Universidad Jaume I de Castellón

Design and Development of Videogames Degree

Technical Report of the Final Degree Project

Design and development of a role-playing videogame

with an expandable world

Student: **Salvador Aparici Martínez**

Tutor: **José Miguel Sanchiz Martí**

Summary

This document presents the technical report of the Final Degree Project of the Bachelor's Degree in Videogame Design and Development. The work developed consists of a three-dimensional computer role-playing videogame made with Unity3D, whose world will be easily expandable. The player has to choose among one of the characters offered by the game to start a story that will be told by the non-player characters through quests. These quests consist of defeating enemies, controlled by an artificial intelligence, that roam around the world and, fighting against them, our character will gain experience to level up and learn new combat skills. The player must defeat the enemies to make his character stronger, since each level will increase his statistics and his new skills will allow him to face more powerful enemies. In addition, the world will be able to expand with new quests and enemies in future updates.

Key Words

Unity3D videogame, Three-Dimensional, Role-Playing Game,

Artificial Intelligence, Expandable world

Table of Contents

Table of Figures

1. Introduction

1.1. Motivation

In the Final Degree Project, I have made a 3D videogame with the characteristics of a Role-Playing Game. In this game, the player must choose among one of the five classes of characters available (warrior, guardian, hunter, barbarian and mage), each class will have different statistics (strength, resistance, dexterity, physique and energy) and five skills to use in the fight. After choosing the character, the player will enter a world divided into several zones, some of those zones will be rest areas and others will be adventure zones. In the rest areas, the player will meet the NPCs who will tell him the story of the game and give him quests to perform. In the adventure zones, be will complete the quests defeating the enemies be encounters. By defeating enemies, the character will gain experience and level up, thus improving his statistics and learning new combat skills.

The motivation to do this Final Degree Project comes from my love of Role-Playing Games, in which each person assumes a role within the story and lives it as if he were a different person. It is great to venture into an unknown world, with creatures never seen before and to improve your character. Each person can choose the character that best suits their tastes and play as be likes, leveling up in low-level areas or quickly going to new areas. It's like having another life, a virtual life, in which you can do whatever you want as you want.

I love videogames in which you can choose the character you want and gradually investigate each corner of the world, discovering new places and new types of enemies to face. It is a very pleasant feeling to see how your character gradually evolves, learning new skills and defeating enemies that at first he could not. This game is a great adventure for the player which will cause him to daydream and imagine all the possibilities that can be found within this world.

1.2. Objectives

- Develop a videogame with a world that is easily expandable in possible future updates. This means that once the game is over, it's easy to add more content without having to program more. If the starting game has ten levels, a new level could be added creating a new city with new enemies and everything would continue working the same way.
- Create several characters with different roles, each with their own characteristics, statistics and skills. Creating different characters makes players feel comfortable choosing the character they like best and gives the game the possibility to replay to try out a new character. Also, learning new skills when the character levels up motivates them to continue playing.
- Implement Non-Playable Characters, controlled by Artificial Intelligence, capable of moving and attacking. Inside the cities there will be citizens who will walk the streets, talk and give quests to the player. Outside there will be enemies wandering around the area that will attack the player when they see him.
- Get a balanced combat system, with mathematical formulas to take into account the statistics of the characters in each attack. In RPG games it is very important to have a balanced combat system in which the difference of the chosen character is noticed without some being more powerful than others. This is achieved by using mathematical formulas that fit the game in the most precise way and making the difficulty increase progressively.
- Offer an interesting story to the player, told through quests, that does not become heavy. There are players who get hooked on a game when the story catches their eye and want to keep playing to see how it will end. It is important to have a good story to tell without it being too difficult to understand or too long, so that nobody gets bored playing it.

1.3. Tools

The game has been developed with Unity 2019.3.6f1 with the Operating System Windows 10. Now it is going to be listed all the tools that have been used on this project:

- **Unity 3D** 2019, has been used to develop the videogame.
- Microsoft **Visual Studio** Community 2017, used for scripting.
- Adobe **Photoshop** CC 2018, has been used to make 2D art.
- Microsoft **Word** 2016, used to create the Final Degree Report.
- Adobe **Acrobat Reader** DC, used to export the Final Degree Report.

1.4. Justification

In this section is going to be explained how the project has a relation with some subjects of the degree. This Final Degree Project has been based on five subjects mainly "Game Engines" (VJ1227), "Artificial Intelligence" (VJ1231), "Videogame Art" (VJ1223), "Videogame Conceptual Design" (VJ1222) and "Hyper Media Narrative and Videogames Analysis" (VJ1218).

- **VJ1227 – Game Engines.** The course introduces the basic architecture of a game engine with special emphasis on the graphics engine, physics engine and programming behaviours of game objects. In this course I have learned to use the Unity editor and to program in C#, as well as other useful techniques for making a videogame.
- **VJ1231 – Artificial Intelligence.** This subject introduces students to the AI techniques, enabling them knowing its foundations and, in the meanwhile, providing an updated list of tools used today in the videogames context. In this subject I have learned Artificial Intelligence techniques that I have applied in the NPCs of the game.

- **VJ1223 – Videogame Art.** The creation of art is one of the most creative tasks in the process of creating a videogame. In this subject I have learned to use Photoshop and it has helped me to make the HUD of the game.

- **VJ1222 – Videogame Conceptual Design.** The design and development of videogames involves understanding the wide range of existing game systems, convergence and evolution.

- **VJ1218 – Hyper Media Narrative and Videogames Analysis.** The course is aimed at analysing and understanding the mechanisms that operate in the narrative design of videogames in order to understand them as a playful medium that can develop narratives.

1.5. Project boundaries

This project consists of the design and development of an RPG videogame, but these types of games are very extensive and impossible to perform by a single person in such a short time. For this reason, the work has focused on the programming of the videogame, taking to the background the art of modelling. All the game code has been programmed by me, as well as the 2D art, design and narrative have also been done by me.

On the other hand, for 3D art, animations and sounds, I needed to go to the Unity Asset Store. Those assets have provided me with models of bodies and structures to be able to make the designs of the characters and the worlds of the videogame. In addition, I have also got several animations with which I have been able to create controllers for the characters in the game.

2. Game Design

2.1. Story and Characters

After creating your character (warrior, guardian, hunter, barbarian or mage), you go to the nearest town. Upon arrival, you find a person asking for help because on the outskirts of the town there is a man being attacked by enemies.

You go quickly to that place and you find the man surrounded by enemies. You fight against them and, by beating them, he thanks you and tells you to go talk to him in the next town.

When you go to the next town, you meet again the same man who thanks you again and asks you to help him get some objects he needs so that his people can survive hunger and cold. From then on, you will go from town to town collecting objects and killing enemies to help the poor man and his people.

In the end, you will discover that this person deceived you and was not really saving his people, but helping his army with which he was going to attack the kingdom. To protect your kingdom, you fight against him in a final battle and end up defeating him, thus achieving peace.

The models of the characters have been obtained from the Unity Asset Store, where Synty Studios has created the Modular Fantasy Hero Characters, an asset in which you can create your own characters through different models of body parts.

The world is divided into humans, elves, goblins, undead, and other races. Some are hostile creatures that will not hesitate to attack you when you approach them. Humans are divided into citizens and five different classes:

Citizen: They are Non-Player Character (NPC) who live peacefully in the villages and are dedicated to walking from one place to another.

Warrior: The huge swords of the warriors cause great damage and make them the perfect attackers. Warriors are the characters with more strength and attack, but with less resistance and defense. They focus on close range combat with their sword.

Guardian: Guardians are slow, but they are ideal for creating an impenetrable barrier thanks to their great defense and their physical and magical attacks. The guardians are the characters with more resistance and defense, but with less dexterity and speed. They focus on close range combat with their hammer and shield.

Hunter: Hunters are experts in long-distance physical attacks and also in causing critical damage to enemies. The hunters are the characters with more dexterity and speed, but with less physique and health. They focus on long-distance combat with their bow and arrows.

Barbarian: Despite his slow attack, the giant barbarian's axe hits where it hurts. The barbarians are the characters with more physique and health, but with less energy and magic. They focus on close range combat with their axe.

Mage: The mages' specialty is offensive magic, which mercilessly strikes enemies from afar. Mages are the characters with more energy and magic, but with less strength and attack. They focus on mid-range combat with their magic staff.

2.2. Level and Environment Design

The game consists of an initial scene as a menu, in which you can see the game title and the buttons: "play", "controls", "options" and "exit". In addition, the game has another ten scenes, one for each level. In each scene, the enemies have a different level, so the game becomes more difficult with each scene you complete.

The game consists of the following scenes:

- **Scene 0:** Initial scene in which we see the logo of the game, the title and a menu with several buttons: “play” (which takes you to the next scene), “options” (which shows the sound controls) and “exit” (to close the application).
- **Scene 1:** Scene in which we begin to play and in which a small tutorial teaches us the basic controls of the game to learn how to move and fight against different enemies. In this scene the enemies are level 1 (the minimum level).
- **Scenes from 2 to 9:** Scenes in which our goal is to complete different quests such as leveling up, collecting some objects, killing several enemies or defeating the boss of the area. In each scene the enemies have a different level: in scene 2 the enemies are level 2, in scene 3 they are level 3, and so on.
- **Scene 10:** Scene in which the enemies are level 10 (the maximum level) and we have to fight the final boss of the game. In this scene we are told the end of the story, but, even if we complete it, we can continue playing.

From scene 0 we can go to scene 1 by pressing the "play" button. Between the scenes from 1 to 10 we can move freely as long as we have the quests completed. The goal is to level up bit by bit while we explore new areas of the game.

The models of the different objects with which the world levels have been formed have been obtained from the Unity Asset Store, where Synty Studios has created the assets Nature Pack, Adventure Pack, Vikings Pack, Knights Pack and Samurai Pack, which have different structures with which you can design cities and exteriors of different types.

Fig. 1 – Levels design

The design of the environment of each scene is similar. Each scene is divided into: the entrance, the town, the outside, the boss zone and the exit. The entrance is a path where you (blue) appear when you move to that scene and it is also the place you have to go to return to the previous scene, it is located near the town or in the town itself. The town is a safe place with houses and roads where the NPCs (yellow) walk and where you can get new quests (orange). The outside of the town is full of trees, rocks and enemies (red) that roam the area and attack you if you get too close to them. The boss zone is a space outside the town where the boss (purple) of the area is located, making it the most dangerous place. The exit is a path that is somewhere outside the town and takes you to the next scene, it is also the place where you appear when you return from a more advanced scene. In each scene the distribution of the elements is different, but the design of the environment is similar.

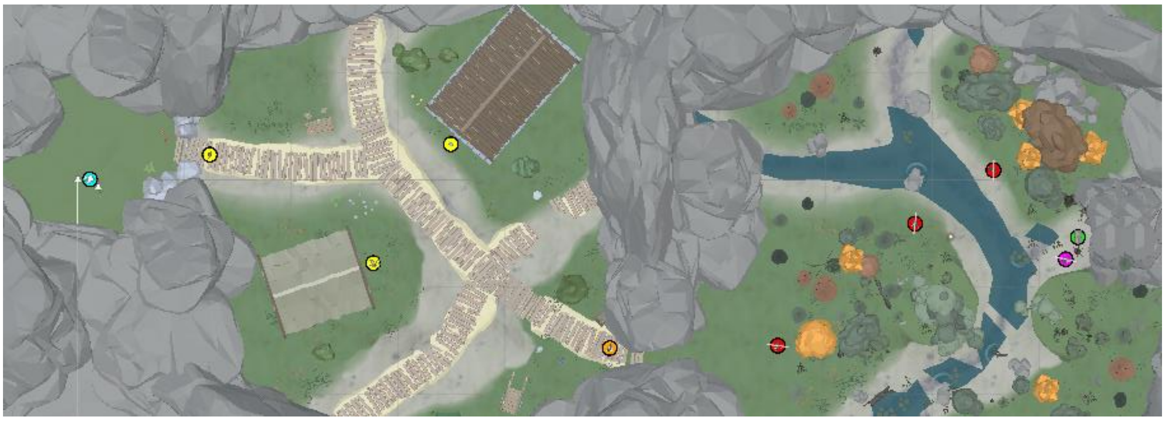


Fig. 2 – Environment design

2.3. Gameplay

Upon entering the game, the first scene is shown, in which we can see the logo and the title of the game and a menu with buttons to start playing, modify the sound options and exit the game. By selecting the option to start playing, we can choose our character from five different classes: warrior (has a lot of strength), guardian (has a lot of resistance), hunter (has a lot of dexterity), barbarian (has a lot of physique) and mage (has a lot of energy).

