IJARSCT

ISSN (Online) 2581-9429

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Impact Factor: 7.53

Volume 4, Issue 8, April 2024

Development of an FPS Game with AI Perception Technology

Abel Prashanth[1] and Mrs Punita. D[2]
Department of Computer Science and Engineering[1,2]
SRM Institute of Science and Technology, Chennai, India

Abstract: *First Person Shooter games typically involve a human player interacting with computer-controlled game agents. The most common approach used to program the behaviour of game agents is a combination of Reinforcement Learning and the A* algorithm. However, this approach leads to a predictable gaming experience. A new FPS game has been developed using an AI perception system to address this limitation. This system enables more intelligent and adaptive game agents to respond to the player's behaviour and environment in real time. The game uses rule-based decision-making and machine learning techniques such as neural networks to enable game agents to perceive their environment and make decisions based on their observations. This creates a more dynamic and unpredictable gaming experience. The AI perception system used in this game can be adapted for other game genres and applications, leading to more sophisticated and intelligent virtual agents. This new first-person shooter game represents a huge leap in AI-based gaming, offering players an even more realistic and engaging experience.*

Keywords: AI Perception, AI Agents, FPS game, Unreal Engine, Perception of Bots.

I. INTRODUCTION

AI (Artificial Intelligence) has become an essential part of game development, with a wide range of applications for creating more immersive, intelligent, and engaging gaming experiences. Game AI uses algorithms, techniques, and data-driven approaches to virtual program agents that can interact with players and the game environment more intelligently and humanistic ally. One of the primary applications of AI in game development is to create more intelligent and adaptive game agents. Game agents can learn from their interactions with players and the game environment using machine learning algorithms and develop more complex and sophisticated behaviours. This makes the gaming experience more immersive and engaging as players face more challenging opponents who can adapt to their tactics and strategies.

Overall, AI has revolutionised how game developers create enemy agents, enabling them to create more intelligent, adaptive, and challenging opponents for players. Game developers can create unique and engaging game play experiences that keep players returning for more using machine learning, behaviour trees, and rule-based systems.

Unreal Engine is one of the industry's most popular and widely used game engines. Developed by Epic Games, Unreal Engine is a powerful tool for game developers, providing a wide range of features and tools to create immersive, high-quality games. In addition to its AI and graphics capabilities, Unreal Engine includes game physics, animation, and multiplayer networking tools. These features make it easier for game developers to create complex, interactive game worlds that support various game play mechanics and modes. Overall, Unreal Engine is a powerful and versatile tool for game developers, providing advanced AI, graphics, physics, and networking capabilities and a supportive community of developers. These features make it ideal for creating immersive, high-quality games that engage players and provide memorable gaming experiences.

II. LITERATURE REVIEW

Reinforcement learning and the A* algorithm are commonly used approaches for programming game agents in FPS games. However, these approaches have several limitations that lead to a predictable gaming experience. This section will critically analyse and synthesise existing research papers that discuss the limitations of these traditional approaches.[1]"Grid-based path-finding." In Conference of the Canadian Society for Computational Studies of

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 8, April 2024

Intelligence, pp. 44-55. Springer, Berlin, Heidelberg, 2002. Square grids are simple yet informative models of both artificial (typically appearing in video games) and physical (involved in robotics) 2D environments used for path planning [Yap, 2002]. Typically, in grid path finding, an agent is presumed to move from one traversable (unoccupied) cell to one of its eight adjacent neighbours. Sometimes diagonal moves are prohibited, restricting the agent's moves to only the four cardinal directions.[3]"A* Search Algorithm in Game AI," published in the Journal of Information Science and Engineering in 2014 by Liu and Zhang, Analyses the limitations of using the A* algorithm for in-game path finding AI. They argue that the A* algorithm can only find the shortest path based on a pre-defined heuristic, which results in predictable agent behaviour. They suggest using dynamic pathfinding techniques, such as potential field-based methods, to make game agents more adaptive to changes in the game environment.[5]Lara-Cabrera, R., Nogueira-Collazo, M., Cotta, C., & Fernández-Leiva, A. J. (2015).Game artificial intelligence: challenges for the scientific community. Traditionally the Artificial Intelligence (AI) of a game has been coded manually using predefined sets of rules leading to behaviours often encompassed within the so-called artificial stupidity, which results in a set of known problems such as the feeling of unreality, the occurrence of abnormal behaviours in unexpected situations, or the existence of predictable behaviours, just to name a few. Advanced techniques are currently used to solve these problems and achieve NPCs with rational behaviour that takes logical decisions like a human player. The main advantage is that these techniques automatically perform the search and optimisation process to find these smart strategies. [6]"Reinforcement Learning for Autonomous Agents in Dynamic and Uncertain Environments," published in the Journal of Artificial Intelligence Research in 2017, Li and Malik discussed the limitations of using reinforcement learning in dynamic and uncertain environments. They argue that reinforcement learning can lead to a lack of exploration and exploitation of the environment, which results in sub-optimal behaviour and a predictable gaming experience. They suggest using a combination of reinforcement learning and other techniques, such as Monte Carlo tree search, to overcome these limitations.[7]"Game AI Perception for NPCs in First-Person Shooter Games," published in the Journal of Computer Science and Technology in 2018, Liu and Wu, Suggest using AI perception systems to enable more intelligent and adaptive game agents in FPS games. They argue that AI perception systems can enable game agents to perceive their environment and make decisions based on their observations, which creates a more dynamic and unpredictable gaming experience.

