ingenieur
wissenschaften
htw saar

Hochschule für
Technik und Wirtschaft
des Saarlandes
University of
Applied Sciences

Bachelor-Thesis

zur Erlangung des akademischen Grades
Bachelor of Science (B. Sc.)
an der Hochschule für Technik und Wirtschaft des Saarlandes
im Studiengang Praktische Informatik
der Fakultät für Ingenieurwissenschaften

The impact of visual effects on player experience in a novel FPS game

vorgelegt von
Mouayad Haji Omar

betreut und begutachtet von
Prof.-Dr. Maximilian Altmeyer

Saarbrücken, 10.05.2024

Selbständigkeitserklärung

Ich versichere, dass ich die vorliegende Arbeit (bei einer Gruppenarbeit: den entsprechend gekennzeichneten Anteil der Arbeit) selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ich erkläre hiermit weiterhin, dass die vorgelegte Arbeit zuvor weder von mir noch von einer anderen Person an dieser oder einer anderen Hochschule eingereicht wurde.

Darüber hinaus ist mir bekannt, dass die Unrichtigkeit dieser Erklärung eine Benotung der Arbeit mit der Note "‘nicht ausreichend"’ zur Folge hat und einen Ausschluss von der Erbringung weiterer Prüfungsleistungen zur Folge haben kann.

Saarbrücken, 10.05.2024

Mouayad Haji Omar

Abstract

This bachelor thesis examines the influence of visual effects, including camera shake and hit feedback, on player experience in a newly developed first-person shooter (FPS) game, Fallen World. I hypothesized that the presence of visual effects in Fallen World would enhance player experience. To test this hypothesis, I created two game levels, the first without visual effects (*Standard*) and the second with visual effects (*Embellished*). After playing each level, participants completed the Player Experience Inventory (PXI) to measure their objective experiences, then completed an exit questionnaire to provide subjective feedback. Results indicated a significant difference in player experience only for the PXI component Challenge. These findings suggest that while visual effects only partially improve player experience across the PXI components, they can contribute to a more challenging and intriguing player experience.

Keywords: player experience, video game development, visual effects

We have seen that computer programming is an art,
because it applies accumulated knowledge to the world,
because it requires skill and ingenuity, and especially
because it produces objects of beauty.

— Donald E. Knuth [11]

Acknowledgements

First and foremost, I would like to sincerely thank Prof.-Dr. Altmeyer for his support during our time working together. I appreciate his guidance and willingness to assist me with my bachelor thesis in the field of game development.

To my parents and siblings, I am grateful for the love and encouragement you have given me throughout my life. I owe all of my accomplishments and success to you.

And to my wife, your love and patience do not go unnoticed. I am thankful for your support through thick and thin; I could not have achieved this milestone without you.

Contents

1 Introduction

Since the development of the very first video game over half of a century ago, video games have rapidly evolved from simplistic pastimes into a multi-billion dollar entertainment industry [14]. Today, video games are a primary form of entertainment worldwide, allowing players to not only become immersed in a fantasy world, but also socialize and compete with other players on a global scale. As a result of technological advancements, video games are now widely accessible and provide players with impressive graphics and immersive experiences that were once unimaginable. These advancements have pushed the focus from purely fun experiences to complex, deeply engaging ones that impact players on a personal level.

1.1 Motivation

In recent years, the field of video game development has placed a substantial focus on visual effects with the goal of improving the gaming environment to enhance player immersion [19]. These visual effects, such as particle effects, blood effects, and lighting, contribute to the creation of an engaging player experience.

The objective of my thesis is to investigate the impact of visual effects on player experience in a novel first-person shooter game. To achieve this, I will first develop a game from scratch using the popular game engine Unreal Engine. By doing so, I will be able to showcase my computer science skills and simultaneously improve my game development skills while gaining practical experience in game engine mechanics. I will examine the steps taken from the beginning of the development process to the final stage of optimization.

I will subsequently create a study based on the development of two FPS game levels. The first level will not include any visual effects, and the second level will incorporate visual embellishments, including camera shake, blood effects, and lighting enhancements. All participants in my study will test both levels and then complete questionnaires to evaluate the impact of each level on their player experience. Additionally, an exit questionnaire will be given to gain feedback into their subjective gaming experience.

The analysis of participant feedback will be used to evaluate the influence of visual effects on player experience. I hypothesize that the addition of visual effects will significantly enhance player experience.

1.2 Research question

In this thesis, I investigate the following research question:

> How do visual effects impact player experience in an FPS game?

To address this question, I created a playable first-person shooter FPS game from scratch, applying my knowledge of computer science and game development to this video game.

2 Approach

2.1 Development process

Video game development is a complex process that begins with the game's conceptualization and ends with the game's release. This process transforms the game from a simple idea to a fully functional, playable game that can be sold and distributed to a wide audience. It is a multi-step process that includes brainstorming and planning, design, implementation, testing, and optimization [2] [15]. Even after the game's release, developers work tirelessly to fix bugs and support game updates. Throughout this process, developers collaborate with programmers, artists, and game testers because bringing the game to life requires a huge team effort. Understanding this complicated process is essential for anyone involved in the creation of video games.

2.2 Project plan

A well-developed project plan is important to begin the video game development process. This plan ensures the project stays organized and progresses the way it should in a timely manner, keeping the game's development on track. It includes selecting a game engine, outlining the game's concept, and budgeting game assets such as characters and objects, animations, sound effects, and visual effects (VFX) [2] [15]. The initial phase of planning ends with the development of the game's storyline and mechanics, which are recorded in the game design document (GDD).

After the planning phase, the project transitions to the pre-production stage, in which the game prototype is developed. The prototype is an early version of the game created for playtesting and obtaining feedback to maintain a practical game concept.

During the next stage, implementation, the game begins to take shape. Enhancements are made to the game prototype by incorporating assets, either self-created or obtained from sources such as Unreal Marketplace, CGTrader, and itch.io. This phase focuses on improving game systems and functionalities. Once the game mechanics are finalized, the creation of levels and maps begins, along with the incorporation of all 3D elements and characters.

User interface is the subsequent step of implementation, which includes the development of the game's main menus and player heads-up display (HUD). After the implementation process is complete, the project progresses to the testing and debugging stage to minimize bugs and optimize the gameplay.

As the game nears completion, it undergoes the final packaging and exporting processes. This progression from initial planning to finalization ensures that the game is fully prepared for its release and ready to engage players.

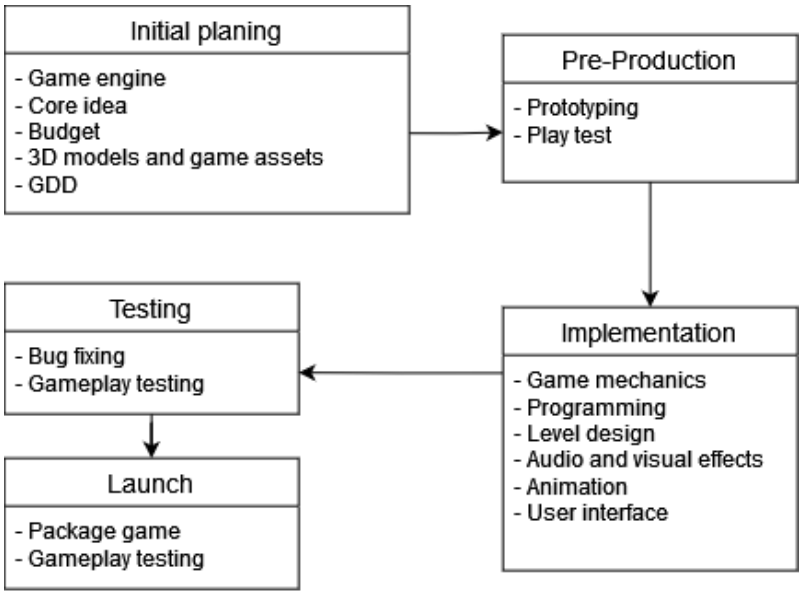

Figure 2.1: Project plan from beginning to end

2.3 Game engine

This first-person shooter game will be developed using Unreal Engine, with which I have nearly three years of experience. Unreal Engine (UE), developed by the video game company Epic Games, is known for its advanced 3D computer graphics [3]. It was first introduced in the 1998 game Unreal and although originally designed for PC first-person shooters, the engine has expanded its use to additional video game genres and is even used in completely different entertainment industries, such as film and television.

Unreal Engine is a C++ based game engine that is extremely portable and supports a wide variety of platforms, ranging from desktop and mobile to console and virtual reality (VR) [17]. The latest version of the engine, Unreal Engine 5, was released in April 2022. Its source code is available on GitHub and can be used commercially with royalties. Epic Games charges a 5% royalty on game revenues exceeding one million United States dollars; however, this fee is waived for titles published on the Epic Games Store, which is Epic Games' digital game distribution service.

UE has advanced as the result of new features acquired by Epic Games, such as the 3D world creation tool, Quixel. Its development has been further supported by the success of Epic Games' battle royale game, Fortnite. Because of these cutting-edge features and my preexisting familiy with the engine, I have chosen UE for the development of my FPS game.

3 Game Design Document (GDD)

The Game Design Document GDD is essentially a blueprint for beginning the video game development journey.

This document introduces the game's basic concept, mechanics, characters, levels, and all other details relevant to the game's implementation [4]. It guides developers, artists, and designers in the intended direction of the game, ensuring that the developing game looks and feels cohesive.

In addition to outlining the game's core components, the GDD typically includes details such as the target audience and platform, game genre, monetization strategies (when applicable), technical requirements, and selling points that distinguish the game from the others on the market [15]. Furthermore, it can include the basic storyline or plots, narrative themes, and world-building elements that contribute to an immersive experience.

As the game's development progresses, the GDD evolves and is frequently modified to reflect new ideas and challenges. It encourages collaboration and consistency between everyone involved in the game's creation. In this section, I explore how I used my GDD to implement a video game and discuss the iterator, the process, challenges, and strategies used to transform my conceptual ideas into a complete game experience.

3.1 Game concept

This section specifies a design for the gameplay of the game titled "Fallen World." This is intended to be read by programmers, artists, and producers. The aim of the game Fallen World is for the player to eliminate all enemies in order to finish the game.

- **Genre:** First-person shooter.
- **Target Audience:** Players of various ages and backgrounds.
- **Target Systems:** PC, Laptops, Windows.
- **Game Graphics and Style:** 3D high/low poly.

3.2 Story

Fallen World is an engaging 3D action game available for Windows users, developed using the latest Unreal Engine 5.2. Set in a post-apocalyptic world, the game immerses players in a desolate environment overrun by zombies. Featuring intuitive controls, players embark on an adventure through compact levels on their quest to survive. Along the way, players can collect weapons and other forms of aid to fend off the zombies. The main objective is to navigate through the hazardous areas, eliminate enemies, and clear zones to find safe havens. As players progress, they build up an arsenal to defend themselves against the relentless zombie horde, ensuring their survival in this fallen world. Each cleared area marks a victory, allowing players to advance to the next stage in their quest for safety.

3.3 Characters and items

- **Characters**
  In Fallen World, a single player character is central to the game's storyline. This character, a former soldier, roams the country in his car, dedicating his life to eliminating zombies and making the land safe for other humans to rebuild their lives. All other characters encountered within the game are enemy AI, each playing a role in this post-apocalyptic environment.

- **Items**
  In the game, players can collect various items scattered throughout the game world, including ammunition, health pickups, and a diverse range of weapons. Each item plays a unique role and is designed to assist the player during their journey.

3.4 Actions

When designing Fallen World, the game's foundational elements must first be defined. This involves deciding which elements are absolutely necessary and which ones could be included to enrich the gaming experience. This approach ensures a clear understanding of the capabilities and actions available to players, enemies, levels, and other game components within the game world. This section explores the core mechanics and systems of Fallen World, showing the wide range of actions available in the game's unique setup. The game is designed to offer a rich and immersive environment that includes complex opportunities for combat and exploration. This section aims to explain the key features of the game and demonstrate how they combine to create an interactive and captivating world that enhances the player's experience.

