VISVESVARAYA TECHNOLOGICAL UNIVERSITY
JNANASANGAMA, BELAGAVI - 590018

Computer Graphics and Visualization Mini Project Report
on

A THIRD PERSON SHOOTING GAME USING UNREAL ENGINE

Submitted in partial fulfillment for the award of degree of

Bachelor of Engineering
in
COMPUTER SCIENCE AND ENGINEERING

Submitted by

ABHIRAM B S (1BG21CS001)
NISCHAL U (1BG21CS052)

Under the Guidance of
Prof. Akshitha Katkeri
Assistant Professor
Department of CSE, BNMIT

Vidyaya Amrutham Ashnuthe

B.N.M. Institute of Technology
An Autonomous Institute Under VTU

Department of Computer Science and Engineering

2023– 2024

B.N.M. Institute of Technology

An Autonomous Institute Under VTU

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Vidyaya Amrutham Ashnuthe

CERTIFICATE

Certified that the mini project report entitled **A THIRD PERSON SHOOTING GAME USING UNREAL ENGINE** carried out by Mr. **ABHIRAM B S (1BG21CS001)** and Mr. **NISCHAL U (1BG21CS052)** are bonafide student of V Semester B.E., **B.N.M Institute of Technology** in partial fulfillment for the Bachelor of Engineering in COMPUTER SCIENCE AND ENGINEERING of the **Visvesvaraya Technological University**, Belagavi during the year 2023-24. It is certified that all corrections / suggestions indicated for internal Assessment have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of Computer Graphics Mini Project Laboratory prescribed for the said degree.

Prof. Akshitha Katkeri
Assistant Professor
Department of CSE
BNMIT, Bengaluru

Dr. Chayadevi M L
Professor and HOD
Department of CSE
BNMIT, Bengaluru

Name and Signature

1. Examiner 1:

2. Examiner 2:

ACKNOWLEDGMENT

The success and final outcome of this project required a lot of guidance and assistance from many people and I am extremely privileged to have got this all along the completion of my project.

I would like to thank **Shri. Narayan Rao R Maanay**, Secretary, BNMEI, Bengaluru for providing the excellent environment and infrastructure in the college.

I would like to sincerely thank **Prof. T J Rama Murthy**, Director, BNMIT, Bengaluru for having extended his constant support and encouragement during the course of this project.

I would like to sincerely thank **Dr. S Y Kulkarni**, Additional Director, BNMIT, Bengaluru for having extended his constant support and encouragement during the course of this project.

I would like to express my gratitude to **Prof. Eishwar N Maanay**, Dean, BNMIT, Bengaluru for his relentless support and encouragement.

I would like to thank **Dr. Krishnamurthy G N**, Principal, BNMIT, Bengaluru for his constant encouragement.

I would like to thank, **Dr. Chayadevi M L**, Professor & Head of the Department of Computer Science and Engineering for the encouragement and motivation she provides.

I would also like to thank **Prof. Akshitha Katkeri**, Assistant Professor, Department of Computer Science and Engineering for providing me with her valuable insight and guidance wherever required throughout the course of the project and its successful completion.

ABHIRAM B S - 1BG21CS001

NISCHAL U - 1BG21CS052

ABSTRACT

Our Unreal Engine-powered third-person shooter game delivers an immersive experience through meticulously designed gameplay, blending intense combat with captivating narrative. With finely tuned shooting mechanics and diverse weapons, players navigate a world filled with peril and promise. Health bars enhance tactical decision-making, while intelligent enemy AI ensures dynamic and challenging encounters. Set in a custom Egypt-themed map, visually stunning environments offer both beauty and depth for strategic gameplay. Character design and animation deepen immersion, driving the narrative forward. Throughout development, player feedback has been integral, resulting in a polished and engaging experience. In an industry driven by innovation, our game stands as a beacon of quality, inviting players to embark on an unforgettable journey filled with action and intrigue.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER – I

INTRODUCTION

1.1 Overview

In today's gaming landscape, the demand for immersive experiences has never been greater. With advancements in technology pushing the boundaries of what's possible, players crave more than just entertainment – they seek escapism, adventure, and the thrill of stepping into new worlds. It is within this dynamic and ever-evolving realm that our project, the Third Person Shooting Game, finds its inspiration.

At its core, our motivation stems from the desire to create a gaming experience that transcends the conventional boundaries of entertainment. We recognize that gaming is not just about pressing buttons and watching pixels move – it is about immersing oneself in a narrative, experiencing adrenaline-pumping action, and feeling a sense of accomplishment with every triumph. Drawing upon the rich tapestry of modern gaming, we aim to weave a story that captivates players from the moment they enter our virtual world.

With Unreal Engine as our canvas, we are afforded a palette of tools and techniques that allow us to bring our vision to life in ways previously thought impossible. From the fluid movement mechanics that make every step feel like a dance, to the responsive shooting systems that put the power of a weapon in the player's hands, every aspect of our game is designed to draw players deeper into the experience.

However, it is not just about the gameplay – it is about the world we create, the environments we construct, and the stories we tell. Our Egypt-style map is more than just a backdrop for gunfights – it is a living, breathing world filled with secrets to uncover, challenges to overcome, and mysteries to solve. From the crumbling ruins of ancient temples to the labyrinthine passages of forgotten tombs, every corner of our world tells a story, waiting for players to discover it.

In addition, it is not just about the action – it is about the strategy, the teamwork, and the camaraderie that comes from working together towards a common goal. Whether players are fighting side by side with their teammates, outsmarting their opponents with clever tactics, or simply enjoying the thrill of the chase, our game offers something for everyone.

In the end, our goal is simple: to create an experience that resonates with players long after they have put down the controller. With its blend of action, adventure, and excitement, our Third Person Shooting Game promises to be an unforgettable journey into the heart of gaming – and we cannot wait to share it with the world.

1.2 Problem Statement

Despite the abundance of third-person shooter games in the gaming market, there remains a noticeable gap in the availability of titles that seamlessly blend immersive gameplay, strategic depth, and captivating narrative elements. Many existing games in this genre often prioritize action over storytelling or lack cohesive mechanics that engage players beyond the superficial level. Additionally, while some games offer multiplayer modes, they often struggle to provide a balanced and dynamic experience that keeps players engaged over the long term. The primary aim of this project is to address these shortcomings by creating an enthralling third-person shooter game using the powerful Unreal Engine. The envisioned game is one that seamlessly blends elements of a compelling storyline, dynamic combat mechanics, challenging enemy encounters, and visually stunning environments to deliver an unforgettable gaming experience. Through meticulous attention to detail in gameplay mechanics, level design, and narrative development, our goal is to develop a game that not only captivates players but also immerses them in a rich and engaging gameplay experience, fostering a sense of immersion, excitement, and camaraderie.

1.3 Motivation

The motivation behind embarking on this project stems from a shared passion for gaming and a deep-seated desire to push the boundaries of what is possible within the third-person shooter genre. We are driven by a vision to create an experience that not only entertains but also resonates with players on a profound level, fostering a sense of immersion, excitement, and camaraderie. Drawing inspiration from the shortcomings observed in existing titles, we are determined to fill the gap in the market by delivering a game that seamlessly blends elements of captivating storytelling, dynamic gameplay mechanics, and visually stunning environments. With Unreal Engine as our canvas, we are excited to embark on this journey, fueled by the prospect of designing an unforgettable gaming experience that leaves a lasting impact on players worldwide.

- **Unleashing Creativity with Unreal Engine:** OpenGL and other frameworks can be used to create games, but Unreal Engine's unmatched ability to produce immersive and aesthetically appealing experiences is what drew us to it. Large-scale gaming studios could not have imagined being able to unleash the creativity and realize their vision as developers can thanks to the extensive toolkit and features that Unreal Engine provides.
- **Immersive Gaming Experiences:** Our goal is to give users a genuinely captivating and immersive gaming experience. With the help of the Unreal Engine, developers can build scenes that are vibrant, dynamically lit, and have realistic dynamics that entice players to explore the game world further.

- **Pushing the Boundaries of Game creation:** Unreal Engine gives developers the freedom to innovate and push the limits of game creation. With features like smooth level transitions and intricate AI behaviors, Unreal Engine provides the technology and tools required to bring big ideas to life. We may experiment with novel gameplay features, narrative approaches, and graphic design elements that enthrall gamers and distinguish our game from the competition by utilizing Unreal Engine.
- **Efficiency and Quick Prototyping with Blueprints:** To expedite iteration cycles and simplify the development process, we decided to use a blueprint-based, no-code approach to game creation. Without requiring a lot of coding, Unreal Engine blueprints offer a visual scripting interface that enables quick prototyping and experimentation. With this method, we can swiftly iterate, test concepts, and improve gameplay mechanics until we get the desired outcomes.
- **Accessibility and Collaboration:** The accessibility and collaboration features provided by Unreal Engine are another aspect that keeps us going. Unreal Engine's extensive asset pipeline and user-friendly interface make it simple for developers of all experience levels to contribute to the project. When it comes to level design, animation, and gaming mechanics, Unreal Engine encourages cooperation and teamwork, which enables us to take use of the wide range of skills within our development team to produce something genuinely remarkable.

In summary, our motivation for embarking on this project stems from a desire to leverage the advanced capabilities of Unreal Engine to create immersive and engaging gameplay experiences that push the boundaries of game design. By embracing a no-code and blueprint-based approach, we aim to streamline development, foster collaboration, and deliver a game that captivates players and leaves a lasting impression in the gaming industry.

1.4 Computer Graphics

Computer graphics is a field of study and practice that focuses on creating, manipulating, and displaying visual content using computers. It encompasses a wide range of applications, including video games, animation, visual effects in movies, virtual reality, scientific visualization, and user interfaces. At its core, computer graphics involves representing and processing graphical data in digital form, which is then rendered onto a display device such as a computer monitor or virtual reality headset. This process often involves techniques such as modelling, rendering, animation, and image processing.

Modelling involves creating digital representations of objects, characters, and environments using geometric shapes, textures, and other visual elements. Rendering is the process of generating 2D or 3D images from these digital models, taking into account factors such as lighting, shadows, and surface materials to create realistic or stylized visuals.

Animation adds movement and dynamics to these digital models, allowing them to come to life through sequences of key frames or procedural algorithms. Image processing techniques are used to enhance or manipulate digital images, such as applying filters, adjusting colors, or removing imperfections. Computer graphics technologies have advanced rapidly in recent years, driven by improvements in hardware capabilities, software algorithms, and artistic techniques. This has led to increasingly realistic and immersive visual experiences across various industries, from entertainment and gaming to education and scientific research.

In summary, computer graphics plays a crucial role in modern computing, enabling the creation of visually compelling and interactive content that informs, entertains, and engages audiences around the world.

1.5 Unreal Engine:

Unreal Engine stands as a robust and adaptable game development platform pivotal to realizing our project's ambitions. Its extensive toolkit, advanced rendering capabilities, and user-friendly workflow significantly expedite our development process while enabling us to achieve visually stunning graphics and immersive gameplay experiences. Notably, Unreal Engine's high-fidelity graphics rendering empowers us to craft visually captivating environments, characters, and effects, enhancing player immersion. Its sophisticated lighting, shading, and particle systems contribute to dynamic and lifelike visuals, enriching our game world with breathtaking detail. Moreover, the Blueprint visual scripting system within Unreal Engine facilitates rapid prototyping and implementation of gameplay mechanics and features without the need for extensive coding knowledge. This no-code approach streamlines our workflow, enabling swift iteration and experimentation to refine our game mechanics in line with our vision.

Additionally, Unreal Engine offers a comprehensive asset library, marketplace, and community support, providing us access to a plethora of resources ranging from pre-made assets and plugins to tutorials and forums. This facilitates leveraging the collective knowledge and expertise of the Unreal Engine community, expediting our learning curve and aiding in overcoming challenges more effectively. Overall, Unreal Engine serves as an indispensable tool in realizing our project's vision, offering a feature-rich development platform that empowers us to create an engaging and immersive third-person shooter game that resonates with players.

1.6 Epic Games:

Epic Games, the driving force behind Unreal Engine, plays a pivotal role in empowering our project by equipping us with the necessary tools, resources, and support to bring our vision to fruition. As a prominent game development company, Epic Games not only provides the Unreal Engine platform but also offers additional benefits that enhance our development process and contribute to our project's success. Notably, Epic Games' commitment to innovation and technological advancement ensures that we have access to cutting-edge tools and features through continuous updates and improvements to Unreal Engine. Moreover, Epic Games provides extensive documentation, tutorials, and learning resources catering to developers of all skill levels, facilitating mastery of Unreal Engine's intricacies and advanced techniques. The company fosters a vibrant and inclusive community of developers through events, forums, and online communities, fostering collaboration, knowledge sharing, and feedback. Additionally, initiatives like Unreal Dev Grants and Epic Mega Grants demonstrate Epic Games' dedication to democratizing game development and supporting indie developers, further enriching our development journey.

Overall, Epic Games' unwavering commitment to empowering developers, fostering innovation, and nurturing the gaming community significantly contributes to our project's success. By providing us with the necessary tools, resources, and opportunities, Epic Games enables us to create a captivating and immersive third-person shooter game that pushes the boundaries of game development.

1.7 Applications of Computer Graphics:

Computer graphics is a multifaceted field encompassing the creation, manipulation, and rendering of visual content using computers. It finds widespread applications in industries such as entertainment, scientific visualization, and design. One of the primary areas of focus in computer graphics is the generation of realistic images, which involves simulating the behavior of light and materials to create lifelike scenes. Techniques such as ray tracing, rasterization, and shading are employed to achieve high-fidelity rendering, enabling the creation of stunning visual effects in movies, video games, and virtual environments. Another key aspect of computer graphics is geometric modeling, which involves representing objects and scenes using mathematical primitives such as points, lines, and polygons. Geometric modeling techniques enable the creation of complex 3D models that can be manipulated and transformed in virtual space. These models serve as the foundation for various applications, including architectural visualization, product design, and animation. In addition to geometric modeling, computer graphics also encompasses techniques for simulating natural phenomena such as fluid dynamics, particle systems, and crowd behavior, allowing for the creation of dynamic and interactive virtual worlds. Moreover, computer graphics plays a vital role in user interface design and human-computer interaction. Graphical user interfaces (GUIs) rely on visual elements such as icons, windows, and menus to facilitate user interaction with software applications. Effective GUI design involves principles of visual hierarchy, feedback, and affordance to create intuitive and engaging interfaces. Furthermore, advances in computer graphics have led to the development of interactive technologies such as virtual reality (VR) and augmented reality (AR), which immerse users in virtual environments or overlay digital content onto the real world. These technologies have applications in fields ranging from training and education to gaming and simulation, revolutionizing how we interact with computers and digital information.

Computer graphics finds application in a wide range of fields, playing a pivotal role in enhancing visual communication, analysis, and entertainment across various industries. Here are some key applications of computer graphics:

- **Entertainment and Media:** Computer graphics revolutionized the entertainment industry, enabling the creation of visually stunning movies, television shows, and video games. From lifelike characters and immersive environments to breathtaking special effects and animations, computer graphics bring stories to life and captivate audiences worldwide.
- **Video Games:** Perhaps the most well-known application of computer graphics, video games rely heavily on advanced graphics technology to deliver immersive gameplay experiences. From rendering realistic 3D environments to simulating complex physics and AI behaviors, computer graphics play a crucial role in creating compelling and interactive gaming experiences.
- **Simulation and Training:** Computer graphics are used extensively in simulators for training purposes, such as flight simulators for pilots or driving simulators for automotive training. These simulations provide a safe and cost-effective way to train individuals in various scenarios and environments, replicating real-world conditions with high fidelity.
- **Architectural Visualization:** Architects and designers use computer graphics to create realistic renderings and visualizations of architectural designs. These visualizations help clients and stakeholders better understand proposed projects, allowing for informed decision-making and feedback before construction begins.
- **Medical Imaging:** Computer graphics are employed in medical imaging techniques such as MRI, CT scans, and ultrasound to generate detailed visual representations of internal anatomical structures. These visualizations aid healthcare professionals in diagnosis, treatment planning, and medical research.