In conclusion, the traditional approaches of programming game agents in FPS games using reinforcement learning and the A* algorithm have several limitations that result in a predictable gaming experience. Researchers have suggested using dynamic pathfinding techniques, reinforcement learning and other techniques, and AI perception systems to overcome these limitations. These approaches enable more intelligent and adaptive game agents that can respond to changes in the game environment and create a more dynamic and engaging gaming experience.

III. ISSUES OBSERVED IN LAST-GENERATION GAMES

One of the most popular FPS games, "Call of Duty: Ghosts," was released in 2013. Despite its high-quality graphics and variety of cinematic effects, the CoD series has continued the modern tradition of publishing AAA projects without sufficient testing procedures. As a result, virtual players can shoot at the ceiling when trying to kill enemies on the floor above due to bugs in wrong scripts or low-level command blocks and determined algorithms that do not cover all game play opportunities. General mesh problems, such as outside areas on stairs and ramps, railings, preventing excessive jumping, and dealing with rotating doors, also exist. When playing a custom map with BOTs for the first time, the generation system will build a .nav file for that map, which can take anywhere from a few minutes to a few hours, depending on the size and complexity of the map. Additionally, automatically detecting enemies can lead to errors in shooter games, such as identifying enemies through walls and BOTs appearing in textures, which does not represent realism. The research aims to create its detection module to avoid these mistakes.

Reinforcement learning and the A* algorithm have been widely used in past game development, particularly in creating game agents or enemies. These algorithms have been used to program the behaviour of game agents, such as determining their movement patterns, decision-making, and interaction with the game environment. One advantage of reinforcement learning and the A* algorithm is that they can provide a structured approach to programming game agents, as they rely on pre-programmed conditions, rewards, and pathfinding techniques to determine agent behaviour. This can make game development more efficient and easier to manage.

IV. APPLICATION OF AI TECHNOLOGY IN GAMING

The use of artificial intelligence (AI) technology has revolutionized the way we approach real-world problems, and it has now also become an essential part of the new generation of gaming. Compared to previous-generation games, new-gen games are leveraging AI technologies to create more immersive and realistic gaming experiences. One of the most significant advances in new-gen gaming is the use of AI perception technology, which has replaced traditional game development algorithms.AI perception technology uses computer vision to create more immersive gameplay by allowing game agents to perceive their environment and make decisions based on their observations in real-time. In contrast, previous-generation games relied on pre-programmed functions, which created a predictable gaming experience for players. This limitation was particularly evident in first-person shooter (FPS) games, where game agents followed specific behaviours and conditions that players could easily exploit. However, the use of AI perception technology has overcome this limitation by enabling game agents to respond to players' behaviour and environment in real time. This creates a more dynamic and unpredictable gaming experience, making the game more challenging and engaging for players.

Several researchers have studied the limitations of the traditional approach of programming game agents in FPS games using reinforcement learning and the A* algorithm, which results in a predictable gaming experience. For instance, in 2018, Yao et al. conducted a study on the limitations of the A* algorithm in RTS games and proposed the use of neural networks to improve the game's AI. Similarly, in 2019, Hsieh et al. discussed the limitations of traditional game development algorithms in creating dynamic and realistic game agents and proposed the use of machine learning techniques to overcome this limitation.