3.4.1 Movement systems

The game allows players to intuitively control the main character using a key action binding system. The main input keys are 'W' and 'S' for backward and forward movement, 'D' and 'A' for left and right movement, 'Space' for jumping, 'Left Shift' for sprinting, the left mouse button for attacking, the right mouse button for aiming, and 'Ctrl' for crouching. The 'F' key allows the player to interact with the game world by opening doors or picking up items, while specific keys enable item usage.

3.4.2 Health and death systems

Health and death systems are essential features in every video game. In Fallen World, the player starts their adventure with a predetermined amount of health. If the player's health is fully depleted, the game over screen will be shown.

3.4.3 Projectiles

In Fallen World, a first-person shooter (FPS) game, the core gameplay centers around the shooting mechanics, which are essential for engaging combat and player immersion. Players can fire bullets, acting as projectiles, to eliminate enemies within the game's environment, which creates an interactive experience. This allows players to strategically navigate through levels, confront enemies, and progress through the game's storyline. To further enhance the projectile feature, the realism of the shooting mechanics is improved by incorporating recoil effects, bullet drop, and varying weapon behaviors. These improvements contribute to the player's tactical decision-making and overall sense of immersion.

3.4.4 Damage systems

- **Player damage**
  The player character can inflict damage on enemies in various ways, including melee attacks (fist damage), weaponry (weapon damage), and explosives (grenades and incendiaries). These abilities are built into the player character's system, enabling the players to effectively defend themselves against enemies.

- **Enemy damage**
  The game features a variety of zombie enemies, each with unique damage capabilities. For example, super zombies cause twice as much damage as normal zombies, signaling to the players that the areas in which these zombies are located are more risky. This prompts players to be more careful and encourages them to strategically prepare for combat or even avoid these areas altogether. Zombies mainly attack using melee (fist damage), posing the greatest threat when they come into close proximity of the player character.

3.4.5 Patrol systems

In the game world of Fallen World, all enemies are designed to patrol through designated areas determined by navigation mesh boundary volumes. These volumes equip them with the ability to move around and avoid certain objects within the game world, such as trees, walls, rocks, and other static meshes. This ensures that they can walk and run smoothly throughout the game environment without getting stuck or falling over obstacles.

3.4.6 Chasing and hearing

In the game, all enemies possess advanced perception abilities, allowing them to both see and hear the player character. This system is designed to enhance the intelligence of in-game opponents, creating a more immersive and challenging experience for players. With this system in place, the game generates a realistic environment in which enemies dynamically respond to visual and auditory prompts from players. This system is implemented using Unreal Engine's AI and perception tools to produce complex AI behaviors that can detect the player character's movements and actions. As a result, stealth becomes a crucial element of gameplay, encouraging players to think strategically and use the environment to influence enemy behavior. This system enhances the gaming experience by providing players with realistic interactions in the game world, creating unique challenges with each enemy encounter.

4 Implementation

After collecting the necessary assets and data, the implementation phase of video game development can begin. In this phase, the initial concepts and designs from earlier phases start to emerge. This process involves activities ranging from character programming and user interface (UI) design to extensive testing and debugging [5]. It is in this phase that all the creative concepts and plans transform into a playable game. Implementation is not merely about combining these components; rather, it involves refining the game mechanics, enhancing visuals, and ensuring all game elements are seamlessly integrated to create a cohesive and engaging gaming experience. This phase is important for turning the initial game concept into a fully functional game that aligns with the expectations of the development blueprint.

4.1 Game system architecture

This section provides an overview of the essential systems developed to create my FPS game using Unreal Engine. This architecture defines the elements of the game, starting with the player character, enemies, game levels, user interface, graphics, and game mode, and ending with testing and debugging each aspect of the game and focus on the optimization process for good gameplay experience.

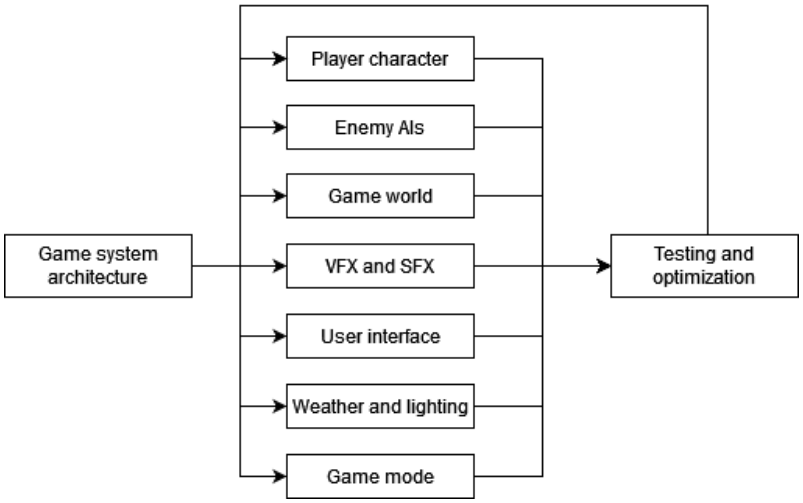

Figure 4.1: Game system architecture

4.2 Player character

The process of developing a player character starts with importing essential assets, including the character model (skeletal meshes) and animations. The next step involves creating a blueprint class for the character that outlines the character's inputs, behaviors, and functionalities. Once the blueprint class is created, the player character camera and spring arm component must be combined. These components work together to replicate the player's viewpoint, effectively creating vision within the game world. Following the setup of the player camera and spring arm, the player character's functionalities are further developed by configuring movement, navigation, and shooting mechanisms.

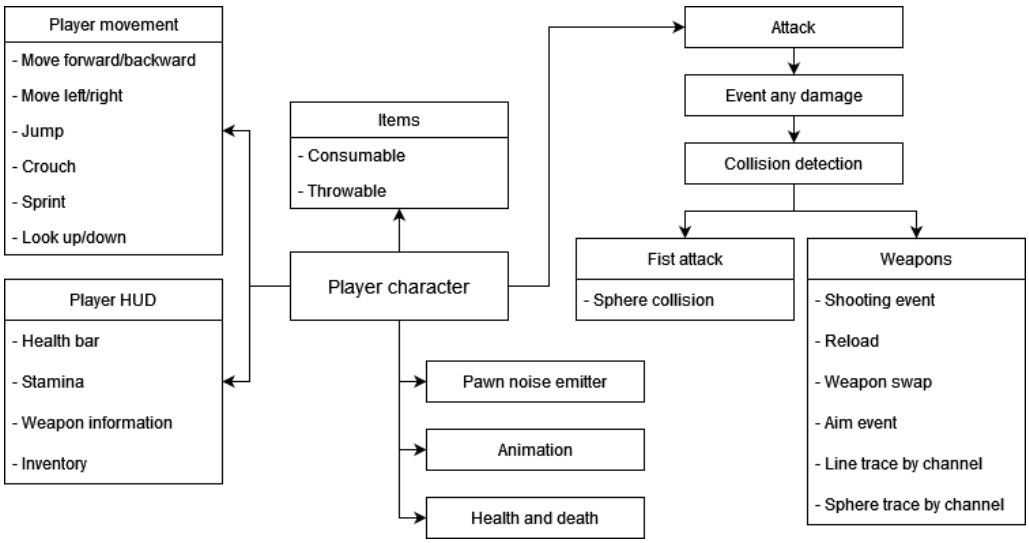

Figure 4.2: Player character functions

4.2.1 Player movement

Unreal Engine 5 has an innovative functionality called "enhanced input." This system allows developers to implement a powerful system for reading and interpreting user inputs. After creating the input actions and input mapping context, the implementation of the player character movement systems can begin. This includes move forward and backward, left and right, look up and look down, and all other inputs. To implement the movement system for the player character, the player location must be updated on the dimensions pitch, yaw, and roll [16].

Key	Input Interpretation	Modifiers
W	Positive scale value on Y-Axis = 1	Swizzle Input Axis Values (YXZ or ZXY)
S	Negative scale value on Y-Axis = -1	Negate and Swizzle Input Axis Values (YXZ or ZXY)
D	Positive scale value on X-Axis = 1	None
A	Negative scale value on X-Axis = -1	Negate

Table 4.1: Player character movement inputs

After defining the input scales for player movement, this can now be applied by using the function in blueprint event graph "Add movement input" along the given world direction vector (usually normalized) scaled by 'Scale Value'. If Scale Value < 0, movement will be in the opposite direction. So if the player presses the 'D' button, the player speed will increase by a positive value on the X-Axis and the player will move to the right. It has the opposite effect for 'A.' If the player presses 'A,' then they move along the X-Axis by a negative value, so the player will move left. The same works for looking up and down on mouse inputs. These are the core mechanics to implement the player character movement in the 3D world.

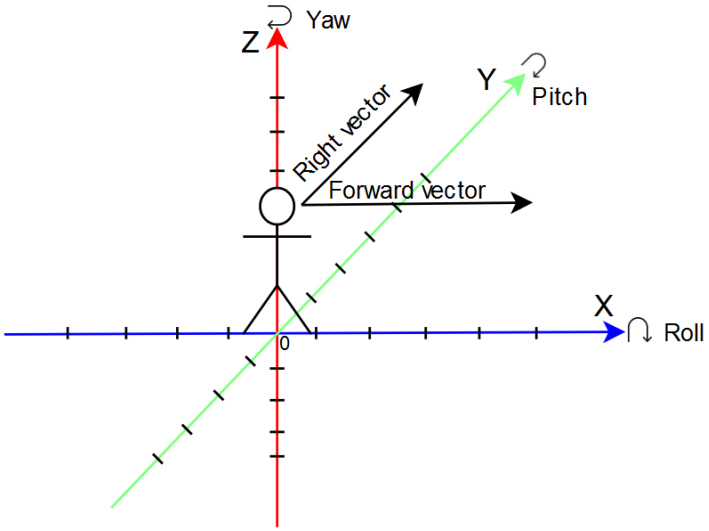

Figure 4.3: Player character movement in 3D space

- **Jump function**
  The jumping function is straightforward. If the player character's velocity on the Z-axis is greater than zero, then the player is in the air and falling. By pressing the space bar button, the player is able to jump.

- **Crouch function**
  The crouch function in the player character movement details panel inside Unreal Engine must first be enabled. After that, the scale of the player character capsule component should be set to half the height of the default scale of that capsule component. Then, the player will be able to crouch whenever they press the 'Ctrl' button.

- **Sprint function**
  This is a simple function that increases the player character's speed whenever the player presses the left 'Shift' button by setting a higher value for walk speed.

4.2.2 Player attack

First, the key inputs for shooting the weapons are defined. The player can then use different type of weapons and others such as grenades and flares. In the game, each weapon has its own skeletal mesh and animation. To implement the shooting mechanic, all sockets must be set to each weapon skeletal mesh. These sockets are the starting point of shooting the bullets from that weapon. Once set for each weapon, a parent blueprint class is created for the weapons, so that all children of that class will inherit the functionalities and variables from that parent class.

- **Fist attack**
  A simple function to apply damage to enemies by adding a sphere collision on the player hands. The collisions will detect the enemy capsule component (collision) and apply a specific amount of damage to them.

- **Weapons**
  First, a socket must be added to each weapon skeletal mesh. The sockets should be placed at the tip of each weapon so they perform as the starting point of shooting the bullets. This shooting method is one of the most common techniques used in FPS games. After adding sockets, the built-in function in Unreal Engine called line trace by channel can be used in the player event graph which is called line trace by channel. This function performs a collision trace along a given line and returns the first object that the trace hits in order to detect enemies. The line trace by channel has a starting point and ending point. The sockets are obtained from the weapons and attached as the starting point for shooting bullets. For the ending point, the player character forward vector is acquired then multiplied by an integer value which will be the length of that line trace. If the line trace overlaps an enemy collision, then it will apply a certain amount of damage on them. If they do not overlap, then the line trace will either be destroyed or will spawn a decal if it hits a static mesh such as wall, ground, or glass. Another method used to detect enemies and apply damage on them when throwing grenades is the sphere trace for objects, which sweeps a sphere along the given line and returns the first blocking hit encountered. This trace finds the objects that responds to the given trace channel.