- **Scientific Visualization:** Scientists and researchers use computer graphics to visualize complex datasets and simulations, enabling them to gain insights into scientific phenomena and communicate their findings effectively. This includes visualizing weather patterns, molecular structures, astronomical data, and more.
- **Advertising and Marketing:** Computer graphics are widely used in advertising and marketing campaigns to create eye-catching visuals, animations, and interactive experiences that attract and engage consumers. From product renderings to immersive virtual experiences, computer graphics help brands tell compelling stories and connect with their audience.
- **Education and Training:** Computer graphics are utilized in educational materials and interactive learning platforms to convey complex concepts and subjects in a visually engaging manner. From interactive 3D models to virtual reality simulations, computer graphics enhance the learning experience and make education more accessible and immersive.

CHAPTER – II

LITERATURE SURVEY

2.1 Introduction on Computer Graphics

In the realm of computer graphics, our project stands as a testament to the transformative power of advanced technologies in shaping immersive gaming experiences. Rooted in the utilization of Unreal Engine, a potent game development platform renowned for its versatility and robust capabilities, our endeavour epitomizes the fusion of artistry and innovation in crafting captivating virtual worlds. Through meticulous attention to detail and leveraging the cutting-edge features of Unreal Engine, we embark on a journey to redefine the boundaries of visual fidelity and gameplay immersion in the realm of third-person shooter games. This introduction serves as a gateway to exploring the intricate interplay between technology and creativity, where our project emerges as a beacon of excellence in the ever-evolving landscape of computer graphics.

2.2 Related Work

- **A Basic Overview on Third Person Shooting Game and its Elements:**

The research paper delves into the realm of game design, particularly focusing on the development of a third-person shooter game using Unreal Engine. It highlights the significance of game design in providing entertainment, education, and immersive experiences for players. With the gaming industry witnessing rapid evolution and technological advancements, the ability to generate innovative ideas for game development has become paramount. The paper aims to address this challenge by presenting a case study of developing a basic third-person shooter game that meets fundamental requirements.

The introduction contextualizes the project within the broader landscape of video games, emphasizing the popularity and appeal of third-person shooter games. It discusses the differences between first-person and third-person perspectives, highlighting the immersive nature of the latter. The paper outlines the various aspects involved in game development, including player movement, shooting mechanics, enemy AI, player status, and win/lose conditions.

The proposed methodology details the steps involved in creating the game, from setting up the environment to implementing character movement, shooting mechanics, and enemy AI. It discusses the utilization of assets from Unreal Engine Marketplace and Megascans for environment creation and character animation. The methodology also covers performance criteria, system configuration, and optimization techniques to ensure smooth gameplay experience across different platforms.

Experimental results showcase the game's performance metrics, including frame rate, memory usage, and hard disk size. Performance optimization strategies, such as SSD utilization, GPU usage, and CPU frequency, are discussed to enhance the game's efficiency and rendering capabilities.

In conclusion, the paper reflects on the evolution of game development, emphasizing the advancements in gaming engines and asset libraries that facilitate easier and more efficient game creation. It highlights the accessibility of game development tools and showcases how even students can embark on game development projects with the right resources and knowledge. Overall, the research paper serves as a comprehensive guide to developing a third-person shooter game and sheds light on the intricacies of game design and development processes.[1]

- **First Person vs. Third Person Perspective in Digital Games: Do Player Preferences Affect Immersion?**

The paper investigates whether playing digital games in first-person perspective (1PP) is more immersive compared to third-person perspective (3PP). It aims to determine if the player's immersion level is affected by their preferred perspective. The study involved participants playing a role-playing game (RPG) in either 1PP or 3PP, naming their preferred perspective, and subjectively evaluating their immersive experience using a questionnaire.

The results indicate that participants experienced higher levels of immersion when playing in 1PP, regardless of their preferred perspective. The study involved 40 participants with varying levels of gaming experience. Skyrim, an RPG with both 1PP and 3PP options, was chosen as the game for the experiment. Participants were interrupted after 15 minutes of gameplay to fill out the immersion evaluation questionnaire.

Overall, the study found a significant difference in immersion between 1PP and 3PP gameplay. Participants who played in 1PP reported higher levels of immersion. However, preferences for a specific perspective also showed a marginal influence on immersion, with some indications of interaction between perspective and preference.

Further analysis revealed that the main influence on immersion was through real-world dissociation, challenge, and cognitive involvement. Players felt more connected to the game world and were more cognitively engaged in 1PP. Additionally, challenge scores were influenced by preference, suggesting that players who preferred 1PP may find it more challenging.

The paper concludes that POV significantly impacts immersion in digital games, with 1PP generally providing a more immersive experience. However, it suggests that gaming experience is multi-faceted, and other factors such as social presence may also play a role in overall gaming experience. Further studies are recommended to explore the importance of POV in different genres of games and its impact on various aspects of gaming experience.[2]

These two papers collectively provide a comprehensive understanding of game design principles and player experience factors, offering valuable insights for building a third-person shooting game using Unreal Engine. The first paper lays out the fundamental elements of game design, emphasizing the importance of player perspective, movement mechanics, shooting mechanics, enemy AI, and performance optimization. It serves as a foundational guide for understanding the key components necessary for creating an engaging and functional game.

The second paper complements this by exploring the specific impact of player perspective on immersion in digital games. By examining the differences between first-person and third-person perspectives and how player preferences influence immersion, it offers actionable insights for designing the player experience in a third-person shooter game. Understanding the psychological and experiential aspects of player perspective can inform design decisions that enhance immersion and overall gameplay enjoyment.

Together, these papers provide a comprehensive toolkit for developers embarking on the creation of a third-person shooting game. By incorporating insights from both papers, developers can ensure that their game not only meets technical requirements but also delivers an immersive and enjoyable experience for players.

CHAPTER – III

SYSTEM REQUIREMENTS

3.1 Hardware Requirements

- **Processor:** AMD Ryzen 7 5800H/ INTEL i7 11800H, a modern multi-core processor, was employed for efficient simulation computations.
- **Memory (RAM):** The system was equipped with 16 GB of 3200MHz RAM, to effectively handle the computational demands of game development and accommodate larger simulations.
- **Storage:** A 256 GB SSD was utilized for storing simulation data, input files, and software libraries. While meeting the minimum requirement, considering the size of game development projects, additional storage capacity may be advantageous.
- **Graphics Processing Unit (GPU):** The system incorporated an Nvidia RTX 3050 with 4GB VRAM, a dedicated GPU supporting general-purpose computing (NVIDIA CUDA), which accelerates simulation algorithms involving graphical rendering or parallel processing, meeting or surpassing the recommended specifications.

Overall, the hardware configuration provided ample processing power, memory, storage, and GPU acceleration capabilities, making it well-suited for the development of a third-person shooting game using Unreal Engine.

3.2 Software Requirements

- **EPIC GAMES**

Epic Games, the company behind Unreal Engine, is instrumental in empowering our project with the tools, resources, and support needed to bring our vision to life. As a leading game development company, Epic Games not only provides the Unreal Engine platform but also offers a wealth of additional benefits that enhance our development process and contribute to the success of our project.One of the key advantages of working with Epic Games is their ongoing commitment to innovation and technological advancement. Through continuous updates and improvements to Unreal Engine, Epic Games ensures that we have access to the latest tools and features, allowing us to leverage cutting-edge technology to create groundbreaking gaming experiences.

Furthermore, Epic Games provides extensive documentation, tutorials, and learning resources that support developers of all skill levels. Whether it's mastering the intricacies of the Unreal Engine's Blueprint visual scripting system or delving into advanced rendering techniques, Epic Games equips us with the knowledge and expertise needed to realize our creative vision.

Additionally, Epic Games fosters a vibrant and inclusive community of developers through events, forums, and online communities. This network provides invaluable opportunities for collaboration, knowledge sharing, and feedback, enabling us to learn from others, gain inspiration, and overcome challenges more effectively.

Moreover, Epic Games' commitment to democratizing game development is evident through initiatives such as Unreal Dev Grants, which provide financial support to promising indie developers, and the Epic MegaGrants program, which awards grants to projects that demonstrate innovation and creativity.

Overall, Epic Games' dedication to empowering developers, fostering innovation, and supporting the gaming community plays a crucial role in the success of our project. By providing us with the tools, resources, and opportunities needed to thrive, Epic Games enables us to create a captivating and immersive third-person shooter game that pushes the boundaries of what is possible in game development.

Figure 3.1: Epic Games Logo

- **UNREAL ENGINE**

Unreal Engine is a powerful and versatile game development platform that is instrumental in bringing our project to life. With its robust set of tools, advanced rendering capabilities, and intuitive workflow, Unreal Engine significantly accelerates our game development process while allowing us to achieve stunning visual fidelity and immersive gameplay experiences.

One of the key benefits of Unreal Engine is its high-fidelity graphics rendering, which enables us to create visually stunning environments, characters, and special effects that captivate players and enhance immersion. The engine's sophisticated lighting, shading, and particle systems allow for dynamic and realistic visuals, bringing our game world to life in breathtaking detail.

Moreover, Unreal Engine's Blueprint visual scripting system empowers us to prototype, iterate, and implement gameplay mechanics and features with ease, without the need for extensive coding knowledge. This no-code approach to game development streamlines our workflow and facilitates rapid iteration, allowing us to experiment with different ideas and refine our game mechanics until they meet our vision.Additionally, Unreal Engine's comprehensive asset library, marketplace, and community support provide us with access to a wealth of resources, from pre-made assets and plugins to tutorials and forums. This enables us to leverage the collective knowledge and expertise of the Unreal Engine community, accelerating our learning curve and overcoming challenges more efficiently.

Overall, Unreal Engine plays a vital role in helping us realize our project's vision by providing a robust and feature-rich development platform that empowers us to create a captivating and immersive third-person shooter game that resonates with players. Its powerful tools, intuitive workflow, and extensive support ecosystem are essential in driving the success of our project and pushing the boundaries of game development.

Figure 3.2: Unreal Engine Logo

CHAPTER – IV

SYSTEM DESIGN

4.1 Proposed System

Creating a third-person shooter (TPS) game involves a comprehensive process encompassing various stages from initial setup to the final polish. This extended methodology delves deeper into each aspect, providing detailed steps and considerations to ensure the successful development of the game.

Installation and Setup: Installing the necessary tools and assets is the first step in starting the development process. This involves:

- Installing the Epic Games Launcher and Unreal Engine, ensuring compatibility with the target platform(s).
- Creating a new project in Unreal Engine using the appropriate template and settings.
- Importing required asset packs such as the Animator Starter Pack, Weapon Bundle, and Stylized Character Kit, and ensuring they are properly integrated into the project.

Project Setup and Character Implementation: Setting up the project environment and implementing the player character are crucial for laying the foundation of the game. This involves:

- Configuring the project directory structure for organization and version control to facilitate collaboration among team members.
- Setting up player character movement, input controls, and camera settings to ensure smooth and responsive gameplay.
- Implementing character animations for various states including locomotion, idle, aiming, shooting, and reloading, utilizing animation blueprints for seamless integration.
- Integrating character customization options such as outfit changes and weapon attachments using provided meshes and materials, and incorporating a system for players to customize their characters as desired.

Gameplay Mechanics Implementation: Developing the core gameplay mechanics is essential for creating an engaging and immersive player experience. This involves:

- Developing player movement mechanics, including walking, running and jumping, with considerations for terrain traversal and obstacle navigation.
- Implementing shooting mechanics with accurate hit detection, bullet trajectory calculations, and feedback systems to provide visual and audio cues for successful hits.
- Integrating weapon-handling mechanics such as recoil, bullet spread, and implementing animations and sound effects to enhance realism and immersion

- Creating a dynamic cover system that allows players to take cover behind objects and surfaces during combat engagements, with intelligent AI behavior to respond to player actions.

User Interface Development: Designing and implementing a user-friendly HUD (Heads-Up Display) is crucial for providing players with important information and enhancing their overall gameplay experience. This involves:

- Designing HUD elements such as health bars, ammunition counters to convey essential information to the player at a glance.
- Implementing team-based UI elements to differentiate between friendly and enemy entities, including indicators for teammate locations, enemy positions, and objectives.
- Integrating audio-visual feedback cues such as hit markers, damage indicators, and sound effects to provide additional context and immersion during gameplay.
- Optimizing the UI layout and visuals for readability and accessibility across different screen resolutions and aspect ratios.

AI Implementation: Developing AI systems for enemy characters is essential for creating challenging and dynamic gameplay experiences. This involves:

- Implementing behavior trees and AI logic to simulate realistic enemy behavior, including patrolling, searching, attacking, and retreating.
- Utilizing sensory perception systems to enable enemies to detect player presence through sight, and other stimuli, with varying levels of awareness and reaction times.
- Incorporating team-based AI coordination to simulate squad tactics and dynamic enemy encounters, including flanking maneuvers, suppressive fire, and coordinated attacks.
- Balancing AI difficulty levels to provide a challenging yet fair experience for players of different skill levels, with options for adjusting AI behavior and aggression based on player feedback.

Level Design and Environment Setup: Designing visually stunning and immersive environments is essential for creating an engaging and memorable gameplay experience. This involves:

- Designing levels that reflect the theme and setting of the game, with attention to detail in environmental storytelling, atmosphere, and ambience.
- Utilizing level design principles such as sightlines, cover placement, and player flow to create engaging gameplay spaces that encourage exploration and strategic thinking.
- Populating the environment with interactive objects, obstacles, and strategic points of interest, including destructible elements and interactive props that react to player actions.

- Optimizing level performance through efficient use of resources, LOD (Level of Detail) systems, and occlusion culling techniques to maintain smooth gameplay and frame rates across different hardware configurations.

Testing, Iteration, and Optimization: Conducting thorough testing and iteration cycles is essential for refining gameplay mechanics, balancing difficulty, and polishing the overall player experience. This involves:

- Conducting playtesting sessions with a diverse group of testers to gather feedback on gameplay mechanics, level design, AI behavior, and overall user experience.
- Iterating on design iterations based on user feedback and market trends to continuously improve the game's quality and appeal, with a focus on addressing player pain points and enhancing gameplay depth.
- Optimizing game performance through profiling, asset optimization, and code refactoring to ensure smooth gameplay and optimal frame rates across various hardware configurations.
- Balancing gameplay mechanics, AI difficulty, and level design elements to provide a satisfying and rewarding experience for players of all skill levels, with options for adjusting difficulty settings and gameplay parameters based on player feedback.

Conclusion: By following this extended methodology, developers can create a compelling and immersive third-person shooter game that meets the specified requirements and exceeds player expectations. Attention to detail, iterative design processes, and a commitment to quality are essential elements in delivering a polished and enjoyable gaming experience. With careful planning, execution, and iteration, developers can create a TPS game that resonates with players and stands out in the competitive gaming market.

4.2 Flowchart

A flowchart is a visual representation of the sequence of steps and decisions needed to perform a process. Each step in the sequence is noted within a diagram shape. Steps are linked by connecting lines and directional arrows. Figure 4.1 is hence depicting the simplified flowchart of the application.

