

**INSTITUT SUPERIEUR D'INFORMATIQUE
ET DES TECHNIQUES DE COMMUNICATION – HAMMAM SOUSSE**

المعهد العالي للإعلامية وتقنيات الاتصال بحمام سوسة



Rapport de stage de fin d'études

Présenté en vue de l'obtention du diplôme de Licence en Science de
l'Informatique (Computer Science)

Spécialité : Informatique et Multimédia

Development of an educational game for Unreal Engine and C++ Game Development concepts

Réalisé par :
Youssef BEN TALEB ALI

Encadrant académique : M. Sami BEN AMOR
Encadrant professionnel : M. Achref DAOUAHI
Société d'accueil



Année universitaire : 2022/2023

Institut Supérieur d'Informatique et des Techniques de Communication Hammam Sousse

ISITCOM

Tél/Fax : +216 73 37 15 71 / +216 73 36 44 11

**INSTITUT SUPERIEUR D'INFORMATIQUE
ET DES TECHNIQUES DE COMMUNICATION – HAMMAM SOUSSE**

المعهد العالي للإعلامية وتقنيات الاتصال بحمام سوسة



Rapport de stage de fin d'études

Présenté en vue de l'obtention du diplôme de Licence en Science de
l'Informatique (Computer Science)

Spécialité : Informatique et Multimédia

Development of an educational game for Unreal Engine and C++ Game Development concepts

Réalisé par :
Youssef BEN TALEB ALI

Encadrant académique : M. Sami BEN AMOR
Encadrant professionnel : M. Achref DAOUAHI
Société d'accueil



Année universitaire : 2022/2023

Institut Supérieur d'Informatique et des Techniques de Communication Hammam Sousse
ISITCOM

Tél/Fax : +216 73 37 15 71 / +216 73 36 44 11

Acknowledgement

While bringing out this thesis form, I came across several people who supported me in various ways throughout my academic career.

To my parents, for their love and support, especially within my last three months of graduate studies. Without them this day would not have been possible.

To my sister, for her encouragement and support. Thank you for being always by my side and for your devoted love.

To my entire family, for the good times in my life and for their devotion which they have always shown to me.

To all my friends and colleagues, who have never stopped supporting me, for the unforgettable moments full of humor, laughter, and sincere friendship.

I dedicate this work to them as a sign of my gratitude, my deep love and sincere thanks.

Youssef BEN TALEB ALI

Special thanks

Frist and foremost, I would like to express my deep sense of gratitude and indebtedness to my supervisor **Prof. Sami BEN AMOR** for his invaluable encouragement, suggestions and support from an early stage of this research and providing me extraordinary experiences throughout the work. Above all, his priceless and meticulous supervision at each phase of work inspired me in innumerable ways.

I am highly grateful to **Mr. Achraf DAOUAHI** head director of “**ENVAST**” for his advice, supervision and the vital contribution as and when required during this research. His involvement with originality has triggered and nourished my intellectual maturity that will help me for a long time. I am proud to record that I had the opportunity to work with an exceptionally experienced person like him.

Youssef BEN TALEB ALI

Table of Contents

General Introduction	1
Chapter 1: General View	2
I. Introduction	3
II. Company presentation	3
III. Project context	4
1. Video Games	4
2. Action-Adventure Games	4
3. Role Playing Games (RPGs)	5
4. First Person Shooters (FPS)	5
5. Strategy Games	6
6. Simulation Games	7
7. Sports Games	7
8. Massively Multiplayer Online (MMO)	8
9. Educational Games	9
IV. Problematic	9
1. Information Overload	9
2. Motivation	10
V. Existence Study	10
VI. Solution	12
VII. Conclusion	13
Chapter 2: System Design	14
I. Introduction	15
II. About The Game	15
1. Backstory	15
2. Executive Summary	15
3. Storyline	15
III. System's Functional View	15
1. Functional Needs	15
2. Non-Functional Needs	16
3. Actor Identification	16
IV. Game Specifications	16
1. Modeling Game flow design	16
V. Game Design	18

1.	Game Concept and Genre	18
2.	Target Audience	19
3.	Early Model Design	19
VI.	Work methodology	20
1.	Extreme Programming	20
2.	Product Backlog	21
VII.	Tools and development environment	23
VIII.	Conclusion	27
Chapter 3:	Implementation	28
I.	Introduction	29
II.	Pre-requirements	29
1.	Unreal Engine	29
2.	Visual Studio	31
III.	Textual Description	31
1.	Game Design Document	31
2.	Character Movement	32
3.	Basic Weapon Mechanics	35
4.	Health System	39
5.	AI Zombie Mechanics	40
6.	Head-up Display	43
7.	Menu UI	45
8.	Laptop	47
IV.	Conclusion	51
General Conclusion		52
References		53

List of Figures

FIGURE 1: "LOGO ENVAST" [1]	3
FIGURE 2: "ASSASSIN'S CREED ODYSSEY" [2]	4
FIGURE 3: "PILLARS OF ETERNITY 2" [3]	5
FIGURE 4: "COUNTER STRIKE: GLOBAL OFFENSIVE" [4].....	6
FIGURE 5: "ENDLESS LEGEND" [5]	6
FIGURE 6: "THE SIMS 4" [6].....	7
FIGURE 7: "MADDEN NFL 2019" [7].....	8
FIGURE 8: "WORLD OF WARCRAFT" [8]	8
FIGURE 9: "SPACE CHEM" [9].....	9
FIGURE 10: "WHILE TRUE: LEARN()" [10]	10
FIGURE 11: "MECHANICA" [11]	11
FIGURE 12: " MAIN GAME SCENE USE CASE DIAGRAM"	17
FIGURE 13: "GAMEPLAY USE CASE DIAGRAM"	18
FIGURE 14: "MAIN MENU"	19
FIGURE 15:" HEAD-UP DISPLAY"	19
FIGURE 16: "EXTREME PROGRAMMING "	20
FIGURE 17: "UE LEARNING LIBRARY" [12]	25
FIGURE 18: "UE MARKETPLACE" [13]	26
FIGURE 19: "UE 5.1 DOCUMENTATION" [14]	26
FIGURE 20: "UE FORUMS" [15].....	27
FIGURE 21: "UNREAL ENGINE VERSION SELECTION"	29
FIGURE 22: "UNREAL ENGINE INSTALLATION OPTIONS"	29
FIGURE 23: "UNREAL ENGINE PLUGINS"	30
FIGURE 24: "UNREAL ENGINE PROJECT CREATION"	30
FIGURE 25: "VISUAL STUDIO INSTALLATION"	31
FIGURE 26: "CUSTOM CHARACTER CLASS"	33
FIGURE 27: "CHARACTER IN VIEWPORT"	33
FIGURE 28: " CHARACTER MOVE / LOOK FUNCTIONS"	34
FIGURE 29: " CHARACTER CROUCH / WALK FUNCTIONS"	34
FIGURE 30: "BINDING FUNCTIONS TO ACTIONS"	35
FIGURE 31: "MOVEMENT INPUT ACTIONS"	35
FIGURE 32: "CUSTOM WEAPONBASE CLASS"	36
FIGURE 33: "CUSTOM AR15 CLASS"	36
FIGURE 34: "AR15 IN THE VIEWPORT"	36
FIGURE 35: "WEAPON ATTACHED TO THE CHARACTER"	37
FIGURE 36: "WEAPON ATTACHED TO THE CHARACTER IN THE VIEWPORT"	37
FIGURE 37: "CHARACTER FIRE FUNCTIONS"	38
FIGURE 38: "CHARACTER RELOAD FUNCTION"	38
FIGURE 39: "CHARACTER ADS FUNCTIONS"	39
FIGURE 40: "CUSTOM HEALTH COMPONENT CLASS"	40
FIGURE 41: " <i>HEALTH COMPONENT DECREASEHEALTH / INCREASEHEALTH FUNCTIONS</i> "	40
FIGURE 42: "CUSTOM ZOMBIEAICONTROLLER CLASS"	41

FIGURE 43: "CUSTOM ZOMBIECHARACTER CLASS"	41
FIGURE 44: "ZOMBIE CHARACTER IN THE VIEWPORT"	42
FIGURE 45: "ZOMBIEAI BEHAVIOR TREE"	42
FIGURE 46: "ZOMBIEAI BLACKBOARD"	43
FIGURE 47: "CUSTOM HUD CLASS"	43
FIGURE 48: "HUD SETHEALTH FUNCTION"	44
FIGURE 49: "HUD SETAMMO FUNCTION"	44
FIGURE 50: "HUD CROSSHAIR FUNCTIONS"	45
FIGURE 51: "CUSTOM HUD IN-GAME"	45
FIGURE 52: "MAIN MENU"	46
FIGURE 53: "PAUSE MENU"	46
FIGURE 54: "SETTINGS MENU"	46
FIGURE 55: "CUSTOM LAPTOP CLASS"	47
FIGURE 56: "LAPTOP IN THE VIEWPORT"	48
FIGURE 57: "INTERACT MESSAGE DISPLAYED"	48
FIGURE 58: "LAPTOP OVERLAP FUNCTIONS"	49
FIGURE 59: "EMPTY DATA TAB"	49
FIGURE 60: "COLLECTED DATA DATA TAB"	50
FIGURE 61: " QUIZ TAB QUESTION"	50
FIGURE 62: "QUIZ TAB SCORE"	51

List of Tables

TABLEAU 1: GENERAL PRESENTATION OF THE COMPANY	3
TABLEAU 2: GENERAL GAME PRESENTATION OF WHILE TRUE: LEARN().....	10
TABLEAU 3: GENERAL GAME PRESENTATION OF MECHANICA	12
TABLEAU 4: PRODUCT BACKLOG	23
TABLEAU 5: TOOLS AND DEVELOPMENT ENVIRONMENT	24

List of abbreviations

(RPGs)Role playing games

(FPS)First person shooters

(MMO)Massively Multiplayer online

(NCPs)non-player characters

(DLC)downloadable content

(UE)Unreal Engine

(XP)Extreme Programming

(HUD)Head-up Display

(PB)Product Backlog

General Introduction

Learning video game development is an ongoing process that requires practice, patience and a willingness to continually update your skills. It is an exciting journey that combines creativity, programming skills and problem-solving skills abilities to bring interactive virtual world to life.

While we can find numerous online resources, tutorials, forums and communities dedicated to game development that can provide guidance and support along the way, learning video game development is a process that takes time and effort. It can be indeed challenging, especially if you are new to programming or have limited experience with the various aspects of game design.

The purpose of this project is to come up with an idea for an educational game that helps it users to learn the basics of video game development, focusing on C++ and Unreal Engine, while playing.

This document will contain 3 Chapters:

- The first chapter “**General presentation**” devoted to the presentation of the start-up, the project problematic, the methodology and plan to which the work will be proceeded.
- The second chapter “**System Design**” model our system using UML, it will be explaining the mechanics and structure of the game.
- As for the last chapter “**Implementation**”, it is the part where we will focus on the tools used in the making of our game. It will also justify our choice of certain softwares over others.

Chapter 1: General View

I. Introduction

This chapter will guide us through the project's overall structure. We'll start out by introducing the host organization: ENVAST. Present the primary issue after which we will discuss the difficulties that prompted the development of and adoption of different solutions in a variety of contexts.

Furthermore, we'll describe the many forms of video games.

II. Company presentation

ENVAST is an IT agency launched in 2016. They create innovative digital products by mixing new technologies with creative designs. They are expertise in the development of high-quality services such as video games, augmented reality, mobile and web applications.



Figure 1: "Logo ENVAST" [1]

Company name	ENVAST
Business sector	Software development
Founding date	2016
Address	Innovation space, Sousse, Tunisia
Phone number	+216.21267722
Website	http://ENVAST.tn/
Company domains	Video games development Augmented reality Mobile and web applications

Tableau 1: General presentation of the company

III. Project context

1. Video Games

Video games are interactive electronic entertainment experiences that involve player participation and engagement. They can be played on different devices such as computers, consoles, mobiles.... Nowadays video games have become a popular form of entertainment captivating players of all ages and backgrounds.

Aside from entertainment, video games have been utilized for educational, training and therapeutic purposes. They can be used to imitate real world circumstances, develop problem solving skills, encourage teamwork and collaboration and provide an interactive learning medium.

2. Action-Adventure Games

An action-adventure game is one that combines aspects from an action game and an adventure game, particularly key components such as puzzles. Many of the same physical talents are required for both action and adventure games.

Action-adventure games typically incorporate a combination of complicated plot components, which are exhibited for players utilizing audio and visual. The story is strongly dependent on the player character's movement, which causes story events and consequently impacts the game flow.