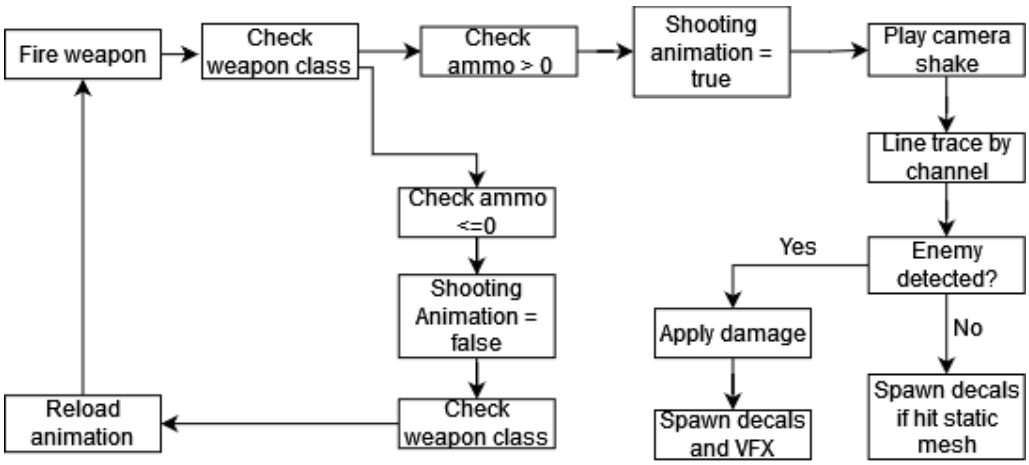

Figure 4.4: Player character shooting mechanic

- **Weapon swap**
  First, a data structure containing all weapon types must be created. Then, a function is defined to check if the weapon is equipped or not. If it is equipped, pressing specific inputs will perform the weapon swap.

- **Aim event**
  For each weapon, a camera is added on the iron sight. The new view target is set to the player by changing the camera from the player camera to the aiming camera with blend when pressing the right mouse button.

- **Camera shake**
  Camera shake refers to camera movement that adds emphasis to a player action or reflects an important game event, which contributes feel to the gameplay, intensifying actions and increasing immersion. The camera shake is an important feature of my study, as it is a widely used visual effect that most modern games implement in their gameplay. The camera shake is a vibration or movement added to the player camera to the pitch, yaw and roll axis of the player camera. In the game Fallen World, camera shake has been implemented in multiple events and actions. For example, when the player character is hit by the enemy, the camera will move in certain directions. When the player character shoots a gun, the camera will also move to give the player the feeling of immersion and realism. Creating camera shake in Unreal Engine is a very simple process. The camera shake base blueprint class of Unreal Engine must be found, then the camera shake effects can be created. In the blueprint editor of camera shake, the settings such as amplitude (which controls the size of the shake pattern) and frequency (which controls the speed of the shake in the X,Y,Z axis) must be adjusted. To implement this in shooting event, the blueprint class is called at the same time as the shooting animation. This gives the player a shaking and vibration effect upon firing the weapon.

4.2.3 Health and death

Those two functions are connected to the enemy AI. The player loses a predetermined amount of health whenever the enemies gets closer and attacks him. Upon losing all health, the game ends and the game over menu displays.

4.2.4 Pawn noise emitter

This function allow the player character to be heard by the enemy. The owning pawn emits a collection of noises that are sensed by the enemy AI. The volume and location of the noises can be altered.

4.2.5 Items

- **Consumable**
  These actors can be found in the game world and picked up by the player character. This includes health kits, which allow the player character to increase their health after taking damage. To create a health pick up item, a blueprint actor class is created and all actor components, such as the static mesh of the health kit, must be defined. In order to to allow the player to interact with these items, this actor class must be casted to the player character, allowing all functions and events of the player character to go inside the item actor class. Then a sphere collision is added to the static mesh of each actor and the function "On component begin overlap" can be utilized. This function is casted to the player character so whenever the player enters the radius of the sphere collision, all widgets can be seen, such as press "F" to pick up item.

- **Throwable**
  Throwable items use the same techinique as the consumble items and include grenades, flares, and flame grenades. An actor class with static mesh can be detected and interact with the player character by using various type of collisions.

4.2.6 Animation

First, a blueprint class is created for the player character and assigned to the player character skeleton mesh, and the player character reference is acquired in the blueprint animation class by casting to the player character in order to expose all player functions and variables to our player character animation blueprint. Next, a state machine is created that holds all the transitions between the player animation states, such as idle, walk, run, jump, and more. One of the best ways to animate a first-person character is to use blend space 1D (direction). This plays animation in one direction for the player movement whenever the speed of the player changes by getting a player reference and checking the velocity of the player. An example of the transition between idle and move forward is as follows: The player reference is acquired, and from that reference the player velocity is obtained. If the velocity is greater than zero, the player is moving forward, but if the movement button is no longer pressed, the player velocity is equal to zero, entering the idle state.

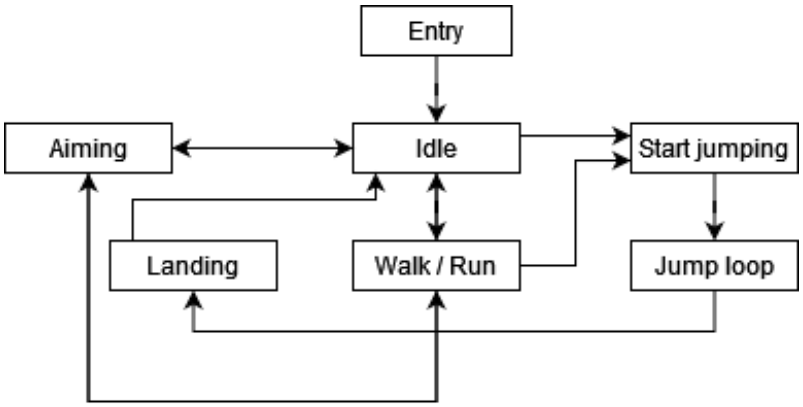

Figure 4.5: Player character animation states transition

For the shooting, death, damage animation, and all internal animations, animation montage classes and animation BP were created for each weapon. The player can play an animation by checking the equipped weapon. Then the blueprint class for the player is fired in order to play the shooting and reloading animations of that weapon.

4.3 Enemy AI

This section discusses the most important functions of the enemies, such as movement, pawn sensing, attacking, and more. First a parent class for all enemies is generated, allowing me to minimize the amount of blueprints running simultaneously for optimization purposes. Then a child blueprint class is created for all enemies, so that they inherit all functions and variables from that parent class.

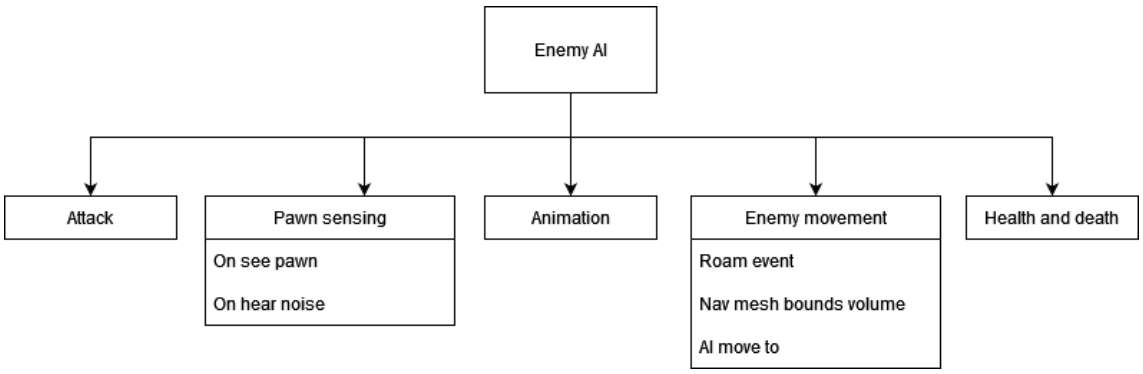

Figure 4.6: Enemies' most important functions

- **Enemy movement**
  To create the enemy movement system, a custom event (function) must first be created in the parent class that will perform the enemy movement. This function begins by checking if the enemy can see the player or if they are dead. If the enemy can see the player, the player character location is obtained and the function "AI Move To" is fired. This function allows the enemies to locate the player character if they can see him or hear noise from his location (This is further explained in Pawn Sensing.). If the enemy is dead before firing this function, then we simply stop the function from excuting so that the enemy's movement is prevented. In order to move in a certain area or location, the function "Get Random Reachable Point In Radius" is used. The radius of movement for each zombie is defined and then applied to the function "AI Move To." For this particular function, "Nav mesh bounds volume" must also be used. This is a special navigation volume which defines the areas in the game world in which navigation meshes are generated. By placing those volumes in the game world, the areas in which enemy AI can move is defined[16].

- **Pawn sensing**
  Pawn sensing is an Unreal Engine system used for AI perception and behavior. It allows enemies to sense and react to different elements of the game world, which includes detecting other characters and actors. This system typically includes components such as hearing, vision, and other inputs. It allows the enemies to make decisions based on what they perceive in their surroundings.

- **On see pawn**
  This is a function that allows the enemies to move to the player location when the player appears in their vision radius. This radius works as collision detection for many types of actors.

- **On hear noise**
  After applying the pawn noise emitter to the player character, the enemies can hear the player character within a specified radius. For example, when the player character shoots a bullet inside the radius of their hearing collision, the enemy moves to the player character faster. By setting a new walking speed for the enemies, they can get closer to the player character and apply damage.

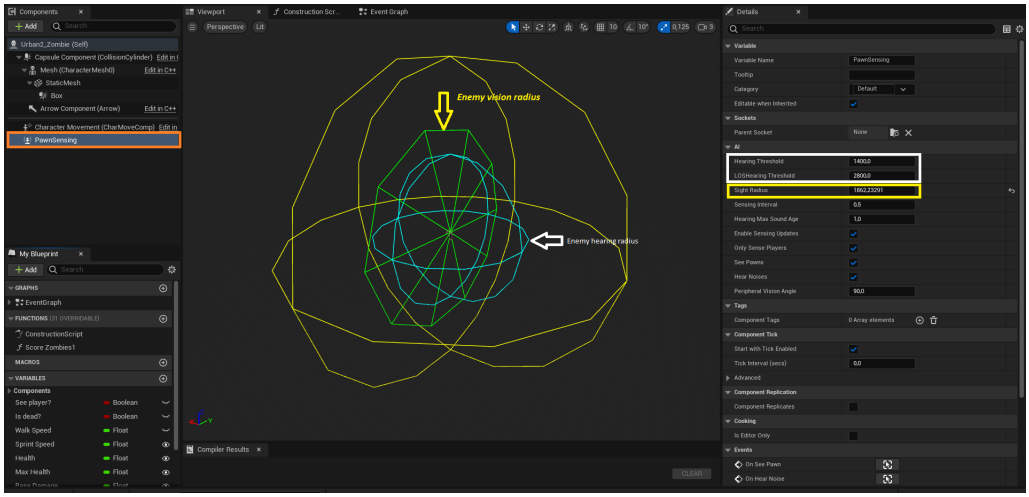

Figure 4.7: Enemy pawn sensing for vision and hear events

- **Attack**
  The enemy attack function is similar to the player character fist attack function. When this collision overlaps the player character capsule component, the player receives a certain amount of damage.

- **Animation**
  Each zombie has a special animation for walking, attacking, and taking damage. In this case, it is difficult to make a parent animation blueprint class for all zombie animations, so the best way to create animations is to use animation montages. These montages are tools inside the engine that combine and selectively play animation that contains a single asset. After creating animation montages for the zombies, they can be referenced in the enemy blueprint graph and applied when necessary in the game.

