Goblin Hunter

TEAM

Our members:

- Panagiotis Triantafillidis - 3200199
- Dimitris Pararas - 3200148
- Evangelos Leftakis - 3200093

Overview

Introduction

The "Goblin Hunter" project is a first-person game developed as part of the Multimedia Technology course at Athens University of Economics & Business in IT Department. The primary goal of this project is to provide an immersive and educational experience for users to explore the realms of audio, lighting, and graphics within the context of multimedia technology.

Purpose

By developing this first-person game, students gain practical insights into the integration of multimedia elements, fostering a comprehensive understanding of the role these components play in creating an engaging and dynamic user experience.

Key Features

- *Audio Exploration*: The project emphasizes the incorporation of diverse audio elements, including ambient sounds, music, and character-specific effects. Students could experiment with audio design to enhance the game atmosphere and user engagement.
- *Dynamic Lighting*: Students explored the manipulation of light sources to influence mood, atmosphere, and gameplay dynamics, showcasing the importance of lighting in multimedia projects.
- Graphics Integration: The game showcases advanced graphics integration, allowing students to experiment with rendering techniques, textures, and visual effects. Through the implementation of high-quality graphics, students learn to optimize visual appeal while maintaining performance standards.

What we are going to analyse

- Architecture & Components
- Techniques
- Team Contribution
- Engine & Libraries
- Resources & Tutorials
- Problem & Solution
- Guide for Installation

Architecture & Components

Story

When the poor lumberjack returns to his house alarmed by screams, He is met with an image of horror, blood, and dismemberment. His family is dead.
Overwhelmed by grief and shock, the poor lumberjack stumbles backward. Amid the carnage, a sinister trail of blood beckons him toward a nearby cave. Desperation mingling with a glimmer of hope, he follows the crimson path into the darkness, determined to unravel the mystery of his family's fate. With each step, the lumberjack's grief transforms into a relentless determination and rage to confront the horrors within, as the cave's ominous depths conceal the answers he seeks.

Scenes

The game is divided into three scenes:

- **Main Menu**: At the start of the game, the user can choose between various option to play his game.
- **Forest**: In the picturesque forest, the player does not fight any enemy. On the other hand, is it an area where the story is introduced based on some events and the player is free to explore the mechanics of the game. If the player is lucky enough, he can find his first weapon!
- **Cave**: In the dark cave, the player encounters the enemies of the games, the goblins. These creatures are patrolling in various areas. Be careful though, if they see you, they can chase you. The aim here is to find the key in this vast area an escape.
- **Dungeon**: Finally, the dungeon. This area is where the player must find the evil boss of the goblins, the Big Goblin, aka Koblin. However, it is not as easy as it sounds because of the goblins that they lurk in some rooms.

Controls

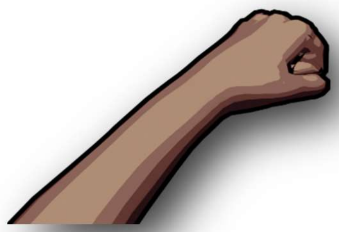
The keyboard input for a player to use are:

- A, W, S, D: Moving left, top, bottom, right.
- Tab: Open the weapon menu.
- Hold Shift and move inputs: Run (be careful, running consumes stamina).
- Space: Jump.
- F8: Save.
- P: Pause.

Architecture & Components

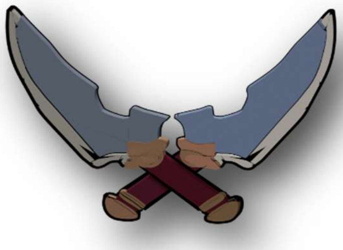
Weapons

The player can battle in four different ways:



1. The basic and most common way. It is the weakest way to battle but consumes little stamina.

2. This weapon is dual and can do a considerable damage. In its case, the stamina consumption is not important.



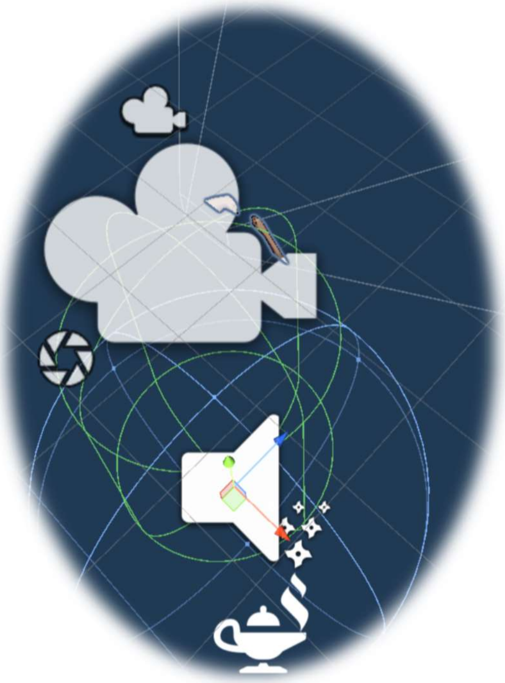
3. Hitting the enemies with a sword can kill them much faster and the stamina consumption is average.

4. The weapon with the most attack power. It is the fastest way to kill your enemies. But with great power comes great responsibility, so be careful as the stamina consumption is high.