Figure 2: "Assassin's Creed Odyssey" [2]

3. Role Playing Games (RPGs)

RPGs are video games in which players interact with the Gameworld through character with backstories and existing motivations. NCPs (non-player characters), side missions, downloadable content (DLC) and broader story arcs are all common features of the RPG genre.

RPGs first appeared on mainframe computers in the 1970s, following enormous popularity of pen-and-paper, tabletop and role-playing games such as Warhammer, cyberpunk and Dungeons & Dragons. Many early PRGs were inspired by sports simulation games, adventure games and fantasy novels.



Figure 3: "Pillars of Eternity 2" [3]

4. First Person Shooters (FPS)

A first-person shooter (FPS) is a type of action video game in which the player controls the protagonist. FPS games often map the gamer's movements and present a view of what an actual person in the game would see and do.

In a FPS, the protagonist's arms are normally seen at the bottom of the screen, carrying whatever weapon is equipped with. The gamer is expected to move his avatar through the game by using the gaming controller to move it forward, backward and sideways. Forward controller actions cause the avatar to move usually with a small left-right rocking motion to replicate the human walk.

Many games feature the sound of breathing and footsteps in addition to the standard sound effects to boost the level of realism.



Figure 4: "Counter strike: global offensive" [4]

5. Strategy Games

A strategy video game is a type of game that focuses on strategic thinking and planning in order to win. It concentrates on strategic, tactical and occasionally logistical issues. Many games also include economic problems and opportunities for investigation. They are generally categorized into 4 sub-types depending on whether the game is turn-based or real-time and whether the emphasis is on strategy or tactics.



Figure 5: "Endless Legend" [5]

6. Simulation Games

A simulation game is a type of video game that attempts to recreate or simulate real-world activities or scenarios in a virtual environment. These games often allow players to take on the role of a specific or entity and engage in activities that mimic real-life experiences or tasks.

Simulation games are intended to give the player a sense of immersion and realism. They often require strategic thinking, decision making and problem solving. A simulation game's purpose might vary based on its genre and theme form: developing and managing a city to operating a corporation or even imitating an individual's day-to-day existence.



Figure 6: "The Sims 4" [6]

7. Sports Games

Sports games are among the most well-known games on the market today. They are simulations of traditional or real-world sports like football, basketball, American football and so on.

In sports games, players typically control individuals' athletes or entire teams and engage in competitive matches or tournaments against computer-controlled or other human players. To simulate challenges and dynamics of the selected sport, the game player frequently includes aspects as team management, strategy, skill-based control and realistic physics simulations.

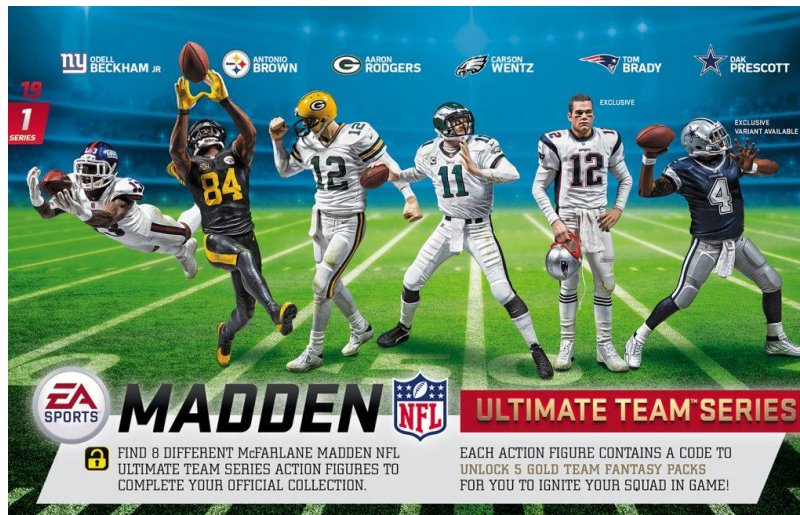


Figure 7: "MADDEN NFL 2019" [7]

8. Massively Multiplayer Online (MMO)

MMO game is a type of video game that allows for a large number of players to play at the same time using an internet connection. These games are typically set in a shared world that the player can access after purchasing or installing the game program.

In MMOs, players create and control virtual avatars or characters, which they use to explore the game world, interact with other players, complete quests or missions. These games often offer vast, open-ended environments that allow for exploration, social interaction, and the development of in-game communities.



Figure 8: "World of Warcraft" [8]

9. Educational Games

An Educational video game is one that provides the player with learning or training value. Edutainment refers to the deliberate combination of video games and educational software into a single product. Unlike traditional educational games, which can encompass various mediums such as board games or card games, educational video games leverage the interactive nature of digital technology to deliver educational content in a dynamic and immersive way. They often utilize graphics, animations, sound effects and interactivity to enhance the learning experience.

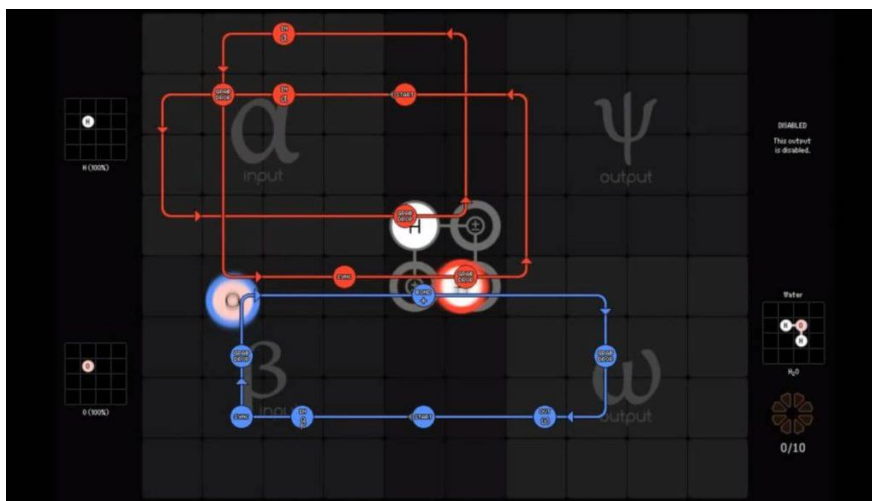


Figure 9: "Space Chem" [9]

IV. Problematic

Starting to learn something new can often be challenging, and this is particularly true in the realm of game development. There are several common problems that aspiring learners may encounter.

1. Information Overload

Learning anything new may be very intimidating as almost every subject known to men is filled with a ton of information which may cause confusion and uncertainty. Game development especially can be challenging due to its intricate blend of art, design, programming, and sound. The multidisciplinary nature of this field can overwhelm newcomers, making it a challenging task.

2. Motivation

Staying motivated and maintaining a consistent learning routine can be a struggle. Game development projects often involve a significant time investment, and the initial excitement can wane as challenges arise. Overcoming these obstacles requires perseverance and structured learning approaches.

V. Existence Study

Numerous video games that teach programming are among the kinds of educational games that have gained popularity in the market.

While True: Learn() :



Figure 10: "While True: Learn()" [10]

Game Presentation :

Publishers	Luden.io, Nival International, Nival Interactive, Inc.
Developer	Luden.io
Release Date	March 28, 2018
Genre	Puzzle Video Game, Simulation Video Game, Indie game
Game Mode	Single-Player
Controls	Touchpad, Mouse and Keyboard, Joysticks

Tableau 2: General game presentation of While True: Learn()

Game Design :

- Scenario :
You play a machine learning specialist who makes neural networks, and you discover that your cat is better at it. Now, to construct a cat-to-human translation system, you must now solve puzzles.
- Gameplay Analysis :
The gameplay consists of solving puzzles with elements similar to visual scripting, meaning nodes that can be attached to each other to represent logic. These puzzles enable the player to learn more about how machine learning and related technologies work.

Criticism :

This game is a fantastic method to learn about AI, big data, neural networks, and machine learning, but it doesn't cover game creation or even programming.

Mechanica :



Figure 11: "Mechanica" [11]

Game Presentation :

Publishers	Deimos Interactive
Developer	Deimos Interactive

Release Date	February 29, 2020
Genre	Indie game
Game Mode	Single-Player / Multi-Player
Controls	Mouse and Keyboard

Tableau 3: General game presentation of Mechanica

Game Design :

- Scenario :

In an open-world, post-apocalyptic map, you play as a sentinel robot, you must strengthen your defenses using a variety of tools to fend off the increasingly aggressive robots. Make use of automation and inventiveness to survive in this hazardous environment.

- Gameplay Analysis :

The gameplay involves using a visual programming system to create any kind of interaction between objects by coding the logic.

This system is very similar to what we can find inside actual game engines such as Unreal Engine and Unity.

Criticism :

Although this game is a lot of fun to play and can aid in a better understanding of game development, it lacks the fundamental concepts which forces players without any prior knowledge to seek assistance elsewhere.

VI. Solution

Video Games are enjoyed by a diverse audience of people of all ages, they offer interactive experiences that cater to a range of interests and inclinations, making their style of entertainment ageless.

The interactive nature of video games provides players with a sense of accomplishment as they complete tasks and overcome obstacles and encourages a sense of progress in them.

By recognizing this, engaging with a video game that teaches basic game development concepts proves to be a valuable solution for easing the learning process.

VII. Conclusion

In this chapter, we outlined the inspiration for this game. We began by giving a very brief overview of video games. Afterward, we defended our game's conformity to the educational game sector by outlining its goals and the means by which it will accomplish them.

The system design utilizing the UML modelling language will be shown in the following section.

Chapter 2: System Design

I. Introduction

This chapter provides an in-depth explanation of the system's intended behavior in order to facilitate its implementation and maintenance. In the first section, we'll go over some specifics regarding the game's scenario as well as the functional and specification requirements for our project's system. Finally, we will present our Product Backlog.

II. About The Game

1. Backstory

Backstory is commonly employed to provide depth or plausibility to a story. In our case, extending the player's experience is beneficial to heighten immersion and put the player in the protagonist's shoes.

2. Executive Summary

You play as the last survival of a postapocalyptic world infested with zombies, and you find yourself fighting for survival while trying to achieve your goals.

3. Storyline

You are the last survivor of a post-apocalyptic world. With all hope lost you try to achieve your long-life dream of becoming a game developer before you die with the only obstacle being the zombie-infested streets stopping you from getting the data necessary to start learning and complete a quiz! Will You be able to succeed ?

III. System's Functional View

In this section, we'll describe the solution's functional and non-functional requirements then analyze and express them.

1. Functional Needs

Every player will be able to:

- Move around the level.
- Access the menu at any time during the game.
- Interact with Objects placed around the level.

- Shoot enemies.

2. Non-Functional Needs

- Maintainability:** The method we devised makes it simple to locate the issue's origin and, consequently, simple to resolve it.
- Scalability:** has potential to add more weapons, interactable objects and more.
- Performance:** Optimized for a smooth pc gaming experience.

3. Actor Identification

Only the Player, the person playing the game, can be identified as an actor in our game.

IV. Game Specifications

1. Modeling Game flow design

a. Modeling language used

Unified Modeling Language or UML is the tool used in this project providing a standard way to visualize the design of a system. Thus, the use cases and a written description of how users will perform tasks on our game which outlines, from a user's point of view, a system's behavior, are provided below.

b. Identification of use cases

We will state the general player interactions throughout the game by illustrating them in diagrams.

