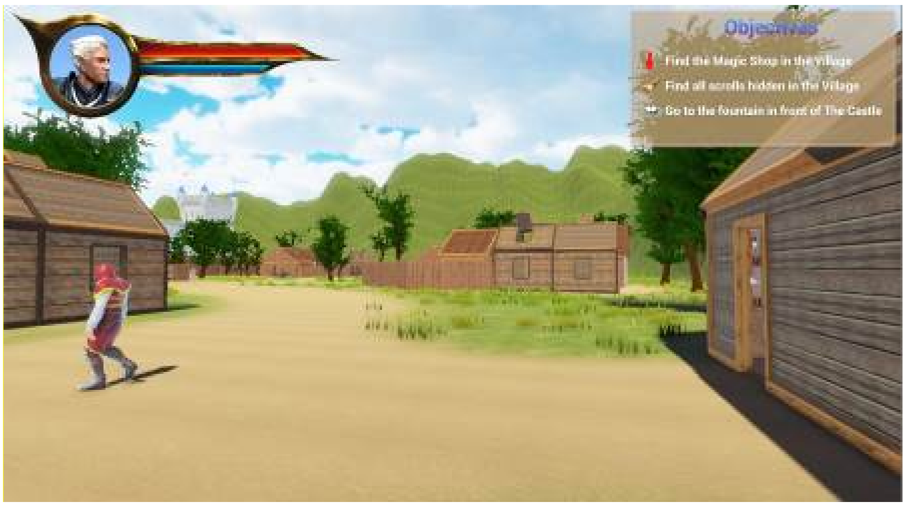

Objectives
Find the Magic Shop in the Village
Find all scrolls hidden in the Village
Go to the fountain in front of The Castle

Fig 5.2.7: Outside of Magic Shop

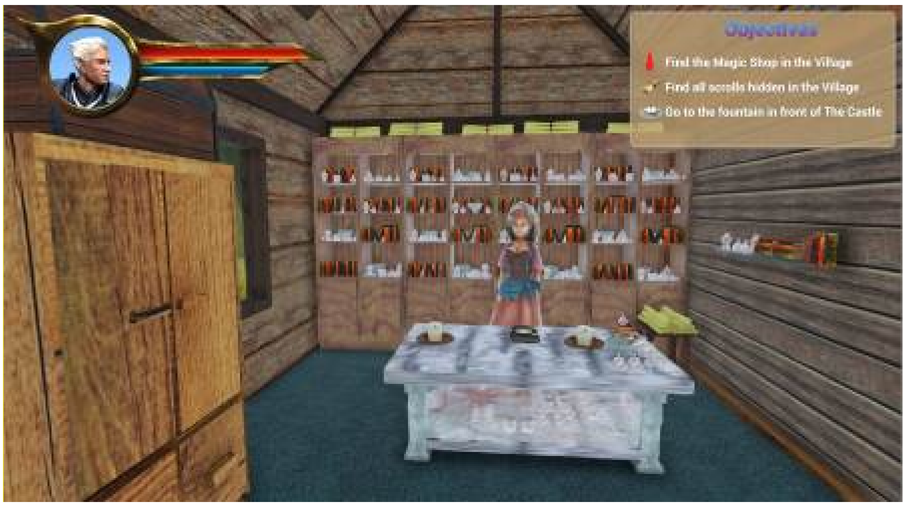
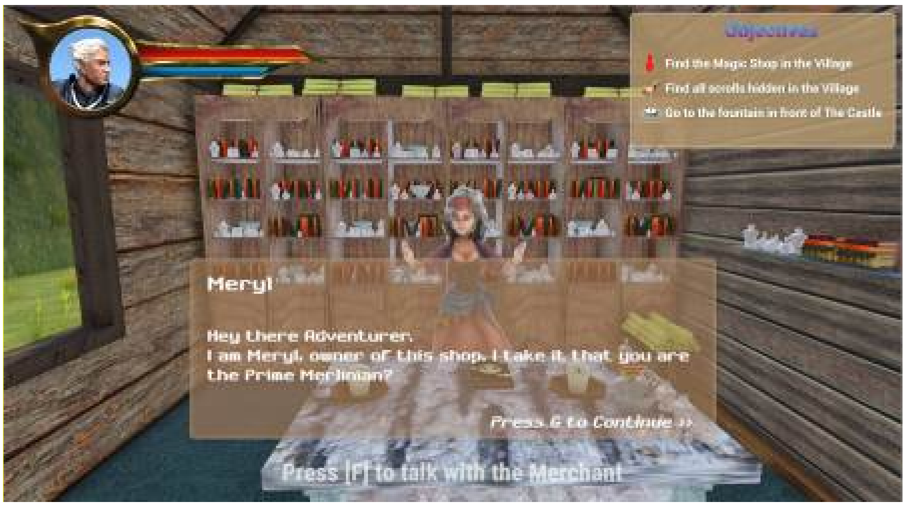

Objectives
Find the Magic Shop in the Village
Find all scrolls hidden in the Village
Go to the fountain in front of The Castle

Fig 5.2.8: Inside the Magic Shop


Objectives
Find the Magic Shop in the Village
Find all scrolls hidden in the Village
Go to the fountain in front of The Castle

Meryl

Hey there Adventurer.
I am Meryl, owner of this shop. I take it, that you are the Prime Merlinian?

Press G to Continue >>

Press [F] to talk with the Merchant

Fig 5.2.9: The Marchant Talking to the Player

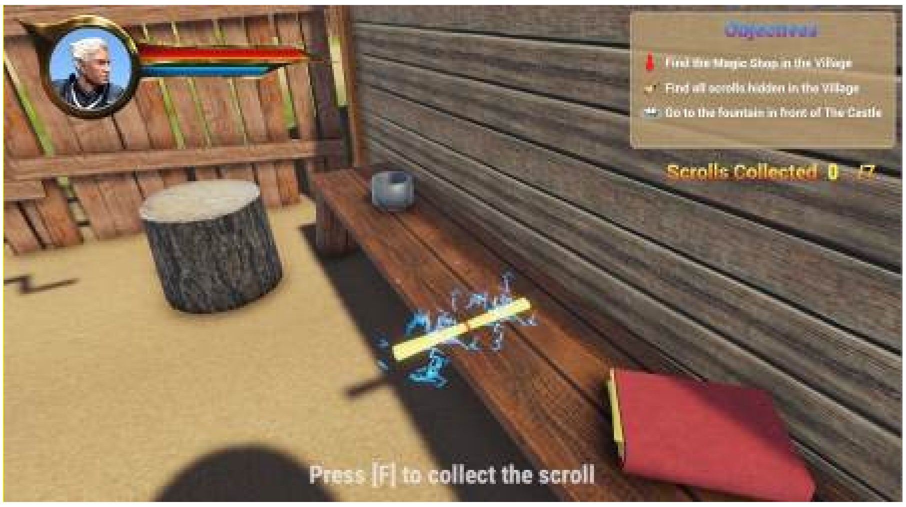

Objectives
Find the Magic Shop in the Village
Find all scrolls hidden in the Village
Go to the fountain in front of The Castle
Scrolls Collected 0 / 7
Press [F] to collect the scroll