Architecture & Components

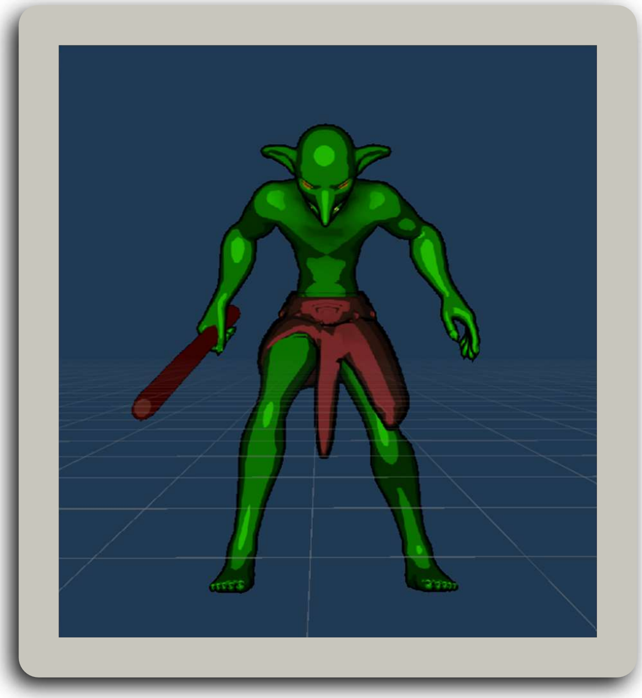
Player Mechanics

As said above, "Goblin Hunter" is a first-person game. In this picture we can see how the component of the player is structured. First it has the **camera** for the player to see. Second, the **UI** is in front of the camera angle to showcase the current weapon (here the hands). Also, UI holds other components like the weapon menu etc and the health and stamina bar. Third, player holds **audio sources** for events (running, walking, attack etc). Last, player contains some special **VFXs** that are triggered by some dynamic events.



Enemies

1. Goblin



Most common enemy in the game. Goblins lurk in the dungeon as well as in the cave. They may be smaller and slower than your average enemy but when they appear as a gang, they are difficult to encounter. Do not worry though, it takes a few hits with a weapon to defeat them.

2. Koblin

The final boss of the game, the evil behind the murder of your wife. Koblin is the last enemy of the game, and your aim is to defeat him.
Be careful! His health is greater than those of the goblins and his strength is considerable enough to strike you down. You do not want risk everything, don't you?

Audio

Audio aspects can be divided into two categories:

- Events: Where small audio parts can be heard and play for a fixed time (screaming, hit by enemy etc)
- Background: The background music is in a time-loop and always used to add an erring feeling to the player

Graphic

Graphic design is applied on the shaders of the game (e.g. bread) and in the texture of some elements (boxes in the dungeon)

Camera

In the case of camera techniques, we used the "Billboard", which allows the objects like the enemies to always look on the player, no matter the angle and seem like 3D.

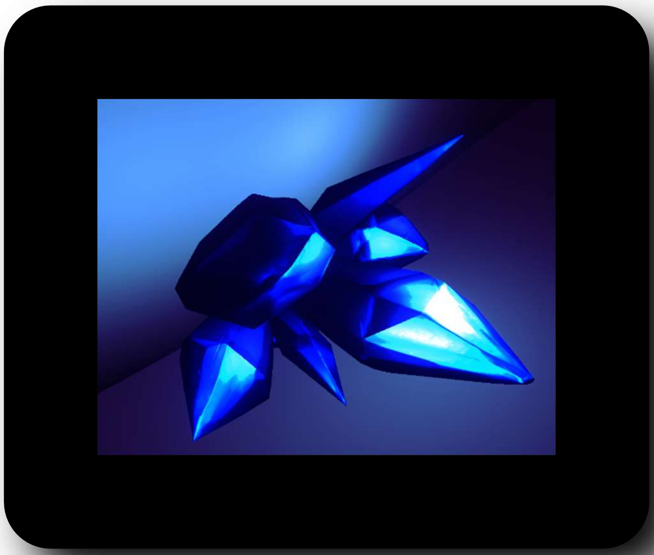
Architecture & Components

Lighting

Lighting techniques are commonly used in a different way in the game.

There are two ways to use lighting:

- point lights have been applied in objects like torches and crystals.
- directional lights have been positioned for bigger spaces like dungeon, to create the perfect atmosphere.



Attributes of the lighting objects have been designed to give the appropriate feeling based on the location of the object.

In the directional lights (for example in the dungeon) where used to manipulate the shadows of the space.

Menu

Main menu and weapon menu (also weapon wheel) are interactive components of the game. These are constructed on the UI tool of the unity. The user can interact with buttons to choose between options.