After choosing our character, we appear at the entrance of the first scene. At this moment a tutorial appears to learn how to move our character. Once the first tutorial is completed, we move on to the first town and there an NPC tells us that there is someone in danger outside the town. We continue advancing and, when leaving the town, another tutorial appears, this time to learn to attack with our character. After completing the last tutorial, we find an NPC surrounded by enemies, which we have to defeat with everything we have learned in the different tutorials. By saving the NPC, he thanks us and leaves. At this point, we can continue exploring the area by defeating enemies to level up or go to the exit to advance to the second scene. Upon reaching the second town, we meet the same NPC, who tells us to help him get some objects.

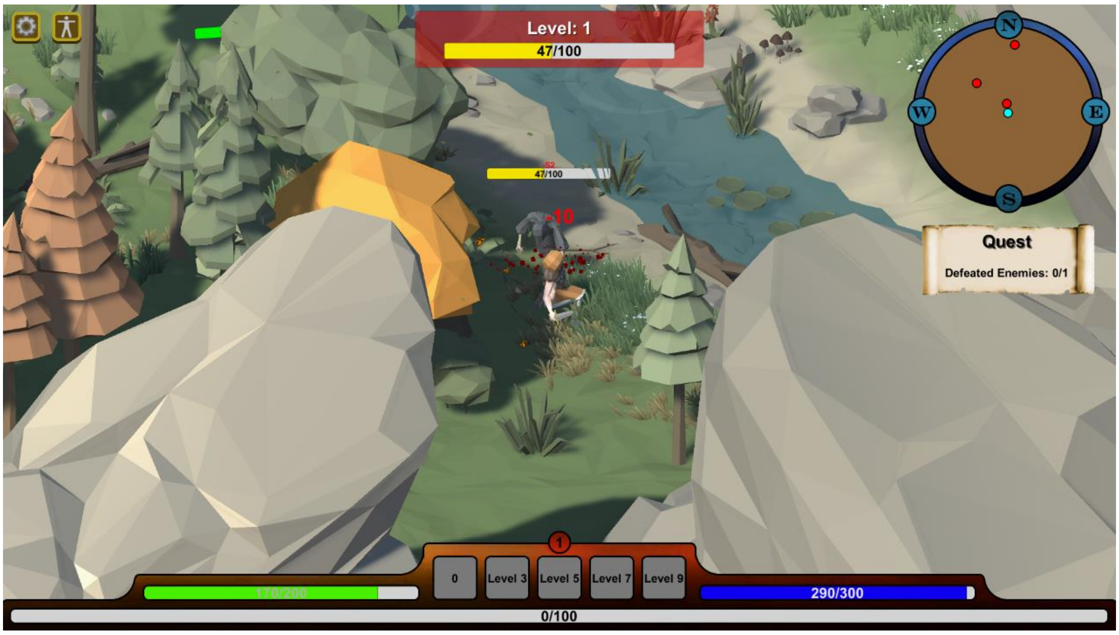


Fig. 3 – Gameplay

From here, our goal is to complete all the quests that the NPC gives us in each of the ten towns. There are different types of quests: leveling up, collecting some items, killing various enemies, defeating the boss of the area, and so on. As we level up, our character's statistics improve and at certain levels we learn new skills: we start the game with one skill, at level 3 we learn the second skill, at level 5 we learn the third skill, at level 7 we learn the fourth and at level 9 the fifth and last. Upon reaching the last town, we face the final boss of the game, the most powerful enemy of all. After beating it, we can continue exploring all the scenes of the game to reach a higher level.

At any time in the game we can pause it to rest or modify some sound options. In the future, new towns and new enemies may be added to expand the game.

2.4. Sound and Music

The different sounds and music of the game, as with art, have been obtained mostly from the Unity Asset Store, the sounds were taken from the asset RPG Voices and the music from the asset Action RPG Music Free. In the main menu, before you start playing, you can configure the options for sound and music.

There is a sound for each action within the game, such as pressing a menu button, attacking with our character or using a skill. There are also sounds for when the character is injured, dies, levels up, picks up an object, completes a quest, and so on.

Regarding the music within the game, there is a different theme for each situation, which creates the perfect environment for the moment. There is a music theme for the initial menu and different themes for the town, the outside, fights against enemies, and so on.

In addition, at any time you can open the pause menu and change the volume of sound and music. The sounds and music make you feel calm in the town, outside you feel like exploring and in a fight you get a little nervous.

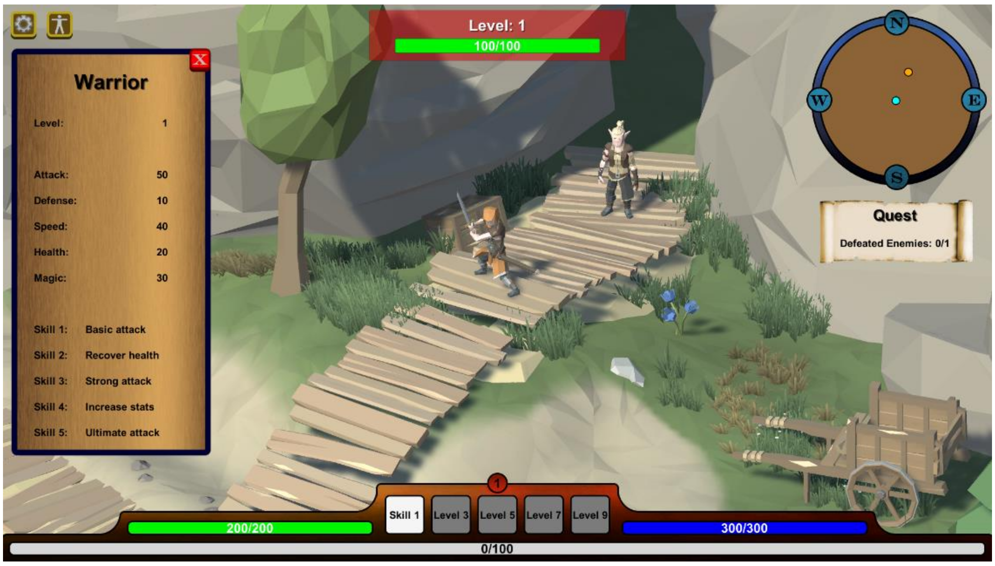
2.5. User Interface and Game Controls

The game's User Interface is quite simple. In the initial scene we can interact with the buttons offered by the menu: "play" to move to the next scene, "controls" to see how to play with the keyboard and mouse, "options" to configure the sound volume and "exit" to close the game.

In game scenes, the UI consists of the information that the game gives the player. In the lower central part of the screen the game shows us our bars of life, magic and experience, as well as the current level of our character.

In addition, next to these bars there are also five icons that represent the five skills of our character, at the beginning only one of the skills is visible, but as you increase your level the rest are unlocked. On the other hand, the different enemies will also have a life bar and an indicator of their level just above each of them.

In the upper left corner, we have the mini map and information about quests, such as how many objects we have left to collect or how many more enemies we must defeat. Finally, in the upper right corner, we have a button to open the pause menu and the character information panel, although it can also be opened with a keyboard shortcut.



Warrior

Level:	1
Attack:	50
Defense:	10
Speed:	40
Health:	20
Magic:	30
Skill 1:	Basic attack
Skill 2:	Recover health
Skill 3:	Strong attack
Skill 4:	Increase stats
Skill 5:	Ultimate attack

Quest

Defeated Enemies: 0/1

Fig. 4 – User Interface

The game is controlled with the mouse and keyboard. All interaction with the menus and buttons is done with the mouse, selecting with the left button. In the game scene, we use the keys W, A, S and D of the keyboard to move around the world and the mouse to rotate our character and point in the direction we want. To launch the different skills of our character we have the keys 1, 2, 3, 4 and 5 that respectively launch the five skills we have. With the space bar we can talk to the NPC that gives us the quests and collect objects. In addition, with the C key we can open the character information panel. Finally, with the escape key we can open the pause menu to go to the options.

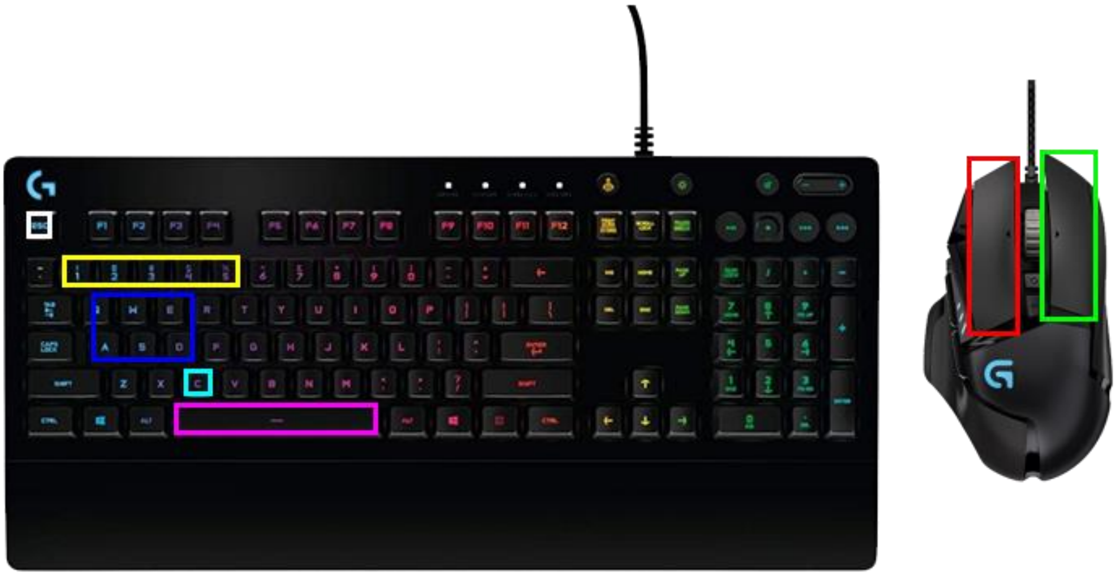


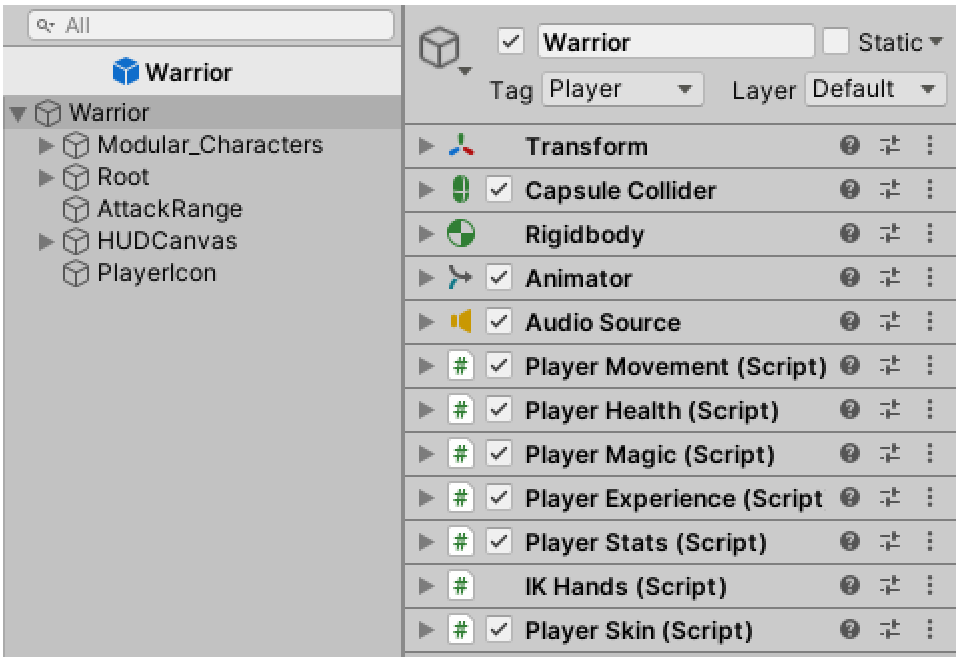
Fig. 5 – Game controls

All it takes to play this game is a computer with a mouse and keyboard. Being a simple game, it is not necessary that the computer we play with is very powerful, so most people can play it without problems. The game is entirely in English, so that it can be understood by the majority of the population worldwide and that more people can play it. Anyway, being a game with basic instructions, it is not necessary to have a very high level of language to understand everything. The game presents a difficulty for all audiences, at first the game is very easy and little by little it increases its difficulty. Thanks to its progression of difficulty, anyone can play it from beginning to end without problems, so it will reach a higher audience.