- **Health and death**
  Each enemy has a predetermined amount of health. When the enemy is damaged, this works with the event any damage which checks the current amount of health of the enemy and subtracts the given amount of damage from it. Once the enemy loses all his health, a death animation montage plays, and after 60 seconds the dead actor is destroyed in order to reduce the amount of actor meshes in the game world for optimization purposes.

4.4 Game world

4.4.1 Levels

The game consists of two levels. Each level has its own setting, but they share the same landscape size, materials, and data.

4.4.2 Landscape

To create a landscape for my game, the mode must first be switched to landscape mode in the engine editor. Then the detail panel of the landscape editor will be shown to the left side of the editor. This panel defines the section size, section per component, number of components, and overall resolution. To create my own landscape, I changed their values and then used the sculpting tools to create hills and mountains, constructing a more detailed environment.

4.4.3 Materials

To make the landscape more colorful and realistic, a landscape material must be created. I began by selecting the texture and materials from the Quixel bridge library, which are free for Unreal Engine users. After importing the necessary materials and textures, I created a blueprint function for each material to use it in the main landscape material function.

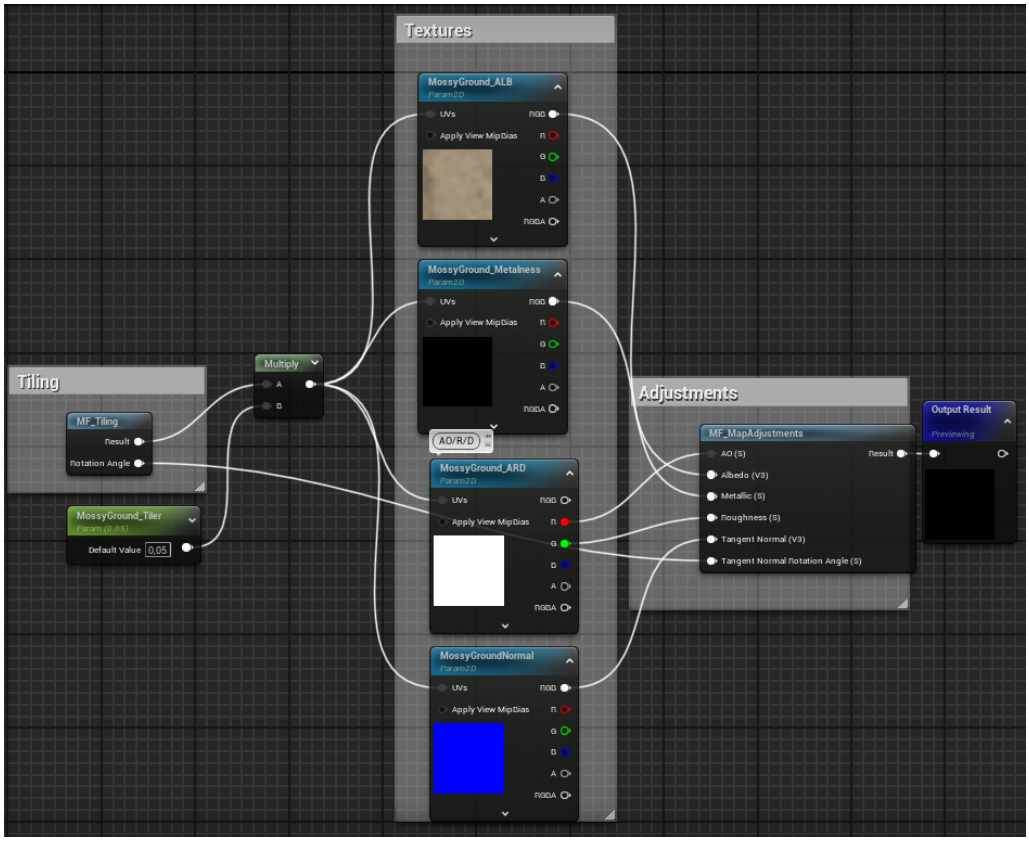

Figure 4.8: A material function with textures for landscape material

This method of creating material functions minimizes the "base pass vertex shader instruction," which is very important for optimization purposes. After creating all the material functions, they are called in the main landscape function, and each layer of the landscape material is defined. Then all important tiling sizes and other colors are set up. Landscape material instances are created and weightblended for each material so they are prepared to be used in the main landscape material in order to begin painting the landscape with more details.

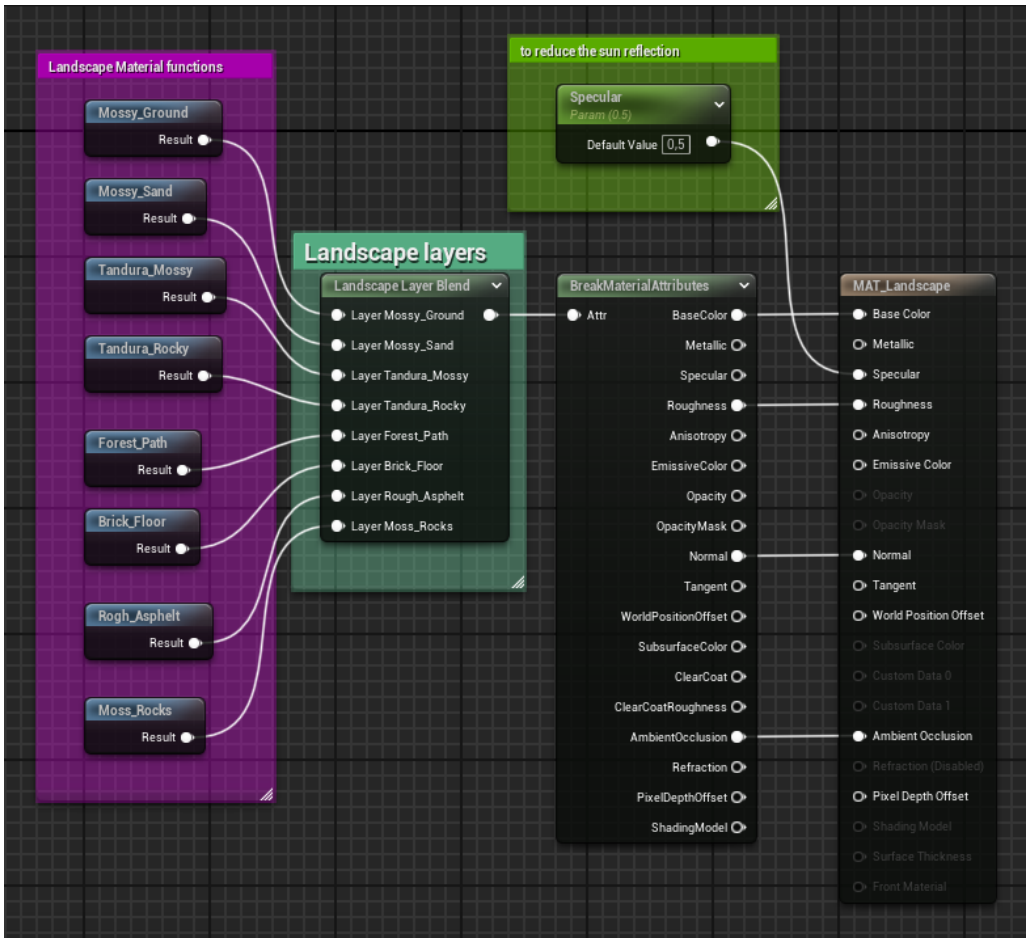

Figure 4.9: Landscape material overview

4.4.4 Objects

After painting and sculpting the landscape, each item (house, tree, wall, fence, etc.) is placed in the landscape by simply dragging and dropping it in its correct place. There should be no clipping or overlap between each actor and object in the game for optimization purposes.

4.5 VFX and SFX

4.5.1 Visual effects (VFX)

In films and video games, visual effects are the tools that create an immersive environment for the audience or players. These effects range from particle effects (such as fire, wind, fog, blood, and more) to more complex systems of visual effects (such as dynamic weather, water effects, and realistic lighting). When creating Fallen World, I implemented the aforementioned particle effects, the most notable of which was blood effects. These appear as a blood screen when an enemy damages the player character and serve to create a more realistic and immersive gaming experience. In order to implement such effects, the assets of those effects must first be added to the game by importing the images to the engine. Then an interactive blueprint widget must be created to manifest UI elements. After adding the necessary UI elements, the blood screen effects were easily animated by using fade in and fade out on taking damage. Then, by casting this widget to the player character, it can be used in the player "event any damage" function by checking if the player is still alive and can be damaged before firing this widget to the viewport.

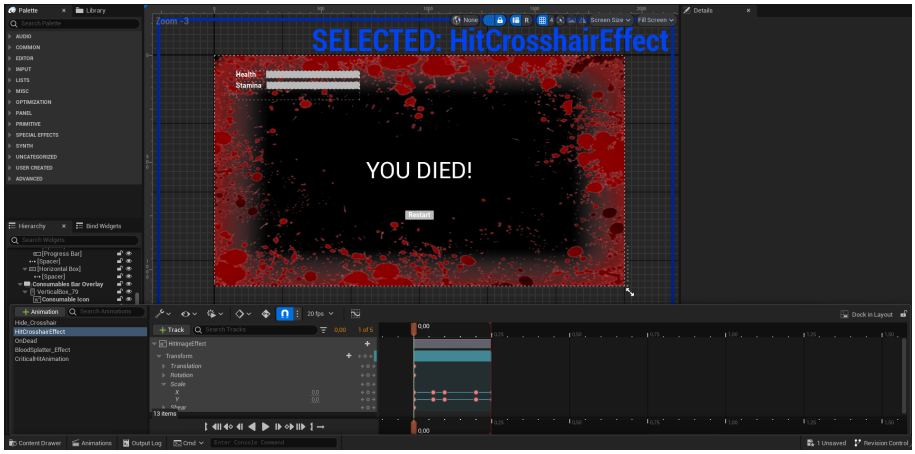

Figure 4.10: Player blood effect widget on taking damage and death

Figure 4.11: Enemy blood hit reaction

4.5.2 Sound effects (SFX)

Sound effects play a critical role in video games as they work to provide the player with an immersive game experience. They can enhance the game's atmosphere and create a realistic player experience. In Fallen World, sound effects ranged from player sound effects to background music and UI SFX. To apply these sound effects to each aspect of the game, I utilized the built-in functions "play sound 2D" and "play sound at location," which are the most commonly used functions in Unreal Engine. However, Unreal Engine 5 recently introduced a new system called metasounds, which is a high-performance system that provides players with an improved audio and sound effects experience. I applied metasounds to the game's actors and functions with the goal of enhancing player experience.

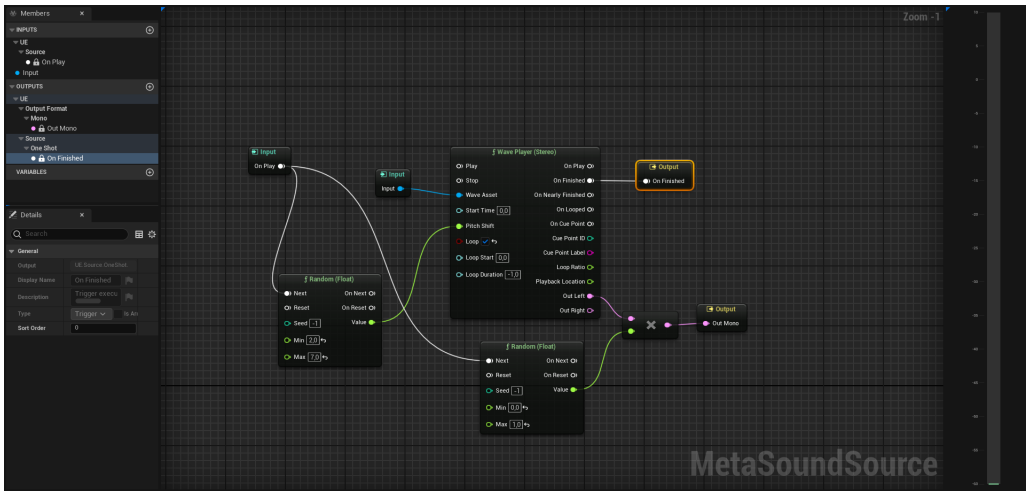

Figure 4.12: Metasounds editor for one of the actors SFX

4.6 User interface (UI)

4.6.1 Main menu

This is the game's main menu with an animated background and the following buttons:

- **Play button:** This opens another user widget that contains two buttons: Tutorial and Test. The Tutorial button opens the game map (Level) where the player plays a tutorial of the game. The Test button opens a new user widget that contains two buttons with the name Test 1 and Test 2. Each of these opens a level where the player can play the game.
- **Option button:** This button contains three user widgets. The Display widget allows the player to configure the game resolution settings, the Audio widget configures the audio settings, and the Key Bind widget modifies the gameplay inputs
- **Credit button:** This display credits the game's contributors, including developers, artists, sound designers, and asset creators.
- **Quit button:** This allows the player to exit the game.