Save System

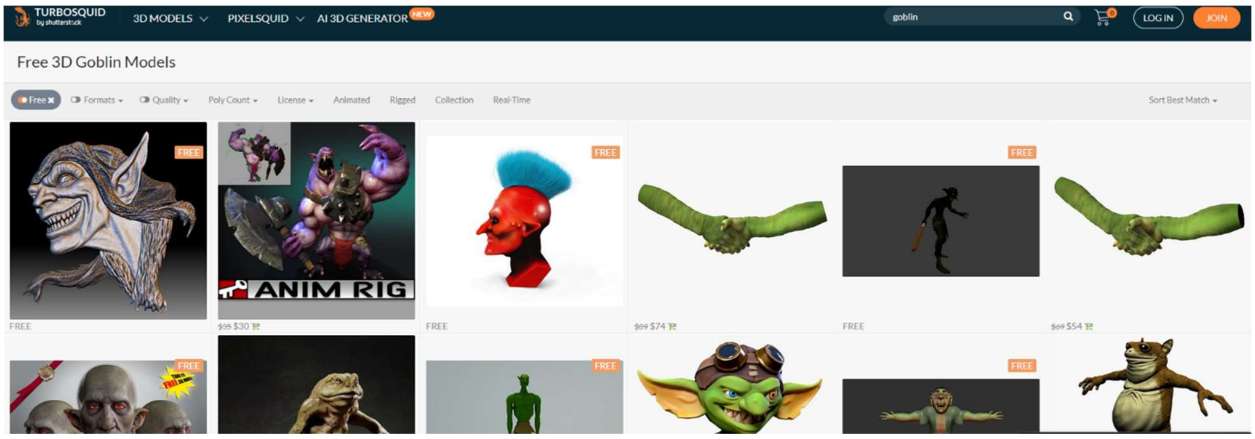
Imagine how annoying it could be for a user to not be able to save his progress. In the save system, the data of the player, enemy, scene etc. is serialized and is stored in the persistent /AppData/LocalLow data path.

Techniques

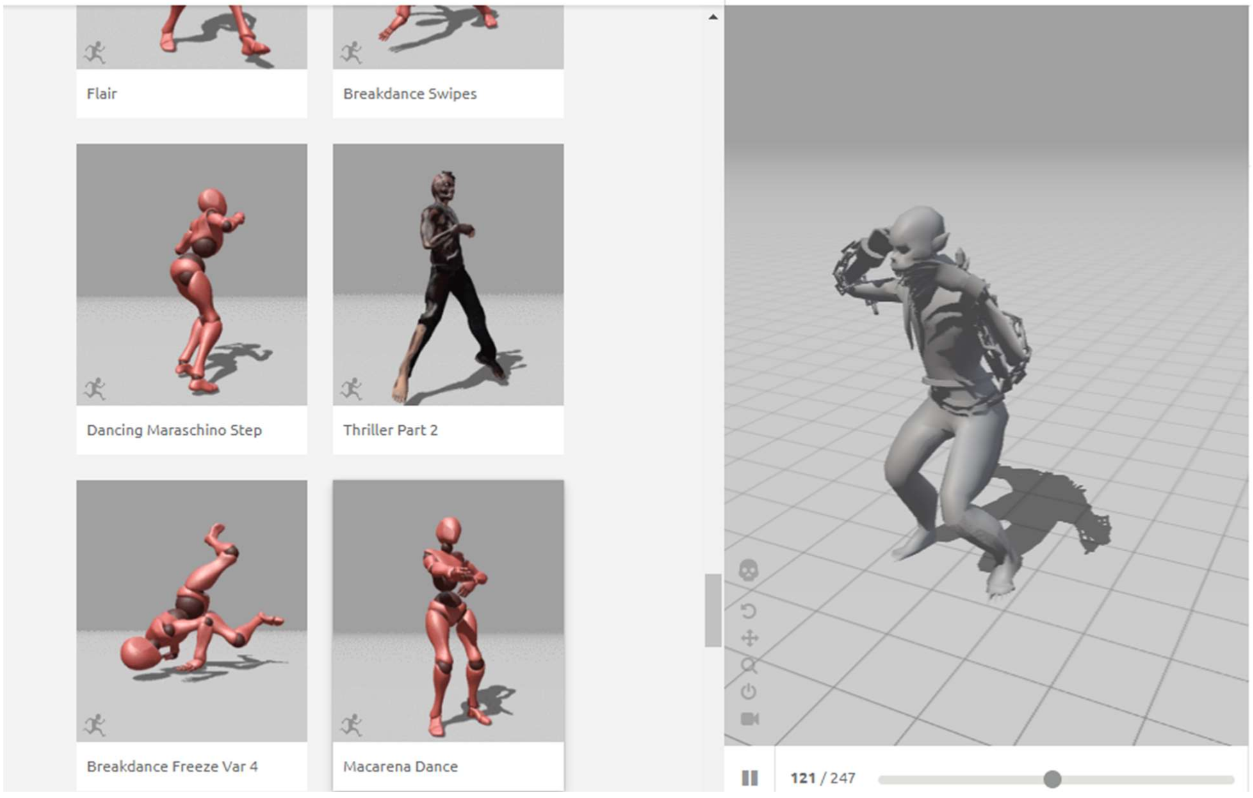
Sprite/Animations of Enemies

To create the enemies, we followed the steps below:

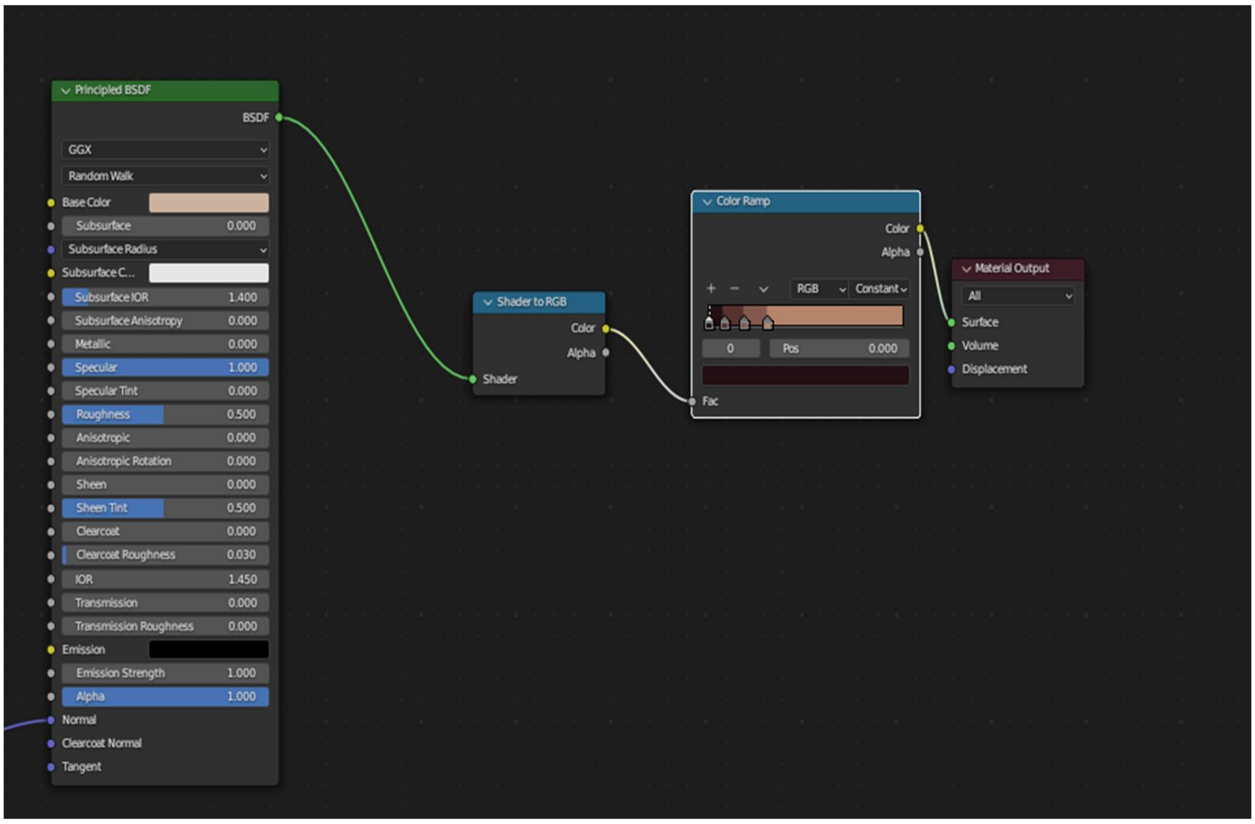
1. We find a free asset of a rigged 3d model in sites like Sketchfab, Turbosquid, cgtrader etc.



2. Then we either animate the 3d like in the case of the players hands or we upload the models to https://www.mixamo.com/ where you can add animation to some preset character 3d models or upload your own 3d model and have it auto rigged and animated. We add the animations to our 3d model.



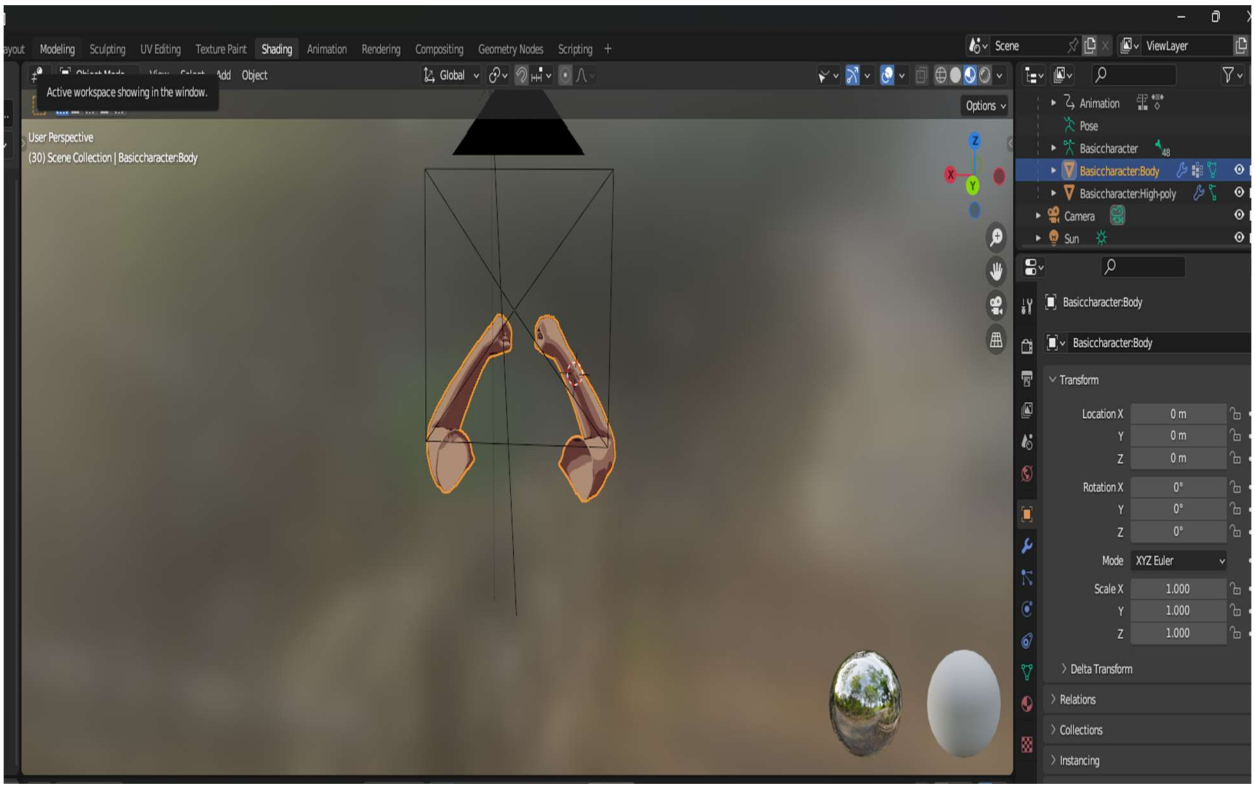
3. Now we Go to blender and stylize the 3d model to give it a cell shaded look. The shader graph used for it is this:


Principled BSDF
BSDF
GGX
Random Walk
Base Color
Subsurface 0.000
Subsurface Radius
Subsurface C...
Subsurface IOR 1.400
Subsurface Anisotropy 0.000
Metallic 0.000
Specular 1.000
Specular Tint 0.000
Roughness 0.500
Anisotropic 0.000
Anisotropic Rotation 0.000
Sheen 0.000
Sheen Tint 0.500
Clearcoat 0.000
Clearcoat Roughness 0.030
IOR 1.450
Transmission 0.000
Transmission Roughness 0.000
Emission
Emission Strength 1.000
Alpha 1.000
Normal
Clearcoat Normal
Tangent

Shader to RGB
Color
Alpha
Shader

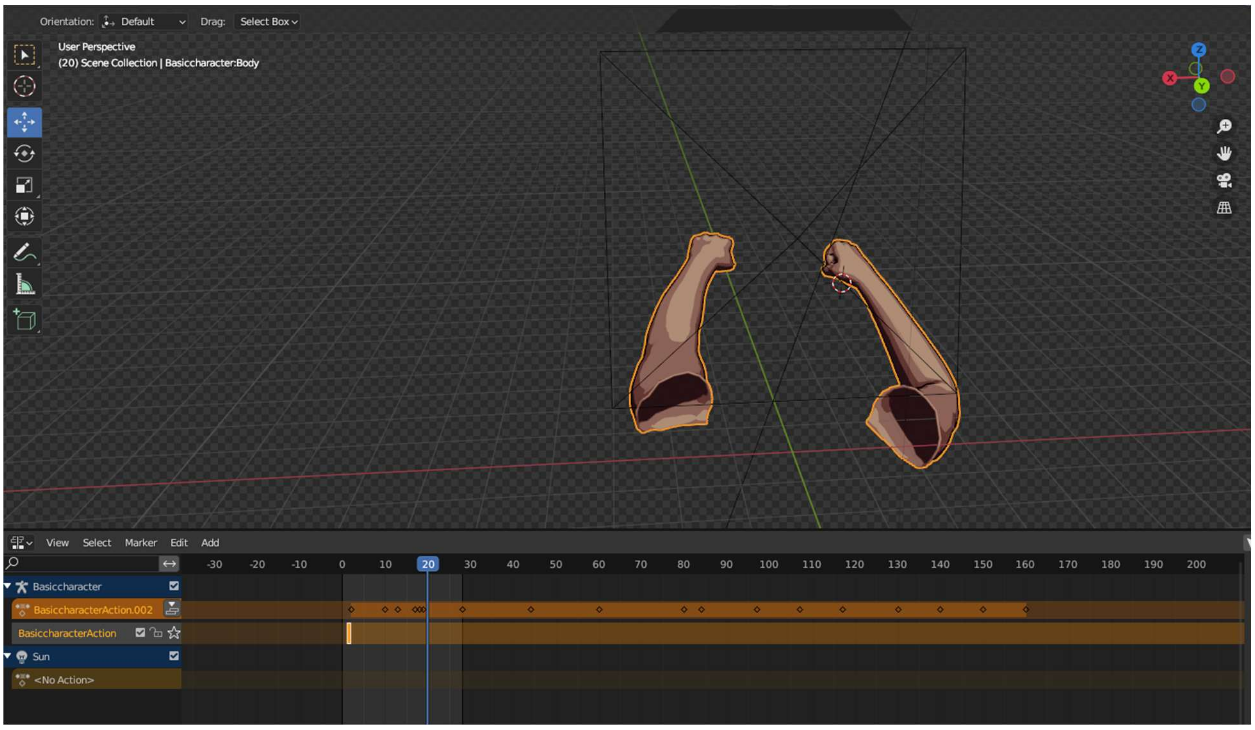
Color Ramp
Color
Alpha
RGB
Constant
0
Pos 0.000
Fac