The diagram below depicts the player's initial encounter with the game. It demonstrates what he can accomplish when he starts the game:

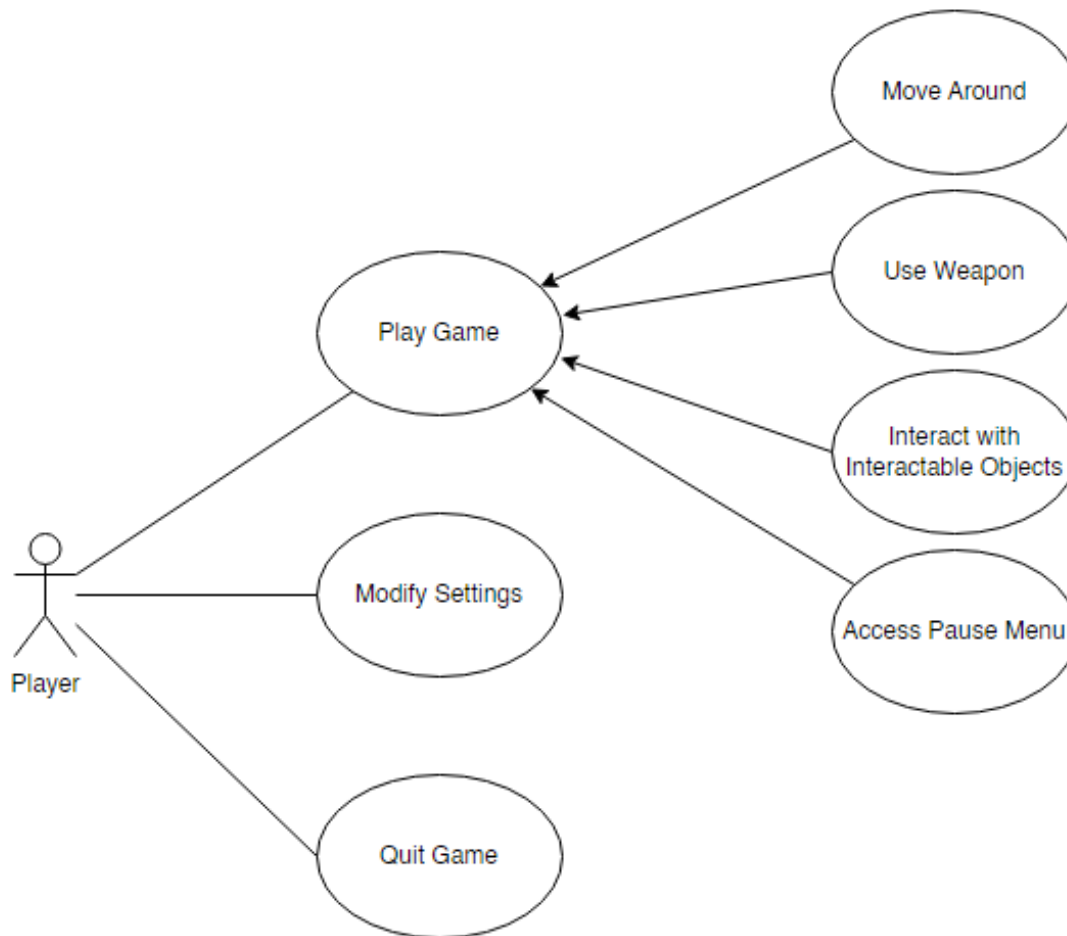


Figure 12: " Main game scene use case diagram"

As the player launches the game, he will be presented by the "Main Menu" in which he will be able to choose between playing the game, modifying the settings or quitting the game.

While playing the player can do the four main in-game actions which are:

- Move around the Level:** using the keyboard and mouse the player can move around the area.
- Interact with different objects:** use laptop, get data and teleport to room.
- Use Weapon:** shoot the zombie enemies, aim down sight and reload.
- Access Pause Menu:** the player can resume game, modify Settings, quit to Main Menu or quit to Desktop.

We will eventually explain the details in the sprints coming later.

Now in this diagram we will look into the details of the main use case diagram of the “GAMEPLAY”:

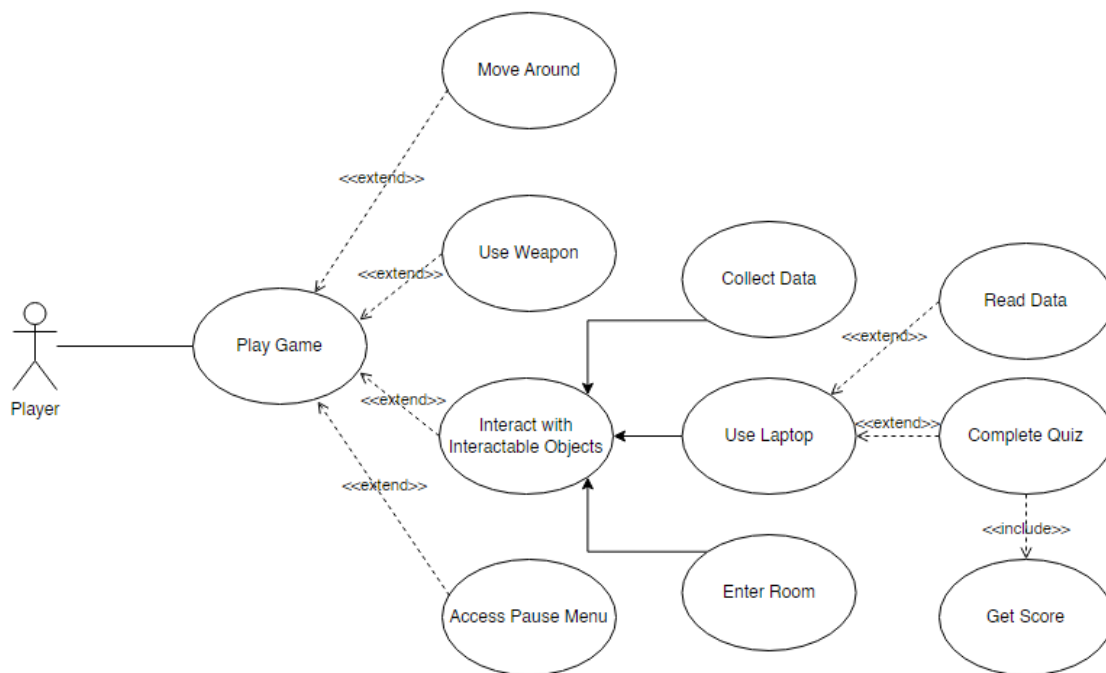


Figure 13: "Gameplay use case Diagram"

- **Main Objective:** Complete the quiz available inside the laptop.
- **Precondition:** The player started the game and pressed the play button.
- **Postcondition:** The player completed the quiz or is dead.
- **Main Scenario:**
 - The player survives the enemy zombies.
 - Collects Data.
 - Completes Quiz.

V. Game Design

1. Game Concept and Genre

Our Game is a 3D FPS educational PC game.

Its core is to survive the zombies, collect data and complete the quiz with as many points as possible.

This game will allow the player to learn game development concepts in general and specifically UE and C++.

2. Target Audience

The game is designed to provide an engaging and educational experience, catering to both beginners and those with prior knowledge of game development. It offers a user-friendly interface and intuitive mechanics that make it accessible to anyone with a passion for video game development at any age.

3. Early Model Design

a. Main Menu

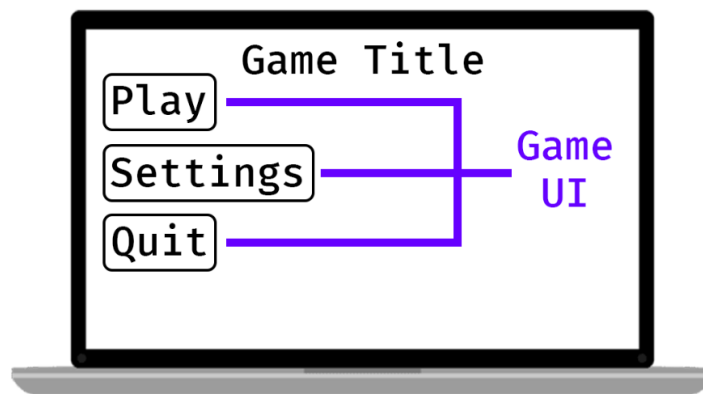


Figure 14: "Main Menu"

The goal here was to create a simple and clean looking menu yet functional and easy to use.

b. In-game UI

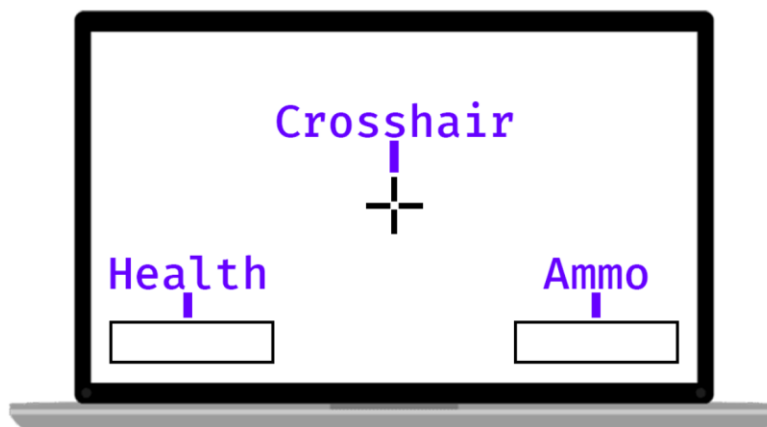


Figure 15: "Head-Up Display"

The aim with this design was to have a very minimalistic looking UI yet providing the all the necessary information needed by the player.

VI. Work methodology

1. Extreme Programming

Taking into consideration being the only developer working on this game and the limited time I have, choosing the Extreme Programming (XP) methodology was a no brainer.

XP is an agile method known for its focus on customer satisfaction, frequent iterations, and continuous improvement.

Some of the core principals of this methodology are:

- Rapid iterations:** short development cycles with frequent iterations to refine features and make improvements quickly.
- Embrace change:** throughout the development process, being open to changes in requirements, design, and scope to adapt to evolving needs.
- Frequent Testing:** prioritize testing to ensure high quality and maintainability of the game's codebase.
- Continuous Integration:** Integrating code and assets regularly to avoid integration issues and ensure a functional, stable build at all times.

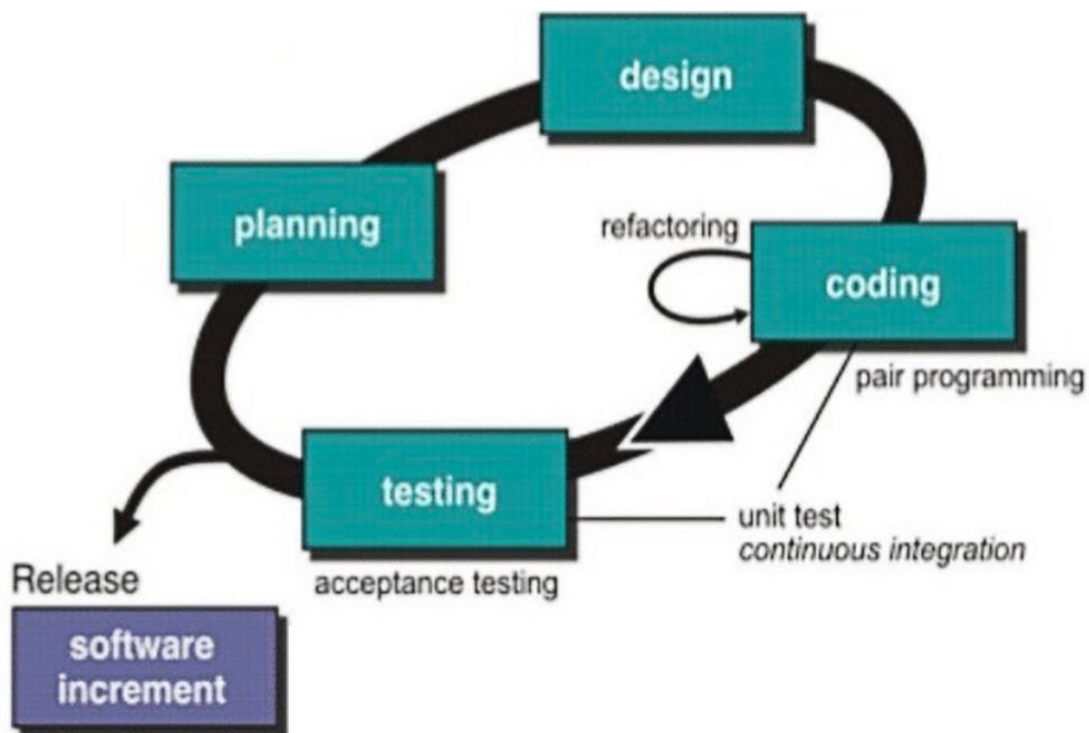
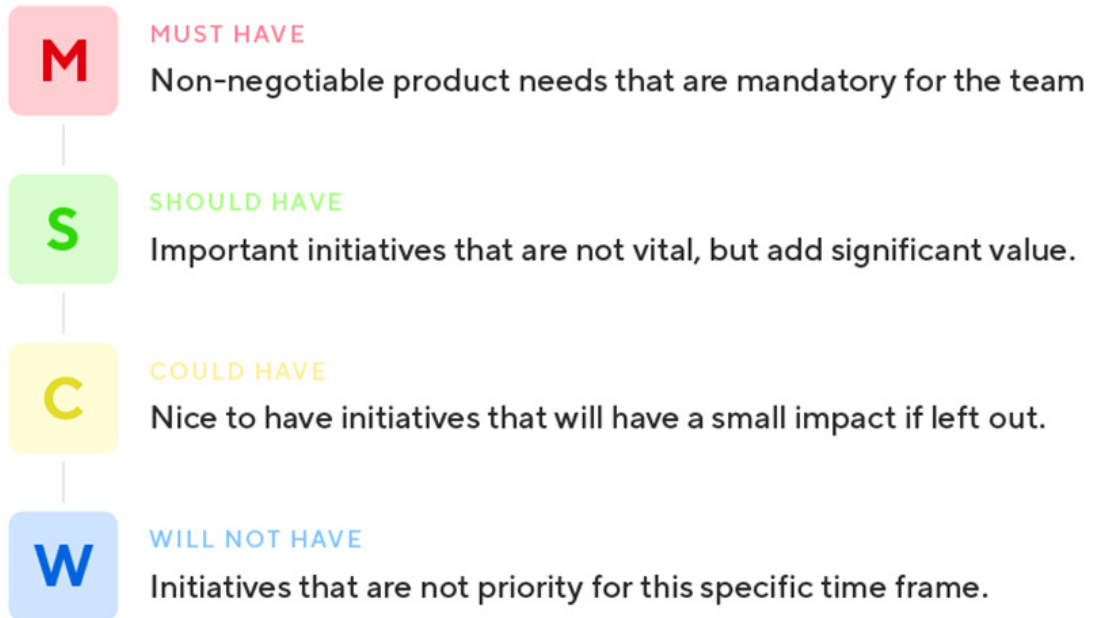


Figure 16: "Extreme Programming "

2. Product Backlog

XP does utilize backlogs as part of its development process which is why we implemented the “MoSCoW” approach as a prioritizing strategy.