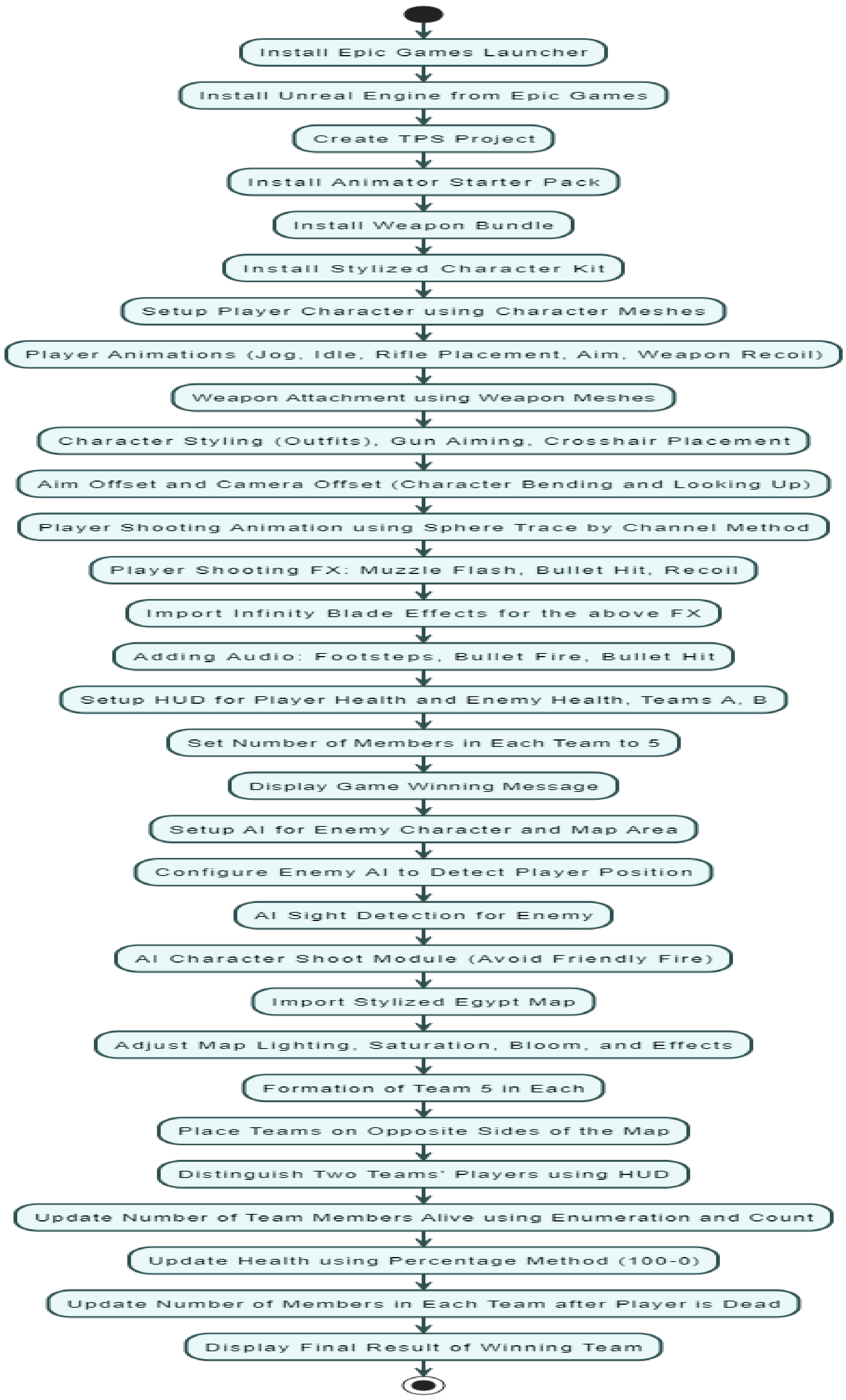
Install Epic Games Launcher
Install Unreal Engine from Epic Games
Create TPS Project
Install Animator Starter Pack
Install Weapon Bundle
Install Stylized Character Kit
Setup Player Character using Character Meshes
Player Animations (Jog, Idle, Rifle Placement, Aim, Weapon Recoil)
Weapon Attachment using Weapon Meshes
Character Styling (Outfits), Gun Aiming, Crosshair Placement
Aim Offset and Camera Offset (Character Bending and Looking Up)
Player Shooting Animation using Sphere Trace by Channel Method
Player Shooting FX: Muzzle Flash, Bullet Hit, Recoil
Import Infinity Blade Effects for the above FX
Adding Audio: Footsteps, Bullet Fire, Bullet Hit
Setup HUD for Player Health and Enemy Health, Teams A, B
Set Number of Members in Each Team to 5
Display Game Winning Message
Setup AI for Enemy Character and Map Area
Configure Enemy AI to Detect Player Position
AI Sight Detection for Enemy
AI Character Shoot Module (Avoid Friendly Fire)
Import Stylized Egypt Map
Adjust Map Lighting, Saturation, Bloom, and Effects
Formation of Team 5 in Each
Place Teams on Opposite Sides of the Map
Distinguish Two Teams' Players using HUD
Update Number of Team Members Alive using Enumeration and Count
Update Health using Percentage Method (100-0)
Update Number of Members in Each Team after Player is Dead
Display Final Result of Winning Team

Figure 4.1: Flowchart of the Game

CHAPTER – V

IMPLEMENTATION

5.1 Module Description

1 **Module 1: Installation of Epic Games and Unreal Engine**

- Installing Epic Games Launcher and Unreal Engine is the initial step towards creating a Third-Person Shooter (TPS) game. Epic Games Launcher serves as the hub for managing game development tools, assets, and projects. To begin, visit the official Epic Games website and download the Epic Games Launcher installer. Once downloaded, run the installer and follow the on-screen instructions to complete the installation process.
- After installing the launcher, you will need to create or sign in to your Epic Games account. This account will grant you access to the Unreal Engine and other Epic Games services. Once logged in, navigate to the "Unreal Engine" tab within the launcher. Here, you will find options to install different versions of Unreal Engine.
- Select the desired version of Unreal Engine and click the "Install" button. The launcher will begin downloading and installing Unreal Engine on your system. Depending on your internet connection speed and system specifications, this process may take some time.
- Once the installation is complete, you will have access to Unreal Engine's powerful suite of game development tools. Launch Unreal Engine from the Epic Games Launcher to start creating your TPS project.

2 Module 2: Creating a TPS Project

- Creating a TPS project in Unreal Engine involves several key steps to set up the project environment and configure essential settings. To begin, launch Unreal Engine and select "New Project" from the main menu. In the project browser window, choose the "Third Person" template as the project's starting point.
- Next, specify the project's location on your system and provide a name for the project. You can also choose additional settings such as project quality settings and starter content inclusion. Once all settings are configured, click the "Create Project" button to generate the project files.
- Upon creating the project, Unreal Engine will initialize the project environment and open the project in the Unreal Editor. Here, you will have access to various tools and features for designing, building, and testing your TPS game.

3 Module 3: Installing Assets from Epic Games Store

- The Epic Games Store offers a wide range of assets, including character models, animations, weapons, effects, and more, which are essential for developing a TPS game. To install assets from the Epic Games Store, open the Epic Games Launcher and navigate to the "Store" tab.
- Here, you can browse through available asset packs and marketplace content. Search for the Animator Starter Pack, Weapon Bundle, and Stylized Character Kit, which contain assets suitable for your TPS project. Once found, click on each asset pack to view more details.
- To install an asset pack, click the "Purchase" or "Install" button, depending on whether the asset is free or requires a purchase. Follow the on-screen instructions to complete the installation process. Once installed, the assets will be available for use within your Unreal Engine project.

4 Module 4: Setting Up Player Character

- Setting up the player character involves integrating character meshes, animations, and controls into the TPS project. Start by importing character meshes from the installed asset packs into the project. This typically involves importing skeletal mesh assets representing the player character's model.
- Once imported, configure the character's animations for various states such as idle, jogging, aiming, and shooting. You can use animation blueprints to blend between different animation states smoothly. Additionally, set up character controls to allow player movement, camera manipulation, and interaction with the environment.
- Implementing character controls may involve setting up input bindings, character movement components, and collision handling. You will also need to configure camera perspectives to provide a suitable view of the player character from behind.

5 Module 5: Weapon Attachment and Styling

- Integrating weapon meshes from the installed asset packs involves importing weapon models and attaching them to the player character's hand socket. This process typically requires creating socket assets on the character's skeleton and assigning the weapon meshes to these sockets.
- Once attached, implement weapon aiming and shooting mechanics. This may involve setting up animations for weapon aiming and firing, as well as implementing logic to handle weapon recoil, bullet trajectory, and hit detection.

- In addition to functionality, focus on weapon styling to enhance visual appeal. This includes applying textures, materials, and visual effects to the weapon models. You can also implement different weapon outfits and customization options to provide players with a variety of choices.

6 Module 6: Aim Offset and Camera Offset

- Aim offset is a technique used to dynamically adjust the character's upper body animations based on the direction of the player's aim. Implementing aim offset involves creating animation blendspaces or animation notifies that smoothly transition between different aiming poses.
- Camera offset adjustments enhance immersion by aligning the camera with the character's gaze during aiming and shooting actions. This involves tweaking camera settings such as position, rotation, and field of view to provide a clear view of the target while maintaining the character's perspective.
- Both aim offset and camera offset contribute to responsive and realistic aiming mechanics, enhancing the player's control and immersion during combat encounters.

7 Module 7: Shooting Effects and Audio

- Shooting effects and audio play a crucial role in conveying the impact and intensity of combat in a TPS game. Implementing shooting effects involves creating particle effects for muzzle flashes, bullet impacts, and weapon recoil.
- Particle effects for muzzle flashes should dynamically emit from the weapon's barrel during firing, while bullet impact effects should spawn at the point of impact on surfaces or enemy characters. Implementing weapon recoil involves animating the weapon model and applying forces to simulate recoil motion.
- Integrating audio effects for footsteps, gunfire, and bullet impacts adds another layer of immersion to the gameplay experience. Footstep sounds should vary based on the character's movement speed and surface type, while gunfire sounds should reflect the type of weapon being fired.
- Bullet impact sounds should also vary depending on the surface material and the force of the impact. By combining visual and audio feedback, shooting effects enhance the player's engagement and provide valuable feedback during combat encounters.

8 Module 8: HUD Implementation

- The Heads-Up Display (HUD) provides essential information to the player, including health status, ammunition count, objective markers, and more. Designing and implementing the HUD involves creating UI elements that convey this information clearly and intuitively.
- Start by designing the layout and visual style of the HUD elements, taking into account factors such as screen space, readability, and aesthetics. Common HUD elements include health bars, ammo counters, minimaps, and status indicators.
- Once the design is finalized, implement the HUD elements using Unreal Engine's UMG (Unreal Motion Graphics) system. This involves creating widget blueprints for each HUD element and binding them to relevant game data such as player health and ammo count.
- Ensure that the HUD elements update dynamically based on gameplay events such as damage taken, ammo expended, and objective progress. Additionally, incorporate visual feedback such as animations or color changes to indicate important events or status changes.

9 Module 9: AI Enemy Setup

- Implementing AI for enemy characters involves creating behavior trees, AI perception components, and navigation meshes to enable intelligent enemy behavior. Start by designing behavior trees that define enemy actions and decision-making logic.
- Behavior trees typically consist of nodes representing actions, conditions, and composite tasks such as sequences and selectors. Define tasks such as patrolling, pursuing, attacking, and taking cover to create varied and challenging enemy behavior.
- Next, set up AI perception components to enable enemies to detect and respond to player presence. This involves configuring senses such as sight and hearing to detect stimuli such as the player character, gunfire, and footsteps.
- Finally, generate navigation meshes to enable pathfinding for enemy characters. Navigation meshes define walkable areas within the game world and allow enemies to navigate complex environments intelligently.
- By combining behavior trees, AI perception, and navigation meshes, you can create lifelike enemy behavior that challenges players and enhances the overall gameplay experience.

10 Module 10: Map Design and Lighting

- The map serves as the backdrop for gameplay in a TPS game, providing environments for combat encounters, exploration, and objective-based gameplay. Importing a stylized Egypt map involves acquiring or creating a suitable map asset and integrating it into the Unreal Engine project.

- Choose a map asset that fits the aesthetic and thematic requirements of the game, such as an Egyptian-themed environment with desert landscapes, ancient ruins, and architectural elements. Ensure that the map asset is compatible with Unreal Engine and meets performance requirements.
- Once imported, adjust lighting settings to enhance the visual quality and atmosphere of the map. Experiment with different lighting sources, such as directional lights, point lights, and skylights, to achieve the desired mood and ambiance.
- Fine-tune settings such as light intensity, color temperature, and shadows to create realistic lighting conditions that complement the map's theme. Pay attention to details such as light direction, shadow casting, and global illumination to enhance immersion and visual fidelity.
- Additionally, adjust settings such as saturation, bloom, and post-processing effects to further enhance the map's visual appeal and stylization. By carefully designing and lighting the map, you can create an immersive and engaging environment for players to explore and experience.

11 Module 11: Team Formation and Placement

- Configuring teams involves organizing players into two opposing teams, each with five members. Start by defining team data structures to store information such as team ID, member list, and team-specific properties.
- Create a mechanism for players to join a team, either manually or automatically based on matchmaking rules. Ensure that teams are balanced in terms of skill level, experience, and other factors to promote fair and competitive gameplay.
- Once teams are formed, place them on opposite sides of the map to create a balanced playing field. Consider factors such as map layout, cover placement, and strategic objectives when determining team placement.
- Ensure that team placement provides equal opportunities for both teams to engage in combat, complete objectives, and secure victory. By carefully balancing team formation and placement, you can create a dynamic and competitive multiplayer experience for players.

12 Module 12: Team Management and Health Update

- Implementing team management involves tracking the number of team members alive and controlling team behavior using enumeration and count methods. Start by defining data structures to represent teams, including properties such as team ID, member list, and current status.
- Create mechanisms for adding and removing players from teams, updating team information, and handling team-related events such as player respawns and eliminations. Ensure that team management logic is robust, efficient, and scalable to accommodate varying team sizes and configurations.
- Track player health using percentage-based methods to accurately reflect damage taken and healing received. Update player health dynamically based on gameplay events such as damage from enemy attacks, healing from health pickups, and other modifiers.
- Incorporate visual feedback such as health bars and status indicators to communicate player health to the player. Ensure that health updates are synchronized across all players in the game to maintain consistency and fairness.

13 Module 13: Final Result Display

- Displaying the final result involves determining the winning team based on gameplay objectives such as eliminating the opposing team or completing specific objectives. Start by defining victory conditions and criteria for determining the winning team.
- Monitor gameplay events and track progress towards victory conditions throughout the match. Update the game state accordingly and determine the winning team once victory conditions are met.
- Display a game-winning message to congratulate the winning team and conclude the match. Provide feedback to all players indicating the outcome of the match and their contributions to the team's success.
- Ensure that final result display logic is accurate, reliable, and free from exploits or loopholes. Handle edge cases and unexpected scenarios gracefully to maintain a positive player experience.

By implementing a robust final result display system, you can provide closure to matches, celebrate player achievements, and encourage continued engagement with the game.