3. Game Development

3.1. Character programming

Being an RPG-style videogame, the character we play with has most of the code. The character needs to have well programmed stats, movements and attacks so that the player feels more immersed in the game. All programming is done from scratch, no template with part of the code has been used. Next, the development of the character with which we play will be shown in steps.



All

Warrior

- Warrior
  - Modular_Characters
  - Root
  - AttackRange
  - HUDCanvas
  - PlayerIcon

☑ Warrior □ Static

Tag Player

Layer Default

Transform

☑ Capsule Collider

Rigidbody

☑ Animator

☑ Audio Source

☑ Player Movement (Script)

☑ Player Health (Script)

☑ Player Magic (Script)

☑ Player Experience (Script

☑ Player Stats (Script)

IK Hands (Script)

☑ Player Skin (Script)

Fig. 6 – Character scripts

- **Player Manager.** When starting a new game, it is necessary to create a new character and for this the Player Manager is used. This script is called from the buttons on the character selection canvas. Once called, the character's gender and class are selected. For each class, its statistics are set in the Player Stats script, which will be seen later. Then the character is instantiated in the starting position and scene scripts are enabled.

- **Player Stats.** This script has all the information on the player's statistics and updates them. You can get the character's gender, the chosen class and also its strength, resistance, dexterity, physique and energy. In addition, it calculates attack, defense, speed, health and magic by multiplying its statistics by a variable and by the level of the character, which is obtained from the Player Experience.
- **Player Experience.** Script in charge of updating the level and experience of our character. The experience required to level up is $100 \times 2^{level-1}$ where level is the current level. This is my own formula obtained by studying various RPG games. This script is also the one that shows the experience bar, the current level, the current experience and the maximum experience on the screen. The Take Experience function is called every time an enemy is defeated and is responsible for adding the experience gained from the enemy to our current experience. If our experience exceeds the maximum experience, the level is raised and the experience returns to zero. In addition, an audio clip corresponding to level up sounds, the text “Level up!” is created to show it on screen with a function of the Player Health script that will be seen later and the appearance of the player is changed with a function of the Player Skin script, which will also be seen later. Finally, the value of the experience bar is updated and the Change Color function is called, which changes the color of the bar and of the text according to the percentage of experience that you have with respect to the total, to make progress visually more striking.
- **Player Skin.** It is the script used to change the appearance of the character as you level up, to give the player a sense of progress. First, the Build Lists function is called to rebuild all item lists by scrolling through the objects to find the parts of the character. Then the Player Skin script disables any enables objects, clears the list of enabled objects and sets the default level 1 selected character by calling the Activate Item function which enables added items. Finally, the Level Up function, which is called from Player Experience every time you level up, disables the enabled objects, clears the list and

activates the objects corresponding to the character level by calling Activate Item.

- **Player Health.** This script is in charge of controlling the health of the player. Every time the player takes damage, a red flash is shown on the screen. At the beginning, the maximum and current health and the maximum and current value of the health bar, which is shown on the screen, are calculated thanks to Player Stats. The Take Damage function is called every time the player takes damage. First, an animation of receiving a hit is shown, then the damage received is calculated by applying the formula $\frac{attack \times attack}{attack + defense} \times 2$ with attack being the enemy attack and defense the defense of the player, this is my own formula obtained from the study of different damage formulas in RPG games. Damage is subtracted from current health, the health bar value is updated, and the audio clip of taking a hit is activated. In addition, the functions Create Text, Change Color and, if the player loses all his health, Death are called, which will now be explained. The Take Health function is similar to the Take Damage function but it serves to heal our character if we use a healing skill. The health recovered is added to the character's health, the health bar is updated and the Create Text and Change Color functions are called. The Death function, called by Take Damage if health reaches zero, activates the death animation, turns on the death audio clip, and disables player movement. The Regeneration function is called after a period of time without taking damage and its purpose is to restore the player's health gradually and then update the health bar and call the Change Color function. The Change Color function changes the color of the health bar and the health text depending on the percentage of health that you have with respect to the total. Finally, the Create Text function, which is also used from the Player Experience script, creates a specific text above the player with a size and color assigned to it depending on the type of data that is displayed and with an animation it makes that the text moves upwards and becomes smaller until it disappears.

- **Player Magic.** It is a script very similar to Player Health, but it is in charge of controlling the player's magic. It has Take Magic, Regeneration and Change Color functions, which work just like the respective functions in Player Health. Take Magic is called every time the player uses a skill and loses some of its magic depending on the type of skill used.
- **Player Movement.** This is the script in charge of the player's movement. The Move function makes the player move in the direction which corresponds to the key pressed. Movement speed is calculated with the formula $3 + \frac{speed}{100}$ where speed is the speed of the Player Stats script. Also, speed is halved if the player is attacking. The Turning function is responsible for rotating the player in the direction the mouse pointer is pointing using a raycast that goes from the camera position to the pointer position and hits the ground layer. Animating is the function that activates the player's movement animations. If the player is moving, the motion animation will be activated with a speed of $0.6 + \frac{speed}{500}$ where speed is the player's speed. In addition, the speed in X and Z is calculated for the type of motion animation. Pressing the space key will activate the animation of picking up an object from the ground. The On Trigger Enter function causes the pick up an object audio clip to sound when colliding with an object, adding an object in the Score Manager script, and destroying the object. If it collides with an invisible level charge object, it will change the scene. Set Attacking is a function that tells whether the player is attacking or not. Finally, the Hit and Shoot functions are called from some of the attack animations, and they launch a message to call functions of the Attack script to deal damage to the enemy. This script and its functions will be seen at a later point.
- **Camera Follow.** It is the script that allows the camera to follow the player. The camera position is interpolated between your current position and the player's position in a progressive way, to give more realism to the movement. In addition, this script is also responsible for making the objects that stand between the camera and the player semi-transparent, so that the player remains visible, even if there is an object that covers it. A raycast is created

from the camera position to the player position. For each hit collider object if it has a renderer component, a transparent shader is applied to its material and the alpha of its color is halved. When the raycast stops hitting that object it goes back to the standard shader and the alpha becomes one again.

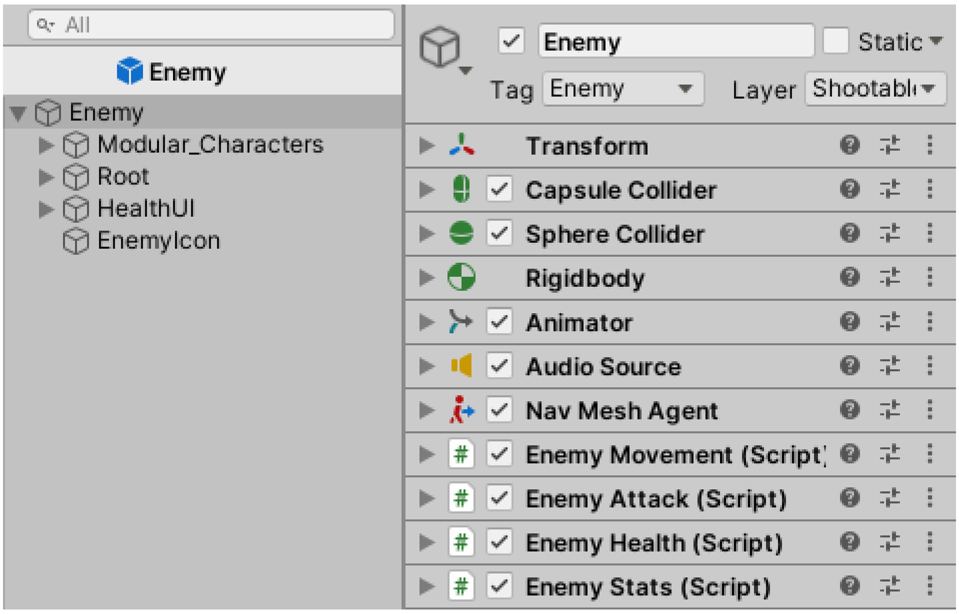
- **Floor Follow.** This script is placed on an invisible plane object with the ground layer and ensures that the plane follows the player. This plane is the one used for the Turning function of the Player Movement script to work correctly. The player will rotate in the direction of the mouse pointer using a raycast from the camera position to the pointer position and hitting the invisible plane with the ground layer. This script is used to correct problems on terrains that are not flat and have variations in the vertical axis, causing the player to not turn properly.
- **IK Hands.** This is the script in charge of uniting the left hand of the Warrior with the hilt of the sword he is holding with his right hand, in order to make him take the sword in two hands, using the On Animator IK function.
- **Character Attack.** This script controls the five skills of each character and, since each character has different skills, the script has some variations depending on the character. First, the time between character attacks is calculated with the formula $1.5 - \frac{speed}{500}$ where speed is the player's speed obtained from the Player Stats script. Also, the time to use each skill is calculated in a similar way with the formula $x - \frac{speed}{50}$ where x is a constant for each skill. Then, the Button Information function is called, which is responsible for putting information on the buttons of the canvas, if the skill has not been unlocked it shows the level at which it is unlocked, if it has already been unlocked it shows the name of the skill, if the skill is unlocked but not yet usable, it displays the remaining time to use it again. Pressing the left mouse button creates a raycast that goes from the camera position and advances through the mouse position. If it collides with an object in the trigger layer, that is, if you click on an enemy, that enemy is selected and its information panel is displayed by calling the Show Enemy Panel function of the Enemy Panel Manager script, which will be seen in the section Artificial

Intelligence programming. If you click where there is no enemy, the enemy is deselected and the information panel is hidden by calling the Hide Enemy Panel function, also from the same script. After that, the health of the selected enemy is obtained from the Enemy Health script and the information panel is updated by calling the Update Enemy Panel function of the Enemy Panel Manager script. If the enemy's health reaches zero, its information panel is hidden. Pressing the number buttons selects the corresponding skill as long as the player has the level necessary to use it. If the time between attacks has passed, the time to cast the selected skill and the player has enough magic it will call the Skill function of that skill. There is a Skill function for each skill of each character. Skill 2 and Skill 4 are common to all characters. Skill 2, which is a healing skill, activates the healing animation, calls the Take Health function of the Player Health script which is responsible for healing the player, activates the healing sound and calls the Take Magic function of the Player Magic script, which is responsible for reducing the player's magic depending on the skill used. Skill 4, which is a skill that increases the player's statistics for a while, activates the stat increase animation, updates the stat to modify depending on the character from the Player Stats script, starts a Wait Seconds coroutine that waits a few seconds and resets the modified statistic to its default value and finally turns on the stat boost sound and calls the Take Magic function of the Player Magic script. Skill 1, Skill 3 and Skill 5 functions are attack skills that activate the corresponding attack animation, set the attack range and damage multiplier, activate the corresponding attack sound, call the Set Movement function of the Player Movement script, to slow movement speed, and call the Take Magic function of the Player Magic script to reduce player magic. In melee attacks, the Skill Hit function is called from the Player Movement script's Hit function, which is called from the attack animation. This feature creates a sphere-shaped collider in front of the player with a specific radius of the skill. For each object that collides with the sphere and is in the trigger layer, the Take Damage function of the Enemy Health script is called by passing the damage multiplier of the skill, a function that is responsible for subtracting health from enemies and that will be seen ahead. In ranged attacks, the Skill Shoot function is called from the Player Movement script's Shoot function, which is called from the attack animation. This function creates a new instance of the attack object and a speed is applied to it. In this object there is a Shot script that destroys the object after a while and that has an On Trigger Enter function that when hitting an enemy calls the Take Damage function of the Enemy Health script to subtract health from the enemy.

- **Player Panel Manager.** This is the script which shows information about the character you are playing with. Pressing the corresponding button calls the Show Player Panel function that activates the information panel for the selected character. The character's class, level, stats, and skill information are continuously updated in the script.
- **Score Manager.** Displays the player's score information, such as defeated enemies and collected items.
- **Game Over Manager.** When the player dies this script is called. It is the script that is responsible for ending the game and displaying a Game Over panel with an animation. If the player's health reaches zero, the Game Over animation jumps and after a while the current scene restarts.