The letters in this method's acronym stand for:



We organized the Product Backlog in a table as shown below:

ID	Feature	User Story	Priority
1	Player Movement	As a player I can move and look around the map.	M
2	Basic Weapon Mechanics	As a player I can fire the weapon.	M
		As a player I can aim down sight (ADS).	
		As a player I can reload the weapon.	
3	Menu UI	As a player I can interact with different menus.	M
4	AI Zombie Mechanics	As player I can face zombies and try to kill them.	M
		We designed an AI system that chases and attacks.	
5	Sound Settings	As a player I can configure the sound settings.	S

6	Video Settings	As a player I can configure the video settings.	S
7	Hideout Room	We designed this room to act as a hideout.	S
8	Inventory System	As a player I can carry items in my inventory	W
9	Game Development Office Room	We designed this room to act as an area containing Data.	C
10	Multiple Weapons	As a player I can use multiple weapons.	C
11	Climbing Mechanics	As a player I Can climb walls and obstacles.	W
12	Crafting Mechanics	As a player I can craft weapons, items and more.	W
13	Use Laptop	As a player I can access and read data.	M
		As a player I can play the quiz.	
14	Teleport To Room	As a player I can teleport to a room.	M
15	Toggle Third Person	As a player I can toggle to third person view.	W
16	Collect Ammo	As a player I can collect Ammo for my weapon.	C
17	Collect Health Packs	As a player I can collect Health packs to heal.	C
18	Kill Zombies	As a player I can kill zombies.	M
19	Collect Data	As a player I Can collect Data.	S
20	Mouse Input System	As a player I can use my mouse to interact with the laptop.	M
21	Multiple Zombie Types	We designed multiple zombie types with different abilities.	C
22	Head-Up Display (HUD)	We designed a simple and effective HUD.	M

23	Health System	We designed a simple Health system for the player and the zombies.	M
24	Advance Weapon Mechanics	We designed and advanced weapon mechanics with different attachments.	W
25	Unique Firing Mechanics	We designed a unique firing mechanics using a random spread.	C

Tableau 4: Product Backlog

VII. Tools and development environment

Material Environment	
	<ul style="list-style-type: none"> •Processor: i7-7700k •RAM: 16 Go DDR4 •SSDs: 124 Go + 1 To •Hard Drives: 1To •Graphics Card: Nvidia 1060 6gb
	<p>Wacom Bamboo Pen Touch Tablet: is a professional and reliable graphics tablet.</p>
Technical Environment	
	<p>Unreal Engine 5.1: Unreal Engine is a powerful and versatile game development engine widely used for creating high-quality, immersive, and visually stunning games.</p>






	<p>Photoshop:</p> <p>Photoshop is a popular and robust image editing software used for enhancing, manipulating, and creating digital images with a wide range of tools and features.</p>
	<p>Substance Painter:</p> <p>Substance Painter is a leading 3D painting software that enables artists to create realistic textures and materials for digital models with a user-friendly interface and powerful painting tools.</p>
	<p>Blender:</p> <p>Blender is a comprehensive and open-source 3D creation suite that offers a range of tools for modelling, animation, rendering, and more, making it a versatile software for various creative projects.</p>
	<p>Visual Studio 2022:</p> <p>Visual Studio is a robust and feature-rich integrated development environment (IDE) widely used by developers for coding, debugging, and deploying software applications across various platforms.</p>
	<p>C++:</p> <p>C++ is a powerful and versatile programming language known for its performance and low-level control, widely used in software development, including game development, to create efficient and complex applications with high-performance requirements.</p>

Tableau 5: Tools and development environment

- **Game Engines**

Game engines are software frameworks for creating video games, offering tools for graphics, physics, audio, and more. They save development time by providing pre-built functionality. They enable developers to focus on game-specific logic and content creation. They support cross-platform compatibility, allowing games to be deployed on various devices. Game Engine examples include Unreal Engine, Unity, and Godot Engine.

- **Why Unreal Engine?**

Among the available open-source game engines offering comprehensive professional functionality for free, Unreal Engine, created by “Epic Games” stands as a perpetual competitor to Unity3D. Deciding between these two robust engines can be challenging as both possess impressive capabilities and offer various advantageous features to suit specific needs. Unreal Engine, known for its wide-ranging capabilities and stunning graphics, provides a substantial community of users and developers, along with an extensive forum and a huge marketplace with a lot of free assets, making it a popular choice for game development.

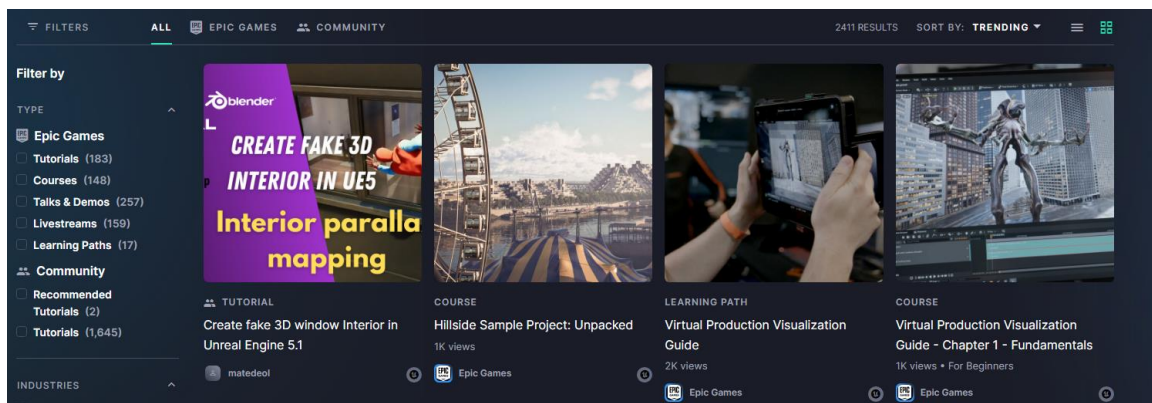


Figure 17: "UE Learning Library" [12]

The UE Learning library contains tons of tutorials, courses, demos and more provided by epic games themselves and by the community ranging from beginner level to expert and across all industries.

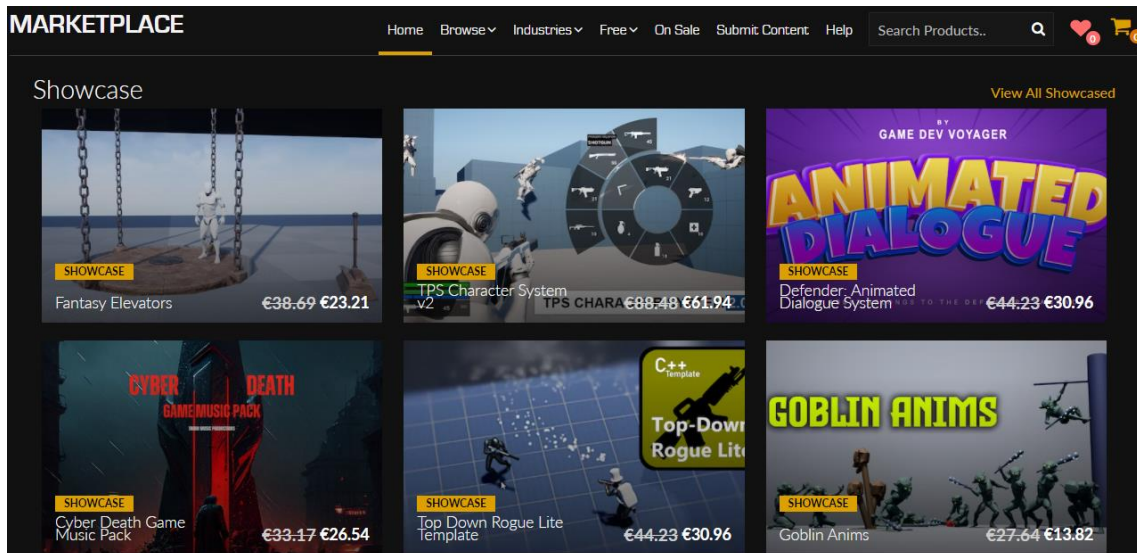


Figure 18: "UE MarketPlace" [13]

The UE Marketplace offers a vast selection of assets, resources, and plugins, paid or free, created by developers worldwide or even by epic games, allowing users to enhance and accelerate their game development process.

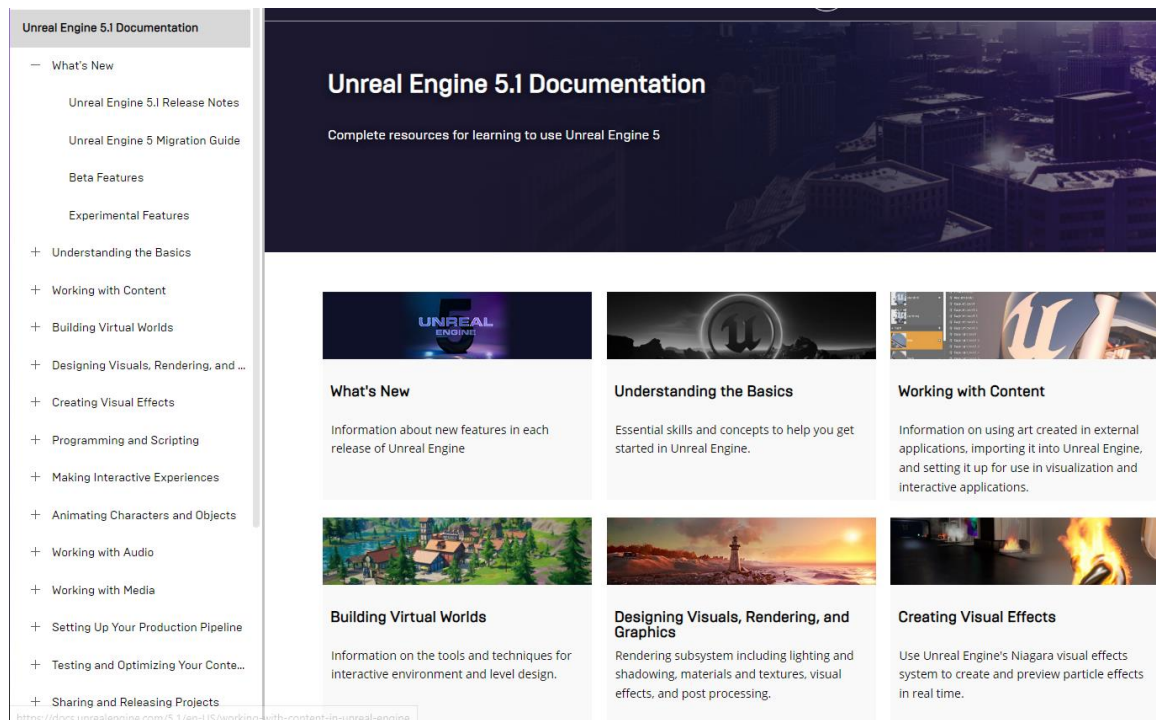


Figure 19: "UE 5.1 Documentation" [14]

UE documentation provides comprehensive and detailed resources, tutorials, and guides to assist developers in understanding and utilizing the features and functionalities of Unreal Engine for effective game development.