5.2 Blueprint Implementation

Third Person Character Blueprint

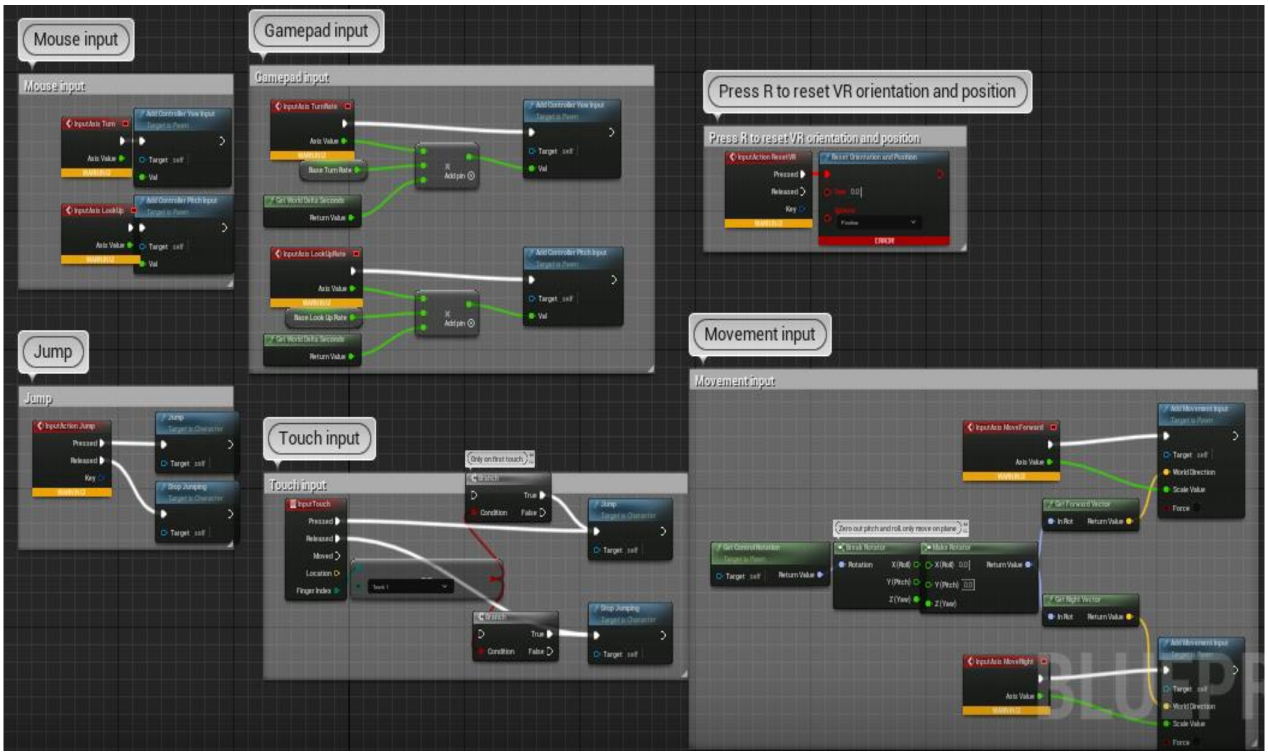

Figure 5.1: Third Person Character Blueprint

Third Person Character Blueprint is the cornerstone of controlling a character within a game environment, likely implemented using Unreal Engine. Let's delve into its components:

- Mouse Input Handling: Responsible for interpreting mouse movements to rotate the character. The character's view direction changes correspondingly to mouse movements, offering fluid control.
- Gamepad Input Handling: Similar to mouse input but tailored for gamepad controls. Utilizes analog sticks on the gamepad to adjust the character's orientation, ensuring intuitive gameplay for controller users.
- Jump Action: Controls the character's jumping behavior. Activated when the player presses the jump button, enabling the character to leap.
- Movement Input Processing: Manages character movement based on player input. Handles forward, backward, left, and right movements, translating player input into character motion for seamless navigation within the game world.
- Touch Input Support: Indicates compatibility with touch-based controls, typically used for mobile devices. Allows players to control the character via touch gestures, providing an alternative input method for mobile gaming.

Overall, this blueprint serves as the fundamental framework for creating a responsive and dynamic third-person character experience.

Add Input Mapping

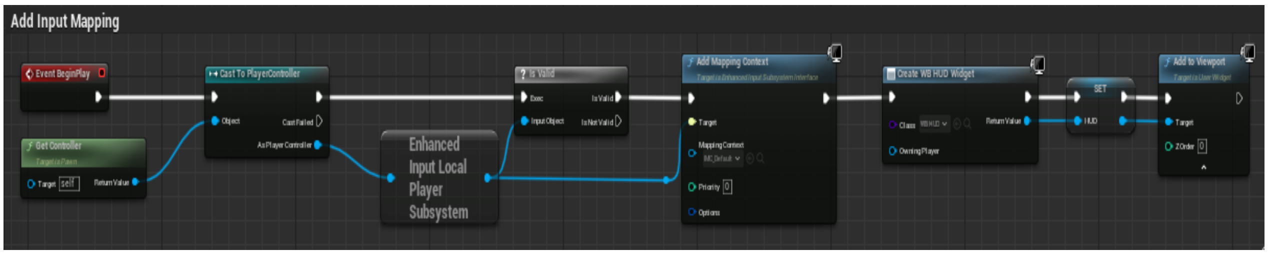

Figure 5.2: Add Input Mapping Blueprint

The "Add Input Mapping Blueprint" played a pivotal role in game development, particularly within the Unreal Engine ecosystem. Let's break down its components:

- Event BeginPlay: The blueprint initiated with an "Event BeginPlay" node. This event typically triggered at the beginning of gameplay, initializing crucial elements and setting the stage for subsequent actions.
- Get Controller and Cast To PlayerController: It retrieved the player's controller.The "Cast To PlayerController" node ensured that the controller being retrieved was indeed a player controller, which was essential for handling input from the player.
- Enhanced Input Local Player Subsystem: This node likely pertained to an advanced input system. Enhanced input systems offered more flexibility and customization for managing player controls, ensuring efficient processing of player input.
- Add Mapping Context: Specific control mappings were added in this section for player inputs.These mappings established the relationship between different input actions (such as keyboard keys, mouse clicks, or gamepad buttons) and their corresponding in-game actions. For instance, associating the "Space Bar" key with the "Jump" action within the game.
- Create HUD Widget and Add To Viewport: The blueprint created a Heads-Up Display (HUD) widget. HUD widgets displayed critical information during gameplay, such as health, score, or ammunition. The "Add To Viewport" node ensured that the HUD widget was displayed on the player's screen during gameplay.

In summary, this blueprint facilitated the setup of input mappings, linked them to the player's controller, and integrated them with the game's Heads-Up Display. It served as a foundational element in crafting an immersive gaming experience, enhancing player interaction and feedback.

Camera Input

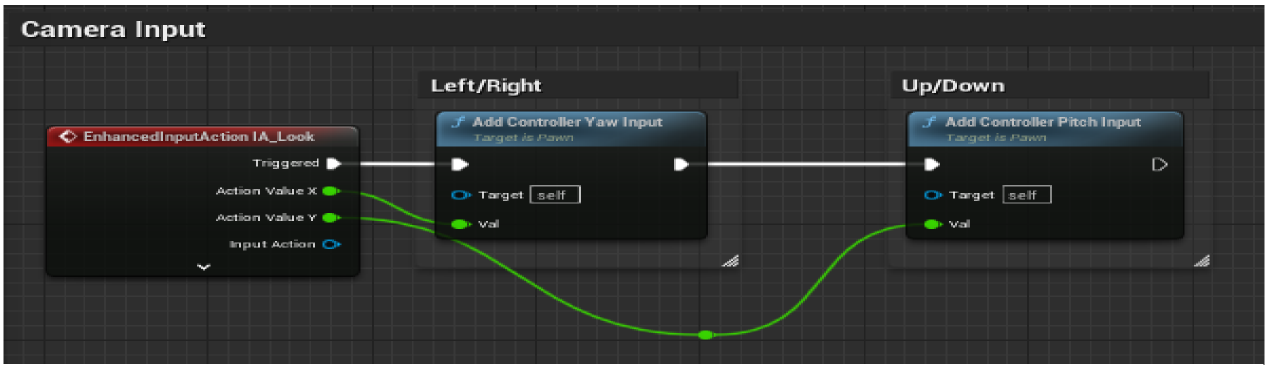

Figure 5.3: Camera Input Blueprint

The Camera Input Blueprint served as a fundamental component in game development, particularly within the Unreal Engine ecosystem. Let's delve into its details:

- Enhanced Input Action 1A_Look: This blueprint commenced with an "Event BeginPlay" node, typically triggered at the start of gameplay, initializing essential elements and setting the stage for subsequent actions. The "Enhanced Input Action 1A_Look" appeared to be activated by specific input actions related to camera control.
- Horizontal (Left/Right) and Vertical (Up/Down) Camera Movement: The two blocks on the right were responsible for handling camera movement. The "Add Controller Yaw Input" node managed horizontal rotation (left/right). The "Add Controller Pitch Input" node controlled vertical rotation (up/down). Both of these nodes affected the player's pawn or character in the game. The presence of the "Target is Pawn" indicated that these actions modified the camera view for the player.
- Flow of Data: Green lines with circular nodes connected these blocks, illustrating the flow of data or commands between them. The background, with its grid pattern, resembled typical blueprint or graph editors used in game development software.

In summary, this blueprint orchestrated camera input, allowing players to pan left/right and move up/down, thereby contributing significantly to creating an immersive gaming experience!

Movement Input

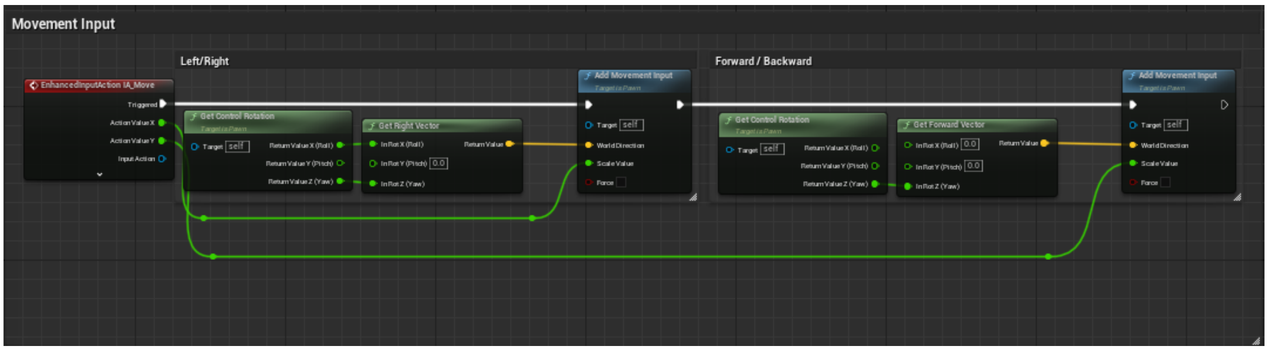

Figure 5.4: Movement Input Blueprint

The Movement Input Blueprint played a crucial role in game development, particularly within the Unreal Engine environment. Let's dissect its components:

- Left/Right Movement: This section of the blueprint managed left/right movement. It began with an "EnhancedInputAction" node triggered by specific input actions, likely corresponding to player controls. Essential information was obtained from nodes such as "Get Control Rotation" and "Get Right Vector". These inputs were then connected to an "Add Movement Input" node, responsible for translating user input into character movement. The presence of the "Target is Self" indicated that the blueprint directly affected the character itself. A "Scale Value" factor of 1 ensured consistent movement.
- Forward/Backward Movement: Similar to left/right movement, this section handled forward/backward motion. It also utilized the "Get Forward Vector" node to obtain necessary directional information. Once again, an "Add Movement Input" node triggered character movement based on user input. The "Target is Self" ensured that the character responded appropriately to the input.

In summary, this blueprint orchestrated character movement in both horizontal and vertical directions, allowing players to explore game worlds seamlessly!

Jump Input

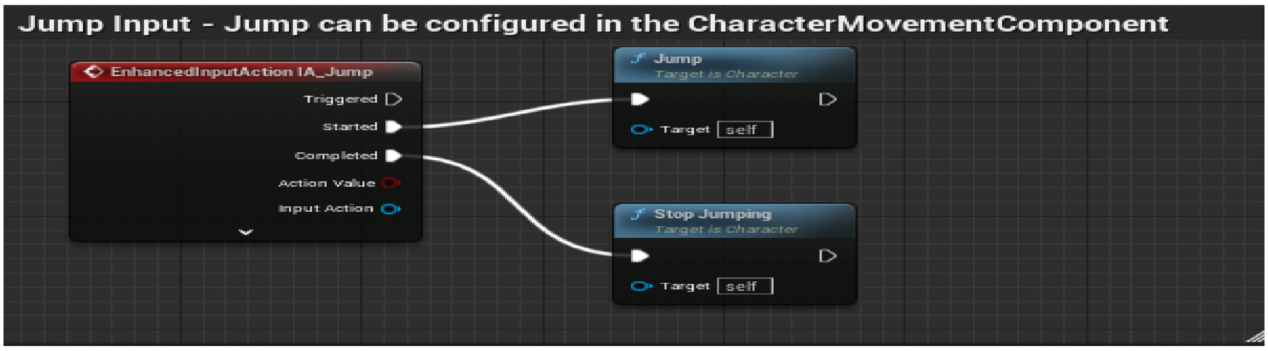

Figure 5.5: Jump Input Blueprint

The Jump Input Blueprint is an integral aspect of game development, especially within the CharacterMovementComponent in Unreal Engine. Let's dissect its components:

- EnhancedInputAction A Jump: This blueprint manages the initiation of a jump action. It features three outputs: Triggered, Started, and Completed. The Action Value and Input Action are also specified, providing additional context. When the player triggers a jump input, this blueprint processes it accordingly.
- Jump and Stop Jumping: Two boxes on the right represent the jump-related actions. Both boxes are tagged with "Target is Character," indicating their association with character behavior.

- The “Jump” box: Connects to both the “Triggered” and “Started” outputs of the A Jump box. Initiates the jump action when triggered by player input.
- The “Stop Jumping” box: Connects to the “Completed” output of the A Jump box. Halts the ongoing jump action when triggered, providing control over the character's mid-air movement.

In summary, this blueprint configures how the character responds to jump inputs, facilitating dynamic and responsive gameplay!

Aim System

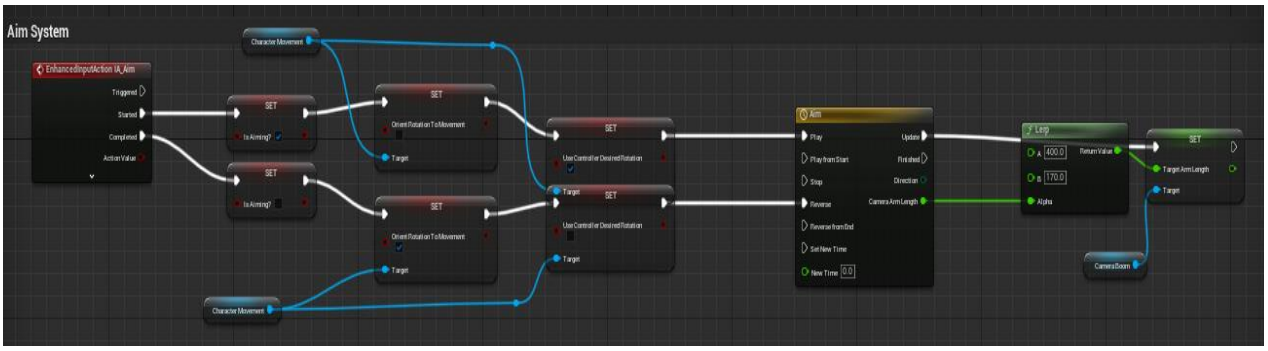

Figure 5.6: Aim System Blueprint

The Aim System Blueprint is a pivotal component in game development, particularly utilized for creating an aiming system within a game. Let's dissect its components:

- EnhancedInputAction_Aim: This node responds to specific input actions related to aiming, likely triggered by player input when aiming is initiated. It features outputs such as "Triggered" and "Pressed," indicating different stages of the aiming process.
- Orient Rotation To Movement: This function is responsible for orienting characters or objects toward a specific direction during movement. It ensures that the character’s rotation aligns with their movement direction, enhancing visual coherence and realism. The “Target” node specifies which object’s rotation is affected, allowing for precise control over orientation adjustments.
- Dynamic Camera Adjustment: The “Update” node is connected to mathematical operations, likely used to calculate the desired camera distance based on certain conditions. The “SET Target Arm Length” node dynamically adjusts the camera’s position, ensuring optimal positioning to follow the character appropriately during aiming.