3.2. Artificial Intelligence programming

In RPG games there are always a wide variety of enemies to be defeated and NPCs that give quests to the player or simply walk around the city. These enemies and NPCs work thanks to Artificial Intelligence, which allows them to move, attack or speak depending on the situation. At this point, we will see the programming done for the enemies and the NPCs of the game.



☑ Enemy □ Static
Tag Enemy
Layer Shootabl

▶ Transform
▶ ☑ Capsule Collider
▶ ☑ Sphere Collider
▶ Rigidbody
▶ ☑ Animator
▶ ☑ Audio Source
▶ ☑ Nav Mesh Agent
▶ ☑ Enemy Movement (Script)
▶ ☑ Enemy Attack (Script)
▶ ☑ Enemy Health (Script)
▶ ☑ Enemy Stats (Script)

Fig. 7 – Enemy scripts

- **Enemy Manager.** This script is the one in charge of creating the enemies and putting them on the scene. First, the Spawn function, which is called from time to time, finds all the enemies in the scene, and if the current number of enemies equals the maximum number of enemies for that scene, the function ends. If not, the Select Stats function is called, which updates the Enemy Stats script with that enemy's statistics information. After that, a random position and rotation is obtained within the limits of the terrain where enemies can appear. Next, a raycast is created from above the enemy's position downward, and if it hits with the ground layer it changes the enemy's position to a position above the ground. This is useful for terrains with different heights, so that enemies always appear on the ground and not inside a mountain or flying through the sky. Once a correct position is found, the enemy is created on the scene. There are two other functions that behave similarly called Spawn Boss and Select Stats Boss to create the final boss of the scene.

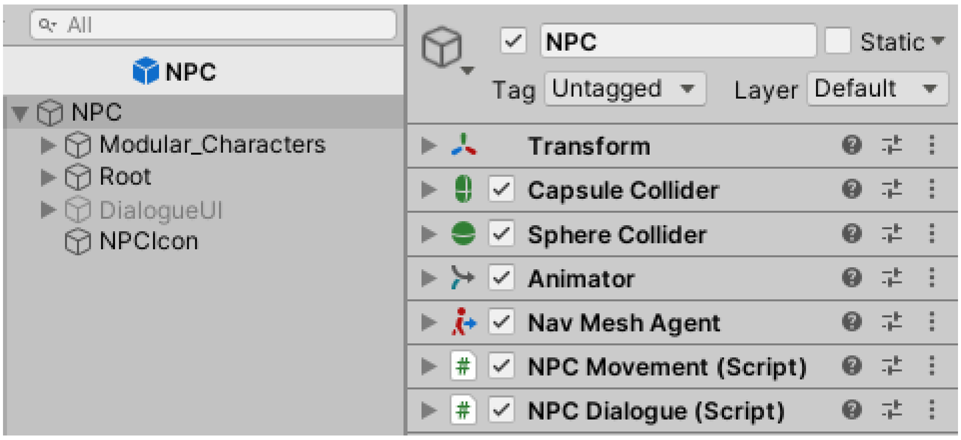
- **Enemy Stats.** This is the script that controls the statistics of the enemies, such as their level, strength, resistance, dexterity and physique. Attack, defense, speed, and health are calculated by multiplying the enemy’s base stats by a variable and its current level.
- **Enemy Health.** This is the script in charge of controlling the health of the enemy. First, the current and maximum health of the character and the health bar are established thanks to the Enemy Stats script and the health text is fixed to show it on the screen. The Take Damage function is called every time an enemy takes damage. The formula for calculating the final damage that the enemy takes is my own formula that has been obtained after studying several RPG games and their combat formulas, this is $\frac{attack \times attack}{attack + defense} + \frac{magic \times magic}{magic + defense} \times skill$ with attack being the player's attack, defense the defense of the enemy, magic the player's magic, and skill the damage multiplier of the player's skill. Then, the probability of giving a critical hit or failure of the attack is calculated using the speed of the Player Stats, if a critical hit is given the final attack has 50% more damage, if the attack fails the damage is zero. Depending on the result, an audio clip and text are assigned with the Create Text function. The player's damage is then subtracted from the enemy's health and health bar, the Change Color function is called, and if the enemy's health reaches zero, the Death function is called. The Death function activates the animation and the death audio, and calls the Start Sinking function, which first adds points in the Score Manager script, then calls the Take Experience function of the Player Experience script to grant the player the experience for defeating the enemy and finally, it causes the enemy to sink and destroy itself. The Change Color

function is used to change the color of the text and the health bar depending on the proportion of the enemy's remaining health to the total. The Create Text function shows on the screen a text with the damage received to which an animation is applied so that, after it appears, it moves upwards while it becomes small and finally disappears.

- **Enemy Movement.** This script controls the movement of the enemy. At the beginning, if the enemy is not well placed in the Nav Mesh, it is destroyed, if not, the Set Move Range function is called to set the range of movement that the enemy has through the scene. The speed of the enemies is calculated with the formula $3 + \frac{speed}{100}$ where speed is the speed of the enemy obtained from the Enemy Stats, if the enemy is attacking the speed is reduced by half. Then, the Set Animation function is called, which is responsible for activating the animations depending on the movement of the enemy. If the player is in the enemy's range of vision, the enemy begins to follow the player, if not, the Random Movement function is called, which sets a random position within the established limits. A raycast is thrown from above the enemy's position downwards, and if it hits the ground layer it changes the enemy's destination to a position above the ground, so there are no problems with terrain of various heights. With the On Trigger Enter function it is known if the enemy collides with an invisible wall placed at the entrance to the city. If so, the Random Movement function is called for the enemy to patrol again, and then the Wait function is called to wait a while before chasing the player if they are close to him.
- **Enemy Attack.** This is the script that controls the attacks of the enemies. With the On Trigger Enter and On Trigger Exit functions it is known if the player is in range for the enemy to attack him. If the player is in range, the enemy turns towards him and, if the entablished time passes between attacks, calls the Attack function. In addition, if the player reaches zero health, the animation of the dead player is activated. The Attack function calls the Set Attacking function of the Enemy Movement script to halve its

speed, activates enemy attack audio, and calls the Take Damage function of the Player Health script to detract from its health.

- **Billboard.** This is the script that makes the health bar located above the enemies face the camera and ensures it is correctly oriented, in order to see the enemy health from any position.
- **Enemy Panel Manager.** This script is used to display enemy information on screen when selected. With the Show Enemy Panel feature, which is called when selecting an enemy, a panel is displayed at the top of the screen with the enemy's level and its health bar. The Hide Enemy Panel function, called when selecting something other than an enemy or when the selected enemy dies, hides the information panel. On the other hand, the Update Enemy Panel function updates the information displayed on the screen taking the data from the Enemy Health and Enemy Stats scripts. It updates the maximum and current value of the health bar, the color of the health bar, the text and the color of the amount of health and the text of the enemy's level.



NPC
- Modular_Characters
- Root
- DialogueUI
- NPCIcon

☑ NPC ☐ Static
Tag Untagged Layer Default

- Transform
- ☑ Capsule Collider
- ☑ Sphere Collider
- ☑ Animator
- ☑ Nav Mesh Agent
- ☑ NPC Movement (Script)
- ☑ NPC Dialogue (Script)

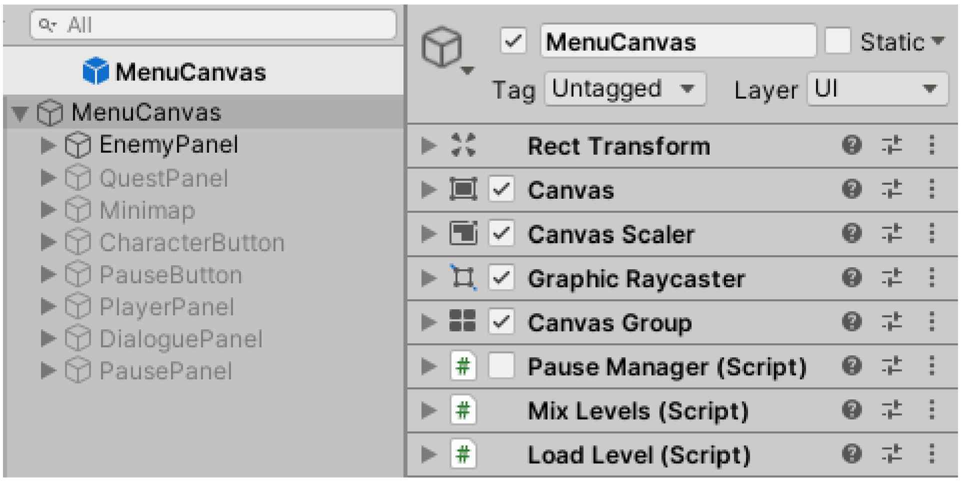
Fig. 8 – NPC scripts

- **NPC Manager.** Script in charge of creating the NPCs and placing them on the scene. The Spawn function, called upon when entering a new scene, creates multiple NPCs depending on the scene. First, a random position and rotation is selected. Thereafter, a raycast is created from above the NPC position downward, and if the beam strikes the ground layer, it changes the NPC spawn position to a position above the ground, to avoid problems in scenes with elevated terrain. Finally, the NPC is instantiated in the correct position and rotation.

- **Quest Manager.** Among the various NPCs are some that give quests to complete in the scene they are in. Show Quest is the function that shows the quest on the screen, Hide Quest hides the quest and Activate Quest is responsible for activating the quest.
- **NPC Movement.** This is the script that is in charge of the movement of the NPCs. If the NPC is not properly positioned in the Nav Mesh it is destroyed, if not, the Set Move Range function is called to set the limits of the movement of the NPC. The speed of the NPC is fixed and every once in a while it is assigned a new random position to go there. A raycast is created equal to that of the NPC Manager script to select a new position, thus avoiding movement problems in terrains with slopes and height variations. Once the destination is correct, the NPC is sent to it and the Set Animation function is called, which is responsible for animating the movement of the NPC based on its actions.
- **NPC Dialogue.** NPCs say random phrases when the player gets close enough to them. This script selects one of the available random phrases and then with the On Trigger Enter and On Trigger Exit functions shows and hides them respectively.
- **NPC Quest.** This is the script to show and hide the quest. With the On Trigger Enter and On Trigger Exit functions it calls the Show Quest and Hide Quest functions of the Quest Manager script to show and hide the quest.
- **Item Manager.** This script is in charge of controlling the objects that the player must collect to complete the object collection quest. The Spawn function creates various objects depending on the scene. It sets a random position and rotation first, then instantiate the object.

3.3. Head-Up Display and Graphical User Interface

RPG games need to display a lot of information on the screen, such as the data of the character being played, be it their level, health, magic, experience or skills. It is also important to show information about the enemies, a mini map, the quests that the player has to carry out, the damage from the attacks or the level up. At this point, the scripts involved in the menus will be explained first, then the menus with which the player can move between scenes and, finally, the information displayed on the screen when playing a game.



All

MenuCanvas

- MenuCanvas
  - EnemyPanel
  - QuestPanel
  - Minimap
  - CharacterButton
  - PauseButton
  - PlayerPanel
  - DialoguePanel
  - PausePanel

☑ MenuCanvas □ Static

Tag Untagged Layer UI

- Rect Transform
- ☑ Canvas
- ☑ Canvas Scaler
- ☑ Graphic Raycaster
- ☑ Canvas Group
- □ Pause Manager (Script)
- Mix Levels (Script)
- Load Level (Script)

Fig. 9 – Menu scripts

- **Load Level.** This is the script in charge of changing between scenes. It has a function for each different scene and another to exit the game. These functions are called from the menu buttons and load the selected scene.
- **Pause Manager.** This is the script that controls the game pause menu. Pressing the button or the pause key calls the Pause function which activates the pause panel and stops the game. If the game is paused and the corresponding button or key is pressed, the Pause function deactivates the pause panel and the game resumes.
- **Mix Levels.** This is responsible for controlling the sound of the game from the pause menu. The Set SFX Level function sets the current volume of the sound

effects through the sound bar in the pause menu. The Set Music Level function sets the current volume of the music through the music bar of the pause menu. The Mute function mutes all sounds in the game or, if they are already muted, turns them on again.