Fig 5.2.10: Collecting Scrolls

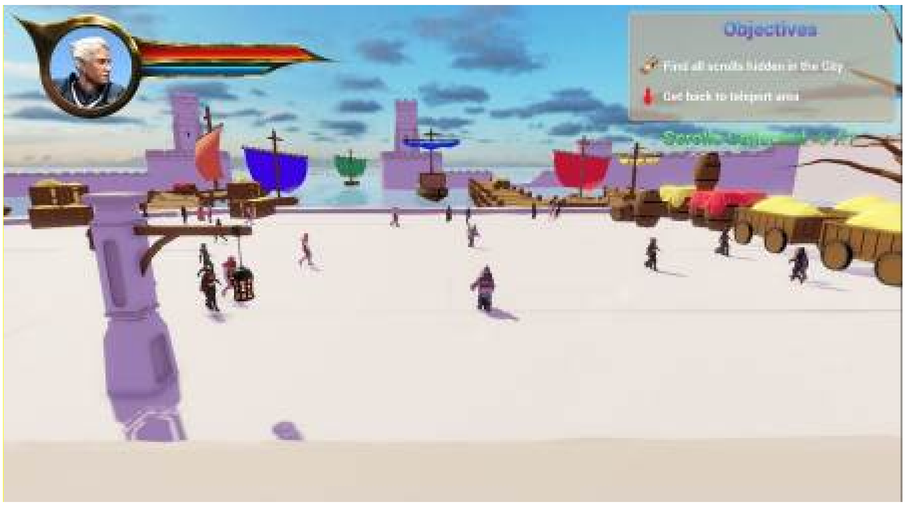

Objectives
Find all scrolls hidden in the City
Get back to teleport area

Fig 5.2.11: Daryas City Port

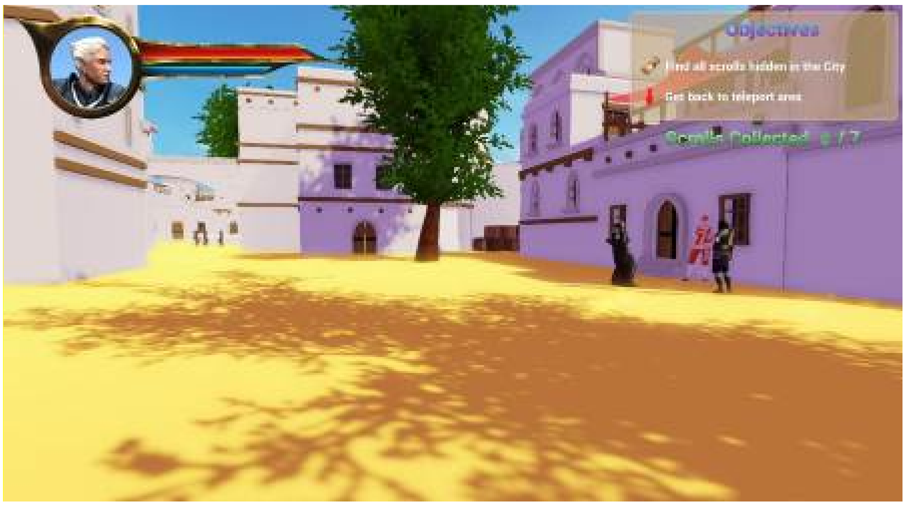

Objectives
Find all scrolls hidden in the City
Get back to teleport area
Scrolls Collected 0 / 7

Fig 5.2.12: Player exploring the City

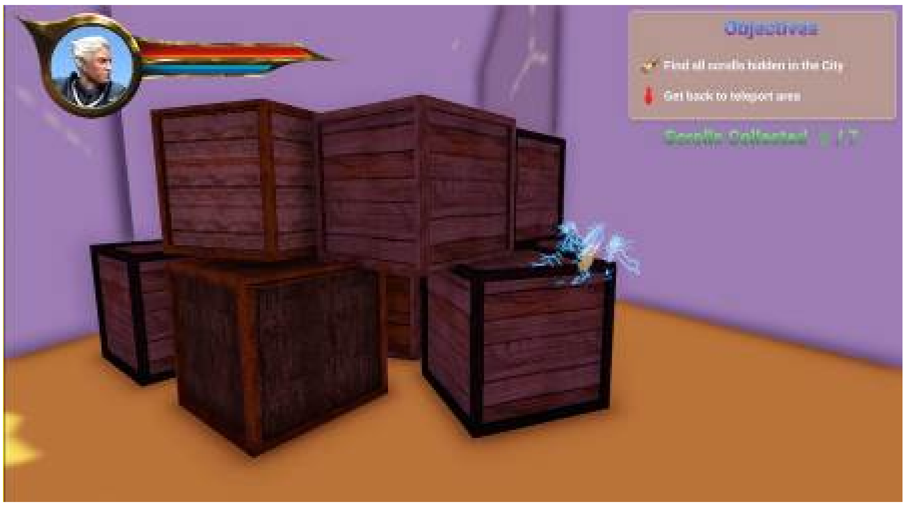

Objectives
Find all scrolls hidden in the City
Get back to teleport area
Scrolls Collected 1 / 7

Fig 5.2.13: Scrolls in the City

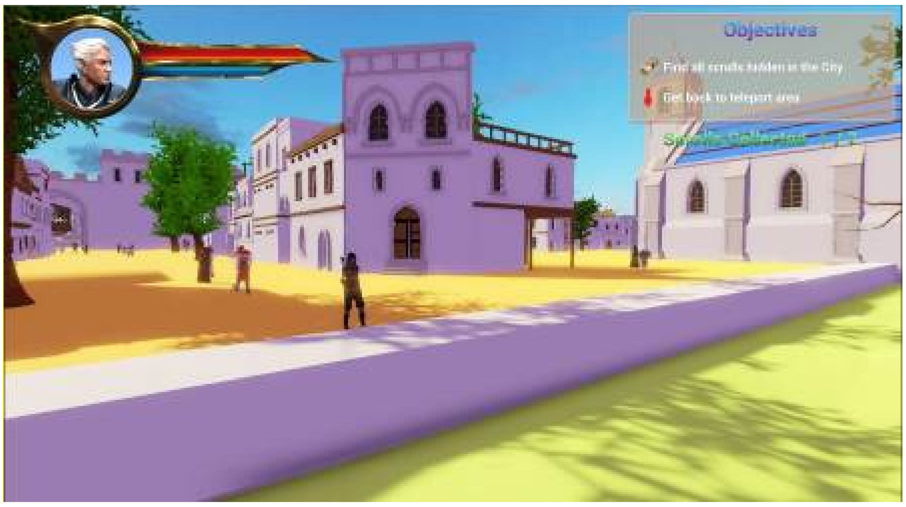

Objectives
Find all scrolls hidden in the City
Get back to teleport area
Scrolls Collected 1 / 7

Fig 5.2.14: Peoples walking around

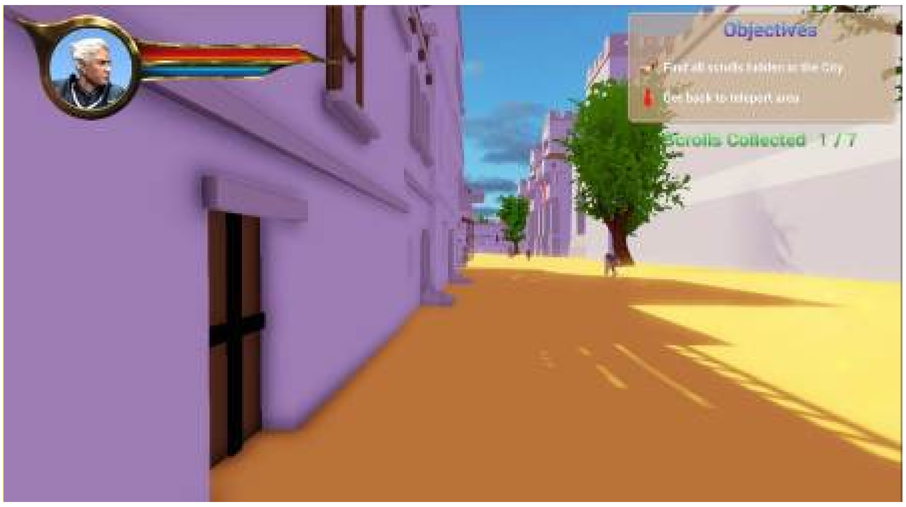

Objectives
Find all scrolls hidden in the City
Get back to teleport area
Scrolls Collected 1 / 7