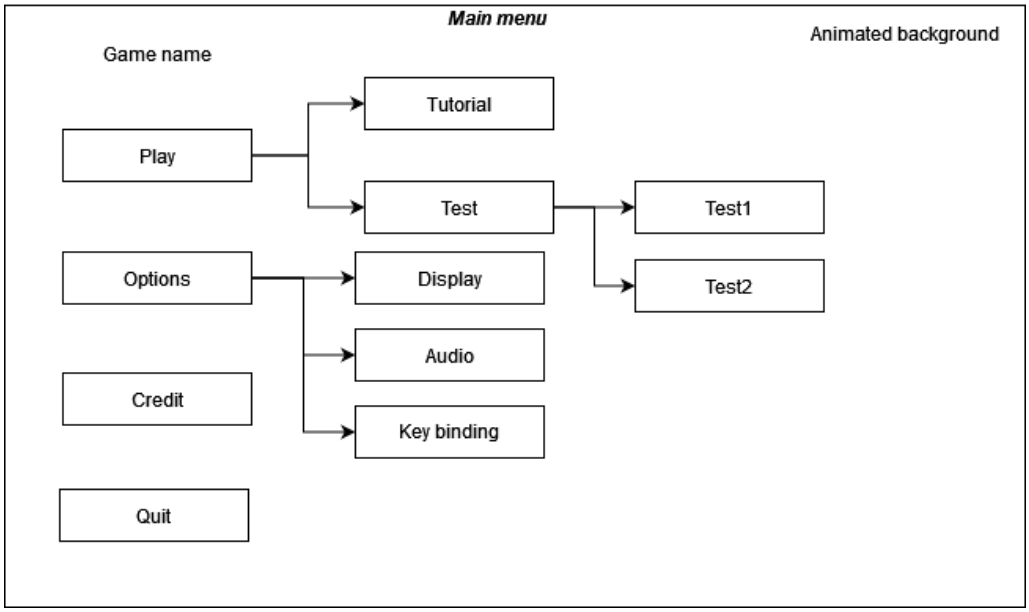

Figure 4.13: Game main menu

4.6.2 Mission widget

The mission completed widget displays when the player finishes the objective before he dies or time runs out. The mission failed widget displays when the objective is not completed.

Figure 4.14: Mission completed after the player cleared the area

Figure 4.15: Mission failed after the player lost all his health

4.6.3 Pause widget

This is a user widget that allows the player to pause at any point in the game to check or change the key inputs, change the game settings, or quit the game.

4.6.4 Player (HUD)

This is the player heads-up display (HUD) widget that contains progress bars to calculate the amount of health and stamina for the player, and these progress bars change with each event or action that occurs during gameplay. It also has a weapon information vertical box that displays the amount of bullets the player has and the gun currently equipped by the player.

Figure 4.16: Player heads-up display

4.6.5 Player inventory widget

This is an inventory widget system that allows the player to choose weapons and other items and store them in the backpack section. The player can only use two weapons and must swap between the primary and secondary weapon. Additionally, the player can store multiple items and use them by dragging and dropping them inside the inventory consumable and throwable sections.

Figure 4.17: Player inventory widget

4.7 Weather and lighting

Because Fallen World is set in a post-apocalyptic environment, the lighting and weather settings of the gameplay must be adjusted to provide the player with a more realistic scene for a more immersive and overall better gaming experience. For the first map (level Test 1), the lighting system was kept as low as possible. In the second map (level Test 2), the game's lighting was improved using the post-processing effects. In Unreal Engine, post-processing volumes can be added to the level as the only way to manipulate post-processing parameters. The most important effects that these volumes can improve in the game include anti-aliasing, auto exposure (eye adaption), bloom, color grading, depth of field, lens flare, vignette, and screen space reflection. By simply adjustning these effects' settings in the post-processing volumes detail panel, each one of these effects can improve the scenes' appearances and create a more immersive gaming environment.

4.8 Game mode

The game mode refers to the game story, task, or elements of the game. I created one identical objective for the player in the two test levels. The main objective is to clean the area of enemies before time runs out by implementing this in each game blueprint. First, the event graph of the game is opened and all actors with a tag (zombies) are acquired. Next, they are casted to the game mode that calculates the zombies in the game world and reduces the number of enemies each time the player kills an enemy. Once the player kills all enemies, the mission complete user widget is displayed in the viewport.

4.9 Testing and optimization

4.9.1 Testing and debugging

Testing and debugging the functions and events of a game are routine parts of the game development process. Each time I created a function or event, I tested it in the editor to ensure smooth gameplay and minimize potential bugs. There are various methods for testing and debugging, each essential for checking different aspects of the game [16].

- **Blueprint debugging**
  One of the simplest yet most effective methods I used was printing a string within a function or event. This prints a message either in the output log or directly on the screen within the editor. It's an invaluable tool for tracking variable values and confirming the execution of specific actions within the blueprint event graph.

- **Breakpoints**
  I also frequently used breakpoints by adding them to blueprint nodes. This pauses execution within the editor, allowing me to inspect variables in real-time. This approach is incredibly helpful for understanding the logic flow and pinpointing potential issues.

- **Other methods**
  Beyond these, there are numerous other testing methods I utilized. For example, console commands monitor frame rate, GPU, and CPU times. Unreal Engine also offers built-in debugging functions, such as drawing debug spheres, points, and lines. I found these particularly useful for testing mechanics such as bullet trajectories, radial damage, and pathfinding. Each of these methods plays a crucial role in ensuring every game component functions correctly, ultimately leading to a smoother and bug-free gaming experience.

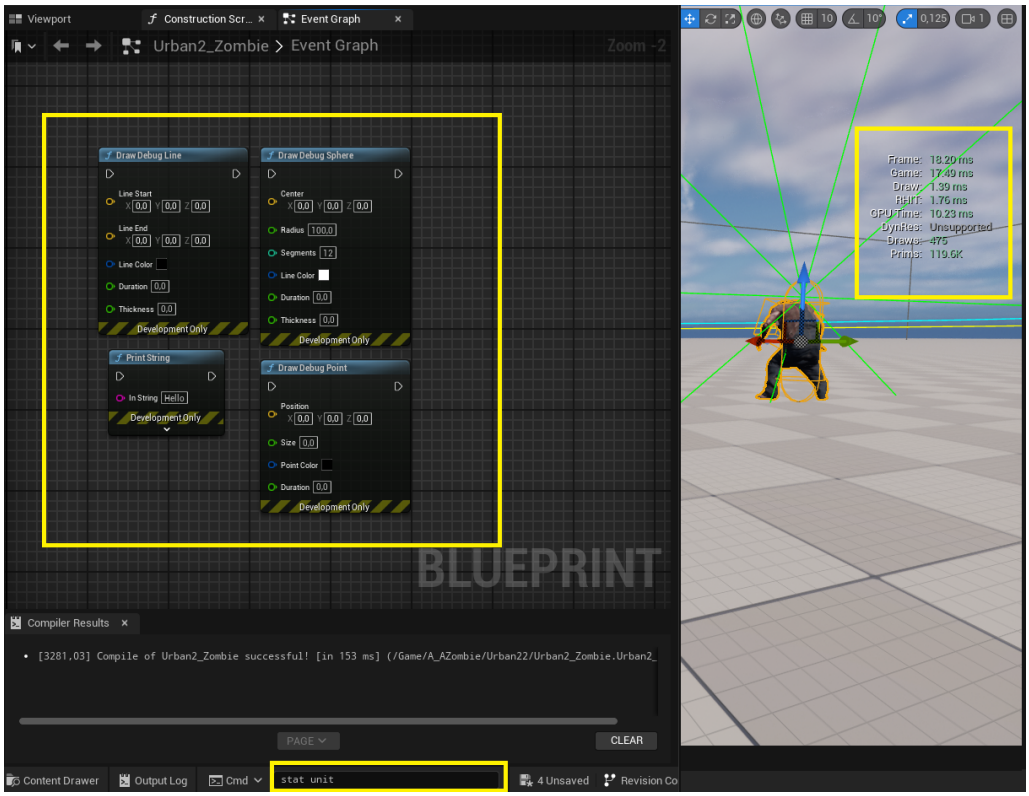

Figure 4.18: Debugging and testing blueprint functions

4.9.2 Optimization

In video game development, optimization is vital to enhance game performance and increase frame rates, load time, and make the game run smoothly across hardware specifications [8]. Multiple methods of optimization were employed in Fallen World to improve the game frame rate. For example, LODs were applied to models and meshes to reduce the count of objects' polygons when the player is far away from an object. By getting closer to that object, the polygon count increases again to provide the player with more detailed object textures. Another method of optimization is to check the material textures and reduce the amount of texture samples in them. In addition to this, lighting can drop the frame rate per second (fps) on a different setting. For example, if the game's sunlight is moveable, then the game fps will dramatically drop, causing lags and bad performance. In this case, the sunlight should be set to static and baked for all geometry to save on real time computation. In general, using static light significantly increases the performance. Event ticks can also drop the fps, so the amount of these running in the game must be minimized to only fire when completely necessary. Another optimization method is culling, which renders the objects in the player field of view within specific distances. By using new technology, such as the Nanite system in Unreal Engine, complex scenes can be rendered without compromising game performance [16].

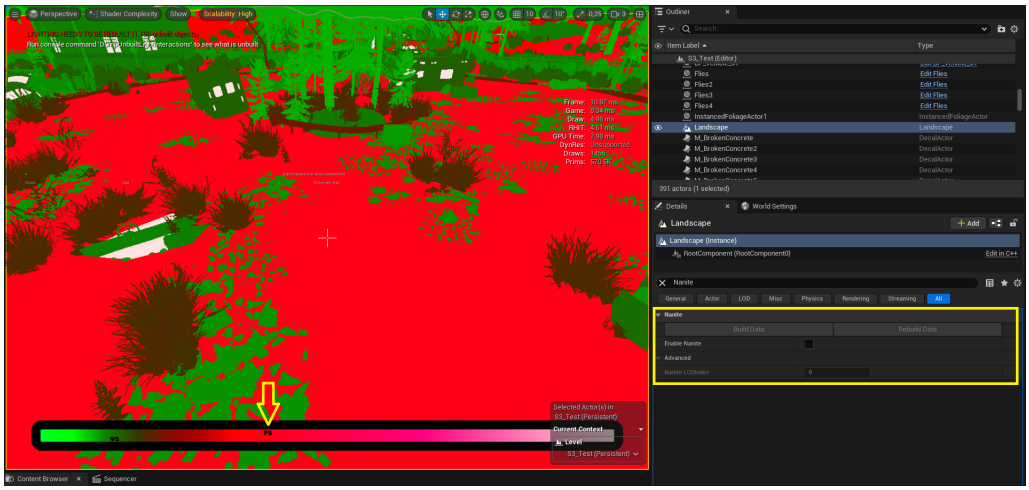

Figure 4.19: Landscape material before using Nanite

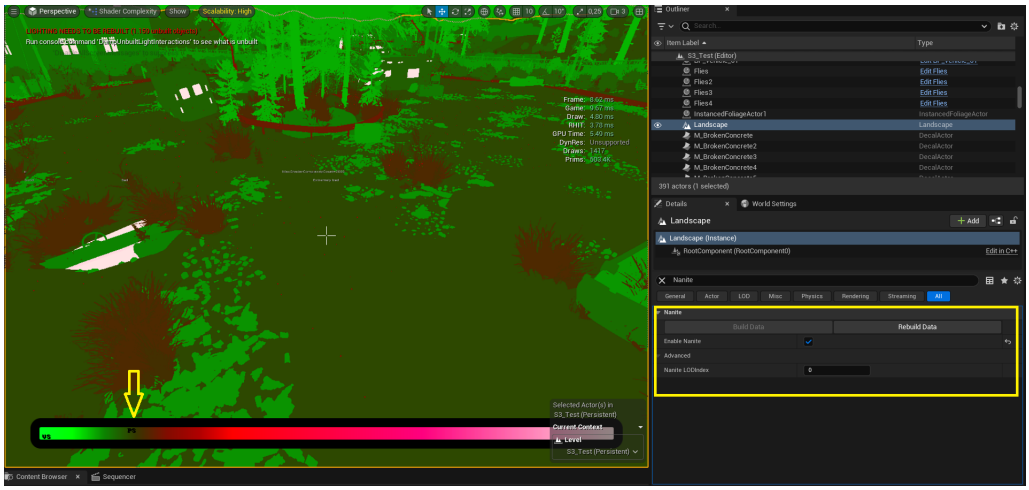

Figure 4.20: Landscape material after using Nanite

When the texture in the shader complexity is in red, it is bad for performance. The most effective way to bring it from red to green and enhance performance is by using Nanite on the landscape material or by reducing the texture resolutions of the landscape materials. When the landscape changes to green in the shader complexity, it benefits performance and render time is gained.