When you open the game, the first scene appears with the title and the Play, Controls, Options and Exit buttons. The Exit button is used to exit the game. The Options button takes you to the options screen where you can change the sound settings. The Controls button takes you to the controls screen where you can see how to play with the mouse and keyboard. The Play button takes you to the character selection screen where the player can choose their character from Warrior, Guardian, Hunter, Barbarian or Mage and can also choose if their character is Male or Female. After selecting the character, you switch to the Level 01 scene and as you progress through the game you will change the scene until you reach the last scene. You can press the button or the pause key at any time, which will pause the game and open the pause panel. The pause panel contains a title indicating that the game is paused, two volume control bars for music and sound effects, a button to mute or re-activate the sound, and two buttons at the bottom to resume the game and to exit to the title screen. In addition, when talking to the NPC of the quests, the panel of the quest will open, which contains the title of the quest, information that the NPC gives us, the requirements to complete the quest and the Accept button to accept the quest and close the panel.

While playing, a lot of information is displayed on the screen to help the player. At the bottom of the screen is a frame that displays character information. At the bottom of the frame is the experience bar, which occupies the thickness of the screen. Above the experience bar is the health bar and magic bar, of a smaller width. Between these two bars are the character's five skills, each in a square. Above skills, inside a circle, is the level of the character. At the top right of the screen is the mini map. The mini map frame is a double circle with four other circles indicating north, east, south, and west of the map. Inside the mini map are icons for the player, NPCs, enemies, and items. Below the mini map there is a small panel that shows the quest currently active, the objectives to be met and the progress the player takes to complete the quest. When an enemy is selected, a panel with the enemy's level and its health bar is displayed at the top center of the screen. At the top left of the screen are two buttons, one of which is used to open the pause menu and the other is used to open the player panel. The player panel opens on the left side of the screen and displays detailed information about the character, such as his class, level, stats, and skills. Also, when approaching an NPC, a small dialog will be displayed above his head. Enemies will always have their health bar on their heads. Finally, when in combat, information will be displayed on the screen such as the damage caused, the critical hits, the failed hits, the damage received, the healing received and the leveling up of the character.

All the HUD and GUI art is done with Adobe Photoshop CC 2018 using the following steps. First, a quick sketch was made to determine the shape and dimensions of the figure. Second, the figure was outlined by following the sketch and adding details such as the text. Third, the figure was colored, adding textures, shadows and lights to make it more striking. Finally, everything was implemented in the game and carefully placed in the proper position. The following image shows the four steps mentioned above.

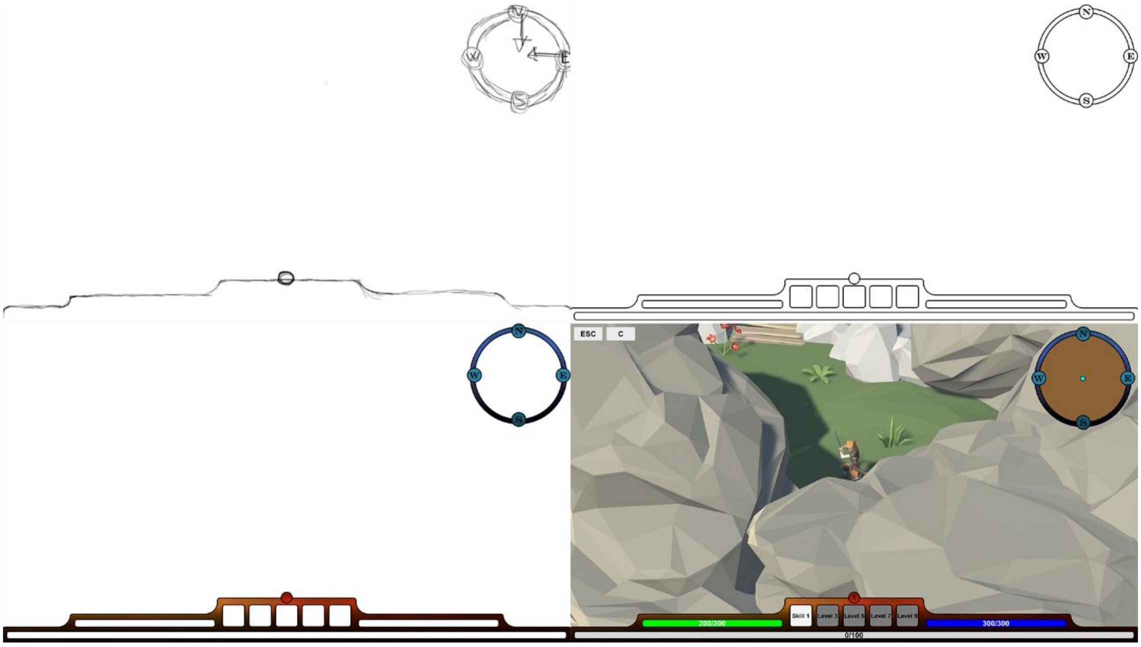


Fig. 10 – HUD evolution

3.4. Character design and animation

Characters are a fundamental part of RPG games because the player has to feel comfortable with the character he is controlling, therefore the characters have to have good design and good animation that is different depending on the class that the player has chosen. The models of the characters have been obtained from the Unity Asset Store, where Synty Studios has created the Modular Fantasy Hero Characters, an asset in which you can create your own characters through different models of body parts. With these models, it was possible to create a hundred different characters, since each character can be Male or Female, both have the five classes Warrior, Guardian, Hunter, Barbarian and Mage, in addition, each of these ten characters have ten different forms when they go up from level one to level ten.

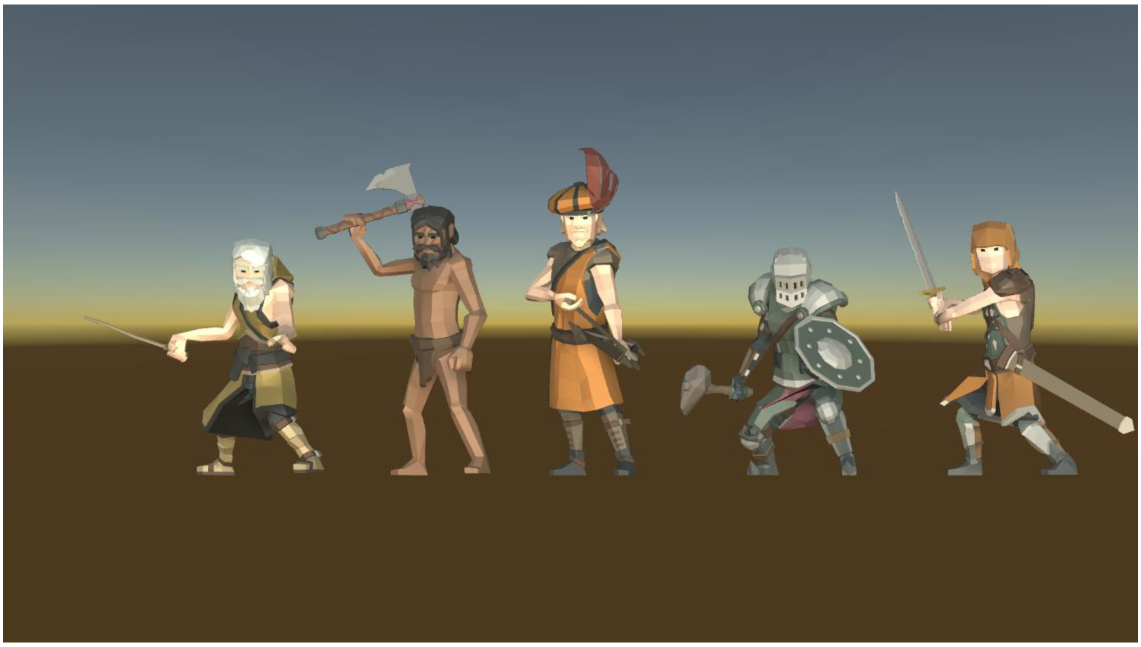


Fig. 11 – Characters at level 1

The Warrior, who is a character focused on strength and attack and attacks at close range melee, has been equipped with a two-handed sword, a sword holster and medium armor. The Guardian, who is a character focused on resistance and defense and attacks at close range melee, has been equipped with a mace, shield, and heavy armor. The Hunter, who is a character focused on dexterity and speed and attacks at a long distance with arrows, has been equipped with a bow, arrows and light armor. The Barbarian, who is a character focused on physique and health and attacks at close range melee, has been equipped with an axe and animal skins. The Mage, who is a character focused on energy and magic and mid-range attacks with magic, has been equipped with a staff and light clothing.

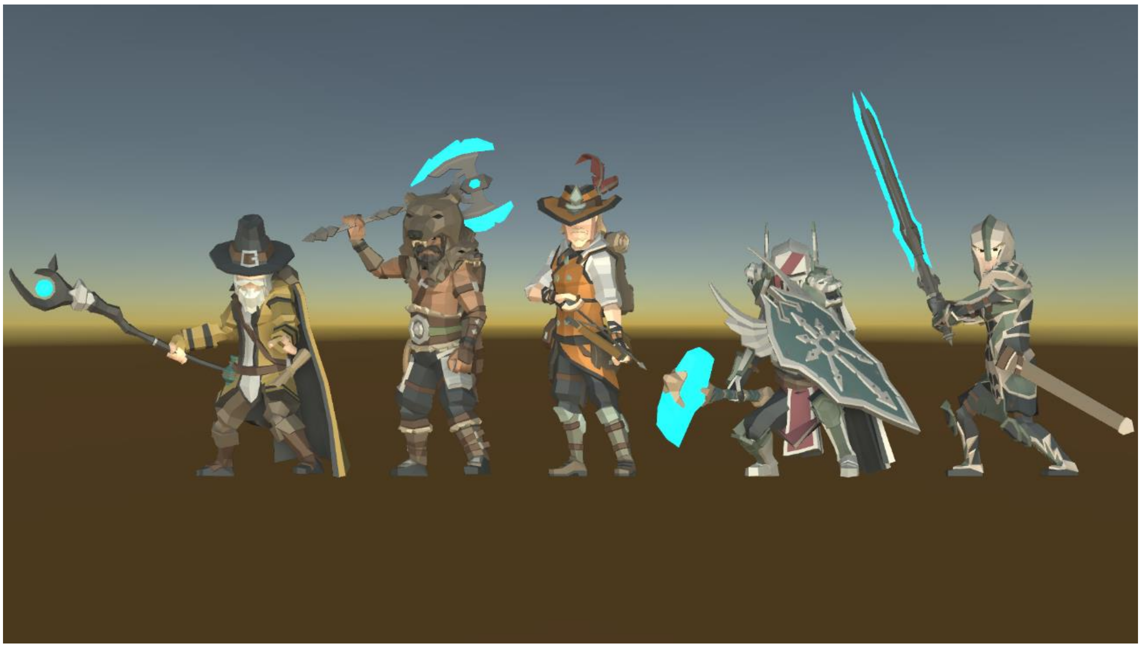


Fig. 12 – Characters at level 10

The characters start at level one with the basic equipment of their class and each time they level up their equipment improves. With each level you climb, your weapons become stronger and your armor becomes more resistant. Level ten is the maximum level and the characters carry their final gear, indicating that they can no longer become stronger. Below is an image with the ten levels of the five character classes. The image only shows the Male characters, but putting the Female characters would have twice as much.