In summary, this blueprint orchestrates aiming mechanics, camera adjustments, and character orientation, all of which contribute to a seamless and immersive gaming experience!

Health System

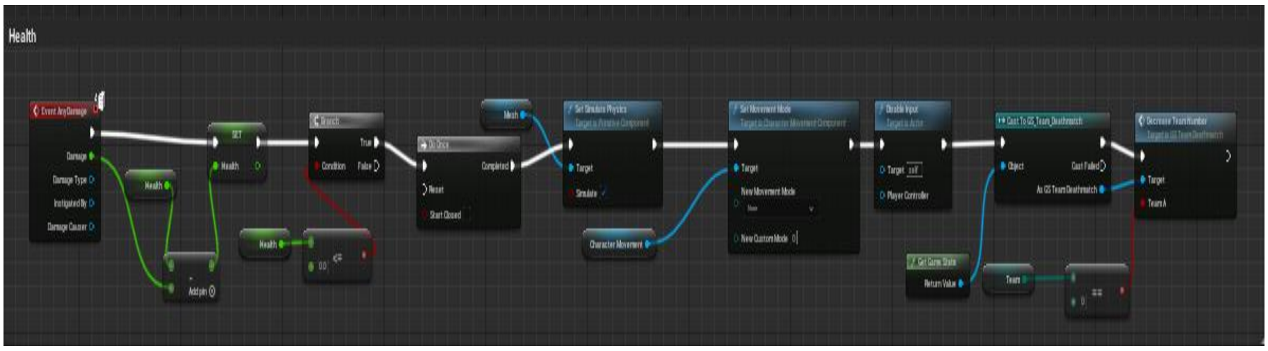

Figure 5.7: Health System Blueprint

The Health Blueprint is indeed a crucial component in game development, particularly for managing a character’s health. Here are its key features:

- Event AnyDamage: This node triggers when the character receives damage, serving as the starting point for health-related logic.
- Health Calculation: The blueprint calculates and updates the character’s health based on the damage received. Conditional checks are employed to determine if the health is less than or equal to 0.
- Health Status: If the health is greater than 0, the game continues and the player is still Alive. If the health reaches 0 or below, the “Death” node is triggered. The “Death” node likely handles character death processes, such as disabling movement and triggering relevant events.

In summary, this blueprint is essential for managing a character’s health, ensuring that the game responds appropriately to damage and character death events, thereby contributing to an immersive and dynamic gameplay experience!

Gun Firing and Target system

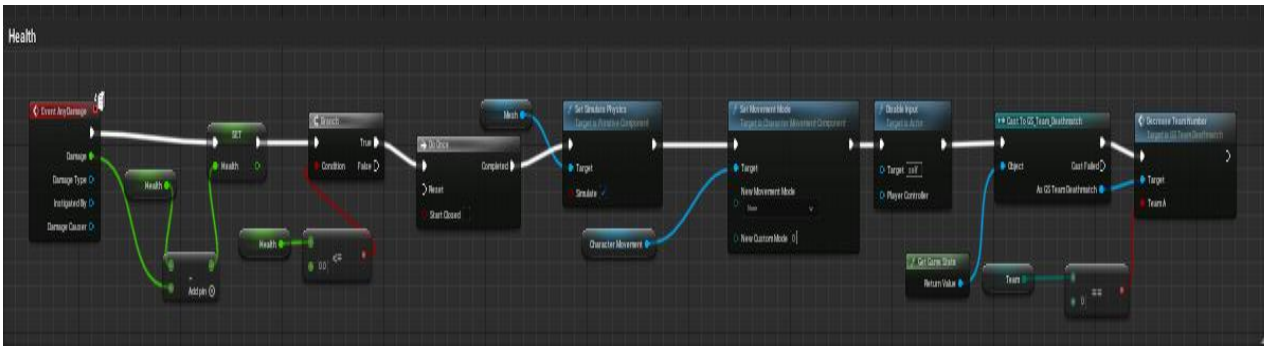

Figure 5.8: Gun Firing and target system Blueprint

The Gun Firing and Target System Blueprint is indeed a visual representation of complex logic flow within a game development environment. Let's dissect its key features:

i. Purpose: The blueprint orchestrates the interaction between a gun (firearm) and a target system, handling processes related to aiming, firing, and hitting targets.

ii. Components:

- Input Trigger (EnhancedInputAction_Aim): This node responds to input actions related to aiming (e.g., right-click or aiming button), initiating the aiming process.
- Health and Damage Calculation: The blueprint may involve health and damage calculations; when the gun fires, it affects the target's health.
- Targeting Logic: Determines how the gun aims at the target, potentially involving adjustments to the gun's orientation based on player input.
- Firing Control Module: Handles firing actions; when the gun is aimed correctly, it triggers the firing mechanism.

iii. Complex Interconnections: The intricate network of nodes represents dependencies and interactions among various game elements, ensuring accurate targeting and firing.

In summary, this blueprint integrates aiming, firing, and target tracking, thereby enhancing the gameplay experience with engaging and dynamic mechanics!

AI Blueprint

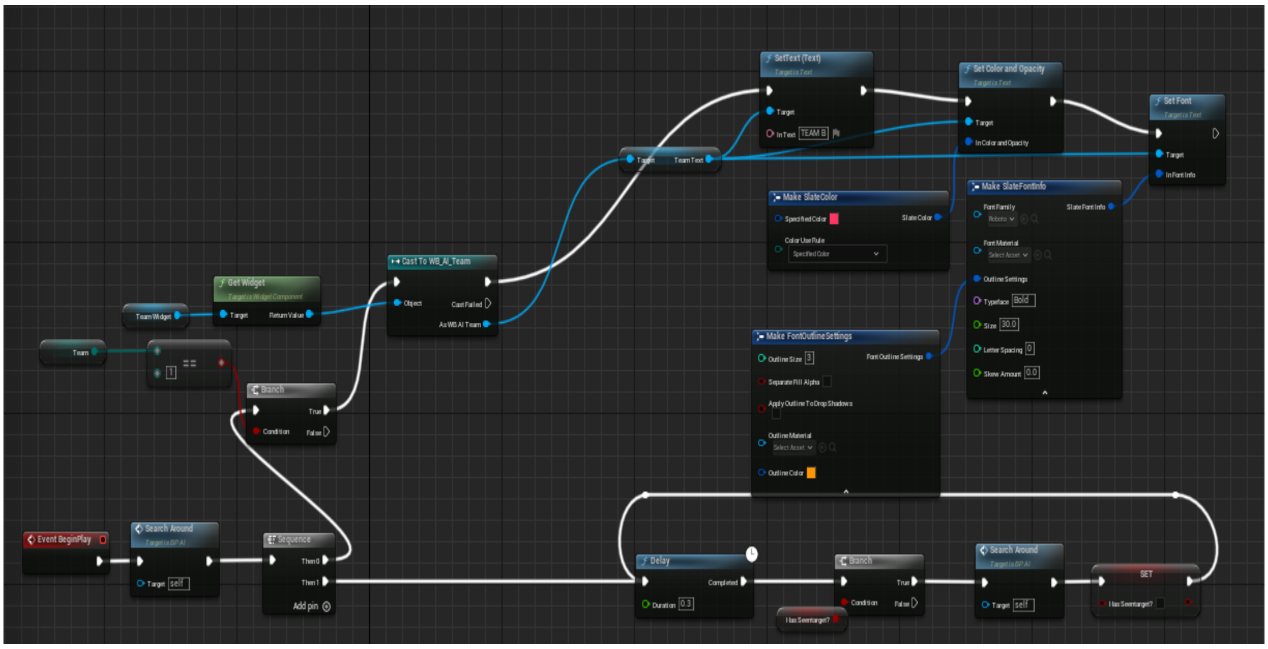

Figure 5.9: AI Blueprint

The Blueprint AI for a third-person shooting game is indeed a critical component that governs the behavior of non-player characters (NPCs) within the game. Let's break down its key features:

- Event BeginPlay: This node initiates the blueprint logic when the game starts, setting the stage for subsequent actions.

- Get Player Character and Cast To ThirdPersonCharacter: These nodes retrieve information about the player character. The “Cast To ThirdPersonCharacter” ensures that the retrieved character is of the correct type (third-person character).
- Branch Node: The branch node acts as a decision point based on certain conditions. If the condition is true (e.g., the player is in sight), one path is taken; otherwise, another path is followed.
- Set Target, Aim At, and Fire at Target: These nodes are related to targeting and shooting mechanics. “Set Target” likely designates the NPC’s target (e.g., the player). “Aim At” ensures that the NPC aims at the designated target. “Fire at Target” triggers the shooting action.

In summary, this blueprint orchestrates NPC behavior, including aiming, shooting, and responding to the player’s presence. It’s essential for creating engaging and challenging gameplay!

Search around Blueprint

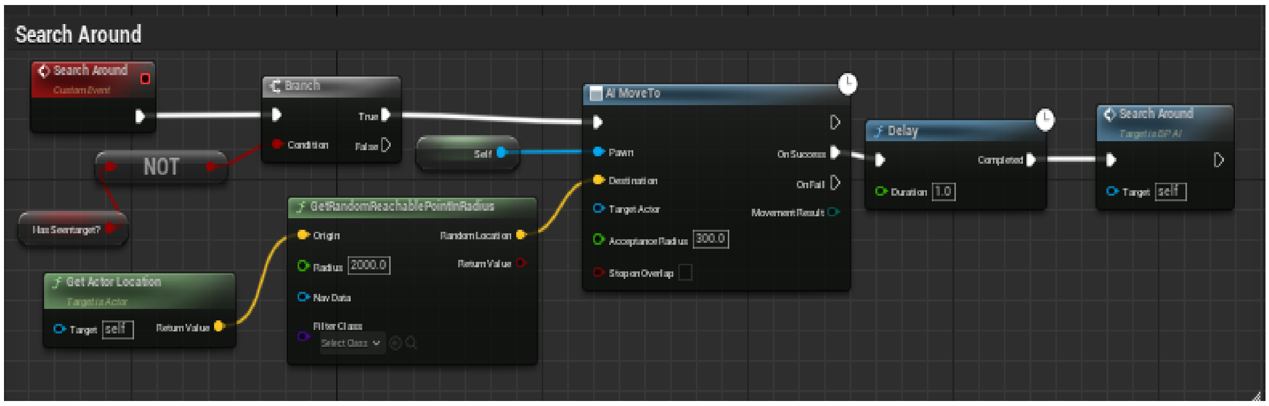

Figure 5.10: Search around Blueprint

The Search Around Blueprint likely represents the logic flow within a game development environment, specifically outlining the behavior of an entity, such as an AI character, when searching its surroundings. Let's dissect its key components:

- Event BeginPlay: This node initiates the blueprint logic when the game starts, setting the stage for subsequent actions related to searching behavior.
- Search Logic: The blueprint may involve conditions (e.g., “Branch”) to determine whether the entity should initiate the search. If specific criteria are met (e.g., “True”), the search process begins. Nodes like “AI MoveTo” suggest that the entity navigates to specific locations during the search, indicating its exploration behavior.
- Delays and Completion: The “Delay” node introduces a time delay, possibly simulating the time taken for searching or adding pacing to the behavior. The “Completed” node likely signifies the end of the search process, indicating that the entity has finished searching its surroundings.

In summary, this blueprint orchestrates how an entity explores its environment, providing a foundation for dynamic and interactive gameplay!

See Pawn (Pawn Sensing)

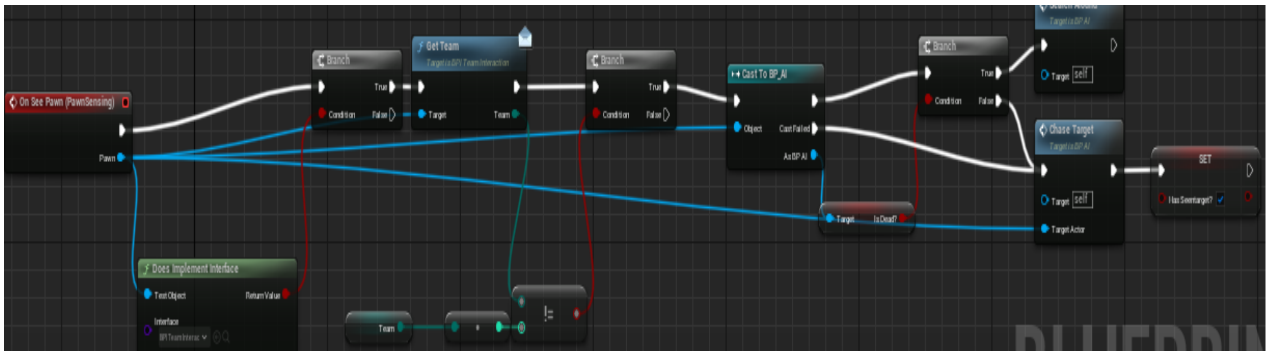

Figure 5.11: See Pawn (Pawn Sensing) Blueprint

The See Pawn (Pawn Sensing) Blueprint is indeed a critical component in game development, particularly within the context of AI behavior. Let's delve into its key features:

- On See Pawn (PawnSensing): This node triggers when the pawn (an AI character) senses or detects another pawn (e.g., the player or an enemy), serving as the starting point for AI reactions based on sensory input.
- Branch Nodes: The blueprint likely involves decision-making, with branch nodes evaluating conditions (e.g., distance, line of sight) to determine the appropriate action. For example: If the sensed pawn is within a certain range, the AI may engage in combat. If not, it may continue patrolling or change behavior.
- Casting and Target Selection: The “Cast To BP_AI” node suggests that the blueprint is casting to a specific AI class (e.g., a custom AI blueprint). The “Choose Target” node likely selects the sensed pawn as the target, crucial for subsequent actions such as attacking or following.
- Property or Variable Setting: The final “SET” node likely modifies a property or variable related to the chosen target, such as updating the AI’s state (alerted, idle, etc.) based on the sensed pawn.

In summary, this blueprint enables AI characters to react to nearby pawns, enhancing the game’s realism and challenge!

Chase Target

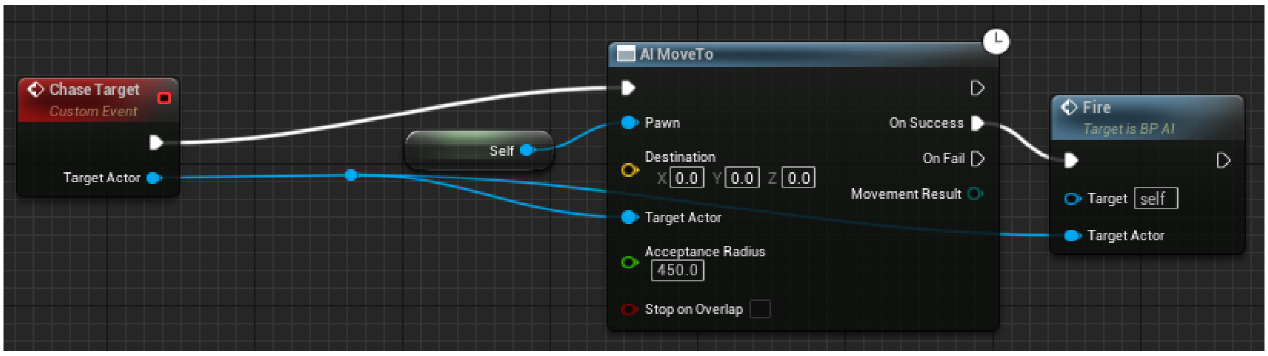

Figure 5.12: Chase Target Blueprint

The Chase Target Blueprint is indeed a critical component in game development, particularly within the context of AI behavior. Let's analyze its key features:

- On See Pawn (PawnSensing): This node triggers when the pawn (an AI character) senses or detects another pawn (e.g., the player or an enemy), initiating AI reactions based on sensory input.
- Branch Nodes: The blueprint likely involves decision-making, with branch nodes evaluating conditions (e.g., distance, line of sight) to determine the appropriate action. For instance: If the sensed pawn is within a certain range, the AI may engage in combat. If not, it may continue patrolling or change its behavior.
- Casting and Target Selection: The “Cast To BP_AI” node suggests that the blueprint is casting to a specific AI class (e.g., a custom AI blueprint). The “Choose Target” node likely selects the sensed pawn as the target, essential for subsequent actions such as attacking or following.