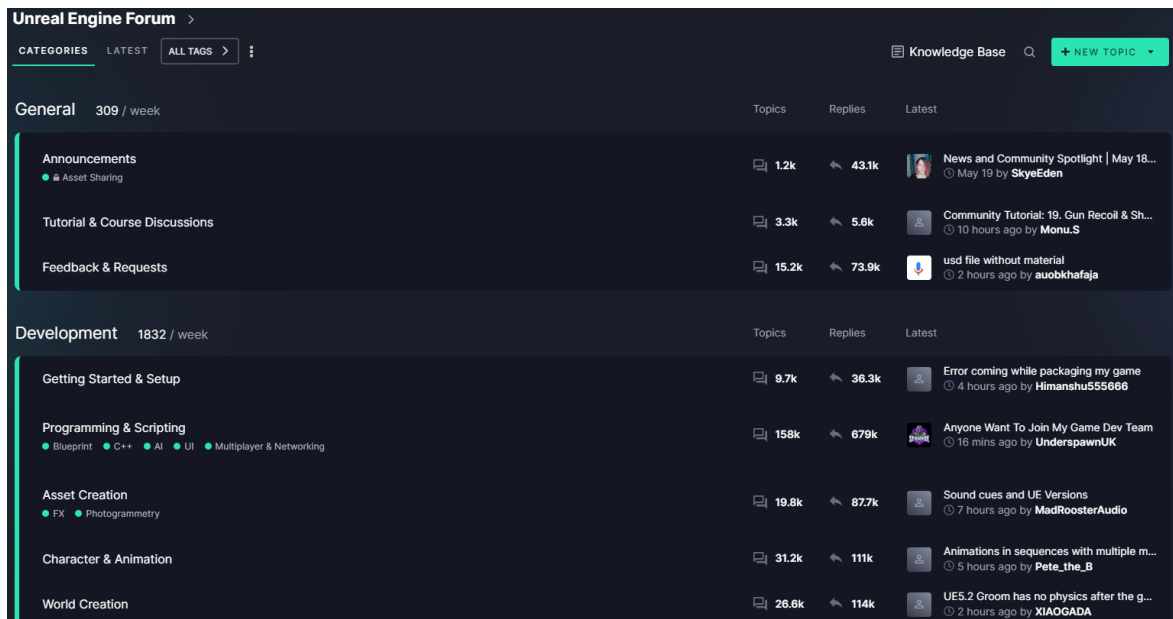


Figure 20: "UE Forums" [15]

The UE forums serve as an online community hub where developers can connect, collaborate, and seek assistance from a diverse community of Unreal Engine users, sharing knowledge, insights, and troubleshooting solutions.

VIII. Conclusion

In this Chapter, we attempted to describe the various functional and non-functional needs, the PB, and to provide more information about the essential elements of our game.

We'll reveal more and more internal game information in the following chapter.

Chapter 3: Implementation

I. Introduction

Throughout this chapter, we'll develop our game piece by piece and discuss the challenges we faced and how we overcame them.

II. Pre-requirements

1. Unreal Engine

- Setup the correct version of UE

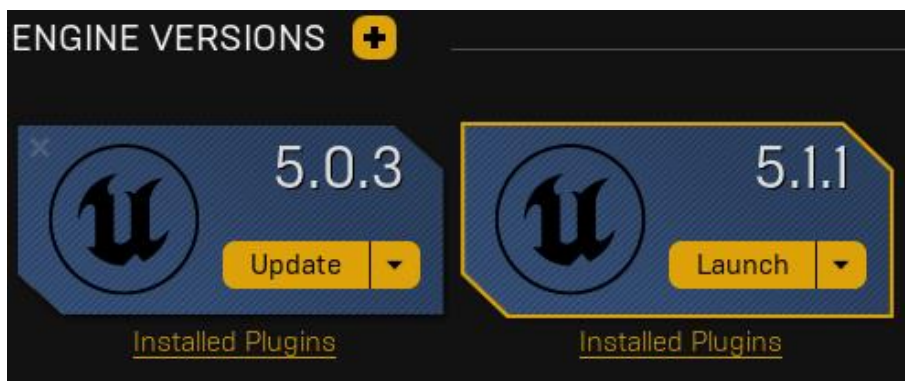


Figure 21: "Unreal Engine Version Selection"

- Edit the Installation Options

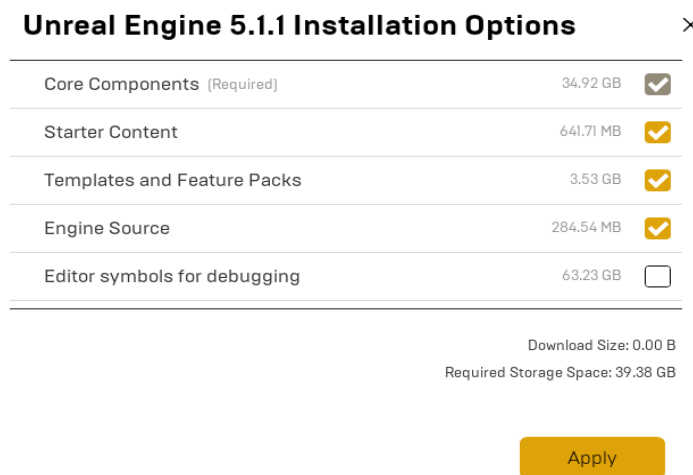


Figure 22: "Unreal Engine Installation Options"

- Install plugins if necessary.

Unreal Engine 5.1.1 Unreal Engine Plugins

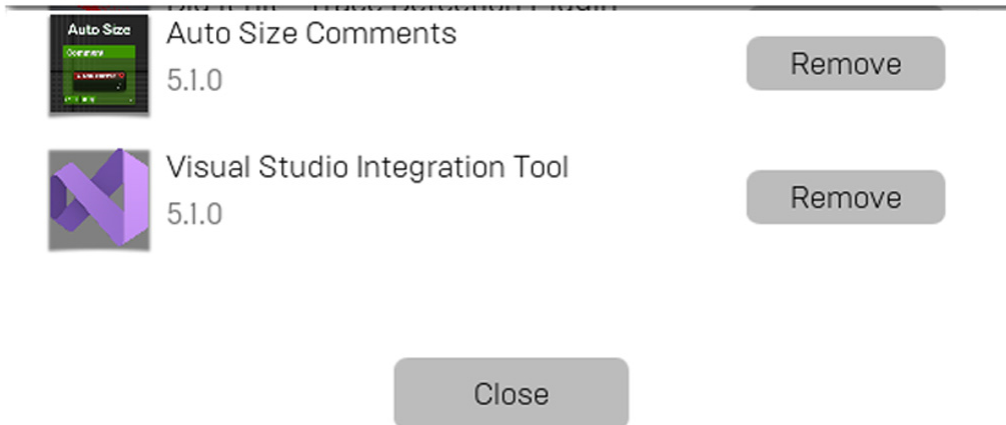


Figure 23: "Unreal Engine Plugins"

- Create your project.

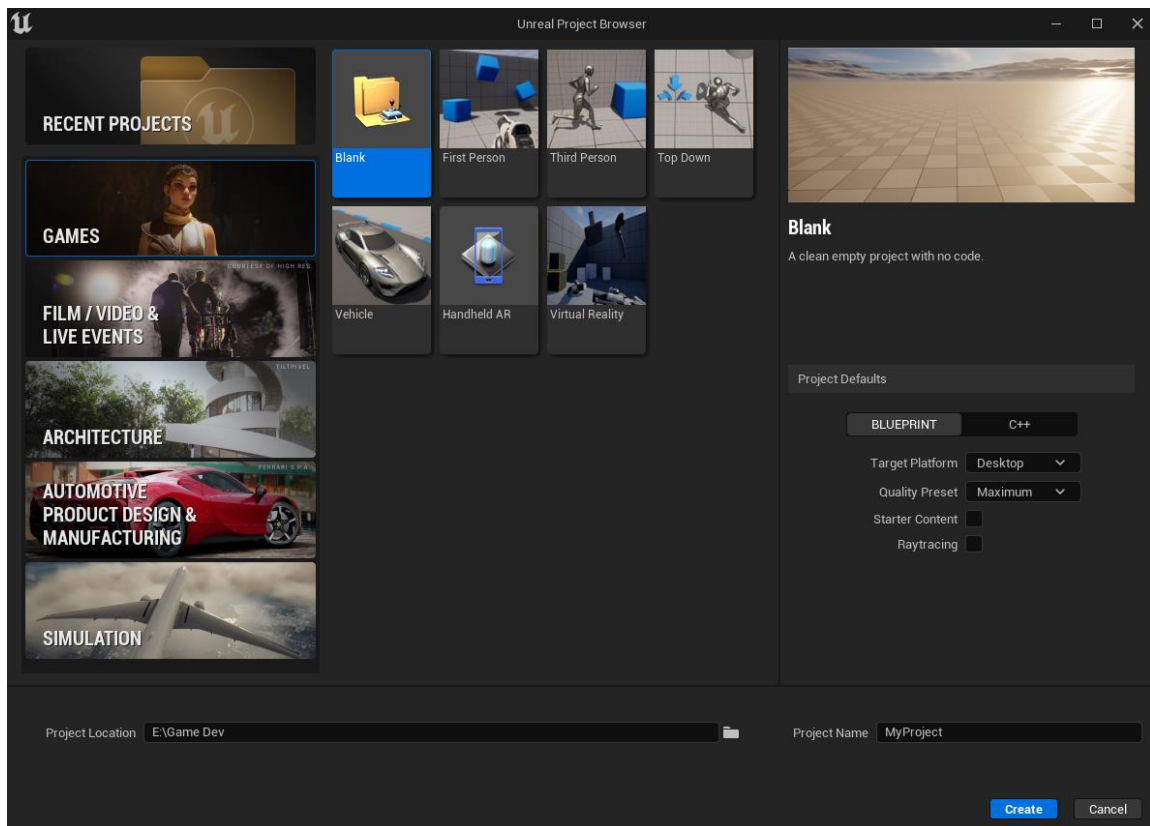


Figure 24: "Unreal Engine project Creation"

To create your project, you need to launch UE, choose a template, choose between Blueprint or C++ (you can make Blueprint projects C++ later if needed), Choose your Target platform and finally Choose the project name and location.

2. Visual Studio

To be able to use Visual Studio with UE you need to first install the right version (for UE 5.1 it is recommended to use VS 2022) then choose these settings:

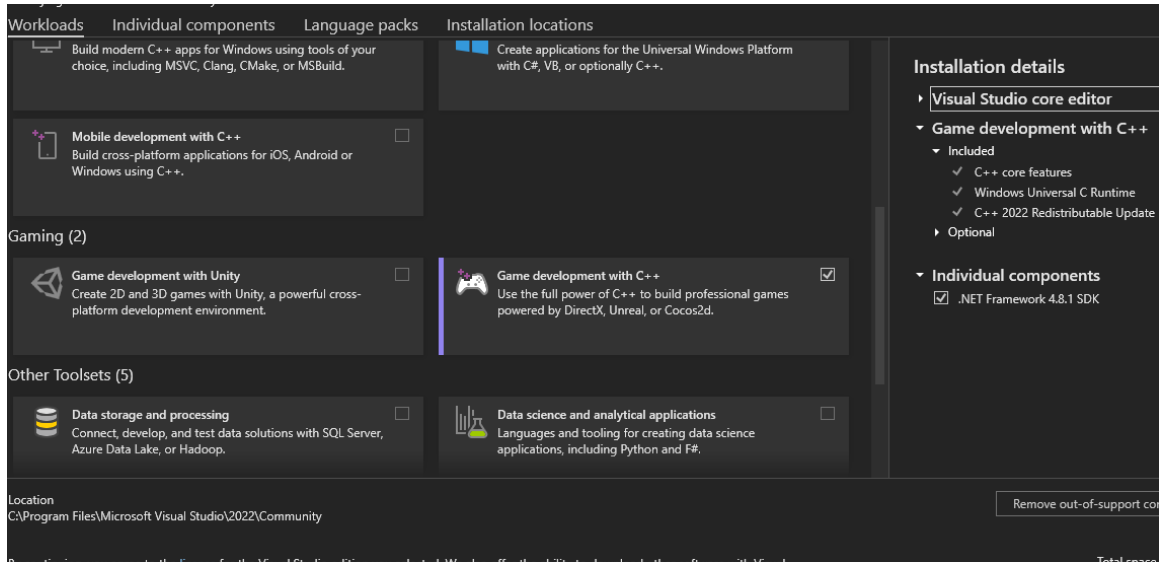


Figure 25: "Visual Studio Installation"

III. Textual Description

1. Game Design Document

A GDD is a living software design document that goes into great detail to describe a video game's design. The game development team's designers, artists, and programmers worked together to create the document, which serves as a roadmap for the entire project.