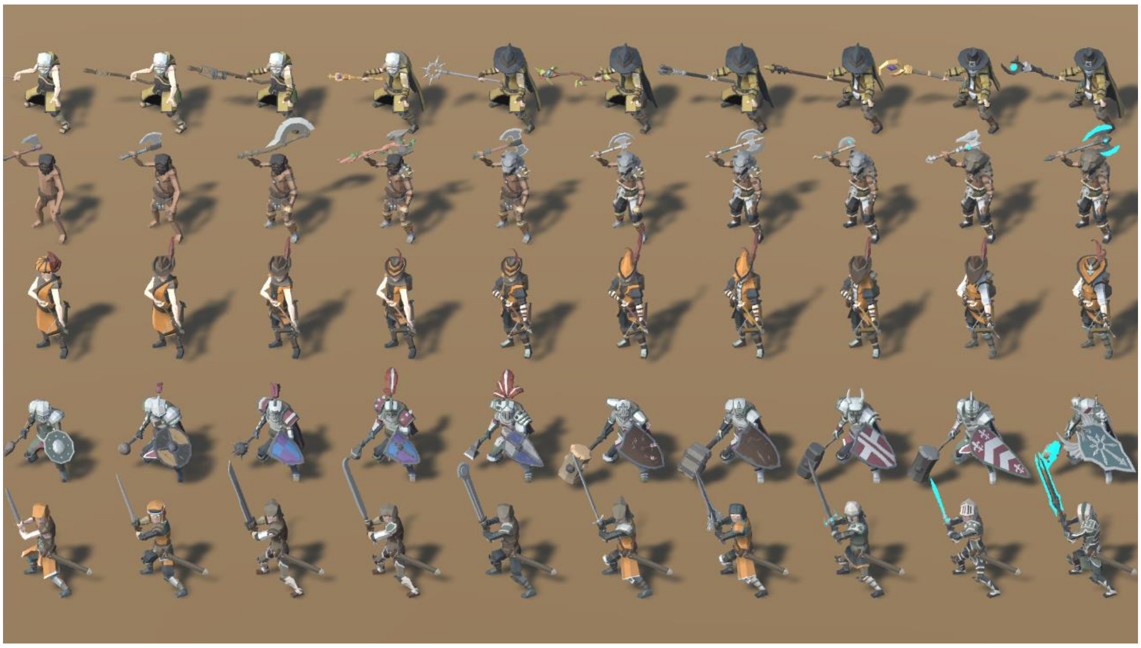


Fig. 13 – All male characters

Animations are also very important to create characters that the player feels comfortable with. The character animations have been obtained from the Unity Asset Store, where Explosive has created the RPG Character Mecanim Animation Pack, an asset in which you can take different animations for your models. All characters have animations for standing, moving, taking damage, and attacking. Next, you will see the schematics of the animations of one of the characters, the Warrior. At first, the standing animation is activated and the character will continue to be with that animation until we interact with it. Pressing the catch button will activate the animation of picking up an object from the ground and return to the standing animation.

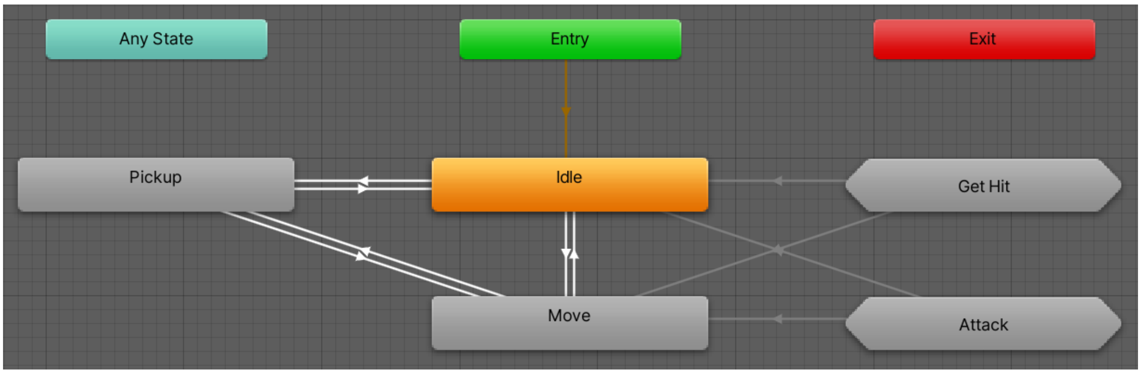


Fig. 14 – Animator (Base Layer)

Pressing one of the character's movement keys will stop them being in the stopped animation and activate the movement animation. The move has a tree of animations that run depending on the character's X and Z speed. In total there are eight animations in the tree, each one for a different direction, which gives more realism when running, since it is not the same to run forward than backward or sideways.

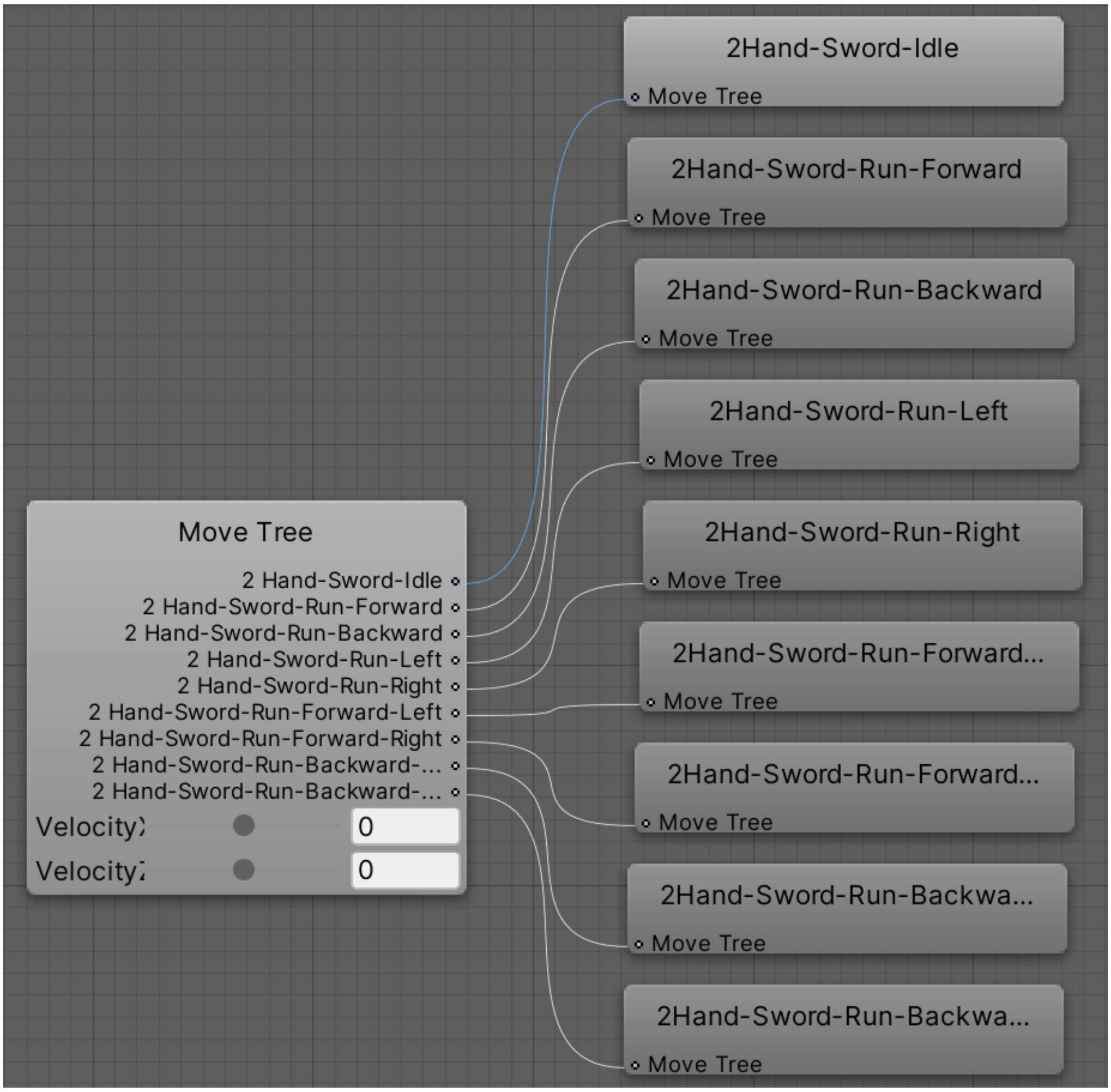


Fig. 15 – Animator (Move)

Taking damage from an enemy is included in the group of animations of taking a hit. When an attack is received, one of the different animations of taking a hit is randomly selected and then it returns to the animation of standing. If the player continues to receive attacks and ends up dying, the death animation is launched.

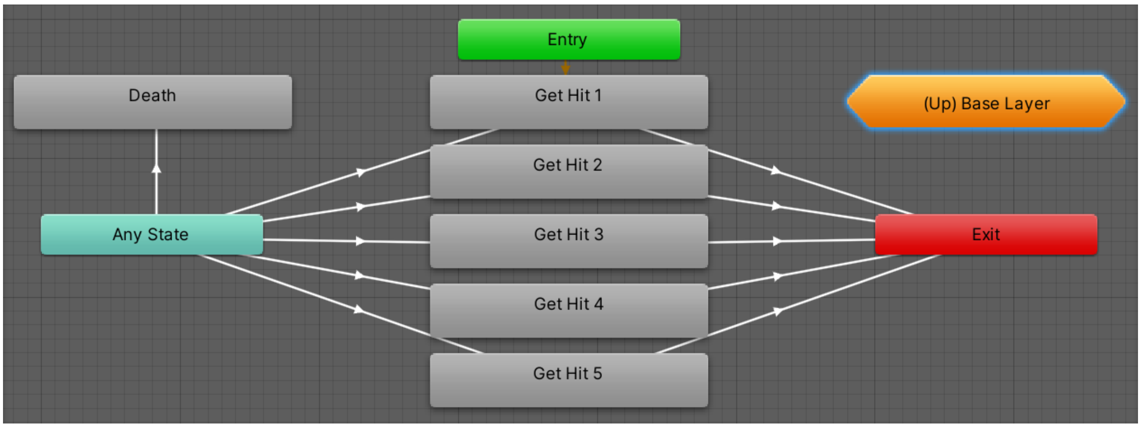


Fig. 16 – Animator (Get Hit)

If the player attacks he enters the attack animation group, where one of the different attack animations is randomly selected. Using a stat boost skill launches the boost animation, like an attack. When using the last skill, a series of animations are launched with a loop, at the end of the animations it returns to the animation of standing.

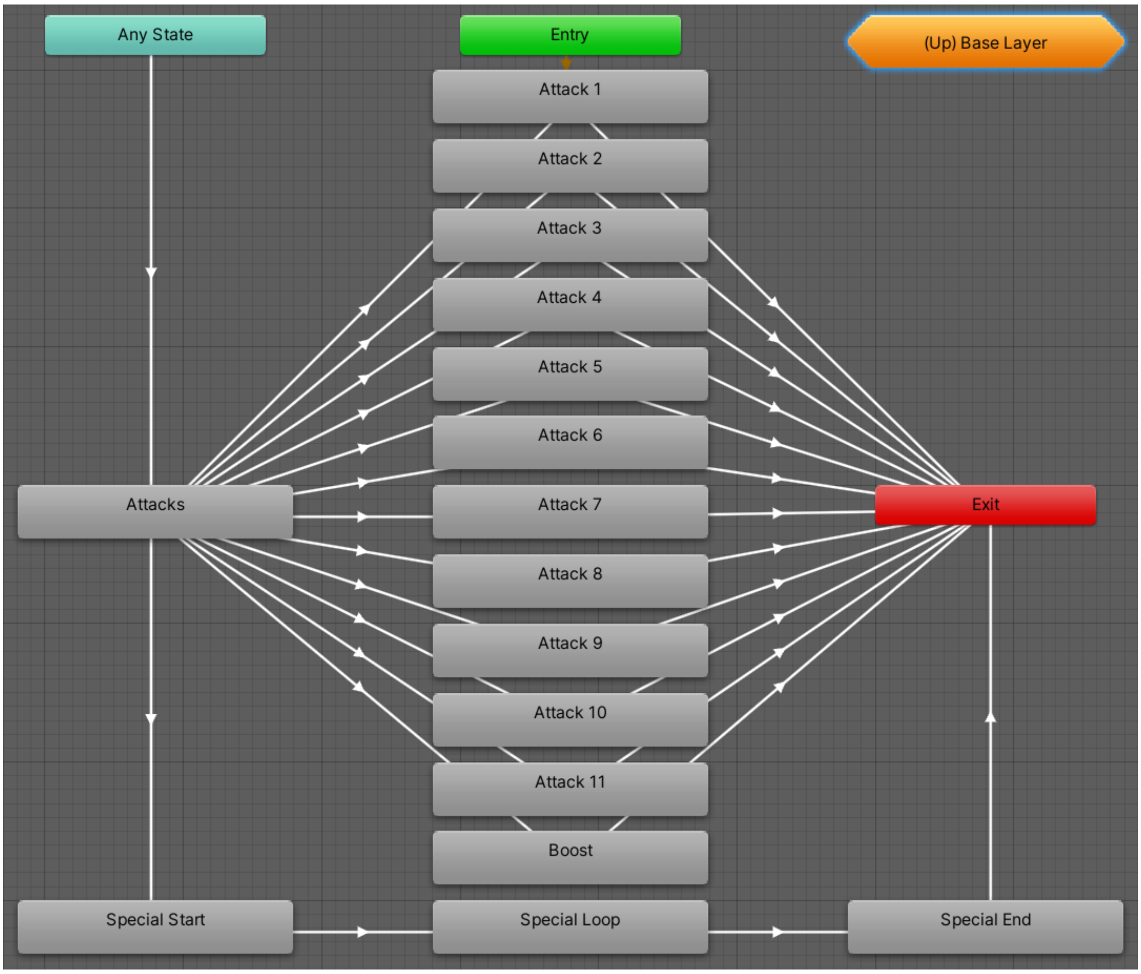


Fig. 17 – Animator (Attack)

3.5. Game narrative, world environment design and combat testing

One day, you suddenly wake up and find yourself in an open field. After investigating the area, you find a path that leads you to a small village. As you enter the village, you see a woman screaming for help because a man is being attacked outside. You follow the path across the village until you reach a forest on the outskirts and the first thing you see is the man being attacked. You decide to help him by defeating the enemies. After the fighting, the man thanks you for your help and introduces himself as Narizu. Before saying goodbye, he tells you that he wants to talk to you, but that area is dangerous, so he will wait for you in the next town. You investigate the forest area a little more and then head to the next town.