5 Player Experience Evaluation

Game developers use various strategies to elevate player experience, ensuring that games are not just enjoyable, but also exciting and highly immersive. Common techniques including narration, character development, and storytelling draw players into the game world and cause them to become invested in the game's outcome. The utilization of advanced graphics and audio design play crucial roles in constructing realistic environments that blur the line between game and reality, enhancing player experience. By streamlining user interface design and incorporating complex game mechanics, developers encourage player engagement and add depth to the experience. Through a combination of these and other innovative techniques, developers strive to create a gaming experience that resonates with their players.

5.1 Literature review

Although many studies have inspected techniques to enrich the gaming experience, the relationship between visual embellishments and player experience still remains of great interest to researchers. In their research of the impact of visual embellishments (VEs) on player experience, Hicks et al. (2019a) explored the idea of "juiciness," or the redundant feedback to player actions, in video games [6]. In their study, Hicks et al. investigated whether visual embellishments increased not only player experience, but also a game's aesthetic appeal, player performance, and perceived competence. Interestingly, their findings concluded that while VEs enhanced the aesthetic appeal of games, they did not affect objective player performance, and the impact of VEs on player competence and player experience varied by game. Ultimately, the ability of VEs to influence aspects of player experience is limited and requires further investigation; however, these findings may be of particular interest to game developers who seek to improve player engagement without altering the mechanics of the game.

These findings have been replicated in similar studies. Hicks et al. (2019b) examined the impact of gamification and juiciness on player experience [7]. Gamification involves the application of game design elements (such as game points and leaderboards) to enhance player engagement. Hicks et al. (2019b) developed four conditions of a VR game simulation: a standard version with very little feedback, a gamified version with gamification elements, a juicy version with audiovisual feedback, and a combined gamified and juicy version [7]. While both the gamified and juicy version improved player experience by increasing satisfaction and engagement, only juiciness significantly fulfilled all basic psychological needs and improved players' intrinsic motivation. Although the players had the tendency to prefer the combined gamified and juicy version of the simulation, gamification and juiciness did not improve their performance. This study highlights the nuanced roles that gamification and juiciness play in enhancing the gaming experience, and their impending ability to influence the future of game development.

In an earlier study of juiciness by Juul and Begy (2016), players played two versions of a matching tile game: a standard version with minimal feedback and a juicy version in which they received audiovisual feedback for their actions [9]. The researchers hypothesized that juiciness would improve the game quality and ease of use, but may also affect player performance as a result of cognitive load. As they predicted, players found that the juicy version was higher quality, but they also scored lower in it. The increase in feedback in the juicy condition could have caused distractions or increased their cognitive load, consequently impairing player performance. Therefore, juiciness can enhance the subjective quality of a game, but excessive audiovisual feedback may also negatively impact player performance. This emphasizes the importance of playtesting to validate game design decisions.

Kao (2020) also explored the concept of juiciness and player experience in a self-developed action role-playing game (RPG) [10]. In his study, participants played four versions of the same game with varying levels of juiciness: None, Medium, High, and Extreme. The results indicated that the absence of juiciness (None) and excessive juiciness (Extreme) decreased player experience, time spent playing (motivated behavior), intrinsic motivation, and game performance (fewer monsters defeated, attacks, and actions) compared to the Medium and High levels. These results suggest there is an optimal level of juiciness that will improve the gaming experience, but too little or too much juiciness can be detrimental to the players' experience. Consequently, juiciness must be carefully balanced when developing a video game to avoid underwhelming or overwhelming the players.

Moreover, research suggests that both immersion and elicited emotions can contribute to player experience. Ravaja et al. (2004) investigated players' emotional reactions and sense of presence (i.e., immersion) while playing four games [13]. The researchers found that strategy games such as Super Monkey Ball 2 elicited higher arousal and more positive emotions to create an engaging gaming experience, while intense first-person shooter FPS games such as James Bond 007: NightFire elicited negative emotions like fear and anger. In addition, the players' sense of presence varied as a function of the game's difficulty and individual differences in player traits such as sensation seeking. These findings indicate that by taking the emotional and psychological traits of the potential players into consideration when optimizing their games, developers can create more meaningful and therefore successful video games.

In an exploration of the relationship between biofeedback and personalized gameplay enhancements, Dekker and Champion (2007) monitored players' biometrics such as skin response and heartrate while playing the horror game, Half-Life 2 [1]. Then, they used the collected data to modify the gameplay (shaders, NPC spawning points, and screen shakes) in real time based on the players' physiological responses. The researchers discovered that biofeedback could enhance players' immersion and feelings of horror. In particular, audio effects greatly affected players' reactions and engagement, with the sound of the player character's heartbeat making the players more anxious. Players also reacted strongly to visual effects such as screen shake, and displaying a black and white screen based on players' heartrates made them calmer. However, the extent of their reactions to visual effects depended on their engagement; engaged players reported that the visuals made them more engaged, while unengaged players felt as if they were distracting. This study emphasizes the promising potential of biofeedback to create a more personalized and engaging gaming experience, particularly in horror games where players' emotional states impact their experience.

Similarly, researchers Nacke and Lindley (2008) studied the concepts of immersion and flow in FPS games by making three modifications to the game Half-Life 2: a control (boredom) level, a level to induce flow through difficulty and pace of challenges, and an immersive level focused on enhancing the audiovisual experience [12]. Using both objective psychophysiological measures and subjective questionnaires, players scored highest on competence and lowest on challenge, immersion, and flow in the control level. The researchers also discovered that the flow level produced higher arousal and positive affect, suggesting that game flow could be related to combative elements of FPS horror games. Additionally, highly immersive games may be connected to more positive emotions about the gaming experience. This research provides insight into how player experience can be influenced by various game design elements, but the relationship between physiological responses and psychological states of the gameplay experience requires further investigation.

While researchers conducted several studies aimed to assess and enhance player experience, further research is required to validate these findings. My study built upon the foundational work of previous studies by continuing the investigation of the relationship between game mechanics, player experience, and visual enhancements through the creation of a novel first-person shooter (FPS) game, Fallen World. I examined how the presence or absence of visual effects, such as hit reactions, altered lighting, and particle effects, contributed to player experience. Furthermore, I closed gaps in recent studies by collecting subjective player feedback to gather suggestions for optimizing the gaming experience. The goal of my study was to use my findings to understand how to create a more immersive gaming experience for players. Through this approach, my research contributed to a deeper understanding of the factors that enhance player experience and offered valuable feedback for future game development research. Additionally, my research highlighted the importance of player involvement in the game development process and emphasized the relationship between game developers and players in creating an engaging gaming experience.

In my study, I focused on one independent variable: the presence of visual embellishments (*Standard*, *Embellished*) in the video game Fallen World. I employed a mixed-methods approach, combining quantitative survey data with qualitative data from open-ended questions. I assessed the dependent variable questions in the survey on a 7-point Likert scale about the degree to which a participant agreed with statements made about their experience while playing Fallen World. I also included open- and close-ended questions pertaining to participants' history of playing video games and feedback for Fallen World. I hypothesized that the presence of visual embellishments would positively enhance player experience.

5.2 Method

5.2.1 Hypothesis

The addition of visual embellishments enhances player experience.

5.2.2 Participant demographics

A convenience sample of 10 university students from the Hochschule für Technik und Wirtschaft des Saarlandes participated in this study. In the 11-question exit questionnaire, participants answered questions related to their previous gaming experience and feedback for the game Fallen World. I collected records of participants' physiological sex and age range. This study involved 10 participants (10 men, 0 women). The age range of participants was 18-34 years old.

Of the 10 participants, 50% indicated that they played video games several times per week, 30% played every day, 10% played once per week, and 10% never played video games. When asked why they played video games, 80% of participants played video games for fun, 40% played for relaxation, 40% for social interaction, 40% for stress relief, 10% for competition, and 10% for other reasons. Among the preferred genre of video game, action games was the most common choice, selected by 80% of participants. This was followed by FPS games (50%), adventure (40%), sport (30%), and strategy video games (30%). 60% of participants indicated that both game mechanics and graphics were the most important aspects of a video game, and 50% indicated that the storyline was another of the most important aspects of a video game. When asked which factors influenced their decision to buy or try video games, graphics (50%), genre (40%), storyline (40%), recommendations (30%), and ratings (20%) influenced participants' decision.

5.2.3 Measures

In this study, player experience was measured using one validated questionnaire, along with an exit questionnaire to collect participants' game feedback.

5.2.3.1 Player experience

Participants completed the 30-question Player Experience Inventory PXI, with 3 supplemental questions about game enjoyment. The PXI assesses the psychosocial and functional factors of player experience, and is divided into 10 sub-scales: Meaning, Mastery, Immersion, Autonomy, Curiosity, Ease of Control, Challenge, Progress Feedback, Audiovisual Appeal, and Clarity of Goals [18]. Each sub-scale contains 3 similarly worded questions about that aspect of player experience. Using the PXI, participants rated statements such as "I understood the objectives of the game" on a 7-point Likert scale from "Strongly disagree" to "Strongly agree."

5.2.3.2 Exit survey

Participants completed an 11-question exit questionnaire in which they answered questions related to their demographics, history of playing video games, and game feedback. The feedback included open-ended questions about how visual effects contributed to the game experience and participants' suggestions for improving the game's visual effects (see Appendix A. for all exit survey questions).

5.2.4 Procedure

At the beginning of this study, participants were given information about the study, had the opportunity to ask questions, and then gave their informed consent to participate (see Appendix B. for consent form). The study was divided into two visual conditions (*Standard*, *Embellished*). The first condition, *Standard*, was the standard version of the game Fallen World, which did not contain any visual embellishments. The second condition, *Embellished*, was the visually embellished or "juicy" version. This version included hit reactions (camera shake, blood screen upon player character damage, and enemy blood effect upon damage), altered lighting (fog and dimmer sunlight), and particle effects (dust and flies) to create a more immersive and visually appealing gameplay experience. In each condition, participants played Fallen World on a provided laptop using a keyboard, mouse, and headset for approximately 10 minutes, or until their player character died. After each condition, participants completed the PXI by hand to assess their experience. The PXI questions were presented to the participants in a randomized order to minimize response bias and order effects and maintain the validity of the results. At the end of the study, participants completed the written exit questionnaire and were thanked for their contributions to my research. From start to finish, each participant's session lasted approximately 60 minutes.