Why is it a good idea to use it even when working alone?

For efficient game development process management, thorough internal documentation must be kept. Without initially producing a game design document, starting to code will almost always result in placeholder art, broken code, and incompatible mechanics.

Final GDD:

- Summary: An educational 3D game that teaches the player concepts of video game development with UE C++.

- Story:
 - Characters: Only 1: the main character.
 - Setting: a post-apocalyptic abandoned city.
 - Narrative: The main character who is the last survivor trying to achieve his dream of becoming a game developer before his inevitable death.
- Gameplay:
 - Core Loop: Fight for survival, collect data and complete a quiz.
 - Mechanics: Moving, weaponry, health system, functional laptop...
 - Enemies: Zombies that roam around, chase the player when they detect him and attack when at close range.
- Level Design:
 - Progression: Fight for survival -> Collect Data -> Play Quiz.
 - Environments: The City: An overgrown post-apocalyptic city with broken cars and buildings.

The Hideout room: A hideout where the player can safely use the laptop.

- Art: Realist and gloomy.
- UI/Game Controls:
 - UI: Should reflect the theme of “Game Development”.
 - Controls: Standard FPS PC controls.
- Audio: No music / creepy ambient sound.
- Targets:
 - Audience: Anyone with a passion for game development and video games.
 - Platform: PC.

2. Character Movement

The Character movement was created by implementing Unreal’s API.

This was possible by making our Character class inherit from Unreal’s ACharacter class.

```
UCLASS(config = Game)
1 Blueprint reference
class ATheLastGameDevCharacter : public ACharacter
{
    GENERATED_BODY()
}
```

Figure 26: "Custom Character Class"

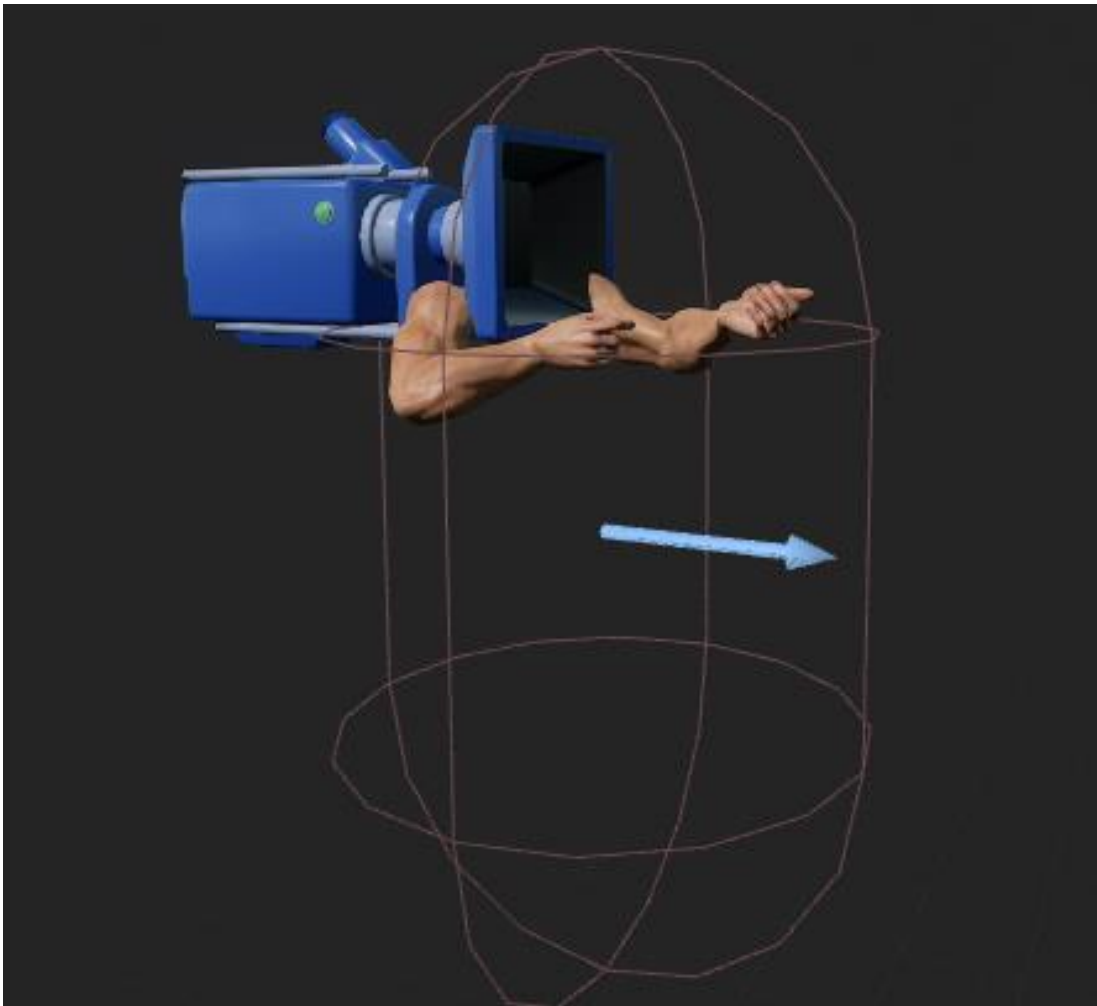


Figure 27: "Character in Viewport"

```

void ATheLastGameDevCharacter::Move(const FInputActionValue &Value)
{
    // input is a Vector2D
    FVector2D MovementVector = Value.Get<FVector2D>();

    if (Controller != nullptr)
    {
        // add movement
        AddMovementInput(WorldDirection: GetActorForwardVector(), MovementVector.Y);
        AddMovementInput(WorldDirection: GetActorRightVector(), MovementVector.X);
        SideMovement = MovementVector.X;
    }
}

void ATheLastGameDevCharacter::Look(const FInputActionValue &Value)
{
    // input is a Vector2D
    FVector2D LookAxisVector = (Value.Get<FVector2D>()) * Sensitivity;

    if (Controller != nullptr)
    {
        // add yaw and pitch input to controller
        AddControllerYawInput(LookAxisVector.X);
        AddControllerPitchInput(-LookAxisVector.Y);
        MouseX = LookAxisVector.X;
        MouseY = LookAxisVector.Y;
    }
}

```

Figure 28: "Character Move / Look functions"

```

void ATheLastGameDevCharacter::StartCrouch()
{
    Crouch();
}

void ATheLastGameDevCharacter::StopCrouch()
{
    UnCrouch();
}

void ATheLastGameDevCharacter::StartWalk()
{
    StopCrouch();

    bIsWalking = true;
    GetCharacterMovement()->MaxWalkSpeed = 150.f;
}

void ATheLastGameDevCharacter::StopWalk()
{
    bIsWalking = false;
    GetCharacterMovement()->MaxWalkSpeed = 600.f;
}

```

Figure 29: "Character Crouch / Walk functions"

The player should be able to move and look around as well as jump and crouch this process is only possible after binding the respective Input Actions.

```

// Jumping
EnhancedInputComponent->BindAction(JumpAction, ETriggerEvent::Started, Object::this, &ACharacter::Jump);
EnhancedInputComponent->BindAction(JumpAction, ETriggerEvent::Completed, Object::this, &ACharacter::StopJumping);

// Moving
EnhancedInputComponent->BindAction(MoveAction, ETriggerEvent::Triggered, Object::this, &ATheLastGameDevCharacter::Move);

// Looking
EnhancedInputComponent->BindAction(LookAction, ETriggerEvent::Triggered, Object::this, &ATheLastGameDevCharacter::Look);

// Crouching
EnhancedInputComponent->BindAction(CrouchAction, ETriggerEvent::Started, Object::this, &ATheLastGameDevCharacter::StartCrouch);
EnhancedInputComponent->BindAction(CrouchAction, ETriggerEvent::Completed, Object::this, &ATheLastGameDevCharacter::StopCrouch);

// Walking
EnhancedInputComponent->BindAction(WalkAction, ETriggerEvent::Started, Object::this, &ATheLastGameDevCharacter::StartWalk);
EnhancedInputComponent->BindAction(WalkAction, ETriggerEvent::Completed, Object::this, &ATheLastGameDevCharacter::StopWalk);

```

Figure 30: "Binding functions to actions"

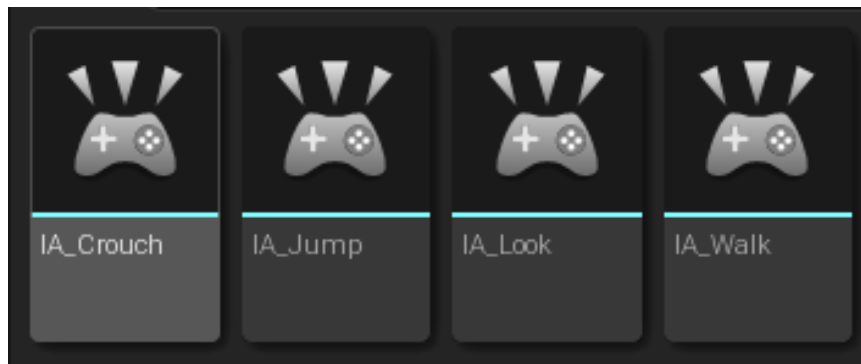


Figure 31: "Movement Input Actions"

3. Basic Weapon Mechanics

The basic Weapon Mechanics needed for any FPS game are Shooting and Reloading and Optionally ADS.

This was achieved by creating a new WeaponBase class inheriting from AActor class and then creating another weapon specific class in our example it's the AR15.

This approach enables us to implement more weapons if needed in the future.

```

UCLASS()
1 Blueprint reference
class THELASTGAMEDEV_API ATheLastGameDevWeaponBase : public AActor
{
    GENERATED_BODY()

public:
    // Sets default values for this actor's properties
    ATheLastGameDevWeaponBase();

```

Figure 32: "Custom WeaponBase Class"

```

UCLASS()
1 Blueprint reference
class THELASTGAMEDEV_API ATheLastGameDevAR15 : public ATheLastGameDevWeaponBase
{
    GENERATED_BODY()

public:
    // Sets default values for this actor's properties
    ATheLastGameDevAR15();

```

Figure 33: "Custom AR15 Class"



Figure 34: "AR15 in the viewport"

To be able to use the weapon we must attach it to the character then call the needed functions inside the character:

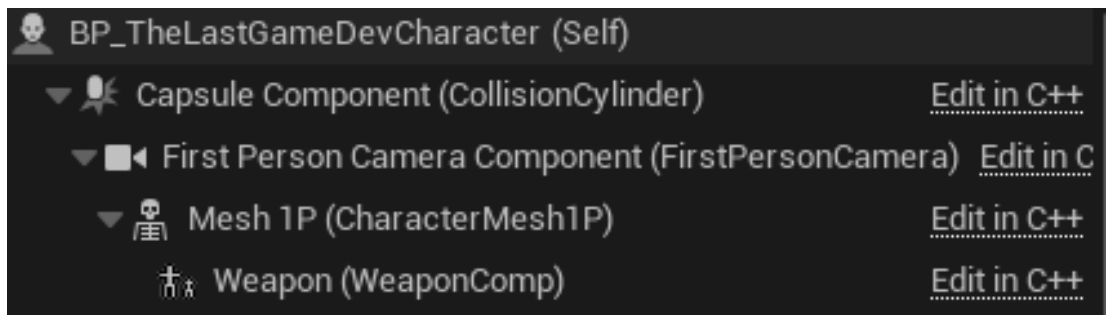


Figure 35: "Weapon attached to the Character"