You arrive at the town, which is a little bigger than the last one, and you meet Narizu. He thanks you again and tells you that he is of the elven race and that he needs your help because his family and his people are dying of hunger, since the humans kicked them out of their lands. You decide to help him to save his people and he tells you that he must first check that you are ready to help him, so he asks you to go up to Level 03 and tells you that he will be waiting in the next town. You train by defeating enemies in the large forest adjacent to the village until you reach the required level and head to the next village.

The third town is larger and there are shops abound. You meet again with Narizu, who congratulates you on getting there and explains to you that during the last years they have not been able to live decently because of the humans, that they have rejected them and exiled them, and now the other elves are dying from hunger, cold and disease. For this reason, he orders several objects that he needs to survive and save his people. Those objects are distributed on the outskirts of each of the towns. From here, you will go from town to town collecting the objects that Narizu asks you and defeating the enemies you see outside.

You will arrive at the fourth town, a viking town surrounded by snow and ice. You will enter the snowy mountains and reach the fifth town, a huge viking town. At last, you will leave the snow behind and reach the sixth place, a city of humans. As you continue to advance, you will come across the seventh place, a huge human fortress. Leaving the fortress behind, you will reach a gigantic mountain where you will find the eighth place, a kingdom of samurai. The ninth place you will arrive at will be the ancient elven kingdom, now invaded by humans, where you will get the last item Narizu needs.

Finally, the tenth and last place is the great kingdom of humans. But this will be the only place where you will not find Narizu. After investigating the area, you will find a huge dungeon and at the end you will see Narizu with an army of elves behind him. Narizu has deceived you, all the objects that you have obtained were not to save his people, but to strengthen his army with which he wanted to attack the human kingdom and destroy all the inhabitants. You decide to defeat Narizu to stop his evil plans and in a bloody final battle you manage to finish him off. Now, the kingdoms are at peace, but there are still enemies in the forests outside, so you dedicate yourself to hunting them.

The design of each level of the world consists of the area of the city where citizens walk and the outer area where the player fights against enemies and collects objects. The models of the different objects with which the world levels have been formed have been obtained from the Unity Asset Store, where Synty Studios has created the assets Nature Pack, Adventure Pack, Vikings Pack, Knights Pack and Samurai Pack, which have different structures with which you can design cities and exteriors of different types. Next, the different levels in which the game world is made up will be shown.

Levels 01, 02 and 03 are made up of small towns surrounded by large forests.

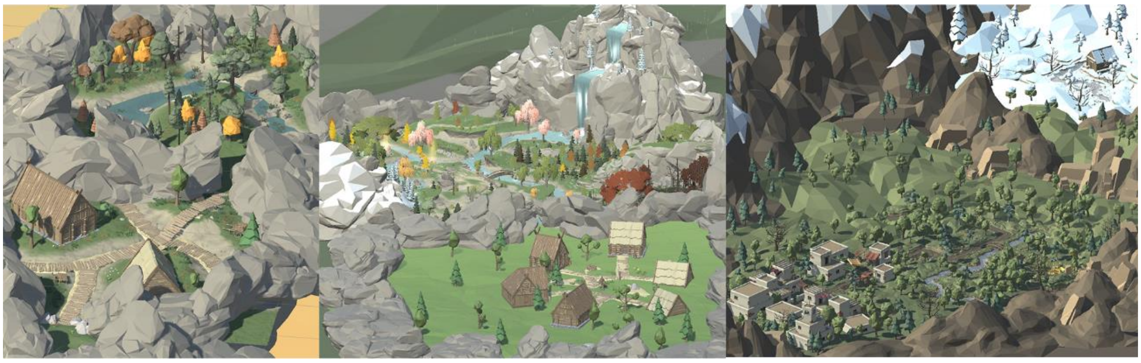


Fig. 18 – Levels 01, 02 and 03

Levels 04 and 05 are made up of viking villages surrounded by snowy mountains.

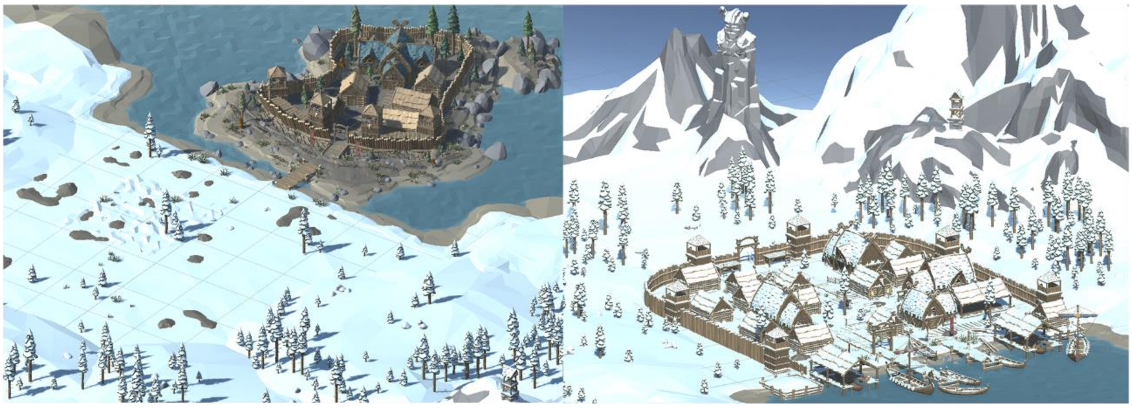


Fig. 19 – Levels 04 and 05

Levels 06 and 07 are made up of cities and fortresses surrounded by large esplanades.

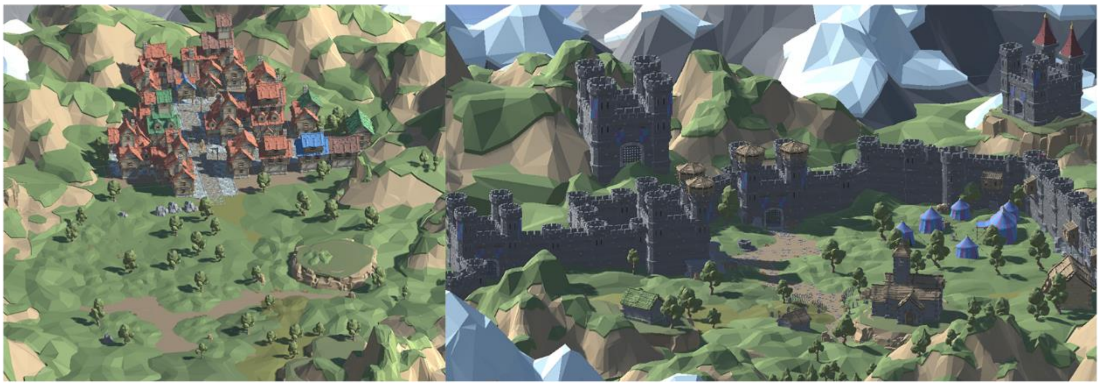


Fig. 20 – Levels 06 and 07

Level 08 is made up of samurai villages and castles surrounded by great mountains.

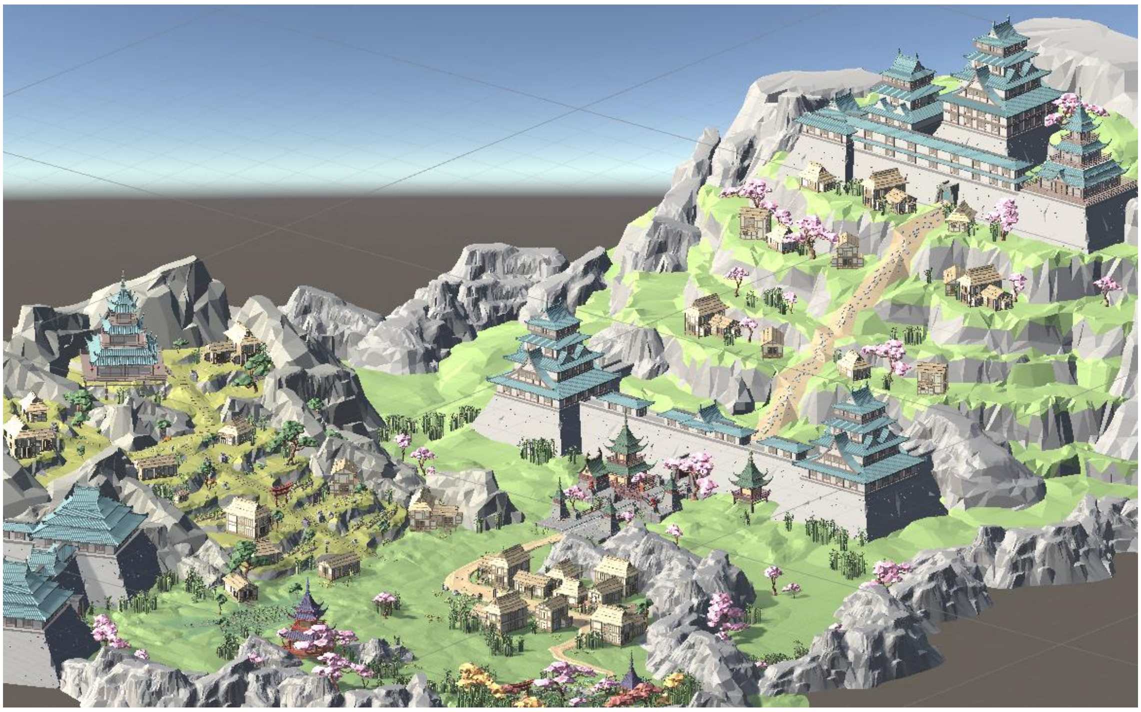


Fig. 21 – Level 08

Levels 09 and 10 are made up of the kingdoms of the elves and humans surrounded by the kingdoms of the goblins and the undead.

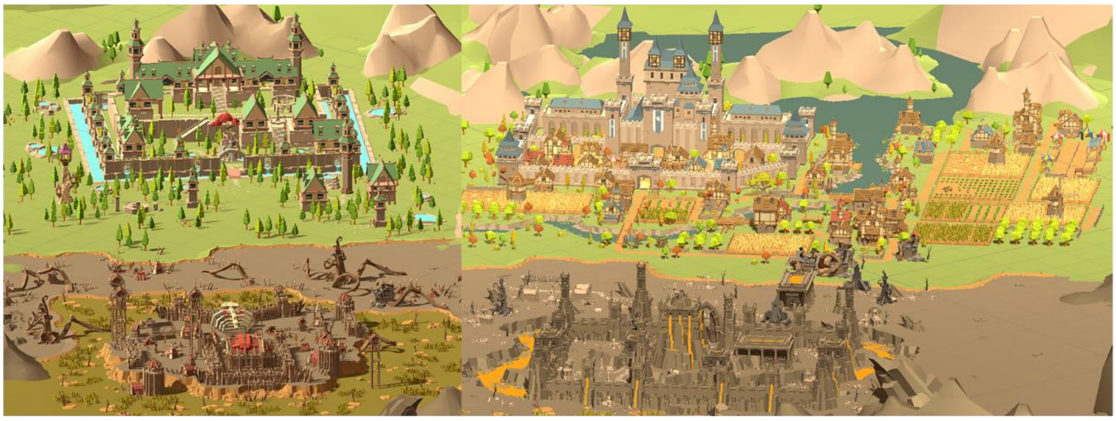


Fig. 22 – Levels 09 and 10

One of the most important things for an RPG is the combat system, since combat is what the player will spend the most time doing in the game. So the game needs a good combat system that is balanced for all character classes and continually scales as you progress through the game. The first step in making this possible has been to research different RPG games and see their formulas for the combat system and the progression of the levels. From here, my own formulas have been obtained that adapt properly to the game and make the combat balanced. First, five different classes have been created and their statistics have been distributed in such a way that each character stands out in one thing, but that they are balanced between them.