In summary, this blueprint empowers AI characters to react to nearby pawns, enhancing the game’s realism and challenge!

Health bar Update

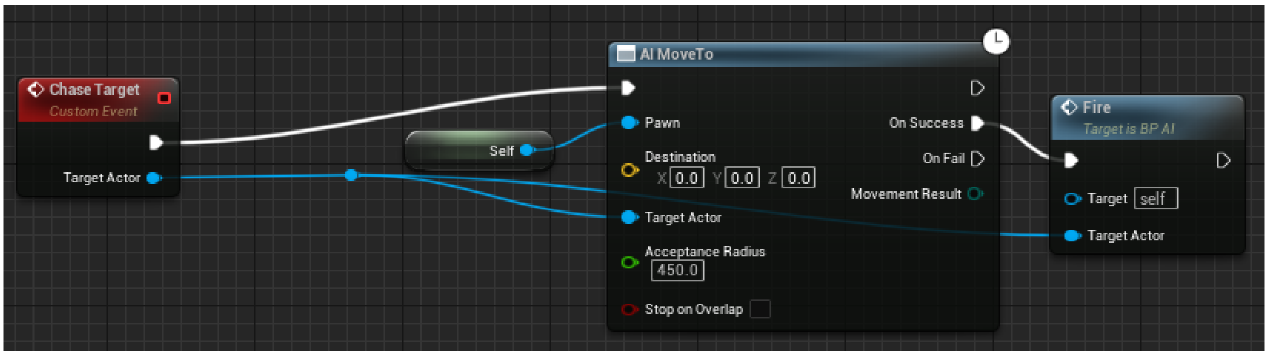

Figure 5.13: Health Bar Update Blueprint

The Health Bar Update Blueprint is indeed a critical component in game development, specifically responsible for managing and updating the visual representation of an AI character’s health. Let's examine its key features:

- Event Tick: This node triggers continuously during gameplay, ensuring that the health bar updates dynamically as the AI character’s health changes.
- Get Widget and Cast To WS_AI_Healthbar: The blueprint retrieves the health bar widget associated with the AI character. The “Cast To WS_AI_Healthbar” node ensures that the retrieved widget is of the correct type, such as a custom health bar widget.
- Set Percent and Add Pin: These nodes handle the visual representation of the health bar. “Set Percent” updates the fill percentage of the health bar based on the AI’s health value. “Add Pin” likely adjusts the health bar’s appearance, such as color or animation, based on additional factors like critical health status.
- World Rotation Adjustment: The blueprint involves getting the AI character’s location and calculating the look-at rotation. The “Set World Rotation” node may adjust the orientation of an object, possibly the health bar widget, within the game world to ensure it faces the player correctly.

In summary, this blueprint ensures that the AI character’s health bar accurately reflects its health status, enhancing player feedback and immersion!

Team Death match Blueprint

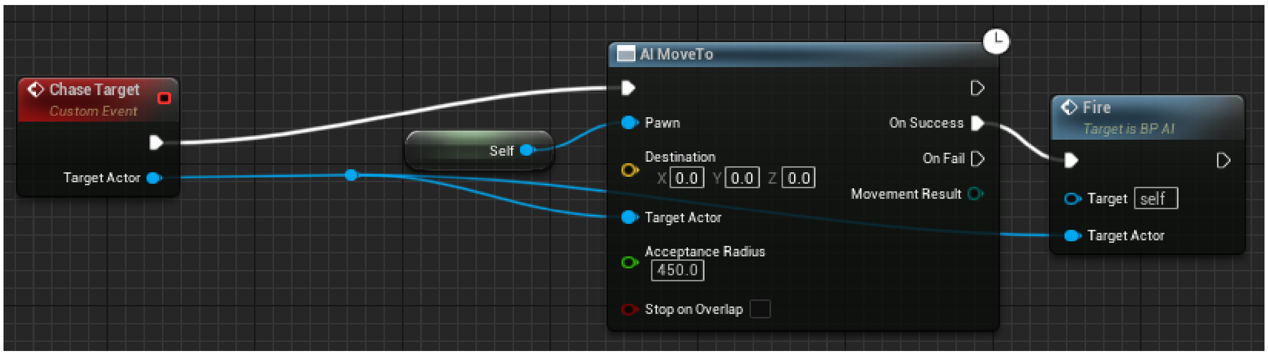

Figure 5.14: Team Death match Blueprint

The Team Deathmatch Blueprint is indeed a critical component in game development, specifically tailored for team-based gameplay. Let's delve into its key features:

- Event BeginPlay: This node triggers at the start of the game, initializing essential elements, including UI widgets, to set the stage for gameplay.

- UI Updates: The blueprint involves creating and updating UI widgets, such as team score displays. "Create Widget" nodes likely generate team-related UI elements, while "Add To Viewport" nodes ensure these widgets appear on the screen during gameplay.
- Team Tracking: The presence of the "Decrease Team Number" function suggests that the blueprint tracks the number of players or scores in each team. It likely adjusts team scores based on game events like kills or objectives, facilitating accurate representation of team performance.
- Determining the Winning Team: Conditional (branch) nodes check certain conditions to determine the winning team. If Team 1 achieves victory (e.g., higher score), the "Set Text" node displays "TEAM 1 WON!" on the UI. Similarly, if Team 2 wins, it displays "TEAM 2 WON!" instead.

In summary, this blueprint efficiently manages team-related logic; UI updates, and determines the winning team in a dynamic Team Deathmatch game, enhancing the overall gameplay experience!

Count of team Members

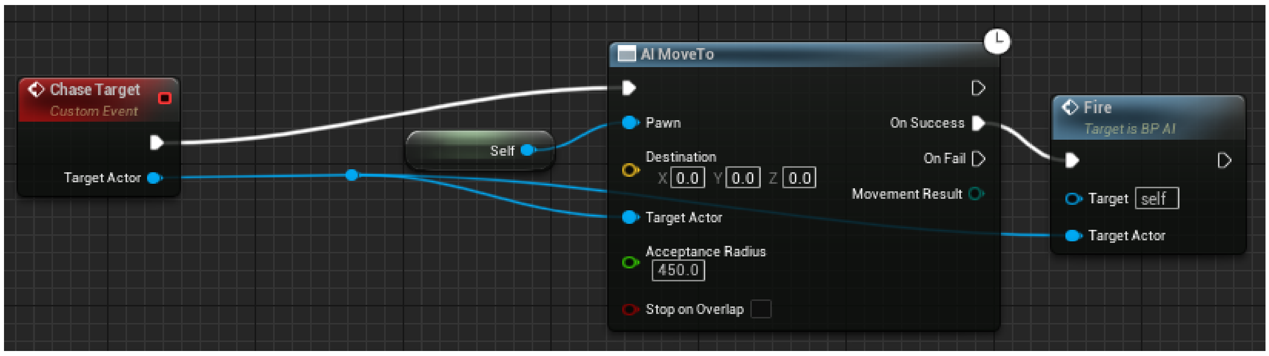

Figure 5.15: Count of team Members Blueprint

The blueprint described here is designed to manage and display the count of team members who are currently alive in a game. Let's examine its key features:

- Event BeginPlay: This node triggers at the start of the game, initializing essential elements, including UI widgets, to set the stage for gameplay.
- Get Player Character and Cast To BP_ThirdPersonCharacter: The blueprint retrieves information about the player's character. The "Cast To BP_ThirdPersonCharacter" node ensures that the retrieved character is of the correct type, such as a custom third-person character.
- Formatting and Display: The "Format Text" nodes likely format the number of team members alive into text for display purposes. The "Set Text (Text)" nodes update UI elements, such as text labels, with the formatted information. This blueprint dynamically

shows the count of living team members during gameplay, providing real-time feedback to players about the status of their team members.

In summary, this blueprint enhances team-based gameplay by providing players with real-time feedback about the status of their team members, thus facilitating strategic decision-making and improving overall gameplay experience!

Different HUD’s

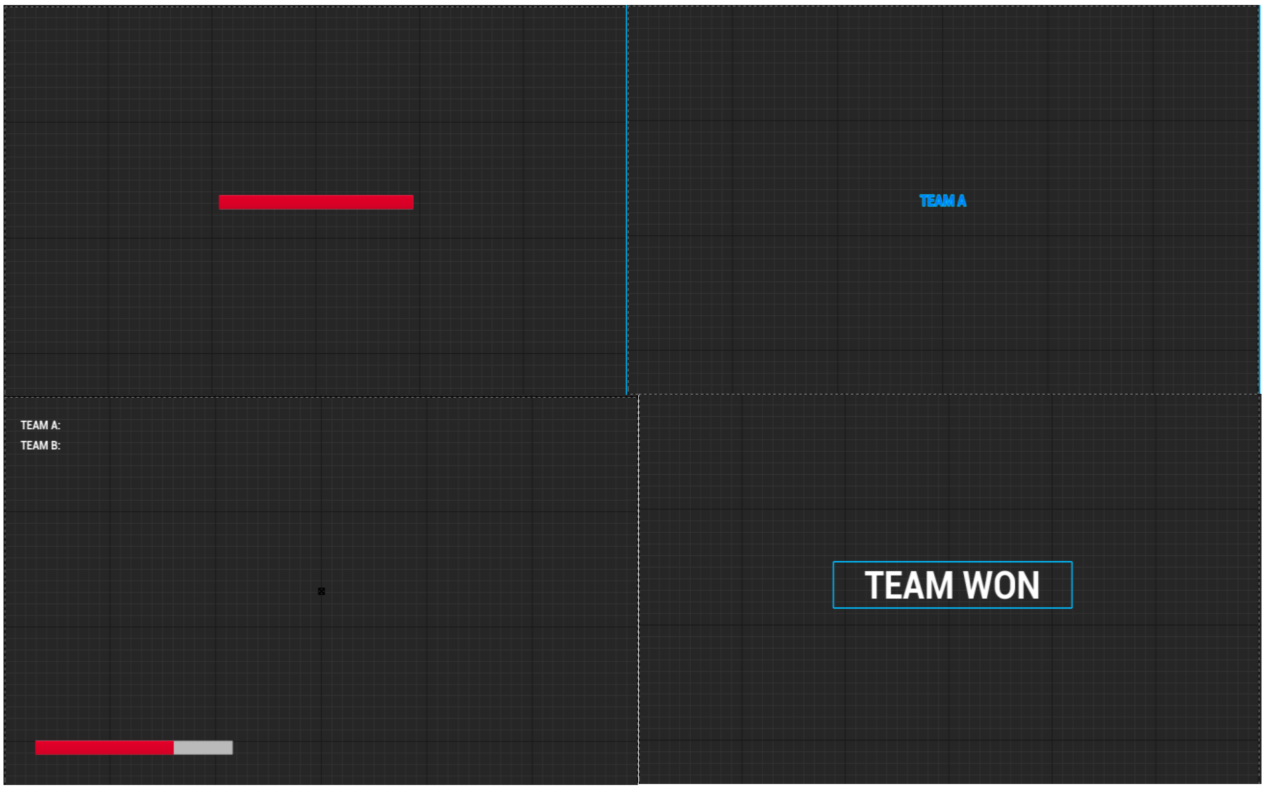

Figure 5.16: Different HUD’s

- **Health Bar HUD:** The Health Bar HUD provides players with a visual representation of their character's health status. Typically positioned in a corner of the screen, it dynamically updates to reflect changes in the player's health during gameplay. This HUD element is crucial for players to monitor their survivability and make strategic decisions during combat encounters.
- **Team Name HUD (A and B):** The Team Name HUD displays the names or identifiers of the competing teams in the game. Positioned prominently on the screen, usually at the top or bottom, this HUD element helps players identify their allies and adversaries easily. It reinforces the team-based nature of the game and enhances players' situational awareness during matches.
- **Team Members Count and Player's Health Bar HUD:** This HUD element serves a dual purpose: it displays the count of living team members alongside the health bar of the

player's character. Positioned strategically, such as in a corner or along the edge of the screen, it provides players with vital information about the current state of their team and their own character's health status simultaneously. This information empowers players to coordinate with their teammates effectively and manage their own survivability during gameplay.

- **Team WON HUD:** The Team WON HUD is a crucial element that signals the conclusion of a match in a team-based game. When one team achieves victory conditions, such as reaching a certain score threshold or completing objectives, this HUD element prominently announces the winning team's name or identifier. Positioned prominently on the screen, it serves as a definitive conclusion to the match and provides closure to players, signaling the end of the game session.

CHAPTER – VI

RESULTS

6.1 Character Physics

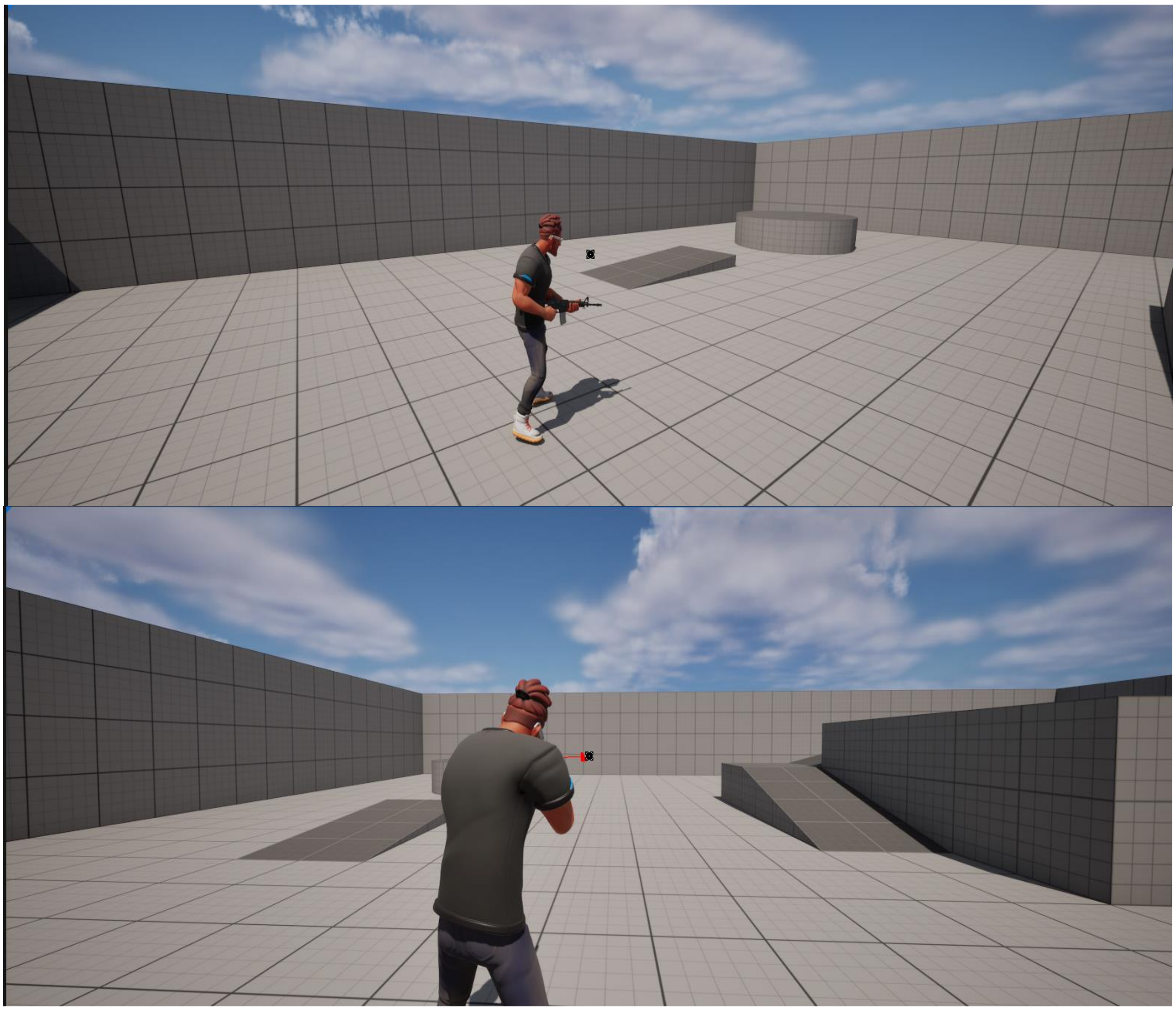

Figure 6.1: Character Physics – Jog, Jump, Weapon, Aiming etc.