Fig 5.2.15: The City Wall

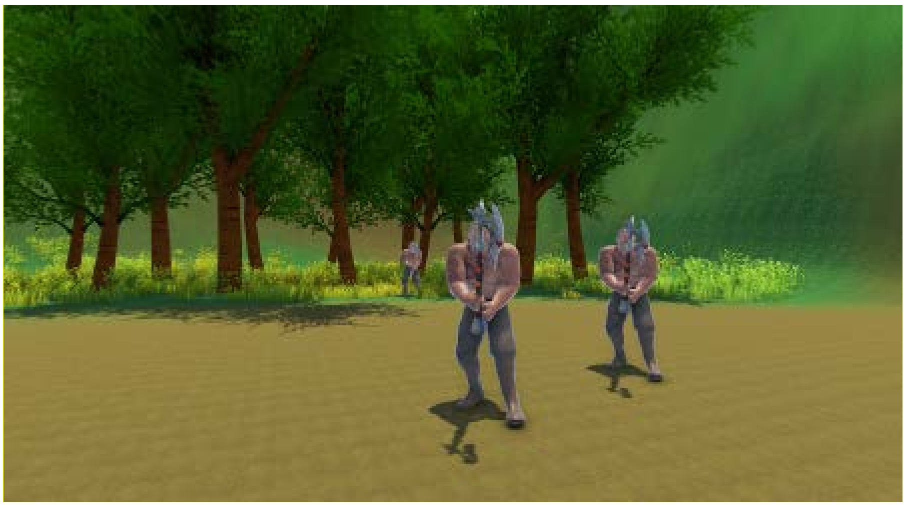


Fig 5.2.16: Woodcutter coming to attack the Player

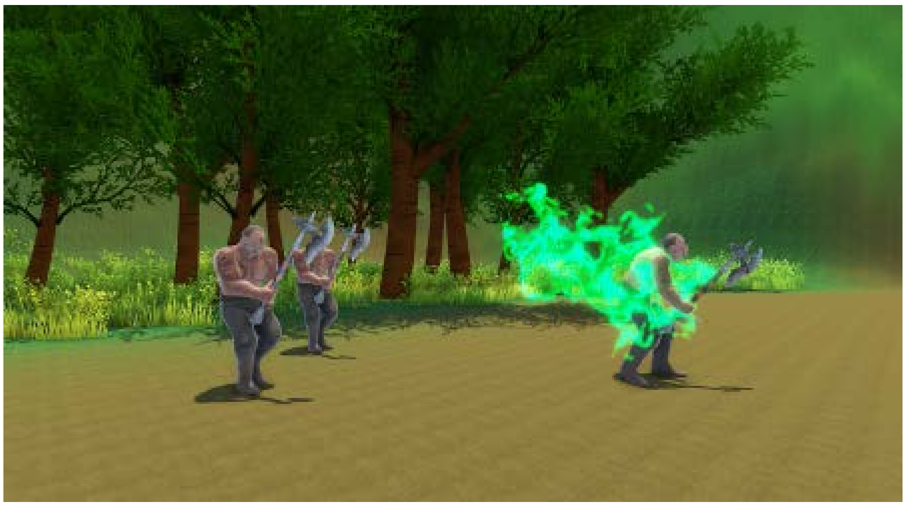


Fig 5.2.17: Player attacking the Woodcutter

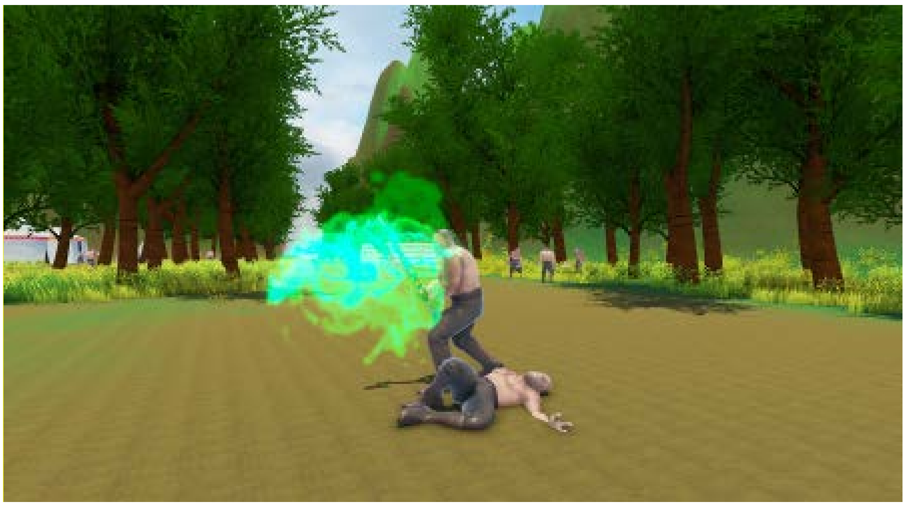


Fig 5.2.18: Woodcutter is Dead

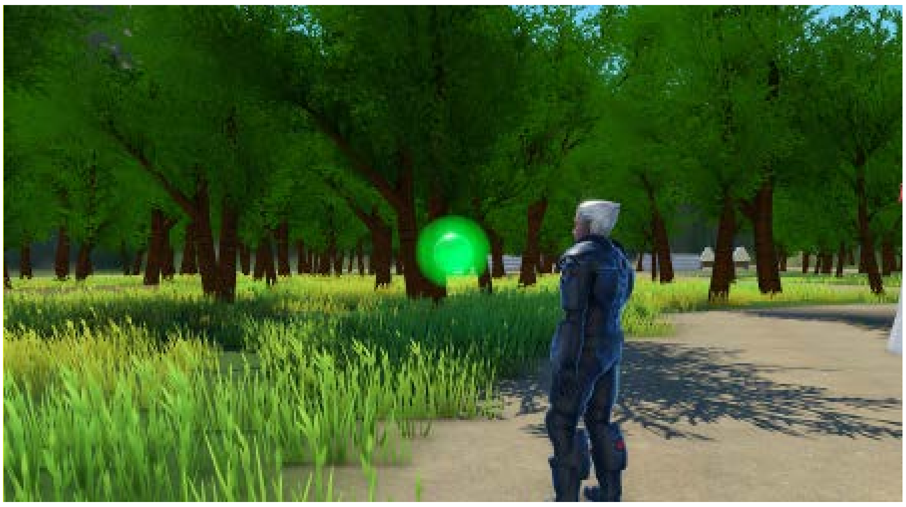


Fig 5.2.19: Player got the Dryad Stone

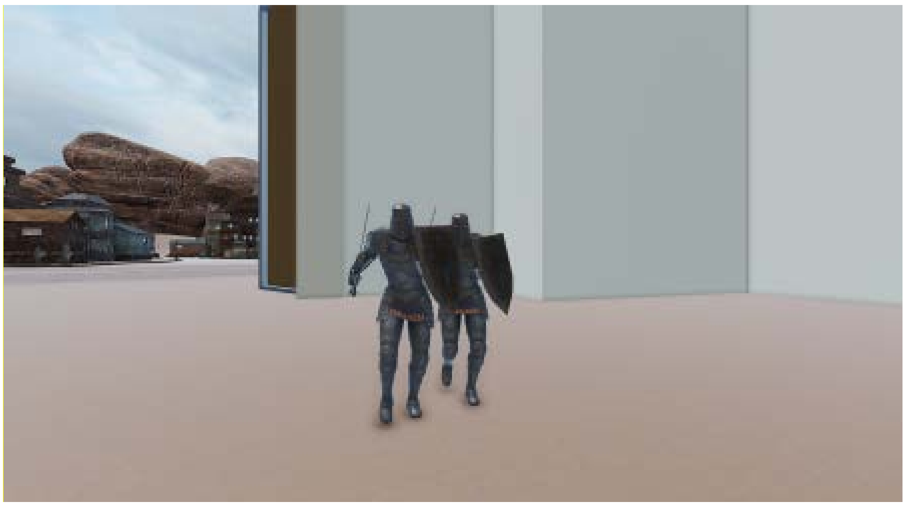


Fig 5.2.20: Paladin Knights attacking the Player

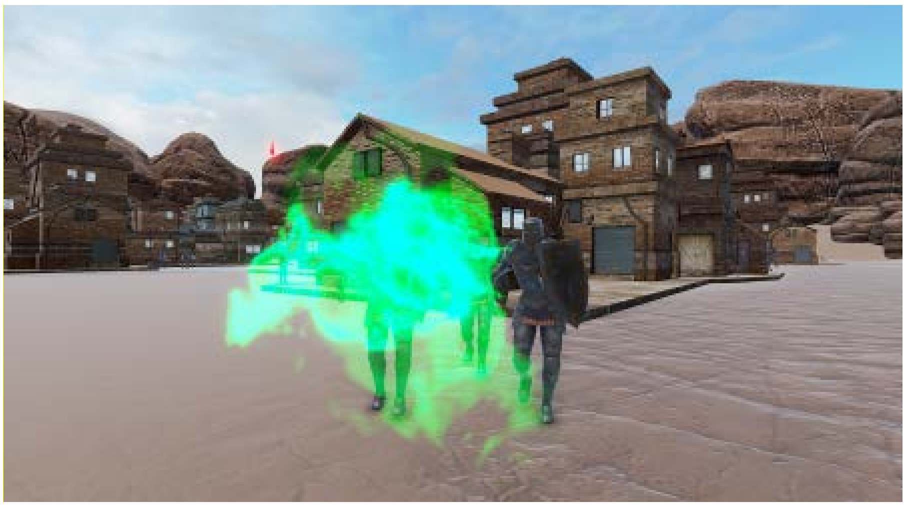


Fig 5.2.21: Player attacking the Paladin Knights

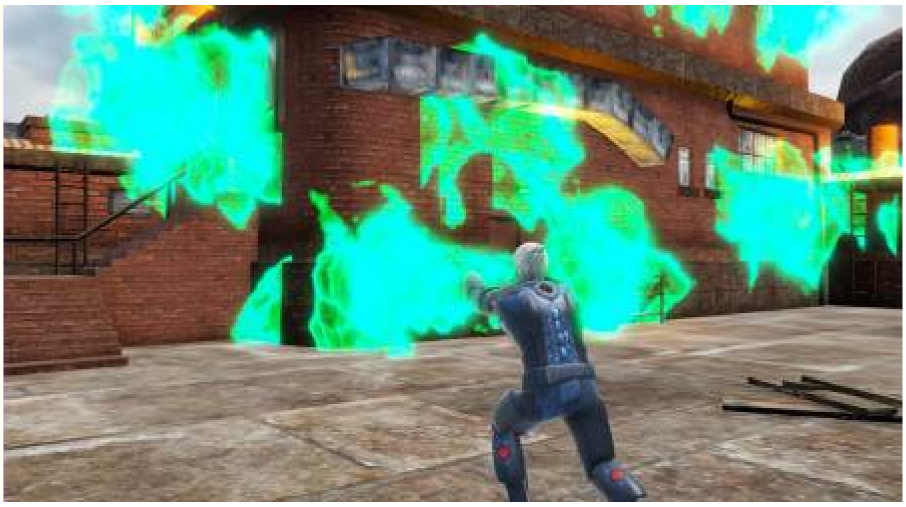


Fig 5.2.22: Player Casting Explosion Spell on the Machine Building

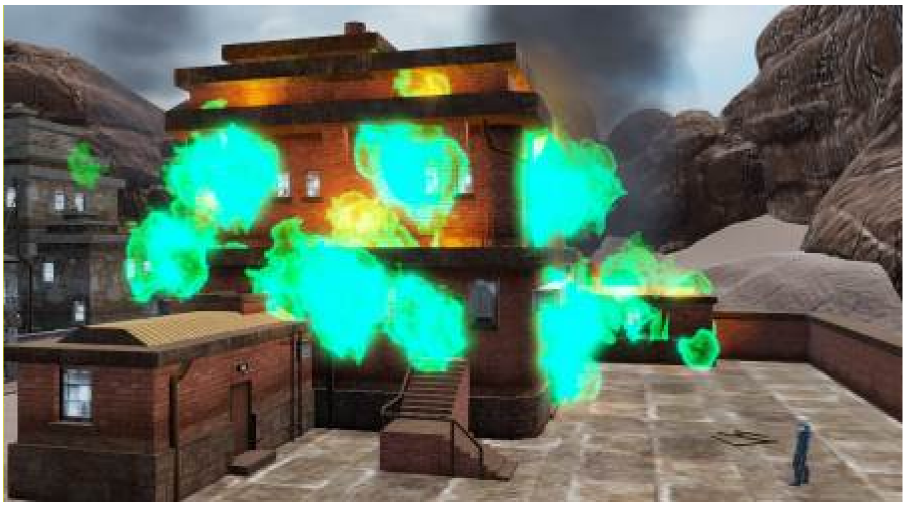


Fig 5.2.23: Machine Building is being destroyed

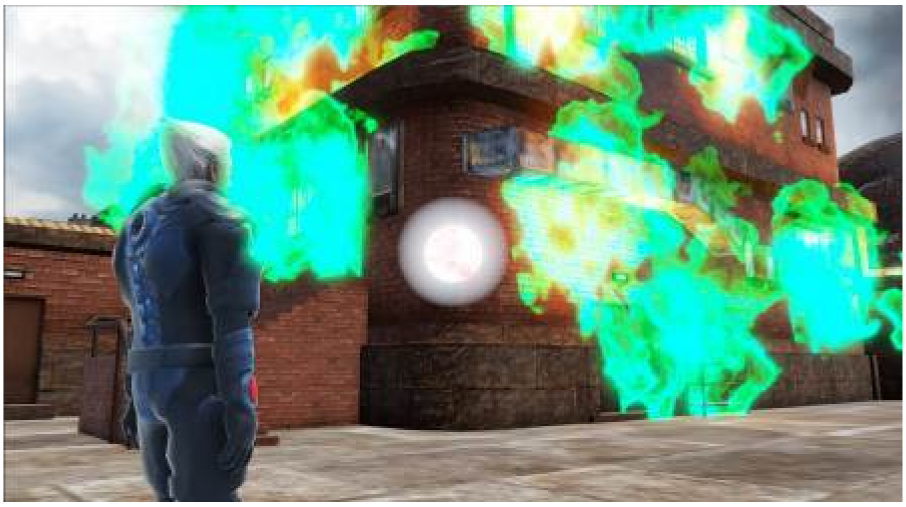


Fig 5.2.24: Player got the Zephyrus Stone

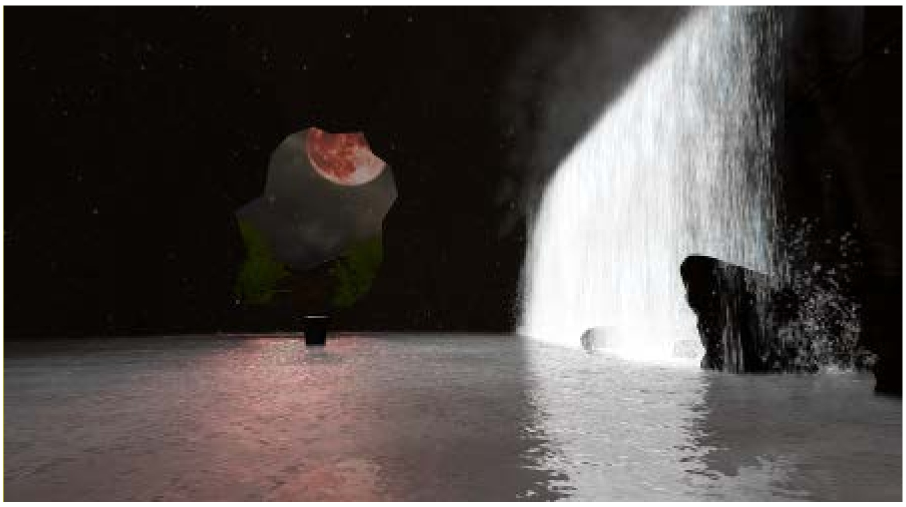


Fig 5.2.25: Player in front of the Sacred Waterfall

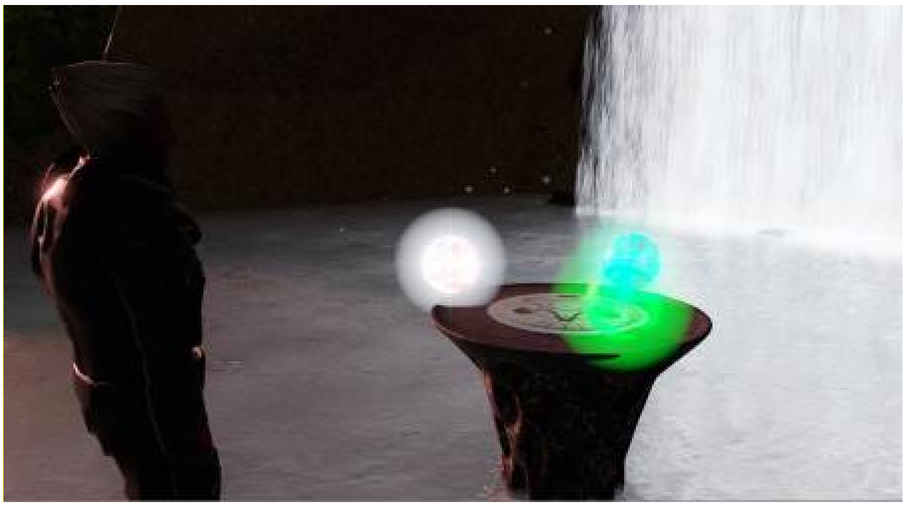


Fig 5.2.26: Player in front of the Ancient Magic Table

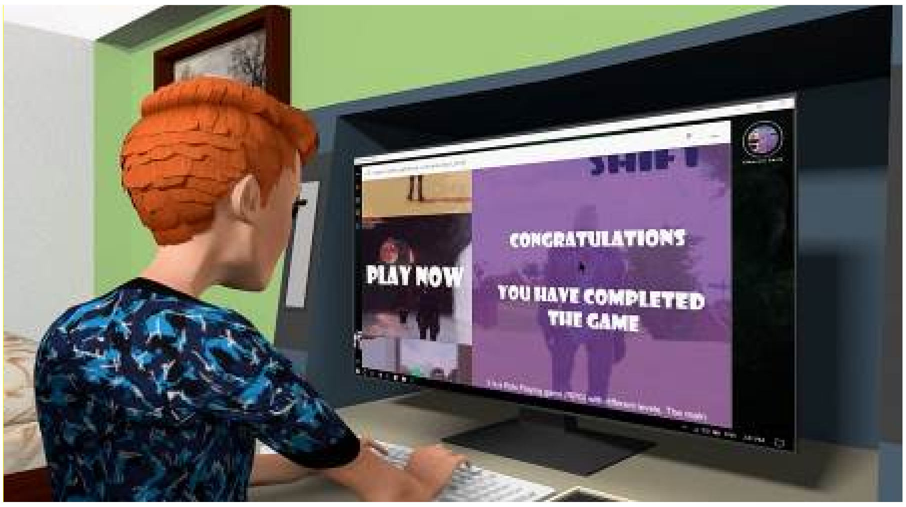


Fig 5.2.27: Game Completed

Chapter 6: Conclusion

A game project needs a lot of experience to complete. In this chapter we summarize the experience gained by our developer team during the whole development process of “Ethoreal Shift”.

6.1 The Obstacles We Faced

1. We had our first idea for the game rejected after four months of development time. So we had only one semester to complete the whole project starting from developing a new concept, planning everything we have to implement in the game, 3D modelling, designing the levels, programming etc.
2. We weren’t given any idea about how much time we would have to finish the project. So we had to set a deadline by ourselves and had to plan the project according to that deadline. Because of this reason, we had to give up on some of the ideas and features of the game to complete it in due time.
3. Unity game engine has lots of necessary features missing because it has a subscription based paid version with lots of additional features, some of which should have been implemented in the free/personal version of the engine that we used. We suffered a lot because of this in this project and our previous games. We had to develop some workarounds to make some features work. But most of the workarounds needs advanced knowledge in C# and we didn’t have enough time to do research and learn more things to fully utilize the features of Unity.
4. We developed these workarounds and broaden our knowledge in game design and programming by video tutorials, text tutorials, internet and learning materials given by the tools themselves, though they had little contribution in our knowledge building. The project was a matter of time, patience and hard work.
5. It is very rational work and it requires lots of time because the game engines try to match the game environment with the real world.