5.2.5 Data analysis

Quantitative data were analyzed in IBM SPSS using a within subjects (repeated-measures) study design. I performed analyses using a two-tailed paired-samples t-test. The independent variable in this study was whether the game was standard or visually embellished. The dependent variable was player experience measured using the PXI. An alpha level of 5% ($p \leq 0.05$) was utilized to determine statistical significance. Furthermore, I operationalized Cohen's d to measure effect size.

Figure 5.1: Standard condition

Figure 5.2: Embellished condition

5.3 Results

I tested whether an association existed between Fallen World's visual embellishments and player experience. Because there was one independent variable for the PXI data (Visual Condition: *Standard*, *Embellished*) and all participants played both conditions, I analyzed the results using a paired-samples t-test. This allowed me to evaluate player experience for each test condition. In addition to the quantitative data, qualitative data collected from the participants' exit questionnaires were also analyzed.

5.3.1 Paired samples *t*-test

To test whether player experience differed as a function of visual embellishments, I performed a paired samples *t*-test. Results across the sub-scales of the PXI generally were not significant and only partially supported my hypothesis (See Table 5.1). The PXI component of Challenge did show a significant difference between the two test conditions across all three questions, $t(9) = -5.51, p < 0.001, r^2 = -1.74$. This suggested that the *Standard* (M = 0.27, SD = 1.50) condition significantly differed from the *Embellished* (M = 1.10, SD = 1.37) for the player experience component Challenge, such that participants found the *Embellished* condition to be significantly more challenging than the *Standard*.

While the PXI component of Progress Feedback did not show a significant difference between the two test conditions, it came close to doing so, $t(9) = 1.87, p = 0.10, r^2 = 0.59$. This suggested that the *Standard* (M = 1.47, SD = 1.36) condition differed from the *Embellished* (M = 1.17, SD = 1.61) for the component Progress Feedback, such that the participants found the *Standard* to provide slightly more progress feedback than the *Embellished*. For the PXI component Curiosity, the results of the *Standard* (M = 1.57, SD = 1.34) condition differed from the *Embellished* (M = 1.87, SD = 0.92), such that participants were more curious in the *Embellished*. For the component Ease of Control, the results of the *Standard* (M = 2.00, SD = 1.14) condition differed from the *Embellished* (M = 1.87, SD = 0.92), suggesting that participants found the *Standard* slightly easier to control than *Embellished*. For the component Audiovisual Appeal, the results of the *Standard* (M = 1.83, SD = 0.93) condition differed from the *Embellished* (M = 2.00, SD = 1.14), such that the audiovisual effects in the *Embellished* condition were slightly more appealing to participants. Finally, for the component Enjoyment, the results of the *Standard* (M = 1.77, SD = 0.90) condition differed from the *Embellished* (M = 1.67, SD = 1.11), such that the players enjoyed the *Standard* condition slightly more than the *Embellished*. The components Meaning, Mastery, Immersion, Autonomy, and Clarity of Goals did not differ at all, or only differed marginally, between the two conditions.

Table 5.1: Results of paired samples t-test

Measure	Test 1: Standard Mean (SD)	Test 2: Embellished Mean (SD)	Effect size	Significance
Meaning	1.43(1.16)	1.40(1.42)	0.07	$t(9) = 0.23, p = 0.82$
Mastery	1.67(1.26)	1.67(1.29)	0.00	$t(9) = 0.00, p = 1.00$
Immersion	1.30(1.14)	1.33(1.41)	-0.08	$t(9) = -0.26, p = 0.80$
Autonomy	1.47(1.33)	1.47(1.08)	0.00	$t(9) = 0.00, p = 1.00$
Curiosity	1.57(1.34)	1.87(0.92)	-0.50	$t(9) = -1.59, p = 0.15$
Ease of Control	2.00(1.14)	1.87(1.32)	0.28	$t(9) = 0.89, p = 0.40$
Challenge	0.27(1.50)	1.10(1.37)	-1.74	$t(9) = -5.51, p < 0.001^*$
Progress Feedback	1.47(1.36)	1.17(1.61)	0.59	$t(9) = 1.87, p = 0.10$
Audiovisual Appeal	1.83(0.93)	2.00(1.14)	-0.42	$t(9) = -1.34, p = 0.21$
Clarity of Goals	2.13(0.82)	2.17(0.85)	-0.10	$t(9) = -0.36, p = 0.73$
Enjoyment	1.77(0.90)	1.67(1.11)	0.20	$t(9) = 0.64, p = 0.54$

5.3.2 Subjective perceptions of the game conditions

In the exit questionnaire, participants disclosed which effects they noticed while playing the game Fallen World. 90% of participants noticed the blood effects, 60% noticed the lighting, 50% noticed the player sound effects, 40% noticed the camera shake, and 20% noticed the background music. Participants then answered the question, "In your opinion, which audio or visual effects improved gameplay the most?". 70% of participants responded that the blood effects improved gameplay, 40% of participants reported that camera shake improved gameplay, 30% that player sound effects improved gameplay, 20% that lighting improved gameplay, and 10% that background music improved gameplay.

When questioned how the addition of visual effects contributed to their game experience and which suggestions they had to improve the game's visual design, the participants reported the information in Table 5.2. Participants responded in both German and English. Overall, the majority of participants who answered Q1 expressed a preference for the visually embellished version of the game. The participants also provided valuable suggestions for improvements to the game design and mechanics (Q2).

Participant	Q1: How did the addition of visual effects contribute to your game experience?	Q2: If you could make suggestions to improve the visual design of the game, what would they be and why?
1	(no response)	(no response)
2	Die Hintergrundmusik von der Bewegung ist unklar wo genau der Gegner ist. Resolution sollte besser sein. Aiming ist gut bei Waffen, bei Deagle ist bisschen nicht gut.	Die Bewegung von Spieler konnte besser sein.
3	(no response)	Ein Vorschlag wäre: Wenn man Items nimmt sodass man Animation sieht wie man die Aktion durchführt.
4	Beim T2 hat mir gefallen, das mehr Realität hat wie Blut oder ein Teil vom Körper beim Schießen wegfallen.	(no response)
5	The upgrade was a good idea, more blood, more things to consider; however more testing is required as it was heavy for most CPUs (overheat problems) and FPS drops.	The texts in a different color; more work on the hitboxes, bunny hopping; the AI in general was basic and collisions at some points are not well programmed. Good job it was great.
6	(no response)	When you pick-up an item, you should look exactly in the middle (should be fixed).
7	(no response)	Die Grafik und Spiel Bedienung konnte besser sein.
8	(no response)	Grafik waren im T2 bisschen langsam. Ich finde die Farben waren bisschen zu hell. Ansonst hat sich das Spiel gut gespielt.
9	Level T2 gab besser Erfahrung.	Bullet Einnahme/Drop (bei F Taste kann man die Waffen bzw. Bullet nicht einfach nehmen.
10	Es hat es spannender gemacht. Auch schwieriger z.B. durch die Beleuchtung man hat Zombies schlechter erkannt.	(no response)

Table 5.2: Participants' written responses

5.4 Discussion

The results partially support my hypothesis that visual embellishments improve player experience. There was no significant difference in player experience across the PXI sub-scales except for the component Challenge. Generally, participants rated the PXI components similarly across both conditions. Because the *Embellished* condition received significantly higher ratings for Challenge and had a very large effect size, this suggests that the addition of visual embellishments made the game significantly more challenging for participants. Qualitative participant responses also reflected this, with one participant observing that the addition of visual effects made the game more difficult, e.g., *"durch die Beleuchtung man hat Zombies schlechter erkannt"* [due to the lighting, zombies are harder to recognize] (See Table 5.2).

However, the *Embellished* condition received lower ratings for Progress Feedback than the *Standard* condition with a moderate effect size. This could suggest that the visual enhancements, while contributing to the perceived challenge, may have inadvertently obscured the indicators essential for participants to effectively monitor their progress in the game. Due to the increased visual feedback, participants may have found it more challenging to recognize achievements and milestones, leading to a lower sense of accomplishment. In addition to this, participants rated Ease of Control higher in the *Standard* condition. These higher ratings indicate that the visual embellishments, while engaging, might also introduce elements of distraction or further complexity that affect the players' ability to control the game. The addition of visual effects could divert their attention away from gameplay mechanics or create a more confusing interface. Similarly, participants rated Enjoyment slightly lower in the *Embellished* condition, once again supporting Juul and Begy's (2016) notion that added visual effects may contribute to more cognitive load [9].

Participants also rated the PXI component Curiosity higher in the *Embellished* condition with a moderate effect size. Although this difference was not quite statistically significant, it suggests that the visual embellishments piqued participants' interest and engagement with the game environment to a greater extent than the standard condition. The more detailed visuals may have introduced elements of novelty that intrigued participants. This effect aligns with the idea that new visual details can contribute to a heightened player experience. Similarly, participants rated the PXI component Audiovisual Appeal higher in the *Embellished* condition. This difference, although not statistically significant, was nevertheless of interest, indicating that visual embellishments can slightly enhance the game aesthetics and connect players to the game's storyline and setting.

While the quantitative data only partially supported my hypothesis, the qualitative data were more nuanced. In the exit questionnaire, most participants who provided written feedback noted that they had a preference for the visually embellished game. The majority of participants (70%) indicated the blood effects improved gameplay the most, while another 40% indicated that camera shake and 20% that lighting improved gameplay. Although the overall quantitative measures did not show a significant difference in player experience, the subjective preferences of the participants demonstrated a preference towards the visually embellished version of the game. The preference for blood effects, camera shake, and lighting enhancements indicates that these elements contributed a degree of realism and excitement to the game that participants subjectively enjoyed. These qualitative responses also highlight the importance of considering player feedback in the

development and refinement of a game's visual design. This can assist developers in effectively designing games to enhance player experience.

I based my original hypothesis on the results of past research. Therefore, my results were generally inconsistent with these previous studies because my hypothesis was only supported by one PXI component, Challenge. The components Meaning, Mastery, Immersion, Autonomy, and Clarity of Goals did not differ at all, or only differed marginally, between the two conditions. Participants' ratings of Curiosity and Audiovisual Appeal were also heightened in the *Embellished* condition, but not quite to a significant extent. Both Ease of Control and Enjoyment were slightly lower in the *Embellished* condition than in the *Standard* condition, and Progress Feedback was lower in the *Embellished* condition.

Hicks et al. (2019a) found that visual embellishments enhanced the game's aesthetic appeal, but their effect on player experience varied by game [6]. In my investigation, I examined this idea through the collection of PXI data and subjective feedback from participants. From my results, I found that the effect of embellishments on player experience also varied across PXI components; however, the results were generally not significant. Conversely, participants in my study reported a subjective, rather than objective, preference for the *Embellished* version of the game. I also based my hypothesis on a study conducted by Kao (2020) in which he exposed participants to four levels of juiciness: None, Medium, High, and Extreme [10]. He found that both the absence and excessiveness of juiciness decreased player experience, signifying there is an optimal level of juiciness. Juul and Begy (2016) similarly discovered that players rated the juicy version of a game to be higher quality yet performed worse in it [9]. In my investigation, participants similarly reported a preference for the visual embellishments, yet the quantitative data generally suggested no significant difference between conditions except in the PXI component Challenge. Perhaps the visual embellishments in my study's *Embellished* condition were too excessive, thus contributing to the convoluted PXI results.

Because my results did not fully align with past research findings, there were many factors that may have affected my results. First, I did not utilize counterbalancing in my study, i.e., I had all participants begin with *Standard* and end with *Embellished*. I did not randomize the order of the tests to control for order effects. Another aspect that may have influenced my results was an imprecision of measures. The PXI used a Likert scale, but this forced participants to select one option even if they could not decide between two different options. The last factor that may have influenced my results was the presence of confounding variables. A confound in my study was whether the participant had prior experience with video games. Participants' player experience may change based on their personal experience with playing video games, such that experienced gamers might respond differently to visual embellishments than novice or less experienced gamers. Likewise, the participants' preference for a certain game genre could have influenced their experience.