The virtual environment depicted in the image showcases a player character in a casual attire consisting of jeans, a t-shirt, and sneakers. The character is holding a gun, with the focus clearly on the weapon. The gun is equipped with a muzzle, indicating its readiness for action. The player character appears alert and focused, suggesting preparedness for potential challenges or encounters within the environment. The scene exudes a sense of readiness and anticipation, with the character poised for action against the backdrop of the expansive, open area surrounded by tall walls. The clear sky with a few clouds overhead adds to the atmosphere, casting natural light that illuminates the entire scene. The juxtaposition of the character against the vastness of the surroundings creates a sense of adventure and excitement, hinting at the potential for thrilling gameplay experiences within this structured environment.

The aiming and firing system described provides players with precise control over their weapon's orientation and realistic feedback during combat encounters. Here's a breakdown of its key features:

- Aiming System: Players can adjust the aim of their weapon using mouse or controller input, allowing for precise targeting of enemies or objects within the game environment. The aiming system offers intuitive controls, enhancing player immersion and facilitating skillful gameplay.
- Firing Mechanism: When the player initiates the firing action by pressing the designated button (e.g., left mouse button), the gun discharges a projectile (e.g., bullet, laser) toward the target. A visually striking muzzle flash occurs at the gun's barrel upon firing, adding visual flair and realism to the shooting experience. Realistic recoil forces accompany the firing action, causing the gun to move backward slightly. This recoil effect simulates the impact and force exerted by the fired projectile, further immersing players in the action.

In summary, the aiming and firing system combines intuitive controls with immersive visual effects like muzzle flash and realistic recoil, enhancing the overall gameplay experience and making combat encounters more engaging for players.

6.2 Egypt Themed Map

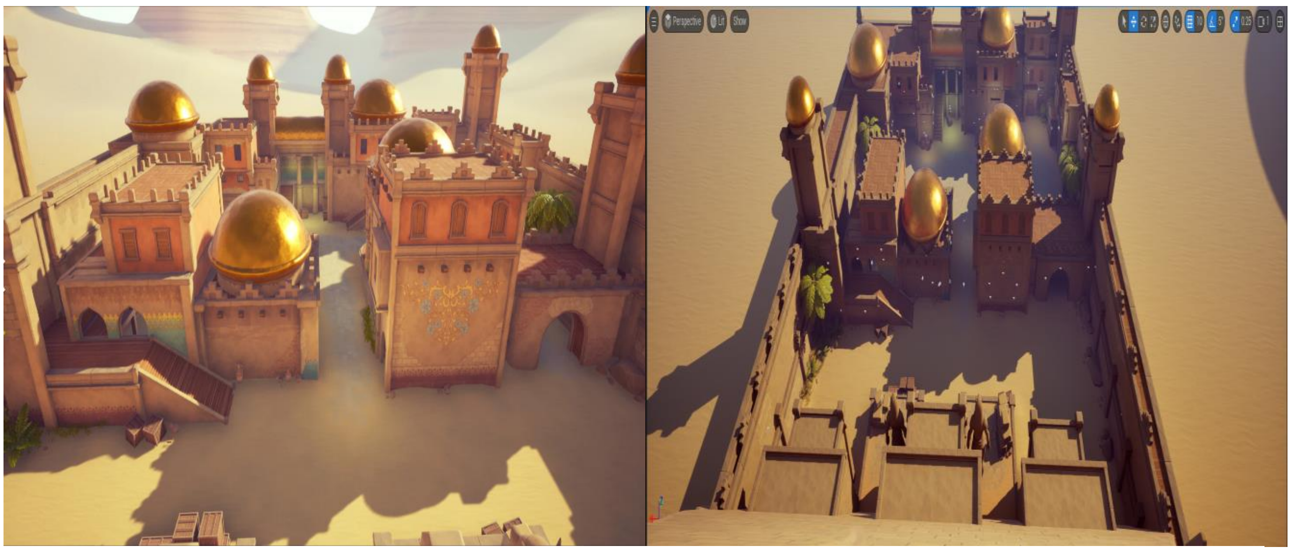

Figure 6.2: Egypt Themed Map.

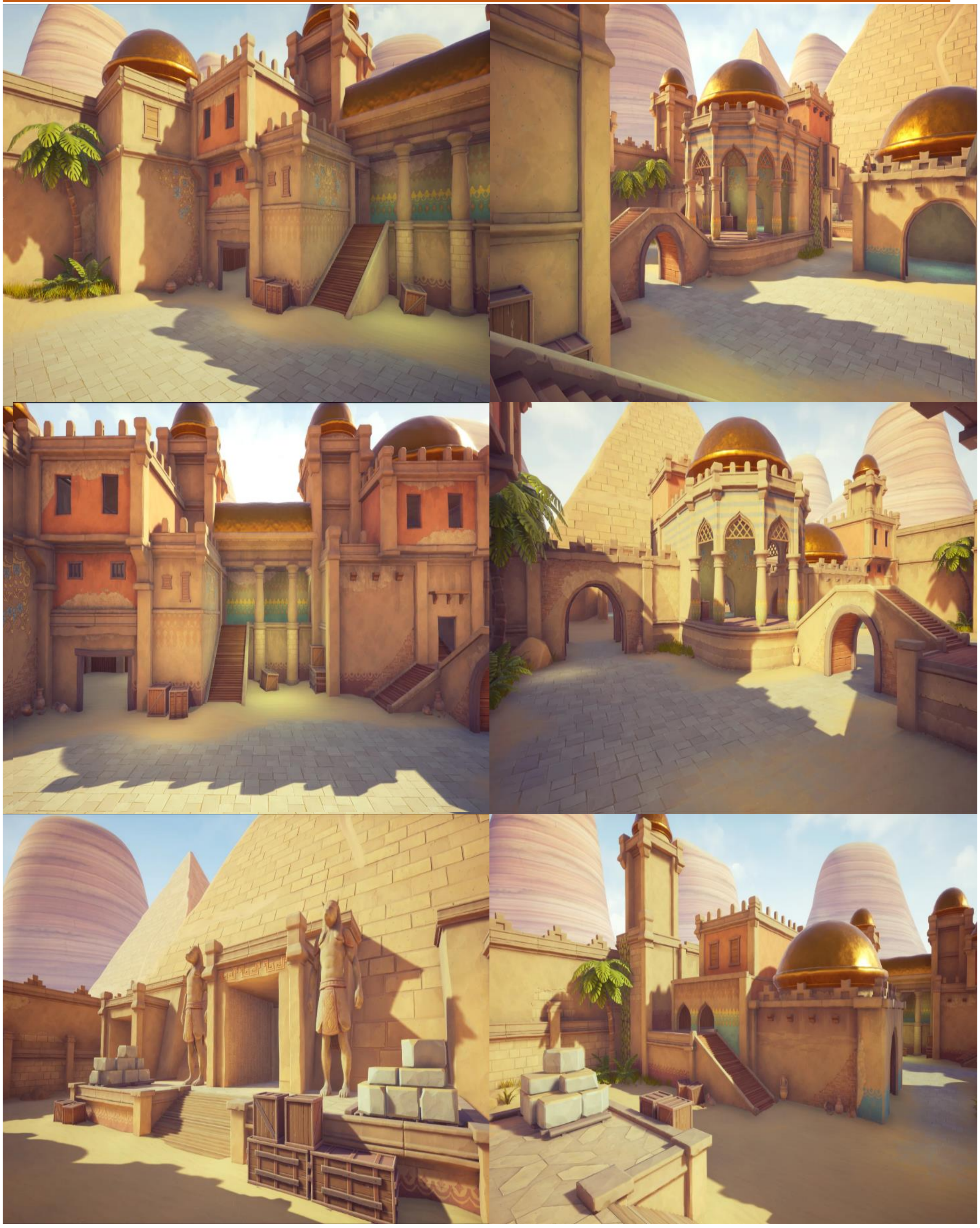

Figure 6.3: Different Scenes in the Map.

The Egyptian-themed map in the game offers players an immersive journey into the world of ancient Egypt, characterized by its rich history, mystique, and architectural wonders. Here are the key features that contribute to the captivating environment:

- Architectural Elements: The map showcases iconic structures such as pyramids, obelisks, and temples, reminiscent of ancient Egyptian civilization's grandeur and majesty.

These architectural marvels immerse players in the world of pharaohs and gods, evoking a sense of awe and wonder.

- Golden Details: Golden domes and intricate designs adorn the structures, adding opulence and luxury to the environment. The golden accents symbolize wealth, power, and the divine, further enhancing the map's visual appeal and historical authenticity.
- Atmosphere: The clear sky and soft lighting create a serene ambiance, transporting players to an era of monumental achievements and ancient mysteries. The atmosphere exudes a sense of tranquillity and reverence, inviting players to explore and uncover the secrets of this ancient civilization.

Enhancing the visual appeal of your Unreal Engine map requires careful attention to lighting and effects. Here are key techniques to consider for creating a stunning environment:

i. Global Illumination (GI): Lumen Global Illumination (UE5): Enable Lumen in your project settings to benefit from dynamic global illumination and reflections, which contribute to realistic lighting. Adjust Lumen’s quality settings to balance performance and visual fidelity, ensuring optimal results.

ii. Light mass: Light mass serves as Unreal Engine’s precomputed GI solution. Adjust lightmap resolutions based on surface details: higher resolutions for intricate surfaces and lower resolutions for distant or less important areas.

iii. Light Types and Mobility: Directional Light simulates sunlight and casts shadows across the entire scene. Point Light creates localized light sources like lamps or torches. Spotlight focuses light in a cone shape, suitable for flashlight effects. Set light mobility based on object behavior: Static lights are baked into lightmaps and ideal for stationary objects. Stationary lights are baked but can change intensity and color during gameplay. Movable lights offer dynamic lighting but can be resource-intensive.

iv. Post-Processing Effects: Bloom adds a soft glow to bright areas, enhancing realism and visual appeal. Ambient Occlusion enhances shadows in crevices and corners, adding depth to the scene. Color Grading adjusts the overall color tone to set the mood and atmosphere. Depth of Field blurs distant or close objects for cinematic focus, adding depth to the scene.

v. Sky and Atmosphere: Utilize atmospheric fog, clouds, and sky settings to create a realistic skybox that complements your environment. Adjust the sun’s position and color to match the time of day, enhancing realism and immersion.

In summary, the Egyptian-themed map offers players a visually stunning and immersive experience, combining architectural splendor, golden details, and atmospheric lighting to evoke the grandeur and mystique of ancient Egypt. It serves as a captivating backdrop for players to embark on epic adventures and unravel the secrets of this fascinating civilization.

6.3 Teams Placement on the Map

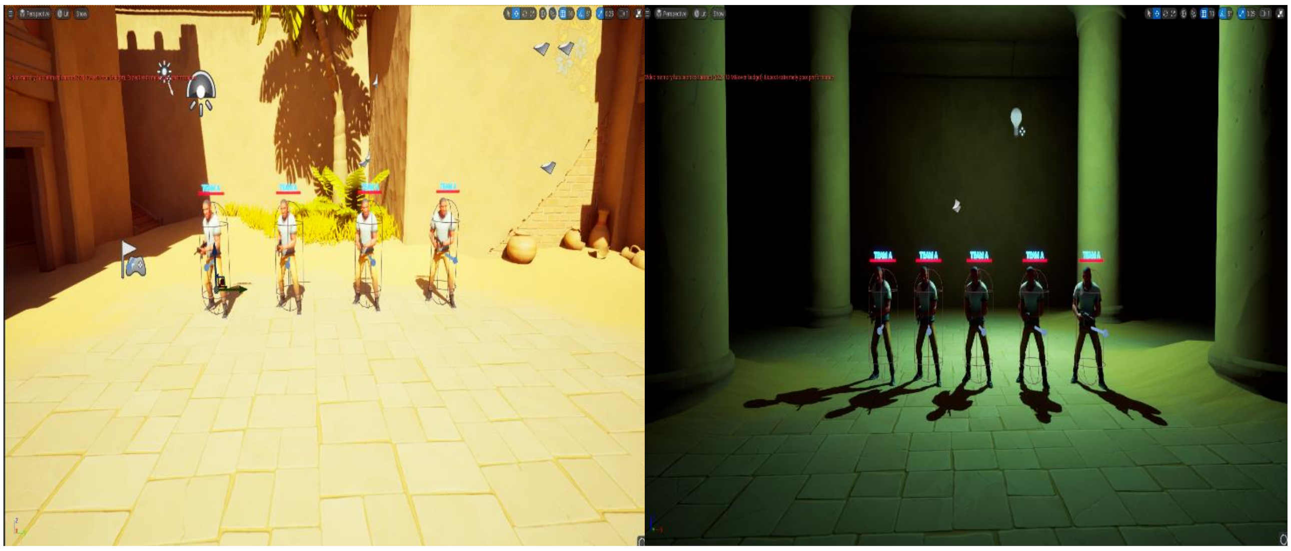

Figure 6.4: Teams Placement on the Map.

On the expansive battlefield depicted in the image, Team A and Team B are strategically positioned on opposite sides, each meticulously arranged and poised for combat. Team A stands steadfast on the left side of the map, their formation disciplined and their resolve evident as they prepare to face their adversaries. Mirroring them across the battlefield, Team B is positioned on the right side, their ranks aligned with precision, ready to engage in the upcoming clash. The strategic placement of both teams sets the stage for an intense and balanced confrontation, where skill, coordination, and tactical acumen will determine the victor.

1. Team A:
   - Composition: Team A comprises five members, each bringing their unique skills and expertise to the battlefield.
   - Formation: The members of Team A are strategically lined up in a disciplined row, demonstrating their readiness for battle and showcasing their coordinated approach to engagements.
   - Equipment: Each member of Team A is armed with A weapon, indicating their proficiency in ranged combat and their ability to control the battlefield from a distance.
   - Position: Positioned on the left side of the map, Team A stands firm, prepared to face their adversaries and defend their position with tactical prowess.
   - Terrain: As the battle unfolds on the sandy ground surrounded by ancient architectural elements like stone walls and pillars, Team A stands as a formidable force, ready to overcome any challenge that comes their way.
   - Health Bars: Above each character of Team A, health bars are prominently displayed, providing real-time feedback on their status and allowing players to monitor their team's health throughout the battle.