6. Creating 3D models for this project was very difficult because we had to work with each and every point of the model to make them look realistic. The characters of the game were most difficult to model as we wanted to make them feel real.
7. Developing models, textures and game levels were much difficult than we had thought because of our under-powered computers.
8. Unity demands vast knowledge about its functions, properties, sections and subsections.
9. The game engines themselves had lots of bugs in them, which made our work much more difficult. We had to develop some alternatives to tackle the bugs of game engines.
10. Some features of Unity that we needed for our project were made available with version upgrade in this year. So we had to improvise some aspects of the game so that we can complete our project. And when we upgraded our Unity version, the upgraded Unity re-imported our project files where it corrupted some files. We had to import those files again, re-modified them for our levels and had to implement them again.
11. The Unity asset store has huge collection of tools, but most of the tools we needed were paid. So We weren’t able to implement some features like we wanted to.

And finally but most importantly, our planned game was not a project that could be completed in 4-6 months by only two people with under-powered computers!

6.2 The Achievements

1. Now we know much more about the Unity game engine. How it works, its features, properties, objects and others, including its bugs.
2. We were able to implement everything we learned in last four years starting from graphics design, motion graphics, 3D modelling, character rigging, animation, to game level design, game programming and scripting etc.
3. We know more about 3D modelling, texturing, look development, rigging and animation.
4. The main thing is that as a game developer, skill and expertise to design game levels, implementing various mechanics and the experience we gained by completing an overall game project by only two people.
5. Dedication and co-operation between group members.
6. Developed better communication skills.
7. Expending creative thinking and imagination capability.

6.3 Future Development Plans

Ethoreal Shift was originally planned to be a free-to-play Massively Multiplayer Online (MMO) fantasy open-world third person action-adventure game where all the players will be magicians and they will be playing the game together online to save the world of the game from the magic beasts of the parallel world. But as we were given pretty short time to develop this game and because of the shortcomings and lack of necessary features in Unity game engine itself, we had to divert from our original plan and cut short most of our ideas to complete the game in time. So our future plan for this game is to rebuild the game from scratch in Unreal Engine using our original plan. We will develop a massive and open fantasy world full of quests, puzzles and adventures using cultural, historical and mythological references, where the players will have to use their own knowledge, intellect and cooperate with each other in order to advance through the quests and finish the game. Also, there will be four elemental stones instead of three and their location won't be revealed until the final phase of the game starts (described further below).

- There would be 9 classes (races) of avatars in the game which the players can choose to play with. Each class will have their own features, powers and special abilities. But they all would have access to the massive weapons arsenal in the game as they level up. The unlock system for avatar, weapons and power upgrades will be based on levels, ranks and amount of in-game gems of the players, rather than pay-to-unlock. Players will level up and collect gems by completing different quests, solving puzzles, in-game trading of gems and items, and defeating the magic beasts.
- The quests and puzzles will be all over the game's open world based on historical events, Greek and Norse mythologies, cultures of different countries and our own original ideas. Some quests would require the players to work together as a team to complete the objectives. Quests based on historical events and mythologies would require the player to gain knowledge about history and mythologies in order to complete them. There would be some fast-paced quests and puzzles that will challenge the intellect and reflex of a player. It will be up to the players to complete those quests and puzzles.

- The enemies would be magic beasts, who would be from parallel world. There will be several races of the beasts and they will have magical powers similar to the magicians of this world. Their powers will differ based on their race, size, and also their level and rank. Some of them will have regenerative abilities, some will have the power to heal others. Some beasts will be huge like a building, very powerful. Those will be the boss of the quests. Some beasts will be the size of small animals and would be found in easy quests. Some beasts would be like regular sized humans or taller, those will be the regular enemies in the quests. Their ranks and levels would decide what amount of gems and experience points will the players get when they defeat the enemies.
- The quests, number and level of enemies, gems, item drops and XP points in cooperative quests would be controlled by a base system or algorithm to keep certain things in the game unpredictable. Some quests will be in this world, and some quests would be in the parallel world. We are planning to give the game itself a time-limit. At first, the game will be in phase alpha, where the players will be given a couple of years to level up, complete quests and find clues to complete the whole game. After that, the final phase of the game will start, which will unlock new quests and objectives that will reveal the locations of the four elemental stones. In this phase, all players will have from 6 months to 1 year to complete all new quests, collect the stones and finish the game. If they can’t finish the game in the given time, the whole game will reset, deleting all player and level data and the game will restart from the beginning, starting a loop until the game is cleared in the final phase. But this isn’t finalized in the plan yet because of the technical difficulties of this process.

Though it will take several years of development and lots of funding to complete this original plan of the game, but we believe that we can develop this into an amazing game if given the opportunity.

6.4 Last Few Words

We learned a lot by working on this project. This project has enriched our knowledge and sharpened our concept of game engines, 3D modeling and animation. We learned a lot about different game documentation. The game we developed is intended to be played by the gamers of all over the world. The success of this project may give pleasure to lots of game lovers among the world. This project has not only tested our knowledge and technical skills but also our temperament. There were few times when we almost lost hope and motivation, but we recovered through continuous concentration, research, teamwork and hard work.

If you have any kind of suggestion, ideas for improvements and more effective development ideas, please feel free to communicate with us.

Appendix

Abbreviation and Acronyms

Term	Definition
Game engine	A game engine is an application designed for the creation and development of video games.
UX	User experience (UX or UE) is a person's interaction and experience with a particular product, system or service.
Animation	Animation is a process by which images or objects are manipulated to appear as moving images.
OpenGL	Open Graphics Library (OpenGL) is a cross-language, cross-platform application programming interface for rendering 2D and 3D vector graphics that is typically used to interact with a graphics processing unit, to achieve hardware-accelerated rendering.
Scripting	A scripting language or script language is a programming language that supports the writing of scripts and programs for a special runtime environment that can interpret and automate the execution of tasks which would otherwise be executed manually by a human operator.
SRS	Software Requirements Specification
UI	User Interface
System	A system is a set of interacting or interdependent components forming an integrated whole.

References

[1] Global video game industry revenue CAGR by category 2015-2020, Retrieved from https://www.statista.com.

[2] ESports, Retrieved from https://en.wikipedia.org.

[3] Role-playing video game, Retrieved from https://en.wikipedia.org.

[4] Software Requirement Specification (2010), Chalmers University of Technology

[5] Project "The Ghost in Town" (2014), Dhaka University

[6] Project "Death Wish" (2008), Viral Light Interactive.

Codes and Scripts Implemented

The scripts and codes we wrote for our game are written below:

First Person Controller:

```
using System;
using UnityEngine;
using UnityStandardAssets.CrossPlatformInput;
using UnityStandardAssets.Utility;
using System.Collections;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
using Random = UnityEngine.Random;

   [RequireComponent(typeof(CharacterController))]
   [RequireComponent(typeof(AudioSource))]
   public class FirstPersonController : MonoBehaviour
   {
      [SerializeField] private bool m_IsWalking;
      [SerializeField] private float m_WalkSpeed;
      [SerializeField] private float m_RunSpeed;
      [SerializeField] [Range(0f, 1f)] private float m_RunstepLenghten;
      [SerializeField] private float m_JumpSpeed;
      [SerializeField] private float m_StickToGroundForce;
      [SerializeField] private float m_GravityMultiplier;
      [SerializeField] private MouseLook m_MouseLook;
      [SerializeField] private bool m_UseFovKick;
      [SerializeField] private FOVKick m_FovKick = new FOVKick();
      [SerializeField] private bool m_UseHeadBob;
      [SerializeField] private CurveControlledBob m_HeadBob = new
CurveControlledBob();
      [SerializeField] private LerpControlledBob m_JumpBob = new
LerpControlledBob();
```

```
        [SerializeField] private float m_StepInterval;
        [SerializeField] private AudioClip[] m_FootstepSounds;    // an array of footstep
sounds that will be randomly selected from.
        [SerializeField] private AudioClip m_JumpSound;           // the sound played
when character leaves the ground.
        [SerializeField] private AudioClip m_LandSound;           // the sound played
when character touches back on ground.
        public float pHealth;
        public float maxHealth = 100f;
        public Transform startPoint;
        public Item item;
        public Text scrolls;
        public GameObject pDeathText;
        public static int scrollCount;
        public static float enemyCount;
        public AudioClip scrollCollect;
        public Image healthBar;
        private Camera m_Camera;
        private bool m_Jump;
        private float m_YRotation;
        private Vector2 m_Input;
        private Vector3 m_MoveDir = Vector3.zero;
        private CharacterController m_CharacterController;
        private CollisionFlags m_CollisionFlags;
        private bool m_PreviouslyGrounded;
        private Vector3 m_OriginalCameraPosition;
        private float m_StepCycle;
        private float m_NextStep;
        private bool m_Jumping;
        private AudioSource m_AudioSource;

        private void Start()
```

```
    {
        m_CharacterController = GetComponent<CharacterController>();
        m_Camera = Camera.main;
        m_OriginalCameraPosition = m_Camera.transform.localPosition;
        m_FovKick.Setup(m_Camera);
        m_HeadBob.Setup(m_Camera, m_StepInterval);
        m_StepCycle = 0f;
        m_NextStep = m_StepCycle / 2f;
        m_Jumping = false;
        m_AudioSource = GetComponent<AudioSource>();
        m_MouseLook.Init(transform, m_Camera.transform);
        pHealth = 100f;
        scrollCount = 0;
        enemyCount = 0;
        pDeathText.SetActive(false);
        scrolls.text = scrollCount.ToString();
        m_CharacterController.transform.position = startPoint.position;           //Moves
the Player to the assigned Start Point.
    }

    private void Update()
    {
        RotateView();
        // the jump state needs to read here to make sure it is not missed
        if (!m_Jump)
        {
            m_Jump = CrossPlatformInputManager.GetButtonDown("Jump");
        }
        if (!m_PreviouslyGrounded && m_CharacterController.isGrounded)
        {
            StartCoroutine(m_JumpBob.DoBobCycle());
            PlayLandingSound();
            m_MoveDir.y = 0f;
            m_Jumping = false;
        }
        if (!m_CharacterController.isGrounded && !m_Jumping && m_PreviouslyGrounded)
        {
            m_MoveDir.y = 0f;
        }
        m_PreviouslyGrounded = m_CharacterController.isGrounded;
        if (startPoint == null)
        {
            startPoint = transform;        //If Start Point is not assigned, player transform will be used.
        }
        if (pHealth <= 0)
            return;
    }
    private void PlayLandingSound()
    {
        m_AudioSource.clip = m_LandSound;
        m_AudioSource.Play();
        m_NextStep = m_StepCycle + .5f;
    }
    private void FixedUpdate()
    {
        float speed;
        GetInput(out speed);
        // always move along the camera forward as it is the direction that it being aimed at
        Vector3 desiredMove = transform.forward * m_Input.y + transform.right * m_Input.x;
        // get a normal for the surface that is being touched to move along it
```

```
            RaycastHit hitInfo;
            Physics.SphereCast(transform.position,              m_CharacterController.radius,
Vector3.down,
out hitInfo,
m_CharacterController.height              /              2f,              Physics.AllLayers,
QueryTriggerInteraction.Ignore);
            desiredMove              =              Vector3.ProjectOnPlane(desiredMove,
hitInfo.normal).normalized;
            m_MoveDir.x = desiredMove.x * speed;
            m_MoveDir.z = desiredMove.z * speed;
            if (m_CharacterController.isGrounded)
            {
                m_MoveDir.y = -m_StickToGroundForce;
                if (m_Jump)
                {
                    m_MoveDir.y = m_JumpSpeed;
                    PlayJumpSound();
                    m_Jump = false;
                    m_Jumping = true;
                }
            }
            else
            {
                m_MoveDir      +=      Physics.gravity      *      m_GravityMultiplier      *
Time.fixedDeltaTime;
            }
            m_CollisionFlags = m_CharacterController.Move(m_MoveDir *
Time.fixedDeltaTime);
            ProgressStepCycle(speed);
            UpdateCameraPosition(speed);
            m_MouseLook.UpdateCursorLock();
        }
```

```
    private void PlayJumpSound()
    {
        m_AudioSource.clip = m_JumpSound;
        m_AudioSource.Play();
    }
    private void ProgressStepCycle(float speed)
    {
        if (m_CharacterController.velocity.sqrMagnitude > 0 && (m_Input.x != 0 ||
m_Input.y != 0))
        {
            m_StepCycle += (m_CharacterController.velocity.magnitude + (speed *
(m_IsWalking ? 1f : m_RunstepLenghten))) *
                         Time.fixedDeltaTime;
        }
        if (!(m_StepCycle > m_NextStep))
        {
            return;
        }
        m_NextStep = m_StepCycle + m_StepInterval;
        PlayFootStepAudio();
    }
    private void PlayFootStepAudio()
    {
        if (!m_CharacterController.isGrounded)
        {
            return;
        }
        // pick & play a random footstep sound from the array,
        // excluding sound at index 0
        int n = Random.Range(1, m_FootstepSounds.Length);
        m_AudioSource.clip = m_FootstepSounds[n];
        m_AudioSource.PlayOneShot(m_AudioSource.clip);
```

```
            // move picked sound to index 0 so it's not picked next time
            m_FootstepSounds[n] = m_FootstepSounds[0];
            m_FootstepSounds[0] = m_AudioSource.clip;
        }
        private void UpdateCameraPosition(float speed)
        {
            Vector3 newCameraPosition;
            if (!m_UseHeadBob)
            {
                return;
            }
            if (m_CharacterController.velocity.magnitude > 0 &&
m_CharacterController.isGrounded)
            {
                m_Camera.transform.localPosition =
                    m_HeadBob.DoHeadBob(m_CharacterController.velocity.magnitude +
                                      (speed * (m_IsWalking ? 1f : m_RunstepLenghten)));
                newCameraPosition = m_Camera.transform.localPosition;
                newCameraPosition.y = m_Camera.transform.localPosition.y –
m_JumpBob.Offset();
            }
            else
            {
                newCameraPosition = m_Camera.transform.localPosition;
                newCameraPosition.y          =          m_OriginalCameraPosition.y          -
m_JumpBob.Offset();
            }
            m_Camera.transform.localPosition = newCameraPosition;
        }
        private void GetInput(out float speed)
        {
            // Read input
            float horizontal = CrossPlatformInputManager.GetAxis("Horizontal");
            float vertical = CrossPlatformInputManager.GetAxis("Vertical");
            bool waswalking = m_IsWalking;

#if !MOBILE_INPUT
            // On standalone builds, walk/run speed is modified by a key press.
            // keep track of whether or not the character is walking or running
            m_IsWalking = !Input.GetKey(KeyCode.LeftShift);
#endif
            // set the desired speed to be walking or running
            speed = m_IsWalking ? m_WalkSpeed : m_RunSpeed;
            m_Input = new Vector2(horizontal, vertical);

            // normalize input if it exceeds 1 in combined length:
            if (m_Input.sqrMagnitude > 1)
            {
                m_Input.Normalize();
            }

            // handle speed change to give an fov kick
            // only if the player is going to a run, is running and the fovkick is to be used
            if (m_IsWalking != waswalking && m_UseFovKick &&
m_CharacterController.velocity.sqrMagnitude > 0)
            {
                StopAllCoroutines();
                StartCoroutine(!m_IsWalking ? m_FovKick.FOVKickUp() :
m_FovKick.FOVKickDown());
            }
        }
        private void RotateView()
        {
            m_MouseLook.LookRotation(transform, m_Camera.transform);
```

```
    }
    public void TakeDamage(float dAmount)
    {
        pHealth -= dAmount;
        healthBar.fillAmount = pHealth / 100f;
        if (pHealth <= 0f)
        {
            pDeathText.SetActive(true);
            StartCoroutine(GameOver());
        }
    }

    void OnTriggerEnter(Collider other)
    {
        if (other.tag == "Scrolls")
        {
            scrollCount = scrollCount + 1;
            m_AudioSource.PlayOneShot(scrollCollect);
            if (scrolls.text != null)
            {
                scrolls.text = scrollCount.ToString();
            }
        }
    }

IEnumerator GameOver()
    {
        yield return new WaitForSeconds(5.0f);
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
    }
    private void OnControllerColliderHit(ControllerColliderHit hit)
    {
        Rigidbody body = hit.collider.attachedRigidbody;
        //dont move the rigidbody if the character is on top of it
        if (m_CollisionFlags == CollisionFlags.Below)
        {
            return;
        }
        if (body == null || body.isKinematic)
        {
            return;
        }
        body.AddForceAtPosition(m_CharacterController.velocity * 0.1f, hit.point,
ForceMode.Impulse);
    }
}
```

Player Magic:

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;

public class PlayerMagic : MonoBehaviour {

    public float pDamage = 10f;             //Primary Attack damage.
    public float cDamage = 30f;             //Charged Attack damage.
    public float range = 50f;               //Attack range.
    public ParticleSystem pAttack;          //Primary Attack effect.
    public ParticleSystem cAttack;          //Charged Attack effect.
    public float pAttackInterval = 1f;      //Primary Attack Interval.
    public float pTimer = 3f;               //Primary Attack Timer.
    public float cAttackInterval = 10f;     //Charged Attack Interval.
    public float cTimer = 10f;              //Charged Attack Timer.
    float pEffectDisplayTime = 1.10f;       //How long the Primary Attack effect will be
displayed.
    float cEffectDisplayTime = 2.80f;       //How long the Charged Attack effect will be
displayed.
    Animator anim;                          //Reference to Player Animator.
   Target target;                           //Reference to Enemy Script.
    public Image manaBar;
    Camera cam;
    AudioSource aSource;
    public AudioClip PAttackSound;
    public AudioClip CAttackSound;

    void Start()
    {
        anim = GetComponent<Animator>();
        aSource = GetComponent<AudioSource>();
        cam = Camera.main;
    }
    void Update ()
{
        pTimer += Time.deltaTime;           //Update the Timer.
        cTimer += Time.deltaTime;           //Update the timer.
        manaBar.fillAmount = cTimer / 10f;
        //Primary Attack.
        if (Input.GetButtonDown("Fire1") && pTimer >= pAttackInterval &&
Time.timeScale
!= 0)
        {
            PrimaryAttack();
        }

        //Charged Attack.
        if (Input.GetButtonDown("Fire2") && cTimer >= cAttackInterval &&
Time.timeScale != 0)
        {
           ChargedAttack();
        }
    }

    //Primary Attack parameters
    void PrimaryAttack()
    {
        pTimer = 0f;                //Resets Primary Attack timer.
        pAttack.Simulate(1);        //Starts the effect.
        pAttack.Play();             //Plays the effect.
        aSource.PlayOneShot(PAttackSound);
        RaycastHit hit;             //Reference to Raycast hit point.
        if (Physics.Raycast(cam.transform.position, cam.transform.forward, out hit, range))
        {
            Debug.Log(hit.transform.name);
            target = hit.transform.GetComponent<Target>();
            if (target != null)
            {
                target.TakeDamage(pDamage);     //Enemy takes damage.
            }
        }
    }

    //Charged Attack parameters
    void ChargedAttack()
    {
        cTimer = 0f;                //Resets Charged Attack timer.
        cAttack.Simulate(1);        //Starts the effect.
        cAttack.Play();             //Plays the effect.
        aSource.PlayOneShot(CAttackSound);
        RaycastHit hit;             //Reference to Raycast hit point.
        if (Physics.Raycast(cam.transform.position, cam.transform.forward, out hit, range))
        {
            Debug.Log(hit.transform.name);
            target = hit.transform.GetComponent<Target>();
            if (target != null)
            {
                target.TakeDamage(cDamage);     //Enemy takes damage.
            }
        }
    }
}
```

Mouse Look:

```
using System;
using UnityEngine;
using UnityStandardAssets.CrossPlatformInput;

  [Serializable]
  public class MouseLook
  {
    public float XSensitivity = 2f;
    public float YSensitivity = 2f;
    public bool clampVerticalRotation = true;
    public float MinimumX = -90F;
    public float MaximumX = 90F;
    public bool smooth;
    public float smoothTime = 5f;
    public bool lockCursor = true;
    private Quaternion m_CharacterTargetRot;
    private Quaternion m_CameraTargetRot;
    private bool m_cursorIsLocked = true;

    public void Init(Transform character, Transform camera)
    {
      m_CharacterTargetRot = character.localRotation;
      m_CameraTargetRot = camera.localRotation;
    }

    public void LookRotation(Transform character, Transform camera)
    {
      float yRot = CrossPlatformInputManager.GetAxis("Mouse X") * XSensitivity;
      float xRot = CrossPlatformInputManager.GetAxis("Mouse Y") * YSensitivity;

      m_CharacterTargetRot *= Quaternion.Euler (0f, yRot, 0f);
```

```
            m_CameraTargetRot *= Quaternion.Euler (-xRot, 0f, 0f);

            if(clampVerticalRotation)
                m_CameraTargetRot = ClampRotationAroundXAxis
(m_CameraTargetRot);

            if(smooth)
            {
                character.localRotation = Quaternion.Slerp (character.localRotation,
m_CharacterTargetRot,
                    smoothTime * Time.deltaTime);
                camera.localRotation = Quaternion.Slerp (camera.localRotation,
m_CameraTargetRot,
                    smoothTime * Time.deltaTime);
            }
            else
            {
                character.localRotation = m_CharacterTargetRot;
                camera.localRotation = m_CameraTargetRot;
            }
            UpdateCursorLock();
        }
        public void SetCursorLock(bool value)
        {
            lockCursor = value;
            if(!lockCursor)
            {//we force unlock the cursor if the user disable the cursor locking helper
                Cursor.lockState = CursorLockMode.None;
                Cursor.visible = true;
            }
        }
        public void UpdateCursorLock()
```

```
{
    //if the user set "lockCursor" we check & properly lock the cursos
    if (lockCursor)
        InternalLockUpdate();
}

private void InternalLockUpdate()
{
    if(Input.GetKeyUp(KeyCode.Escape))
    {
        m_cursorIsLocked = false;
    }
    else if(Input.GetMouseButtonUp(0))
    {
        m_cursorIsLocked = true;
    }

    if (m_cursorIsLocked)
    {
        Cursor.lockState = CursorLockMode.Locked;
        Cursor.visible = false;
    }
    else if (!m_cursorIsLocked)
    {
        Cursor.lockState = CursorLockMode.None;
        Cursor.visible = true;
    }
}

Quaternion ClampRotationAroundXAxis(Quaternion q)
{
    q.x /= q.w;
```

```
        q.y /= q.w;
        q.z /= q.w;
        q.w = 1.0f;
        float angleX = 2.0f * Mathf.Rad2Deg * Mathf.Atan (q.x);
        angleX = Mathf.Clamp (angleX, MinimumX, MaximumX);
        q.x = Mathf.Tan (0.5f * Mathf.Deg2Rad * angleX);
        return q;
    }
}
```

Target (Enemy):

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Target : MonoBehaviour
{

    public float health = 50f;          //Enemy health
    public float damage = 20f;          //Damage done to the Player
    Animator anim;                      //Reference to Enemey Animator
    float startDelay = 1.5f;            //Start delay of death animation
    public Transform player;            //Reference to Player position
    FirstPersonController fpc;          //Reference to Player Control script
    float enemyGravity;
    AudioSource aSource;
    public AudioClip walkSound;
    public AudioClip attackSound;

    void Start()
    {
        anim = GetComponent<Animator>();
        player = GameObject.FindGameObjectWithTag("Player").transform;
        fpc = player.transform.GetComponent<FirstPersonController>();
        aSource = GetComponent<AudioSource>();
    }

    private void Update()
    {
        //Enemy animation
        anim.SetBool("isRunning", false);
        anim.SetBool("isIdle", true);
        anim.SetBool("isAIdle", false);
        anim.SetBool("isWalking", false);
        anim.SetBool("Dying", false);
        anim.SetBool("Attack", false);
        anim.SetBool("gotHit", false);
        enemyGravity += Physics.gravity.y * Time.deltaTime;
        Vector3 direction = player.position - this.transform.position;        //Determining
enemy position from Player
        float angle = Vector3.Angle(direction, this.transform.forward);          //Enemy
moves forward

        //Checks distance from player
        if (Vector3.Distance(player.position, this.transform.position) < 20 && angle <
300 && fpc.pHealth >= 0 && Time.timeScale >= 1)
        {
            if (health <= 0)
                return;
            anim.SetBool("isRunning", false);
            anim.SetBool("isIdle", false);
            anim.SetBool("isAIdle", true);
            anim.SetBool("isWalking", false);
            anim.SetBool("Dying", false);
            anim.SetBool("Attack", false);
            anim.SetBool("gotHit", false);
            this.transform.rotation = Quaternion.Slerp(this.transform.rotation,
Quaternion.LookRotation(direction), 1f);

            //Keeps moving toward the player until reached at a certain distance
            if (direction.magnitude > 2)
            {
                this.transform.Translate(0, 0, .09f);
                anim.SetBool("isRunning", false);
                anim.SetBool("isIdle", false);
                anim.SetBool("isAIdle", false);
                anim.SetBool("isWalking", true);
                anim.SetBool("Dying", false);
                anim.SetBool("Attack", false);
                anim.SetBool("gotHit", false);
            }

            //Attacks the player when in range
            else if (direction.magnitude <= 2)
            {
                anim.SetBool("isRunning", false);
                anim.SetBool("isIdle", false);
                anim.SetBool("isWalking", false);
                anim.SetBool("Dying", false);
                anim.SetBool("Attack", true);
                anim.SetBool("gotHit", false);
                StartCoroutine(PlayerDamage());
            }
            if (fpc.pHealth <= 0)
            {
                anim.SetBool("isRunning", false);
                anim.SetBool("isIdle", true);
                anim.SetBool("isWalking", false);
                anim.SetBool("Dying", false);
                anim.SetBool("Attack", false);
                anim.SetBool("gotHit", false);
            }
        }

        //If player is not in range, enemy stays idle
        else
```

```
        {
            anim.SetBool("isRunning", false);
            anim.SetBool("isIdle", false);
            anim.SetBool("isAIdle", true);
            anim.SetBool("isWalking", false);
            anim.SetBool("Dying", false);
            anim.SetBool("Attack", false);
            anim.SetBool("gotHit", false);
        }
    }

    //Takes damage
    public void TakeDamage(float amount)
    {
        health -= amount;                    //Reduces Enemy health by the Player's damage
amount.
        anim.SetBool("isRunning", false);
        anim.SetBool("isIdle", false);
        anim.SetBool("isWalking", false);
        anim.SetBool("Dying", false);
        anim.SetBool("Attack", false);
        anim.SetBool("gotHit", true);

        //When health reaches zero, enemy dies
        if (health <= 0f)
        {
            Die();
        }
    }

    //Death Sequence
    void Die()
    {
        anim.SetBool("isRunning", false);
        anim.SetBool("isIdle", false);
        anim.SetBool("isWalking", false);
        anim.Play("Death");
        anim.SetBool("Attack", false);
        anim.SetBool("gotHit", false);
        PlayerControl.enemyCount = PlayerControl.enemyCount + 1;
        //Destroys game object after 7 seconds
        Destroy(gameObject, 5);
    }

    IEnumerator PlayerDamage()
    {
        yield return new WaitForSeconds(.5f);
        fpc.TakeDamage(damage);       //Damages the player
    }
}
```