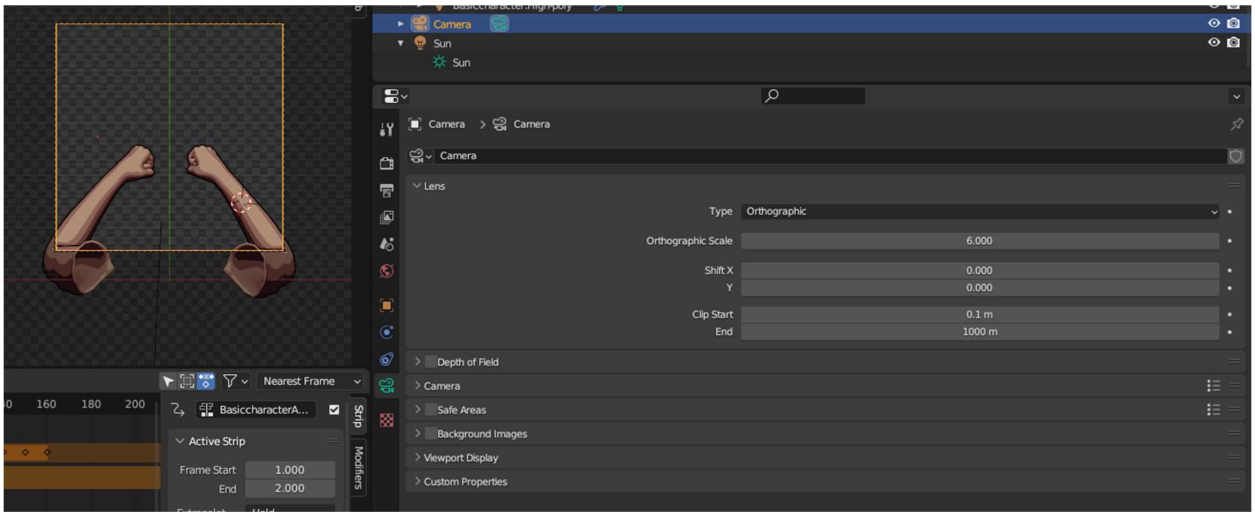
Material Output
All
Surface
Volume
Displacement



Basiccharacter:Body
Transform
Location X 0 m
Y 0 m
Z 0 m
Rotation X 0°
Y 0°
Z 0°
Mode XYZ Euler
Scale X 1.000
Y 1.000
Z 1.000

4. Then we setup a camera and captured the rendered results in a sequence of pngs. The number of images to be exported is decided to create sprite sheets that are no larger than 4096x4096:





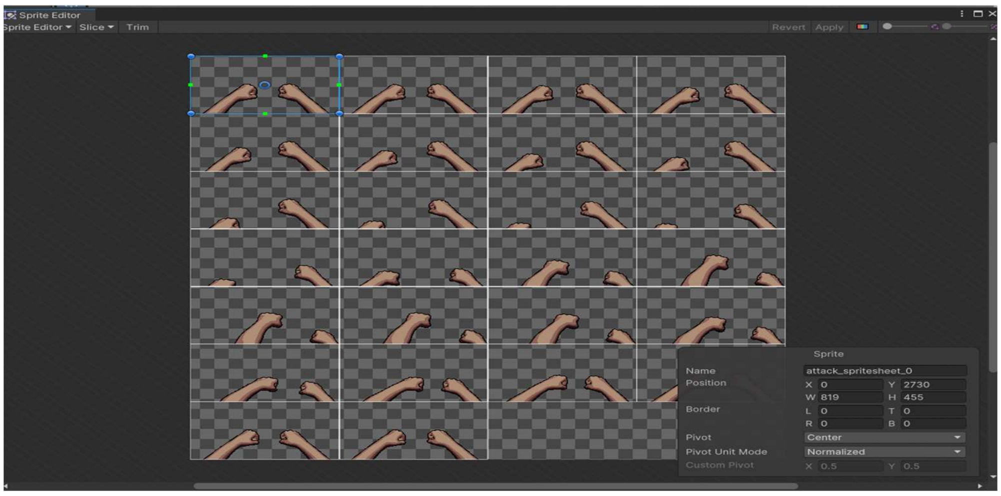
Camera > Camera

Lens

Type: Orthographic
Orthographic Scale: 6.000
Shift X: 0.000
Y: 0.000
Clip Start: 0.1 m
End: 1000 m

Active Strip
Frame Start: 1.000
End: 2.000

5. Finally, the images are imported to unity and made into a sprite sheet with Unity's Sprite Editor tool:



Sprite

Name: attack_spritesheet_0
Position: X 0, Y 2730, W 819, H 455
Border: L 0, T 0, R 0, B 0
Pivot: Center
Pivot Unit Mode: Normalized
Custom Pivot: X 0.5, Y 0.5

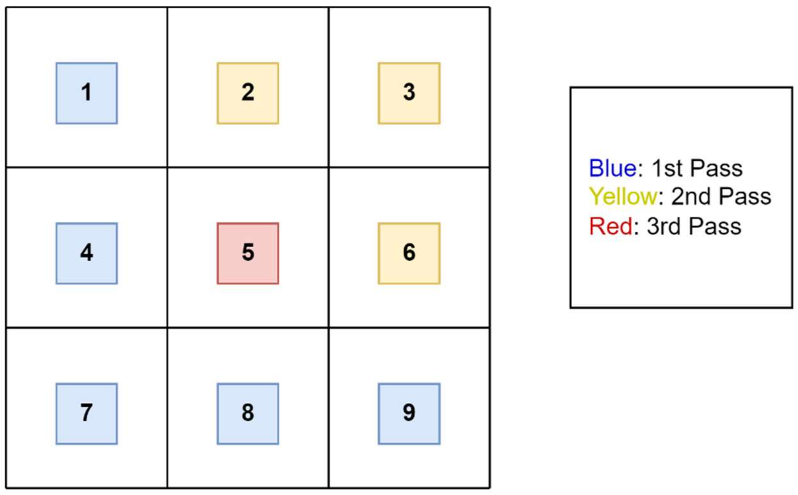
Techniques

Dungeon Generator

The generation of the dungeon is procedural, meaning that is created dynamically. Also, the rooms of the dungeon with every game can be positioned randomly, **but how do this work?**