5.4.1 Limitations

There are many limitations to my research, the most notable of which is the small sample size. A sample size of only 10 participants is not generalizable to the population. Another way in which my research was restricted was through selection bias. The questionnaires were distributed to only Hochschule für Technik und Wirtschaft des Saarlandes students, and all of those students were men. Therefore, the participants are not representative of the target population. Additionally, as a student myself, my resources for conducting this study were limited. This constrained access to technological equipment, resources, and budget could have influenced the results. Because the study was also conducted in a controlled lab setting on the HTW Saar campus, it does not completely replicate a natural gaming experience. Participants' experience in this controlled setting may differ from those in their usual gaming environment. Finally, my study did not account for potential interactions between the visual embellishments and other game elements (e.g. sound, game mechanics, etc.).

5.5 Conclusion

This study serves as a preliminary exploration of the complex impact of visual embellishments and lays the groundwork for more extensive investigations into how a game's visual effects influence player experience in the rapidly evolving field of game design. My findings suggest that while visual embellishments can make a game more intriguing and challenging, they do not universally improve player experience as hypothesized. The decrease in ratings for Progress Feedback and slight reductions in Enjoyment and Ease of Control in the embellished condition highlight the potential drawbacks of over-embellishment, which may lead to confusion and an increased cognitive load. Although participants rated the component of Challenge significantly higher and the components Curiosity and Audiovisual Appeal slightly higher in the embellished condition, this was not enough to confirm the benefits of visual embellishments across all aspects of player experience. Because of these mixed outcomes, it is clear that visual embellishments must be carefully developed to avoid overwhelming the players, which could lead to neutral or even negative experiences. Developers must find a balance between maintaining practicality and enhancing the game's visual appeal in order to enrich player experience.

5.5.1 Future research

The results of this study may provoke further discussion about the impact of visual embellishments on player experience. In the future, researchers may consider exploring how visual embellishments and other game elements interact to affect player experience. Researchers may also decide to conduct comparative studies that compare the effects of different types of visual embellishments (e.g. lighting, particle effects, hit reactions) to investigate how varying levels of visual complexity impact player experience. Because this study was constrained to only 10 students who attended HTW Saar in Germany, researchers may consider broadening the sample size and including participants from different cultural and regional backgrounds to explore how culture and visual embellishments can interact to influence player experience.

6 General Conclusion

Evidently, game development is an iterative process that involves extensive planning, coordination, and continual adaptation. When creating a video game, developers must consider everything from the game's conceptual design to its technical implementation to ensure the game elements are cohesive. This requires testing and retesting the game, gathering player feedback, and making adjustments as necessary. Working on this project as a team of one, I performed the roles of project manager, developer, programmer, designer, artist, and tester. Each role presented unique challenges, but I persisted and successfully developed a game that was generally well-received by the research participants. By singlehandedly balancing these roles, I gained a deeper understanding of the world of game development across several disciplines. Furthermore, I learned that a video game's success depends not just on an innovative design, but also on an awareness of players' preferences through feedback. These insights have prepared me for the challenges yet to come in the ever-evolving field of game development.

Bibliography

[1] A. Dekker and E. Champion. *Please biofeed the zombies: Enhancing the gameplay and display of a horror game using biofeedback*. Vol. 4. 2007. DOI: 10.25917/5d1443e8af4a0.

[2] Y. Denisyuk. "What are the stages of game development?" In: *Pingle* (2022). URL: https://pinglestudio.com/blog/full-cycle-development/game-development-stages.

[3] Epic Games. "About Epic Games". In: (2024). URL: https://www.epicgames.com/site/en-US/about?lang=en-US.

[4] J. French. "How to write a game design document (with examples)". In: *Game Dev Beginner* (2024). URL: https://gamedevbeginner.com/how-to-write-a-game-design-document-with-examples/.

[5] J. Gregory. *Game Engine Architecture*. CRC Press: Taylor & Francis Group, 2015. URL: http://ce.eng.usc.ac.ir/files/1511334027376.pdf.

[6] K. Hicks, K. Gerling, P. Dickinson, and V. Vanden Abeele. *Juicy game design: Understanding the impact of visual embellishments on player experience*. 2019, pp. 185–197. DOI: 10.1145/3311350.3347171.

[7] K. Hicks, K. Gerling, G. Richardson, T. Pike, O. Burman, and P. Dickinson. "Understanding the effects of gamification and juiciness on players". In: *IEEE Conference on Games (CoG)*. 2019. DOI: 10.1109/CIG.2019.8848105. URL: https://doi.org/10.1109/CIG.2019.8848105.

[8] Intel. "Game optimization methodology". In: (n.d.). URL: https://www.intel.com/content/www/us/en/docs/gpa/user-guide/2022-4/game-optimization-methodology.html.

[9] K. Juul and P. Begy. *Good feedback for bad players? A preliminary study of 'juicy' interface feedback*. https://api.semanticscholar.org/CorpusID:198609848. 2016.

[10] Kao. "The effects of juiciness in an action RPG". In: *Entertainment Computing* 34 (2020). DOI: 10.1016/j.entcom.2020.100359.

[11] Donald E. Knuth. "Computer Programming as an Art". In: *Communications of the ACM* 17.12 (1974), pp. 667–673.

[12] L. Nacke and C. A. Lindley. *Flow and immersion in first-person shooters: Measuring the player's gameplay experience*. 2008. DOI: 10.1145/1496984.1496998.

[13] N. Ravaja, M. Salminen, J. Holopainen, T. Saari, J. Laarni, and A. Järvinen. *Emotional response patterns and sense of presence during video games: Potential criterion variables for game design*. 2004. DOI: 10.1145/1028014.1028068.

[14] S. Roy. "From Pong to virtual reality: The evolution of gaming through the ages". In: *Medium* (2023). URL: https://medium.com/@subhojeet.roy0807/from-pong-to-virtual-reality-the-evolution-of-gaming-through-the-ages-e939d7599306.

[15] N. Stefyn. "What is the game development pipeline?" In: *CG Spectrum* (2022). URL: https://www.cgspectrum.com/blog/game-development-process.

[16] *Unreal Engine Documentation*. Unreal Engine. 2024. URL: https://docs.unrealengine.com/4.27/en-US/.

[17] Unreal Engine. "Features". In: (2024). URL: https://www.unrealengine.com/en-US.

[18] V. Vanden Abeele, K. Spiel, L. Nacke, D. Johnson, and K. Gerling. "Development and validation of the player experience inventory: A scale to measure player experiences at the level of functional and psychosocial consequences". In: *International Journal of Human-Computer Studies* (2019). DOI: 10.1016/j.ijhcs.2019.102370.

[19] Bojan Veselinovikj. In: *VFX in gaming: The ultimate guide to visual effects in video games. Boris FX. Link* (2024).

List of Figures

List of Tables

List of Abbreviations

FPS	first-person shooter
VFX	visual effects
GDD	game design document
HUD	heads-up display
UE	Unreal Engine
UI	user interface
VEs	visual embellishments
PXI	Player Experience Inventory

Appendix

A. Demographics and Game Feedback Survey

Demographics & Game Feedback

Bitte füllen Sie den folgenden Fragebogen vollständig aus. Vielen Dank für Ihre Teilnahme.

Q1 Alter

- ○ unter 18 Jahre
- ○ 18 - 24
- ○ 25 - 34
- ○ 35 - 44
- ○ 45 oder älter

Q2 Geschlecht

- ○ männlich
- ○ weiblich
- ○ nichtbinär/drittes Geschlecht
- ○ keine Angabe

Q3 Wie oft spielen Sie Videospiele?

- ○ täglich
- ○ mehrmals pro Woche
- ○ einmal pro Woche
- ○ einmal im Monat und seltener
- ○ nie

Q4 Warum spielen Sie Videospiele? (Bitte kreuzen Sie alle zutreffenden Gründe an.)

- [ ] Entspannung
- [ ] Stressabbau
- [ ] Wettbewerb
- [ ] Spaß
- [ ] soziale Interaktion
- [ ] andere (bitte angeben) _____
- [ ] nicht zutreffend

Q5 Welches Genre von Videospielen bevorzugen Sie?

- ○ Action
- ○ Adventure
- ○ First Person Shooter (FPS)
- ○ Strategie
- ○ Sport
- ○ andere (bitte angeben) _____
- ○ nicht zutreffend

Q6 Welcher Aspekt eines Videospiels ist für Sie am wichtigsten?

- ○ Spielmechanik
- ○ Grafik
- ○ Handlung
- ○ andere (bitte angeben) _____
- ○ nicht zutreffend

Q7 Welche Faktoren beeinflussen Ihre Entscheidung, ein neues Spiel zu kaufen oder auszuprobieren? (Bitte kreuzen Sie alle zutreffenden Faktoren an.)

- [ ] Genre
- [ ] Grafik
- [ ] Handlung
- [ ] Empfehlungen
- [ ] Bewertungen
- [ ] andere (bitte angeben) _____
- [ ] nicht zutreffend

Zur Beantwortung der folgenden Fragen (Q8-11) denken Sie bitte an die zwei Levels, die Sie gerade gespielt haben.

Q8 Welche Effekte sind Ihnen aufgefallen?

- ○ Kamerawackeln
- ○ Bluteffekt
- ○ Hintergrundmusik
- ○ Beleuchtung
- ○ Spieler-Soundeffekt
- ○ andere (bitte angeben) _____

Q9 Nach Ihrer Meinung, welcher Audio- oder Visuelleffekt hat das Gameplay am meisten verbessert?

- ○ Kamerawackeln
- ○ Bluteffekt
- ○ Hintergrundmusik
- ○ Beleuchtung
- ○ Spieler-Soundeffekt
- ○ andere (bitte angeben) _____

Q10 Auf welche Weise hat die Hinzufügung von Visuelleffekten zu Ihrem Spielerlebnis beigetragen?

_____

_____

_____

_____

_____

_____

_____

Q11 Wenn Sie Verbesserungsvorschläge für das visuelle Design des Spiels machen könnten, welche wären das und warum?

_____

_____

_____

_____

_____

_____

_____

B. Participant Consent Form

Einverständniserklärung für Teilnehmer:

Hiermit erkläre ich, _____, dass ich mich freiwillig dazu entschieden habe, an der Studie zur Bewertung des Spielerlebnisses in einem FPS-Spiel teilzunehmen, die im Rahmen der Bachelorarbeit von Mouayad Haji Omar durchgeführt wird. Mir ist bewusst, dass die Teilnahme an dieser Studie freiwillig ist und ich das Recht habe, meine Teilnahme jederzeit ohne Angabe von Gründen abzubrechen, ohne dass dies negative Konsequenzen für mich hat.

Ich verstehe, dass die Informationen und Antworten, die ich während der Studie bereitstelle, vertraulich behandelt werden und ausschließlich für wissenschaftliche Zwecke verwendet werden. Meine persönlichen Daten werden anonymisiert und nicht an Dritte weitergegeben.

Ich wurde darüber informiert, dass ich das Recht habe, Fragen zu stellen und weitere Informationen über die Studie zu erhalten, bevor ich meine Teilnahme bestätige. Mir ist auch bewusst, dass ich das Recht habe, meine Einwilligung zur Teilnahme an der Studie jederzeit zurückzuziehen, ohne dass dies Auswirkungen auf mich hat.

Des Weiteren bestätige ich, dass ich keine Anzeichen von epileptischen Anfällen habe und mich in einem angemessenen geistigen und Zustand befinde, um an der Studie teilzunehmen.

Ich erkläre mich damit einverstanden, an der Studie teilzunehmen, und habe die oben genannten Informationen verstanden und akzeptiert.

Datum: _____

Unterschrift: _____

Kolophon

Dieses Dokument wurde mit der LaTeX-Vorlage für Abschlussarbeiten an der htw saar im Bereich Informatik/Mechatronik-Sensortechnik erstellt (Version 2.23, März 2022). Die Vorlage wurde von Yves Hary und André Miede entwickelt (mit freundlicher Unterstützung von Thomas Kretschmer, Helmut G. Folz und Martina Lehser). Daten: (F)10.95 – (B)426.79135pt – (H)688.5567pt