Item:

```
using UnityEngine;
 [CreateAssetMenu(fileName = "New Item", menuName = "Inventory/Item")]
public class Item : ScriptableObject
{
    new public string name = "New Item";
    public Sprite icon = null;
    public bool showInInventory = true;
    public virtual void Use()
    {    }
    public void RemoveFromInventory()
    {
        Inventory.instance.Remove(this);
    }
}
```

Consumables:

```
using UnityEngine;
[CreateAssetMenu(fileName = "New Item", menuName = "Inventory/Consumable")]
public class Consumable : Item
{
    public int healthGain;
    public GameObject player;
    public override void Use()
    {
        PlayerControl playerControl = player.GetComponent<PlayerControl>();
        playerControl.Heal(healthGain);
        RemoveFromInventory();
    }
}
```

Inventory Scripts:

Inventory:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class Inventory : MonoBehaviour
{
    public static Inventory instance;
    void Awake()
    {
        instance = this;
    }
    public delegate void OnItemChanged();
    public OnItemChanged onItemChangedCallback;
    public int space = 10;
    public List<Item> items = new List<Item>();
    public void Add(Item item)
    {
        if (item.showInInventory)
        {
            if (items.Count >= space)
            {
                return;
            }
            items.Add(item);
            if (onItemChangedCallback != null)
                onItemChangedCallback.Invoke();
        }
    }
    public void Remove(Item item)
    {
        items.Remove(item);
        if (onItemChangedCallback != null)
            onItemChangedCallback.Invoke();
    }   }
```

Inventory Slot:

```
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.Events;
using UnityEngine.EventSystems;
public class InventorySlot : MonoBehaviour
{
    public Image icon;
    public Button removeButton;
    Item item;
    public void AddItem(Item newItem)
    {
        item = newItem;
        icon.sprite = item.icon;
        icon.enabled = true;
        removeButton.interactable = true;
    }
    public void ClearSlot()
    {
        item = null;
        icon.sprite = null;
        icon.enabled = false;
        removeButton.interactable = false;
    }
    public void RemoveItemFromInventory()
    {
        Inventory.instance.Remove(item);
    }
    public void UseItem()
    {
        if (item != null)
        {
            item.Use();
        }   }   }
```

<u>Inventory UI:</u>

```
using UnityEngine;
using System.Collections.Generic;
using UnityEngine.UI;
public class InventoryUI : MonoBehaviour
{
    public GameObject inventoryUI;
    public Transform itemsParent;
    Inventory inventory;
    void Start()
    {
        inventory = Inventory.instance;
        inventory.onItemChangedCallback += UpdateUI;
    }
    void Update()
    {
        if (Input.GetButtonDown("Inventory"))
        {
            inventoryUI.SetActive(!inventoryUI.activeSelf);
            UpdateUI();
            Debug.Log("Inventory");
        }
    }
    public void UpdateUI()
    {
        InventorySlot[] slots = GetComponentsInChildren<InventorySlot>();
        for (int i = 0; i < slots.Length; i++)
        {
            if (i < inventory.items.Count)
            {
                slots[i].AddItem(inventory.items[i]);
            }
            else
            {
                slots[i].ClearSlot();
            }     }    }   }
```

Items Pickup:

```
using UnityEngine;
public class ItemPickup : Interactable
{
    public Item item;
    public override void Interact()
    {
        base.Interact();
    }
    void OnTriggerEnter(Collider Other)
    {
        if (Other.tag == "Player")
        {
            Inventory.instance.Add(item);
            Destroy(gameObject);
        }   }   }
```

Interactable:

```
using UnityEngine;
using UnityEngine.AI;
public class Interactable : MonoBehaviour
{
    public float radius = 3f;
    public Transform interactionTransform;
    bool isFocus = false;
    Transform player;
    bool hasInteracted = false;
    public virtual void Interact()
    {        }

    void OnDrawGizmosSelected()
    {
        if (interactionTransform == null)
            interactionTransform = transform;
        Gizmos.color = Color.yellow;
        Gizmos.DrawWireSphere(interactionTransform.position, radius);
    }
}
```

Dialogue System:

Dialogue:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
[System.Serializable]
public class Dialogue
{
    public string name;
    [TextArea(3, 10)]
    public string[] sentences;
}
```

Dialogue Manager:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class DialogueManager : MonoBehaviour
{
    public Text nameText;
    public Text dialogueText;
    public Animator animator;
    public Animator mAnim;
    private Queue<string> sentences;
    void Start()
    {
        sentences = new Queue<string>();
    }
    public void StartDialogue(Dialogue dialogue)
    {
```

```
    animator.SetBool("IsOpen", true);
    mAnim.SetBool("Idle", false);
    mAnim.SetBool("Talk", true);
    nameText.text = dialogue.name;
    sentences.Clear();
    foreach (string sentence in dialogue.sentences)
    {    sentences.Enqueue(sentence);
    }
    DisplayNextSentence();
}
public void DisplayNextSentence()
{
    if (sentences.Count == 0)
    {
        EndDialogue();
        return;
    }
    string sentence = sentences.Dequeue();
    StopAllCoroutines();
    StartCoroutine(TypeSentence(sentence));
}
IEnumerator TypeSentence(string sentence)
{
    dialogueText.text = "";
    foreach (char letter in sentence.ToCharArray())
    {
        dialogueText.text += letter;
        yield return null;
    }    }
void EndDialogue()
{
    animator.SetBool("IsOpen", false);
```

```
        mAnim.SetBool("Talk", false);
        mAnim.SetBool("Idle", true);
    }   }
```

Dialogue Trigger:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class DialogueTrigger : MonoBehaviour {
    public Dialogue dialogue;
    public GameObject dialogueText;
    private void Start()
    {
        dialogueText.SetActive(false);
    }
    private void OnTriggerStay(Collider other)
    {
        if (other.gameObject.tag == "Player")
        {
            dialogueText.SetActive(true);
            if (dialogueText.activeInHierarchy == true && Input.GetKeyDown
(KeyCode.F))
            {
                FindObjectOfType<DialogueManager>().StartDialogue(dialogue);
            }       }   }
    private void OnTriggerExit(Collider other)
    {
        dialogueText.SetActive(false);
    }
}
```

Other Scripts:

Quest Marker:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class QuestMarker : MonoBehaviour {
   public float degreesPerSecond = 15.0f;
   public float amplitude = 0.5f;
   public float frequency = 1f;
   Vector3 posOffset = new Vector3();
   Vector3 tempPos = new Vector3();
   void Start()
   {
      posOffset = transform.position;
   }
   void Update()
   {
      transform.Rotate(new Vector3(0f, Time.deltaTime * degreesPerSecond, 0f),
Space.World);
      tempPos = posOffset;
      tempPos.y += Mathf.Sin(Time.fixedTime * Mathf.PI * frequency) * amplitude;
      transform.position = tempPos;
   }
}
```

Teleportation:

```
using UnityEngine;
using UnityEngine.SceneManagement;
public class TeleportToDC : MonoBehaviour {
    public GameObject teleportText;
    public string levelToLoad;
    public GameObject qMarker;
    void Start () {
        teleportText.SetActive(false);
        qMarker.SetActive(false);
    }
    private void Update()
    {
        if (PlayerControl.scrollCount >= 7)
        { qMarker.SetActive(true);    }
        else {    qMarker.SetActive(false);      }
    }
    private void OnTriggerStay(Collider other)
    {
        if (other.gameObject.tag == "Player" && PlayerControl.scrollCount >= 7)
        {
            teleportText.SetActive(true);
            if (teleportText.activeInHierarchy == true && Input.GetKeyDown
(KeyCode.F))
            {
                SceneManager.LoadScene(levelToLoad, LoadSceneMode.Single);
            }       }    }
    private void OnTriggerExit(Collider other)
    {
        teleportText.SetActive(false);
    }      }
```