- **Depth First Search**

  With the help of DFS, the dungeon generator script first creates a path from the start room to the boss room. However, it doesn't stop there as it continues to execute the same algorithm to fill as many rooms as possible. In our game, we have set the dimension to be 3x3 (9 rooms). That means that even if it finds the path from the start room to the boss room (1st pass: 1->4->7->8->9), it will run again the algorithm for each of the nine rooms to have been placed. The root of the tree metaphorically is the start room, and the aim of the algorithm is to find the leaf of the tree that is the boss room.



- **Random Placement**

  The procedural part comes from the random seed that is created from the Game Data at the start of the game and is passed at the dungeon generator from the Player Profile. The seed is used to decide various components of the dungeon like the objects of the dungeons and the objects of the corridors, even the number of the enemies in a room.

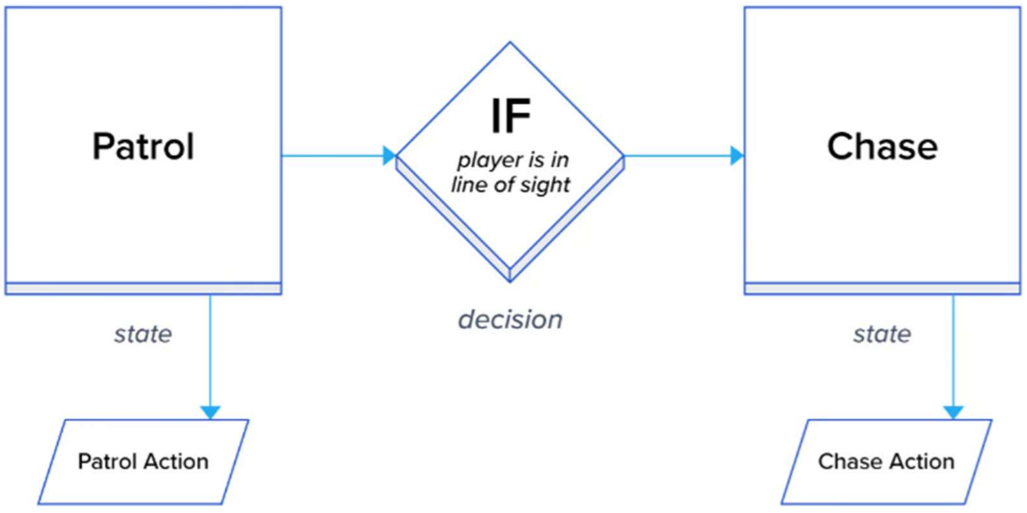
Techniques

Controlling enemies

We spoke before about having enemies in our game, **but how exactly they know how to move and what to do?** This answer is based on two parameters:

- **Navigation Mesh**
  Navigation mesh (aka NavMesh) is a tool in Unity that helps define the walking area of our enemies. NavMesh exist in the cave and in the rooms of the dungeon. The enemy can only walk and chase the player only in the defined navigation mesh. There are a lot of meshes in the game.

- **Goblin State Machine**
  As we know, humans act based on their environment. That depends on the senses that we have that help us take decisions fast and change from one state to another. The same can be said for the enemies. Enemies have states that present their current action. For example, the default state of the enemy is **Patrol**. In this state, the goblin chooses to walk around in a specific diameter of its centre point. If the player goes inside the field of view of the goblin, now the goblin transfers into the **Chase** state. This method is also applied to the enemy's animations.



Team Contribution

Evangelos Leftakis

- **Graphic & Scene Designer:** One of the main scenes of the game and most enormous was the Cave scene, with large models, complex lighting, and dark sound. This scene was created only by Evangelos with the help of Blender tool and other prefabs and models in the game. He also created the design of the enemies.
- **Backend developer:** His main task was to build the save system of the game from scratch, a difficult part without the help of build-in tools in unity. He also wrote the code for the weapon menu he is responsible for the transition between the scenes.
- **UI Designer and Developer:** Most of the work of Evangelos in this field was to design the main menu, the interface of the player in many interactions with events and to write the code for the connection between the player and the environment. Some of his works also were the hands of the players and the weapon menu.
- **Animation Designer:** In this filed, Evangelos designed the animation of the daggers.
- **VFX designer:** Many of the events that are occurring in the game, like the blood of goblins, the sounds of hittable enemies are his work.

Dimitris Pararas

- **Graphic & Scene Designer:** Behind the quaint design of Forest, with rocks, rivers, the hunter's hut and a graveyard, a spooky fog, and scary events (screaming of a woman, blood trail) is Dimitris. Its unique layout, lighting and theme made the forest the perfect scene for the player to start their adventure on the adventure.
- **Shader Designer:** His main role was to write the code for many shaders that were used in the game.
- **Backend Developer:** Dimitris specialized in the basic mechanic construction of the player behind the movement and the attack. Moreover, the interacting abilities of the player with the weapons, the animation scripts of the enemies, the interactable objects and the sprite billboard were his main work.
- **Animation Designer:** Except from the code, Dimitris also created the sprite sheets for the animation of the enemies and the animation of the sword and hands.