Overall, the use of AI perception technology in new-gen gaming has the potential to revolutionize the gaming experience for players, making it more immersive, realistic, and engaging. By enabling game agents to perceive their environment and make decisions based on their observations in real time, AI perception technology has overcome the limitations of traditional game development algorithms, creating a more dynamic and unpredictable gaming experience for players.

V. COMPUTER VISION

Computer vision is a vital component of AI perception technology that game developers can use to create more intelligent and adaptive game agents in Unreal Engine. In this context, computer vision algorithms are used to enable game agents to perceive their environment and make decisions based on what they see. This enhances the game's realism, making it more immersive for players. To us computer vision algorithms in Unreal Engine, game developers can leverage its built-in Computer Vision API, which provides a set of functions for implementing computer vision in games. The API includes features such as image processing, object detection, and motion tracking that can be used to track game agents' movements and actions.

To track agents, the computer vision algorithms first detect and identify the agents' visual features, such as their body shape, clothing, or other distinctive characteristics. Once identified, the algorithms track their movements using motion-tracking techniques, allowing the game engine to continually update their position and orientation in real time based on their movements and actions.

In addition to tracking agents, computer vision algorithms can also be used to identify and track other objects in the game environment, such as weapons, obstacles, or interactive elements. This information can then be used to inform the game agents' decisions and behaviours, making them more intelligent and adaptive. Computer vision algorithms can also help create more dynamic and unpredictable game play, making the game more challenging and engaging for players. For example, game agents can react to changes in the environment, such as the player's movement or new obstacles, resulting in a more realistic and challenging experience. In summary, the computer vision part of AI perception technology in Unreal Engine is an essential tool for creating more immersive and realistic gaming experiences. By enabling game agents to perceive their environment and make decisions based on what they see, computer vision algorithms can enhance game play and create more intelligent and adaptive game agents.

VI. GAME ENVIRONMENT

The 3D mesh models of the AI enemy agents and the game's visual environment characters are created in Blender 3D, a popular open-source 3D modelling software. Blender offers a variety of tools and features for creating high-quality 3D models, including the ability to create complex textures and animations.

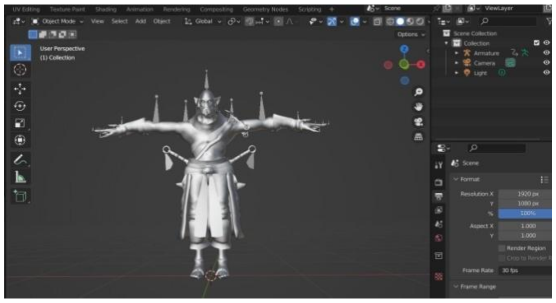


Fig 1: 3d Character mesh

Once the 3D models are completed, they have then exported to the Unreal Engine, a powerful game development platform that offers a range of features for building immersive and interactive games. In Unreal Engine, game developers can use various tools and features to enhance the game's visual and interactive elements, such as lighting, physics, and animation.

The following are key points related to the objects included in the game's environment process:

- Walls: In the game, a physical border is set up to limit the playable area. Walls are used to create this border and define the playable area, ensuring that the game agents and players stay within the set boundaries
- Doorways: Doorways are used as endpoints for the game, where the player or game agents need to reach to complete the game. These provide a clear objective for the player or game agents to accomplish and give a sense of progression in the game.
- Obstacles: Obstacles are used to make the game more immersive and challenging. They can include objects such as tables, chairs, and other environmental objects that the player or game agents need to navigate around to progress through the game.
- Guns: Guns are a tool for the player to use to damage the AI enemy agents. They are essential for the player to defend themselves and progress through the game.
- Obstacles: Obstacles are used to make the game more immersive and challenging. They can include objects such as tables, chairs, and other environmental objects that the player or game agents need to navigate around to progress through the game.
- Guns: Guns are a tool for the player to use to damage the AI enemy agents. They are essential for the player to defend themselves and progress through the game.
- Projectiles: Projectiles are circular forms like bullets that are projected from the gun. They are used to damage the AI enemy agents and are an integral part of the combat system in the game. The projectiles' properties, such as speed and trajectory, can be adjusted to create different levels of difficulty and immersion in the game.