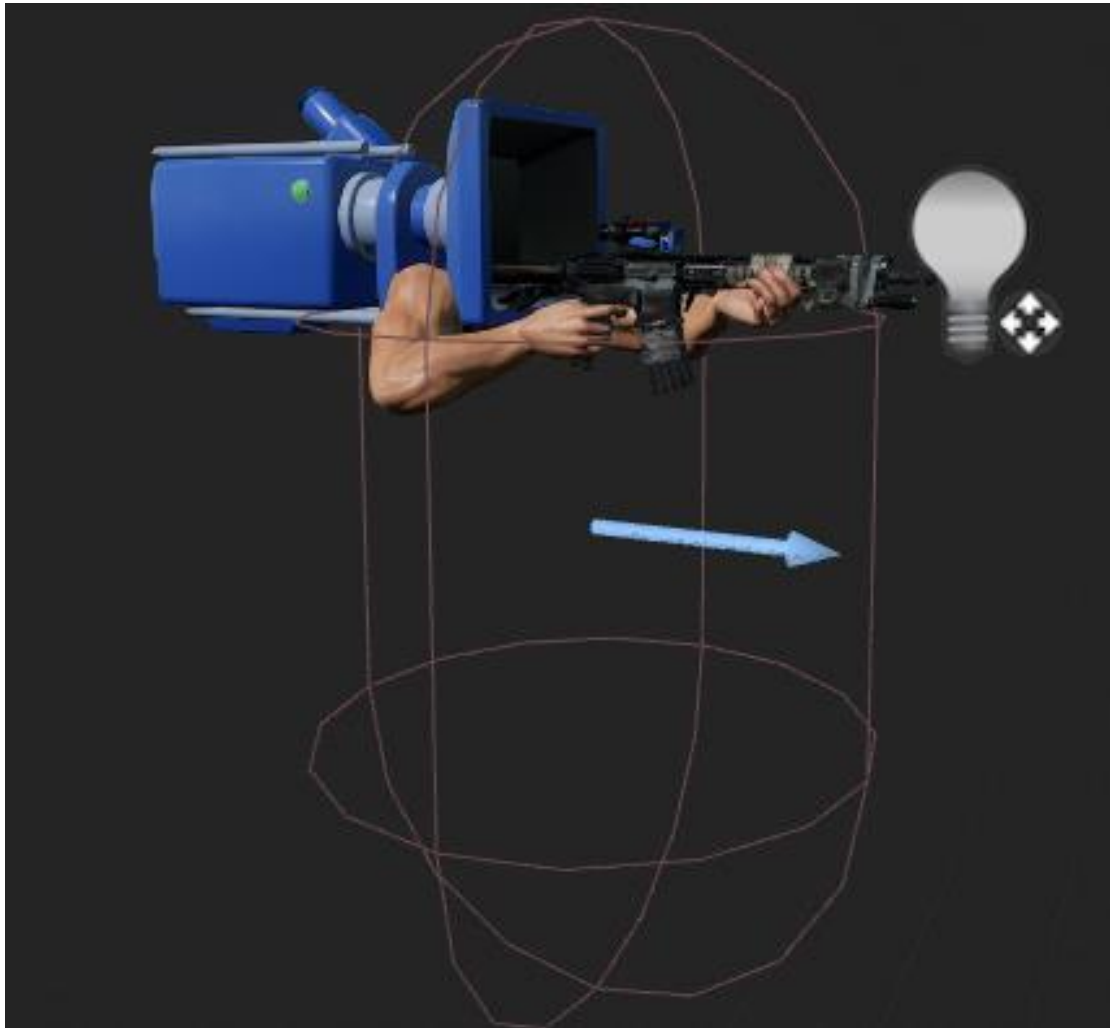


Figure 36: "Weapon attached to the Character in the viewport"

```

void ATheLastGameDevCharacter::StartFire()
{
    if (!bIsReloading)
    {
        bIsShooting = true;

        FireShot();

        GetWorldTimerManager().SetTimer([0] FireTimerHandle, InObj: this,
    }
}

void ATheLastGameDevCharacter::StopFire()
{
    GetWorldTimerManager().ClearTimer([6] FireTimerHandle);

    bIsShooting = false;

    GetWeaponChildActor()->StopFire();
}

void ATheLastGameDevCharacter::FireShot()
{
    GetWeaponChildActor()->Fire();
}

```

Figure 37: "Character Fire functions"

```

void ATheLastGameDevCharacter::Reload()
{
    if (GetWeaponChildActor()->AmmoCount == GetWeaponChildActor()->MaxAmmo)
    {
        return;
    }

    GetCharacterMovement()->MaxWalkSpeed = 150.f;

    if (bIsADSing)
    {
        StopADS();
    }

    if (bIsShooting)
    {
        StopFire();
    }

    GetWeaponChildActor()->Reload();

    if (!bIsWalking)
    {
        GetCharacterMovement()->MaxWalkSpeed = 600.f;
    }
}

```

Figure 38: "Character Reload function"


```

void ATheLastGameDevCharacter::ADS()
{
    if (!bIsReloading && !GetCharacterMovement()->IsFalling())
    {
        GetCharacterMovement()->MaxWalkSpeed = 150.f;

        bIsADSing = true;

        if (bADSBulletSpread)
        {
            TempBulletSpread = GetWeaponChildActor()->BulletSpread;
            GetWeaponChildActor()->BulletSpread = 0;
            bADSBulletSpread = false;

            ADSCameraZoom(bIsADSing);
            ADSCameraZoomTimeline->SetLooping(false);
            ADSCameraZoomTimeline->PlayFromStart();
        }

        if (HUD)
        {
            HUD->UnDrawCrosshair();
        }
    }
}

void ATheLastGameDevCharacter::StopADS()
{
    GetCharacterMovement()->MaxWalkSpeed = 600.f;
    bIsADSing = false;

    GetWeaponChildActor()->ADS(bIsADSing);

    GetWeaponChildActor()->BulletSpread = TempBulletSpread;
    bADSBulletSpread = true;

    ADSCameraZoom(bIsADSing);
    ADSCameraZoomTimeline->SetLooping(false);
    ADSCameraZoomTimeline->PlayFromStart();

    if (HUD)
    {
        HUD->DrawCrosshair();
    }
}

```

Figure 39: "Character ADS functions"

4. Health System

Both the character and the Zombies need to be able to use the Health system and to achieve that the simplest and most efficient way is to create a custom component inheriting Unreal's UActorComponent Class.

```
UCLASS( ClassGroup=(Custom), meta=(BlueprintSpawnableComponent) )
0 Blueprint references
class THELASTGAMEDEV_API UHealthComponent : public UActorComponent
```

Figure 40: "Custom Health Component Class"

This health component has 2 functions that can be called, when necessary, inside the character class that uses the health component: DecreaseHealth() and IncreaseHealth()

```
public:
    // Called to decrease Health base
    UFUNCTION(BlueprintCallable)
    0 Blueprint references
    void DecreaseHealth(float Damage);

    // Called to increase Health base
    UFUNCTION(BlueprintCallable)
    0 Blueprint references
    void IncreaseHealth(float Heal);
```

Figure 41: "Health Component Decreasehealth / Increasehealth functions"

The purpose of IncreaseHealth() function here is just to enable the implementation of a healing item in the future, for example a health pack.

5. AI Zombie Mechanics

The AI zombie mechanics are by far the most intricate system used in this project.

To be able to implement a Zombie AI we need to create 2 main classes:

A ZombieAIController and a ZombieCharacter.

➤ ZombieAIController:

This class is inherited from Unreal's AIController class and it is responsible for controlling how AI characters behave and make decisions in the game world. It can also give them senses like sight and hearing.

```

class UAISenseConfig_Hearing;
class UAISenseConfig_Sight;
class UBehaviorTree;
class UAIPerceptionComponent;

/**
 *
 */
UCLASS()
1 Blueprint reference
class THELASTGAMEDEV_API AZombieAIController : public AAIController
{
    GENERATED_BODY()
}

```

Figure 42: "Custom ZombieAIController Class"

➤ ZombieCharacter:

Like our custom Character class this class also inherits its functionality from Unreal's ACharacter class and this is where we implement the health system and attack action of the zombies.

```

class UHealthComponent;

UCLASS()
1 Blueprint reference
class THELASTGAMEDEV_API AZombieCharacter : public ACharacter
{
    GENERATED_BODY()
}

```

Figure 43: "Custom ZombieCharacter Class"



Figure 44: "Zombie Character in the viewport"

To be able to connect the ZombieAIController to the ZombieCharacter we need to use a visual scripting system provided by UE Called "Behavior Tree".

This Behavior Tree allows us to define and organize AI behavior and decision-making.

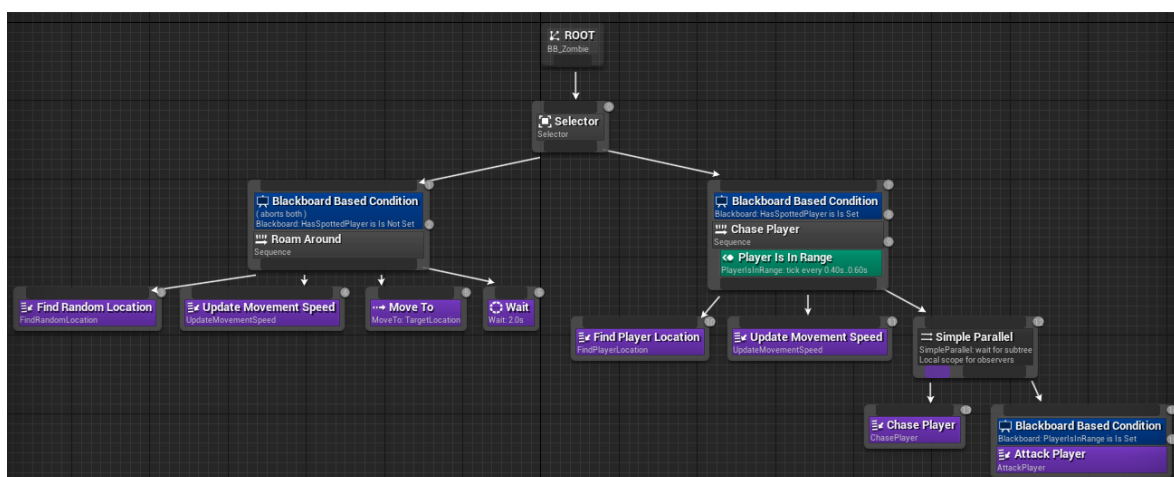


Figure 45: "ZombieAI Behavior Tree"

Using a Behavior Tree requires providing it with a data storage called "Blackboard" that acts as shared memory space for multiple AI agents of the same type.

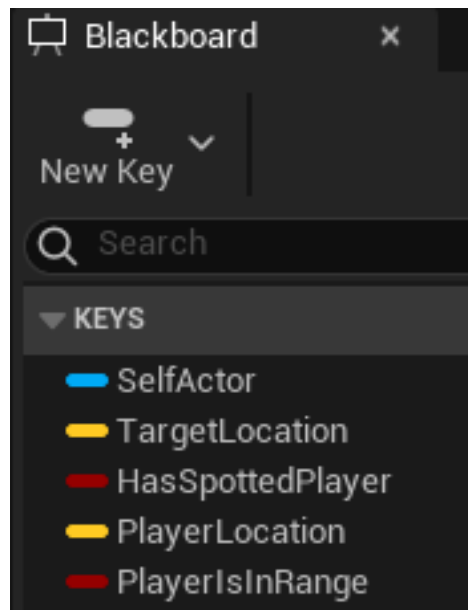


Figure 46: "ZombieAI Blackboard"

6. Head-up Display

The implementation of a HUD can be achieved with two methods in UE, either by inheriting from the UHUD class or from UUserWidget.