Team Contribution

Panagiotis Triantafillidis

- **Graphic & Scene Designer:** Panagiotis main work on this field was the design of the **Dungeon**, the last location of the game. He created and designed each room as best as possible to give a touch a mystery and horror in the game. This can be seen from the audio theme of the dungeon, the lighting of the dungeon and the unique characteristics in each room.
- **Backend Developer:** He set the structures for the enemy controller, wrote the states of the enemy (patrol, chase, and attack) and helped with the organization of the interactable objects (chest & doors).
- **Animation Designer:** A small but useful contribution of him is the animation of axe.
- **Algorithm Enthusiast:** Unlike the other scenes, Dungeon is the only procedural scene. That means that the positions of the rooms are totally random, based on the DFS (Depth First Search) algorithm, written, and implemented by Panagiotis. It is a singular feature of this scene, that gives the player the feeling to start again and again the game an in each single play there is a different dungeon layout.

Summary

Even though each member had specific central parts, nothing was completely done by one member. Every member helped on the debugging of other members, suggested design ideas, and created models for other members to help them on their task.

Engine & Libraries

- To create the game: **Unity (version 2022.3.8f1)**
- To serialize the objects: **Newtonsoft.Json (dll)**

Resources & Tutorials

- **Our inspiration came from this game:**
  https://www.youtube.com/watch?v=drm-rFh37D4&t=31s
- **The combat system came as an idea from this tutorial:**
  https://www.youtube.com/watch?v=yZhKUViKS_w&t=288s
- **This is the first dummy tutorial to learn unity:**
  https://www.youtube.com/watch?v=Kgjth3nRsFc&list=PLiyfvmtjWC_Uzpqn_76LTJYa9cn3DzUk6
- **A tutorial for creating the basic animations of the enemies in the start of the development:**
  https://www.youtube.com/watch?v=iVA9xvbUP7Y
- **A tutorial for learning state machines and creating logic behind enemies:**
  https://www.toptal.com/unity-unity3d/unity-ai-development-finite-state-machine-tutorial
- **The idea of the DFS came from this video:**
  https://www.youtube.com/watch?v=gHU5RQWbmWE&t=985s
- **Also, the game reminds of the Chamber Dungeons in the links awakening.**
  https://blog.turtlebeach.com/ca/how-the-chamber-dungeon-maker-in-links-awakening-works/
- **Created the fog, inspired by this video:**
  https://youtu.be/RBI9hQWD_xY?si=Y5g-0niWpztWfG85
- **Blood Effect Tutorial:**
  https://www.youtube.com/watch?v=IVj7-usBxHo
- **Save & Load System:**
  https://www.youtube.com/watch?v=XOjd_qU2Ido&t=888s

Problem & Solution

Among the various problems that we encountered when we developed the game, there was one that stood out. I am speaking about the performance, that was dependent on the frame rate.

Problem

Because of the many 3d objects & the number of game objects around the scenes, there were a lot of times where the frame rate dropped and that could be seen visually (screen trembles, cracking in audio)

Solution

To find a solution to this serious problem, we applied the “Level of Detail” (LOD). Level of Detail is a technique used to optimize the rendering performance of 3D models by adjusting their detail based on factors like distance from the camera. The idea is to use simplified versions of objects when they are farther away from the viewer and switch to more detailed versions as they come closer. Unity's LOD system dynamically switches between highly and simplified levels of detail based on the distance of the object from the camera. When an object is far away, a less detailed version is rendered, and as it comes closer, Unity seamlessly transitions to more detailed versions. By using LOD, Unity reduces the number of polygons and textures rendered, which can significantly improve performance. This is crucial for maintaining a high and consistent frame rate, especially in scenes with many objects.

But how we implemented it?

In Unity, LOD is often implemented using the LOD Group component. This component allows developers to associate multiple LOD levels with a single GameObject. Unity then automatically manages the transition between these levels based on the camera's distance. For us, we applied it in most of the objects in the dungeon, in the trees of the forest and in the crystals inside the cave.

Guide for Installation

In Editor

1. Extract the Unity-Project.rar.
2. The unity scenes which correspond to the levels can be found int the Assets\Scenes Folder.
3. To start and test the game through the editor you need to open the MainMenu scene.
4. The game can only be started from the main menu. Initiating Play mode in other areas such as forest, cave and dungeon will not work since the game relies on its save system to spawn the player and enemies in the correct positions with the correct data.
5. In order to test all areas without needing to progress through the game normally the save files (.json & .bin) should be extracted from the saves.rar file to unity's persistent path "C:\Users*user\AppData\LocalLow\DefaultCompany*project name" to access the appdata folder which is hidden you need to type %appdata% on the file explorer. If the path Doesn't exist the game should be run once in order for it to be generated (NOTICE when extracting the saves from saves.rar it is possible that a warning is shown due to the saveLocations.bin file being serialized in binary format). When the game opens there should be save files for each area and some special places in the game in the "Load Game" menu.
6. Due to graphic content in the first area an option to turn of gore is available in the options menu.
7. To see the controls go to the controls panel in the options Menu.
8. Simply double click on a save file.

In Build

1. Extract the Build.rar
2. Open the Build Folder
3. Double Click on the My Project.exe

Link to download the project:

https://drive.google.com/drive/folders/10Kmu7f5Cp0nt8BDuUIKHS78JDwoFrxO6