2. Team B:
    - Composition: Mirroring Team A, Team B also consists of five skilled members, each contributing to the team's strength and strategy.
    - Formation: Strategically aligned in a disciplined row, the members of Team B exhibit their preparedness for combat and their commitment to working together as a cohesive unit.
    - Equipment: Armed with Weapons, Team B demonstrates their proficiency in ranged combat, employing precision and accuracy to gain the upper hand in the battle.
    - Position: Positioned on the right side of the map, opposite Team A, Team B faces their adversaries with determination and resolve, ready to engage in intense combat and emerge victorious.
    - Terrain: Amidst the sandy ground and ancient architectural structures, Team B stands poised for action, leveraging the environment to their advantage as they confront their opponents in a battle that will test their skills and resilience.
    - Health Bars: Like Team A, health bars hover above each member of Team B, providing essential information about their status and enabling players to monitor their team's health throughout the heated confrontation.

In this symmetrical clash between Team A and Team B, both sides exhibit strategic prowess, skilful coordination, and unwavering determination as they vie for victory on the ancient sands of the battlefield.

6.4 Team Death match

In the immersive world of the third-person shooting game crafted using Unreal Engine, players engage in exhilarating team deathmatch battles set amidst the captivating backdrop of an ancient Egyptian-themed map. As the match begins, two rival factions, Team A and Team B, comprising five formidable members each, spawn at opposite ends of the map, strategically positioned to commence their quest for victory. Armed with powerful guns, players are equipped to navigate the intricate terrain and confront their adversaries head-on.

The game interface provides players with crucial information, prominently displaying details such as player health and the count of members on both teams. This real-time feedback allows players to gauge their team's status and strategize accordingly throughout the intense firefight.

With the stage set, both teams embark on a thrilling search across the sprawling Egyptian landscape, utilizing their keen senses and tactical prowess to locate and eliminate enemy combatants. As players navigate through ancient ruins, sandy dunes, and labyrinthine corridors, every corner becomes a potential battleground, and every encounter a test of skill and reflexes.

The dynamic gameplay unfolds as players engage in intense firefights, exchanging gunfire amidst the crumbling architecture and towering structures of the map. Team coordination, communication, and individual skill are paramount as players work together to outmaneuver and outgun their opponents.

As the battle reaches its climax, the tension mounts, and every elimination brings the victorious team one step closer to glory. Finally, when all members of one team are vanquished, the screen lights up with the triumphant Team Winning message, signaling the end of the exhilarating match and celebrating the valiant efforts of the victorious team.

In this immersive and action-packed team deathmatch experience, players are transported to a world of ancient mysteries and intense combat, where every shot fired and every decision made could mean the difference between victory and defeat.

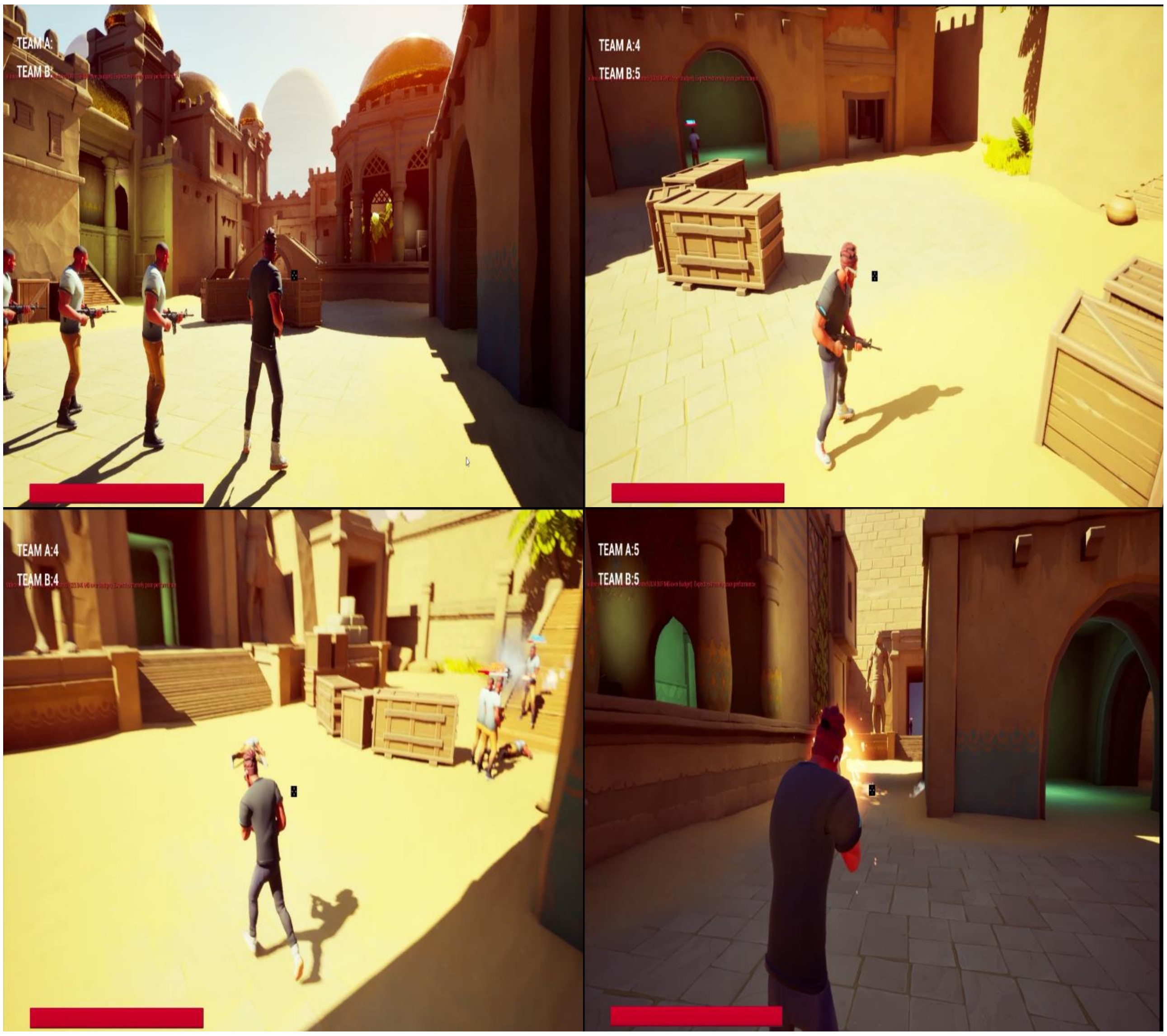

Figure 6.5: Gameplay of the Death Match.

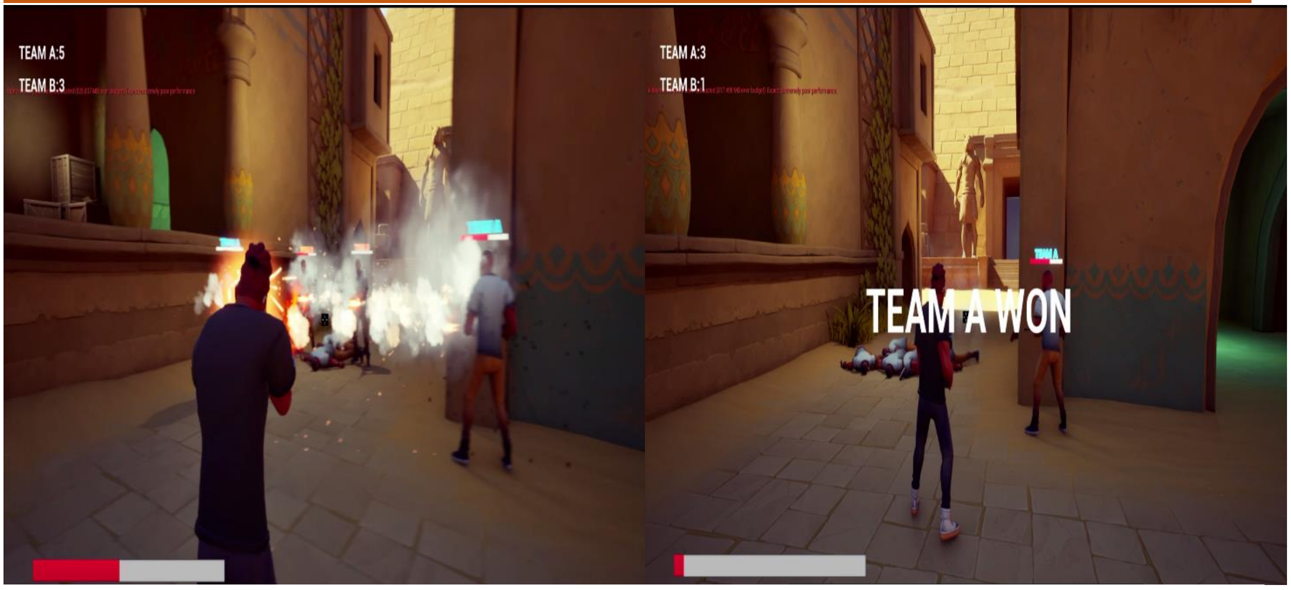

Figure 6.6: Battle between two teams and winning of team A.

In this adrenaline-fueled third-person shooting game, players engage in intense team deathmatch battles on an ancient Egyptian-themed map. With each team comprising five skilled members armed with powerful guns, players must navigate diverse terrain, from ancient ruins to sandy dunes, strategically eliminating opponents to secure victory. Real-time feedback, including player health and team member counts, informs strategic decisions. As tension mounts and heart-pounding action unfolds, teamwork, precision aiming, and quick reflexes are essential for triumph. The climactic moments culminate in the crowning of the victorious team, delivering an immersive and thrilling gaming experience in every match.

CHAPTER – VII

CONCLUSION AND FUTURE ENHANCEMENT

7.1 Conclusion

The culmination of efforts in developing the Third-Person Shooter (TPS) game has resulted in a meticulously crafted gaming experience, marked by a blend of immersive gameplay mechanics, captivating visuals, and dynamic challenges. From the inception of the project to the integration of assets and the implementation of features, each phase has been approached with meticulous attention to detail, aiming to create a cohesive and engaging player experience.

The successful setup of player characters, weapon systems, and AI adversaries lays a solid foundation for intense and thrilling combat encounters within the richly designed Egypt-themed map. Through the seamless integration of visual and audio effects, players are transported into a world where every gunshot, every footstep, and every explosion feels visceral and real. The inclusion of a Heads-Up Display (HUD) serves as a vital tool, providing players with essential information and enhancing their immersion in the game world.

As we reflect on the journey thus far, it is evident that the TPS game holds promise as a captivating gaming experience, offering players the thrill of intense firefights, strategic team-based gameplay, and immersive exploration. However, as with any endeavor, there are areas for improvement and opportunities for growth.Looking ahead, the roadmap for the TPS game is filled with possibilities. From expanding the roster of weapons and characters to refining AI behaviors and introducing new gameplay modes, there is ample room for innovation and enhancement. By listening to player feedback, embracing new technologies, and remaining dedicated to delivering high-quality content, the TPS game has the potential to evolve into a landmark title in the gaming industry.

In conclusion, the journey of developing the TPS game has been one filled with challenges, triumphs, and endless creative possibilities. With a passionate and dedicated team behind it, the game stands poised to offer players an unforgettable gaming experience that will keep them coming back for more. As the game continues to evolve and grow, it is our hope that it will leave a lasting impact on the gaming landscape, inspiring and delighting players for years to come.

7.2 Game Features

- Immersive Player Experience: Players are drawn into the action-packed world of the TPS game, navigating visually stunning environments and engaging in intense firefights with realistic weapon mechanics and responsive controls.

- Dynamic AI Adversaries: AI-controlled enemies present a formidable challenge, utilizing advanced detection mechanisms and tactical decision-making to keep players on their toes. Each encounter offers a unique and thrilling experience.
- Stylized Egypt Map: Set against the backdrop of an intricately designed Egypt-themed map, players explore ancient ruins, sandy dunes, and mystical landscapes, adding depth and ambiance to the gameplay.
- Strategic Team-Based Gameplay: With teams of five players each, strategic team placement and coordination are essential for victory. The HUD provides vital information, including team member status and objective markers, enhancing teamwork and communication.
- Immersive Audiovisual Experience: From the thunderous roar of gunfire to the subtle crunch of footsteps on sand, the game's audiovisual elements work together to create a rich and immersive gaming experience that pulls players into the heart of the action.Features

7.3 Future Enhancements

- **Expanded Content:** Introducing additional weapons, character customization options, and map variations will diversify gameplay and cater to different player preferences.
- **Advanced AI Behaviors:** Further refining enemy AI to exhibit more nuanced behaviors, including adaptive tactics, environmental interactions, and dynamic difficulty scaling, will enhance the challenge and depth of gameplay.
- **Multiplayer Integration:** Implementing robust multiplayer functionality, including matchmaking, leaderboards, and social features, will allow players to compete and collaborate in online matches, fostering a vibrant and competitive community.
- **Enhanced Visuals:** Continuously improving graphical fidelity, adding advanced lighting and particle effects, and optimizing performance will elevate the game's visual quality and immerse players in the game world even further.
- **Expanded Narrative:** Integrating a compelling storyline, engaging characters, and immersive lore will enrich the game world and provide players with a deeper sense of immersion and connection to the game's universe.
- **Community Features:** Implementing features such as player-generated content, modding support, and community challenges will empower players to contribute to the game's ecosystem and foster a strong and dedicated player community.
- **Regular Updates:** Providing regular updates, patches, and content expansions will keep the game fresh and exciting, ensuring long-term player engagement and retention.

The development process for the Third-Person Shooter (TPS) game has been meticulous, focusing on crafting an immersive and dynamic gaming experience. From the installation of essential tools to the integration of assets and mechanics, each step has contributed to shaping a cohesive gameplay environment. The successful setup of player characters, weapon systems, and AI adversaries sets the stage for exciting combat encounters within the stylized Egypt map. Visual and audio effects heighten immersion, while the inclusion of a Heads-Up Display (HUD) ensures players remain informed and engaged throughout their journey.

Reflecting on the development journey, it's clear that the TPS game holds promise as a captivating gaming experience. Players are drawn into the action-packed world, navigating visually stunning environments and engaging in intense firefights with realistic weapon mechanics and responsive controls. Dynamic AI adversaries present a formidable challenge, utilizing advanced detection mechanisms and tactical decision-making to keep players on their toes. The stylized Egypt map provides a rich backdrop, with players exploring ancient ruins, sandy dunes, and mystical landscapes.

Strategic team-based gameplay adds depth, with teamwork and coordination essential for victory. The immersive audiovisual experience, from gunfire to footstep sounds, pulls players into the heart of the action, enhancing the overall gameplay experience.

Looking ahead, there are opportunities for expansion and refinement. Introducing additional weapons, character customization options, and map variations will diversify gameplay. Further refining enemy AI behaviors and implementing robust multiplayer functionality will enhance challenge and community engagement. Continuously improving graphical fidelity and integrating compelling narratives will enrich the game world and foster a dedicated player community. With dedication and innovation, the TPS game has the potential to become a landmark title in the genre, providing countless hours of entertainment and enjoyment for players worldwide.

BIBLIOGRAPHY

[1] G. Saini, U. Verma, and A. K. Saini, “Third Person Shooter Game,” *Journal of Pharmaceutical Negative Results* ¦, vol. 13, 2019, doi: 10.47750/pnr.2022.13.S10.671.

[2] A. Denisova and P. Cairns, “First person vs. Third person perspective in digital games: Do player preferences affect immersion?,” in *Conference on Human Factors in Computing Systems - Proceedings*, Association for Computing Machinery, Apr. 2015, pp. 145–148. doi: 10.1145/2702123.2702256.

[3] Wikipedia: https://en.wikipedia.org/wiki/Unreal_Engine

[4] Unreal Engine: https://docs.unrealengine.com/4.27/en-US/Resources/Templates/ThirdPerson/

[5] YouTube: https://www.youtube.com/watch?v=Ymsabf0ZQuE

[6] YouTube: https://www.youtube.com/watch?v=whHWEObyIbQ