	Strength	Resistance	Dexterity	Physique	Energy
Warrior	5	1	4	2	3
Guardian	2	5	1	3	4
Hunter	3	4	5	1	2
Barbarian	4	2	3	5	1
Mage	1	3	2	4	5

Fig. 23 – Character class statistics table

In the table, no number is repeated in any row or column, so no character has any statistics repeated with another class. A character's final stats are derived from his base stats and current level. These are the formulas used to calculate attack, defense, speed, health, and magic.

$$Attack = strength \times level \times 10$$

$$Defense = resistance \times level \times 10$$

$$Speed = dexterity \times level \times 10$$

$$Health = physique \times level \times 10$$

$$Magic = energy \times level \times 10$$

Attack, defense and magic are used for the damage formula when the player attacks an enemy or vice versa. Speed is used to calculate movement speed, time between attacks, time between skills, the probability of delivering a critical hit, and the probability of missing a hit. Health and magic are used to calculate maximum health and magic and to calculate health and magic regeneration.

$$Experience\ to\ level\ up = 100 \times 2^{level-1}$$

$$Experience\ gained\ by\ enemy = 10 \times 2^{level-1}$$

In the experience to level up, the level variable refers to the current level of the player, while in the experience gained by the enemy, the level variable refers to the level of the enemy that has been defeated by the player. This way, if the player defeats enemies that are at the same level, he would need to defeat ten enemies to go up a level. The final bosses in each zone give the player double the experience of a normal enemy.

$$Player\ attack = \frac{attack \times attack}{attack + defense} + \frac{magic \times magic}{magic + defense} \times skill$$

$$Enemy\ attack = \frac{attack \times attack}{attack + defense} \times 2$$

The attack and the magic of the formula belong to the aggressor, while the defense belongs to the one who receives the blow. The enemy attack differs from the player attack because the enemies do not have the magic stat, so the result is multiplied by two so that it is balanced. The skill variable can take the values 0.5, 1 or 2 depending on the power of the skill used, thus achieving that characters like the warrior or the barbarian are stronger at the beginning with basic attacks and that the magician is more powerful at the end with attacks that use a lot of magic.

$$Movement\ speed = 3 + \frac{speed}{100}$$

$$Time\ between\ attacks = 1.5 - \frac{speed}{500}$$

$$Time\ between\ skills = skill - \frac{speed}{50}$$

$$Critical\ hit = 10 + \frac{speed}{100}$$

$$Miss\ hit = 20 - critical\ hit$$

Thanks to the speed of the character that goes up with each level, it increases the speed of movement and decreases the speed between attacks and the speed between skills. The skill variable of the time between skills is a different time for each skill and can take the values 12, 24, 36 or 48 depending on the type of skill. Also, with speed the probability of making a critical hit and the probability of missing a hit are calculated.

$$Maximum\ health = health \times 10$$

$$Maximum\ magic = magic \times 10$$

$$Health\ regeneration = \frac{health}{400}$$

$$Magic\ regeneration = \frac{magic}{200}$$

Maximum health and magic are calculated by multiplying the health and magic stat by ten, so if the character has ten health he would have one hundred maximum health. Health and magic regeneration is calculated from maximum health and magic, so the more health the character has, the faster he will regenerate. Magic regenerates twice as fast as health, so you can use skills faster.

4. Results

When I came up with the idea for this project I was thinking of an expandable RPG videogame that would tell a story for a single player, with NPCs and enemies controlled by Artificial Intelligence and with a system of levels, skills and quests that can be expanded in future updates. This is what I have achieved. Although it is true that during the process new ideas have arisen and some of them I have not been able to implement due to lack of time, the truth is that I have been able to fulfil everything that I set out to do in the conception of the game.

It is an RPG game, in which the player plays a certain role, in this case being able to choose between different classes of characters. It tells a story for a single player, the player must choose a character class and delve into the story that is being told. The NPCs and enemies are controlled by Artificial Intelligence, the NPCs tell you the story through quests and the enemies will attack you when you approach them. The game has a complete system of levels, skills and quests, the character will grow stronger as he defeats enemies and completes quests, gaining experience, climbing levels and learning new combat skills. Also, this game can be easily expanded in future updates by adding a new world, a new story, and more NPCs, enemies, levels, skills, quests, and classes. Everything is developed to make it easy to expand the game when necessary.

As a result of this, I think that the game is as I had originally imagined it, even with additions that have occurred to me during the process. It is a game designed for RPG lovers who want to discover a new fantasy world and venture against enemies. The work now is for many people to try the game and tell me what they think, in order to correct all the errors, improve the aspects that do not please the public and expand the content.

Game link: https://simmer.io/@Narizu1994/rpg-project

Code link: https://github.com/Narizu/RPG-Project

5. Project deviations and objectives done

- ✓ Develop a videogame with an easily expandable world in possible future updates. This means that once the game is over, it's easy to add more content without having to program more. If the starting game has ten levels, a new level could be added creating a new city with new enemies and everything would continue working the same way.
- ✓ Create several characters with different roles, each with their own characteristics, statistics and skills. Creating different characters makes players feel comfortable choosing the character they like best and gives the game the possibility to replay to try out a new character. Also, learning new skills when the character levels up motivates them to continue playing.
- ✓ Implement Non-Playable Characters, controlled by Artificial Intelligence, capable of moving and attacking. Inside the cities there will be citizens who will walk the streets, talk and give quests to the player. Outside there will be enemies wandering around the area that will attack the player when they see him.
- ✓ Get a balanced combat system, with mathematical formulas to take into account the statistics of the characters in each attack. In RPG games it is very important to have a balanced combat system in which the difference of the chosen character is noticed without some being more powerful than others. This is achieved by using mathematical formulas that fit the game in the most precise way and make the difficulty increase progressively.
- ✓ Offer an interesting story to the player, told through quests, that does not become heavy. There are players who get hooked on a game when the story catches their eye and want to keep playing to see how it will end. It is important to have a good story to tell without it being too difficult to understand or too long, so that nobody gets bored playing it.

6. Hours control and planning

November 5. I start writing documents.

November 21. I start programming.

December 24. I have made a basic movement controller for the character, I have made enemies with a basic AI that attack the player and that can also die if attacked and also something from the HUD.

January 28. Basically what I have developed is the basis of the game: a controllable character and enemies to fight against. To play: you can move with the WASD keys and with the mouse you have several attacks (left click melee attack and right click remote attack). Also, with the ESC key you can open the pause menu to modify the sound settings or to close the game.

February 3. In the executable 0.2 I added that you can select your character at the beginning of the game, that the enemies have health bars and that the player also has a magic and experience bar.

February 7. In executable 0.3 I added that each character has different stats (attack, defense, speed, health and magic) and I improved the HUD with numbers to make it easier to understand.

February 14. In the executable 0.4 I have added different skills that are learned by leveling up (that are used with keys 1, 2, 3, 4 and 5) and I have added another HUD improvement in which the damage caused by the screen appears.

February 21. This week I added the executable 0.5 to Google Drive in which I have included information in the HUD about the use of the skills (at what level they are learned, how long it takes to re-launch ...), new remote skills for the Hunter and the Mage (skills 1, 3 and 5), AI for the enemies that follow random routes but if they see the player they start to chase and attack him, items randomly distributed by the map that can be collected and displayed in the HUD, NPCs with random routes (that in the future will have dialogues / quests) and a test of a boss (a bigger enemy and with better stats).

February 28. I have added a mini map, limits on the 4 sides for the character and the camera, enemy information (level and health) when selected, player information (level, stats and skills) when pressing a button or the C key, an NPC that gives you the quests, a quest that is activated when approaching the NPC, random dialogues for all the NPCs and speed reduction when attacking (both for the player and the enemies).

March 6. I have started with art. Made HUD for the health, magic and experience bars and for the five skills and level. Made the HUD for the mini map and made the icons to represent the player, the enemies, the boss, the NPCs and the NPC of the quest. Testing modelling and animations in another project.

March 13. Added the characters with their 2 sexes, their 5 classes and their 10 levels. Prefabs, models, materials, textures and animations.

March 20. Start menu to select the level to be played. Selection of the class and sex of the character to be played with. Various animations for each character. Change of appearance of the character when leveling up. Level 01 art finished.

March 27. Finished the art of the levels from 02 to 08. Made the colliders and the nav mesh. Made a menu scene to select levels. Made both enemies and NPCs appear above and move around the ground layer (even though there are higher surfaces than others). Added the fact that when an object is placed between the camera and the player, this object appears semi-transparent.

April 3. Finished the art of levels 09 and 10. Done the dungeon of level 05, half the dungeon of level 07 and thought about the dungeons of levels 09 and 10. Done that when you reach the entrance of a dungeon you change to the scene of said dungeon. Made an executable in WebGL and uploaded to simmer to be able to play without having to download the game. I start with the initial memory of the project.

May 3. Finished the initial version of the project report to deliver to the tutor.

Tasks	**Hours**
Character programming: movement, attacks, skills, statistics, interaction	100
Artificial Intelligence programming: routes, dialogues, quests, combats	90
Head-Up Display, Graphical User Interface, menus, buttons, sounds	50
Character and enemy design, movement and skill animations	60
Game narrative, world environment design, combat testing	70
Technical proposal, analysis and design document, memory, presentation	80
Total	**450**

Fig. 24 – Task and temporary planning

7. Conclusion

I am very happy with the result of the game, it is just what I hoped it would be when the idea occurred to me. Also, I had a lot of fun programming it and seeing how it was gradually taking shape, especially when I had to think of mathematical formulas for the combat system and the experience system. I'm looking forward to people trying it out and telling me what they think of the game, then correcting any bugs they find and improving the experience.

Also, during the process of creating the game I was coming up with more ideas to implement, some I have had time to do and others stay to do them in the future. These ideas motivated me to continue making the game because I really wanted to develop them and see them working in the world I have created. Yes, it is true that there have been desperate parts of the code, but in the end I have known how to solve all the problems and I liked the result.

For years I have been a player of different MMORPGs because I am excited to discover new worlds with their adventures and their stories. So I wanted to do something similar and since the MMO issue was getting out of hand, I decided to make a single player RPG. Although who knows, maybe in the future it will have enough capabilities to expand the game to be multiplayer.

Now, as I play the finished game, I keep coming up with new ideas to implement and I'm already thinking about how to do it. This is the videogame that I have most enjoyed developing from start to finish. I hope that the people who play it have a little fun and feel that sense of adventure that I felt years ago when I played RPG games.

8. Bibliography

[1] Unity User Manual

https://docs.unity3d.com/Manual/index.html

[2] Unity Learn

https://learn.unity.com/

[3] Levels in RPG

http://howtomakeanrpg.com/a/how-to-make-an-rpg-levels.html

[4] RPG Formulas

https://forum.yoyogames.com/index.php?threads/rpg-formulas-infinitely-scaling-damage.7516/

[5] Learn to make videogames with Unity

https://www.youtube.com/playlist?list=PLOqjKhfbQ8FXKEkk294y1LAsyuPip7Euy

[6] Unity RPG Tutorial

https://www.youtube.com/playlist?list=PLX-uZVK_0K_6JEecbu3Y-nVnANJznCzix

[7] Unity Forum

https://forum.unity.com/

[8] Unity Asset Store – Synty Studios

https://assetstore.unity.com/publishers/5217

[9] Unity Asset Store – RPG Character Mecanim Animation Pack

https://assetstore.unity.com/packages/3d/animations/rpg-character-mecanim-animation-pack-63772

[10] Unity Asset Store – Action RPG Music Free

https://assetstore.unity.com/packages/audio/music/action-rpg-music-free-85434