For our project we chose UUserWidget class as it is newer, more powerful and versatile compared to UHUD.

```
UCLASS()
1 Blueprint reference
class THELASTGAMEDEV_API UTheLastGameDevHUD : public UUserWidget
{
    GENERATED_BODY()
}
```

Figure 47: "Custom HUD Class"

Our Custom HUD class contains the code that sets the health amount and the ammo amount.

```

void UTheLastGameDevHUD::SetHealth(const float CurrentHealth, const float MaxHealth)
{
    if (HealthBar)
    {
        HealthBar->SetPercent(CurrentHealth / MaxHealth);
    }
    if (HealthTextBlock)
    {
        // Define the text content with markup tags
        const FString HealthString = FString::Printf(Fmt: TEXT("<Class>float</> Health<
        // Set the text content
        const FText HealthText = FText::FromString(HealthString);
        HealthTextBlock->SetText(HealthText);
    }
}

```

Figure 48: "HUD SetHealth function"

```

void UTheLastGameDevHUD::SetAmmo(const float CurrentAmmo, const float MaxAmmo)
{
    if (AmmoTextBlock)
    {
        if (CurrentAmmo == MaxAmmo)
        {
            // Define the text content with markup tags
            const FString AmmoString = FString::Printf(Fmt: TEXT("<Class>float</>
            // Set the text content
            const FText AmmoText = FText::FromString(AmmoString);
            AmmoTextBlock->SetText(AmmoText);
        }
        else
        {
            // Define the text content with markup tags
            const FString AmmoString = FString::Printf(Fmt: TEXT("<Class>float</>
            // Set the text content
            const FText AmmoText = FText::FromString(AmmoString);
            AmmoTextBlock->SetText(AmmoText);
        }
    }
}

```

Figure 49: "HUD SetAmmo function"

The Hud also contains Crosshair Logic which is coded in a way where it is possible to make it dynamic in the future by changing a few lines of code.

```

void UTheLastGameDevHUD::DrawCrosshair()
{
    LeftSlot->SetSize(FVector2D(InX: CrosshairLength, InY: CrosshairThickness));
    LeftSlot->SetPosition(FVector2D(InX: -(CrosshairLength + (CrosshairGap / 2)), InY: -(CrosshairThickness / 2)));

    RightSlot->SetSize(FVector2D(InX: CrosshairLength, InY: CrosshairThickness));
    RightSlot->SetPosition(FVector2D(InX: (CrosshairGap / 2), InY: -(CrosshairThickness / 2)));

    TopSlot->SetSize(FVector2D(InX: CrosshairThickness, InY: CrosshairLength));
    TopSlot->SetPosition(FVector2D(InX: -(CrosshairThickness / 2), InY: -(CrosshairLength + (CrosshairGap / 2))));

    BottomSlot->SetSize(FVector2D(InX: CrosshairThickness, InY: CrosshairLength));
    BottomSlot->SetPosition(FVector2D(InX: -(CrosshairThickness / 2), InY: (CrosshairGap / 2)));
}

void UTheLastGameDevHUD::UnDrawCrosshair()
{
    LeftSlot->SetSize(FVector2D(InF: 0.f));
    RightSlot->SetSize(FVector2D(InF: 0.f));
    TopSlot->SetSize(FVector2D(InF: 0.f));
    BottomSlot->SetSize(FVector2D(InF: 0.f));
}

```

Figure 50: "HUD Crosshair functions"

Creating our HUD class by inheriting from UUserWidget allows us to customize the look of it to our hearts content.

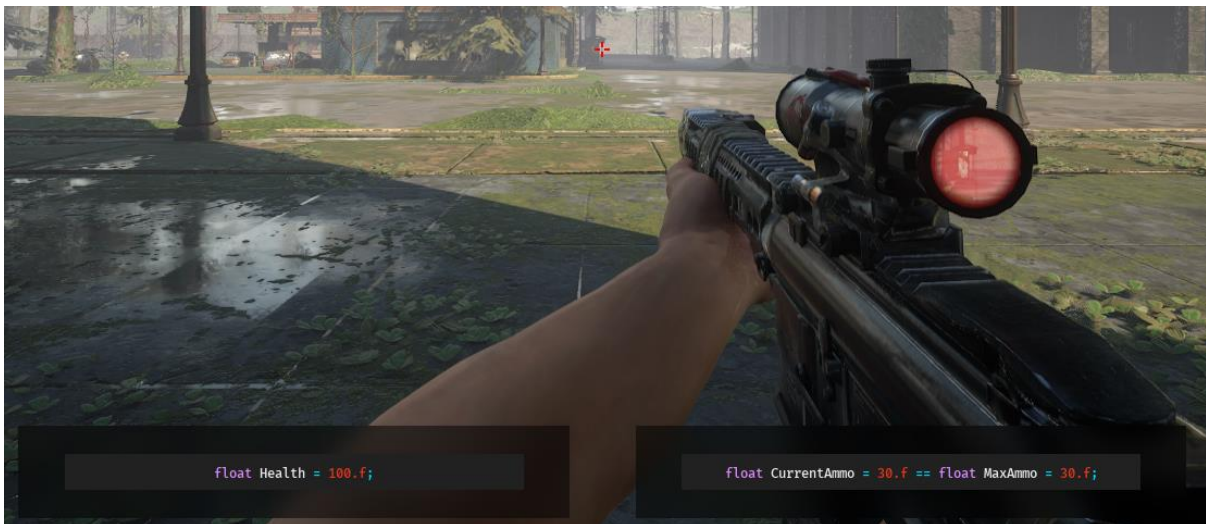


Figure 51: "Custom HUD in-game"

For the look of our HUD, we decided to keep it minimalistic, modern and following the theme of game development by taking inspiration from VS Text Editor and.

7. Menu UI

The design for the menu UIs like the Main Menu, Pause Menu and Settings menu was chosen specifically for easy use and provide an idea of the syntax of UE and C++ development, this not only gives our game a unique look, but also achieves the goal of our project which is developing an educational game.

```

Class TheLastGameDev{

    UButton* Play;

    UButton* Settings;

    UButton* QuitGame;

}

```

Figure 52: "Main Menu"

```

Class TheLastGameDev{

    UButton* Resume;

    UButton* Settings;

    void Quit(UButton* ToMainMenu, UButton* ToDesktop);

}

```

Figure 53: "Pause Menu"

```

Class TheLastGameDev{

    void WindowMode(UButton* FullScreen, UButton* Windowed UButton* WindowedFullscreen );

    void Resolution(UComboBox( 1920x1080- ));

    void ViewDistance( UButton* Near, UButton* Medium, UButton* Far, UButton* Epic );

    void PostProcessing( UButton* Low, UButton* Medium, UButton* High, UButton* Epic );

    void AntiAliasing( UButton* Low, UButton* Medium, UButton* High, UButton* Epic );

    void TextureQuality( UButton* Low, UButton* Medium, UButton* High, UButton* Epic );

    void ShadowQuality( UButton* Low, UButton* Medium, UButton* High, UButton* Epic );

    void OptimalSettings( UButton* SetOptimal );

    void SaveSettings( UButton* SaveSettings );

    UButton* Back;

}

```

Figure 54: "Settings Menu"

8. Laptop

The laptop is the highlight of our project, so we took the time to find a solution that is simple to understand and implement yet attractive to the player.

The solution we found to a complex idea such as usable laptop inside a video game is to utilize UUserWidget to represent the display and AActor to enable the player to interact with it.

```
UCLASS()
1 Blueprint reference
class THELASTGAMEDEV_API ALaptop : public AActor
{
    GENERATED_BODY()

public:
    // Sets default values for this actor's properties
    ALaptop();

protected:
    // Called when the game starts or when spawned
    virtual void BeginPlay() override;

public:
    // Called every frame
    virtual void Tick(float DeltaTime) override;

    UPROPERTY(EditAnywhere)
    Changed in 1 Blueprint
    UStaticMeshComponent* LaptopMesh;

    UPROPERTY(VisibleAnywhere, BlueprintReadWrite)
    Changed in 1 Blueprint
    UBoxComponent* CollisionBox;
```

Figure 55: "Custom Laptop class"

The laptop itself is just a mesh with a collision box that notifies the player that he can interact with it when there is an overlap by displaying a message on the screen.



Figure 56: "Laptop in the viewport"



Figure 57: "Interact message displayed"

```

void ALaptop::OnOverlapBegin(UPrimitiveComponent* OverlappedComp
int32 OtherBodyIndex, bool bFromSweep, const FHitResult& Swee
{
    if (Cast<ATheLastGameDevCharacter>(OtherActor))
    {
        bIsInRange = true;

        ToggleLaptop();
    }
}

void ALaptop::OnOverlapEnd(UPrimitiveComponent* OverlappedComp,
int32 OtherBodyIndex)
{
    if (Cast<ATheLastGameDevCharacter>(OtherActor))
    {
        bIsInRange = false;

        Character = nullptr;

        ToggleLaptop();
    }
}

```

Figure 58: "Laptop Overlap Functions"

When the player presses the interact key the laptop screen will appear as a widget where the player can use it and interact with it.

This screen contains 2 tabs, one for the data which will be empty if the player didn't collect it and if he did he will have multiple chapter and concepts that he can read to help him with the quiz, the other tab is for the Quiz where he can play the quiz and get a score at the end.

```

Class TheLastGameDev{                               UButton* Quit;
    Data.txt      Quiz.cpp
    // Title

    UButton* Back;
}

```

Figure 59: "Empty Data tab"

```

Class TheLastGameDev{                               UButton* Quit;
  Data.txt      Quiz.cpp
  // GameDevData
  // Introduction :
  // Basic C++ Programming Concepts :
  // C++ Programming for Game Development :
  // Building the Game World :
  // User Interface and Interaction :
  UButton* Back;
}

```

Figure 60: "Collected data Data tab"

```

Class TheLastGameDev{                               UButton* Quit;
  Data.txt      Quiz.cpp
  // 1. What is Unreal Engine?
  // a. A programming language
  // b. A game engine
  // c. A software application
  // d. A hardware device
  UButton* Next;
}

```

Figure 61:" Quiz tab question"

```
Class TheLastGameDev{                               UIButton* Quit;
  Data.txt      Quiz.cpp
  // Result
  int32 Score = 9;
  UIButton* Replay;
}
```

Figure 62: "Quiz tab score"

IV. Conclusion

In this chapter, we offer more details regarding the game design document before moving on to gameplay mechanics, UI and the highlight item, the laptop and its functionality.

General Conclusion

In this project We were able to produce an educational game that will teach interested persons who wish to start game creation the fundamental concepts of Unreal and C++.

With this goal in mind, we were able to design a functional instructional computer game that assists players in beginning their path into game development. We managed to construct an exciting game that was both amusing and educational by using Unreal as a game engine and other Softwares.

With the time we were given to complete this project, we were able to implement the majority of our game ideas. While working on this project, however, many more ideas for improving the game surfaced, such as adding new gameplay aspects like multiple weapons and different enemies.

We discovered a new attitude while working on this project; building a game from scratch is very different from playing it.

Playing video games demands simply focusing on the game narrative, but while building it, we were mindful of the players' needs at every stage, particularly in terms of amusement.

Coordination with my lecturer and supervisor in ENVAST was critical to making this idea a reality.

Finally, our efforts were rewarded with such a fantastic game.

References

[1] official website of ENVAST, consultation date: 24 May 2023

<https://envast.tn/en/#Welcome>

[2] Steam store, consultation date: 20 May 2023

https://store.steampowered.com/app/812140/Assassins_Creed_Odyssey/

[3] Obsidian website, consultation date: 20 May 2023

<https://eternity.obsidian.net>

[4] Steam store, consultation date: 20 May 2023

https://store.steampowered.com/app/730/CounterStrike_Global_Offensive/

[5] Steam store, consultation date: 20 May 2023

https://store.steampowered.com/app/289130/ENDLESS_Legend/

[6] EA website, consultation date: 20 May 2023

<https://www.ea.com/games/the-sims/the-sims-4>

[7] EA website, consultation date: 20 May 2023

<https://www.ea.com/games/madden-nfl/news/hall-of-fame-edition>

[8] Blizzard website, consultation date: 20 May 2023

<https://worldofwarcraft.blizzard.com/en-us/>

[9] Steam store, consultation date: 20 May 2023

<https://store.steampowered.com/app/92800/SpaceChem/>

[10] Epic Games store, consultation date: 20 May 2023

<https://store.epicgames.com/en-US/p/while-true-learn>

[11] Steam store, consultation date: 20 May 2023

<https://store.steampowered.com/app/1226990/Mechanica/>

[12] Epic Games website, consultation date: 20 May 2023

<https://dev.epicgames.com/community/unreal-engine/learning>

[13] Unreal Engine marketplace, consultation date: 26 May 2023

<https://www.unrealengine.com/marketplace/en-US/store?sessionInvalidated=true>

[14] Unreal Engine website, consultation date: 26 May 2023

<https://docs.unrealengine.com/5.2/en-US/>

[15] Unreal Engine website, consultation date: 26 May 2023

<https://forums.unrealengine.com/categories?tag=unreal-engine>

Development of an educational game for Unreal Engine and C++ Game Development concepts

Abstract

The primary objective of this project is to create a PC video game that will enable players of various ages and backgrounds to understand and learn the fundamentals of game development while playing.

Résumé

L'objectif principal de ce projet est de créer un jeu vidéo sur ordinateur qui permettra à ses utilisateurs de différents âges et milieux d'évoluer et apprendre les principes fondamentaux de la programmation des jeux vidéo tout en jouant.

الملخص

الهدف الرئيسي من هذا المشروع هو إنشاء لعبة إلكترونية تسمح لمستخدميها من مختلف الأعمار والخلفيات بتعرف وتعلم أساسيات البرمجة الألعاب أثناء اللعب.