These objects are placed strategically in the game environment to create a challenging and engaging game play experience. For instance, walls and doorways can be used to create cover and obstacles that players and agents need to navigate around, while guns and projectiles can be used to defeat enemies and progress through the game. To ensure that the game play experience is controlled and within a set area, the movement of the user's and AI enemy agents' characters is limited by a physical border. This ensures the characters stay within the playable area and do not move outside the game's boundaries.

Overall, the use of Blender and Unreal Engine in game development enables game developers to create highly realistic and immersive game environments. With the ability to create high-quality 3D models, textures, and animations in Blender and leverage Unreal Engine's tools and features for building interactive game elements, game developers can build exciting and engaging games that offer a unique and dynamic game play experience.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 8, April 2024

VII. GAMEPLAY MECHANICS

Unreal Engine allows developers to create games using C++ programming and visual scripting with Blueprints. C++ is a powerful programming language that allows for more complex systems and optimal performance, while Blueprints provide a more user-friendly interface for creating game logic. To program in Unreal Engine using C++ and Blueprints, developers can create a new C++ class in the engine editor, write the necessary code, and then compile it to create a new module that can be added to the project. Developers can also create a new Blueprint in the Blueprint editor and add nodes to the graph to implement its functionality. To add C++ code to a Blueprint, developers can create a new C++ function in the class that the Blueprint
is based on and then call the function from a node in the Blueprint graph. This allows developers to combine the power of C++ with the ease of use of Blueprints. Debugging is an important part of programming in Unreal Engine, and developers can use the Visual Studio debugger to debug C++ code and the Blueprint debugger to debug Blueprint graphs.

In the initial phase of creating a C++ class for an Unreal Engine AI FPS shooting game, the first step would be to define the data sets required for the game's character in .cpp and .h files. This involves creating separate files for each game character type, such as the ENEMY, SHOOTERCHARACTER (USER), PROJECTILE, SHOOTERGAMEMODE, and UI HUD FILES. The .cpp file contains the implementation code for the character's behaviour and actions, while the .h file contains the class definition and function prototypes. These files are then compiled into a module, which can be integrated into the Unreal Engine project.

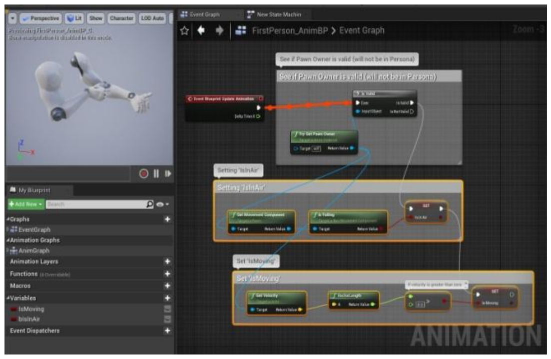
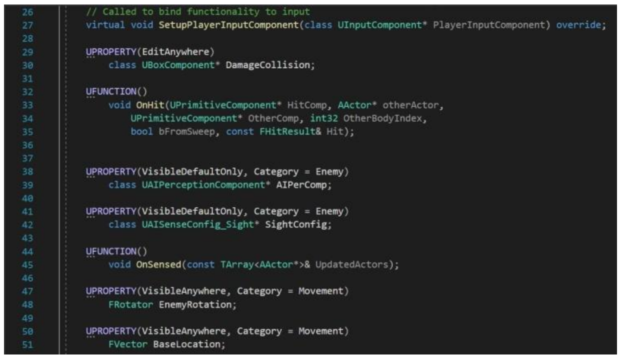


Fig 2: Spring Arm Component.

The ENEMY class would define the characteristics and behaviour of the enemy character, such as its movement speed, health points, and attack power.

The ENEMY CLASS is a crucial module in an Unreal Engine-based AI shooting game. The first step is to create a 3D character mesh for the enemy, which is then exported to the game engine. Next, general class sets are done using C++ coding. The initial step involves defining the general data to be placed in the game environment. This includes the enemy's view radius, peripheral vision angle in degrees, and action-angle when it detects a target at sight. The major part of creating the enemy class involves adding its main character as the AI perception for visualisation. This means the enemy will use AI to detect the target (USER) and take appropriate actions. The enemy AI's field of vision is set to 180 degrees FOV, allowing it to detect the target if it comes into this field. However, one setback is that the enemy cannot see the target if it hides behind a mesh. Since the game focuses on level-based areas, the enemy AI is set to follow the target (USER) only when it is under its FOV. When the target hides behind any mesh in the game environment, the AI enemy is set to go back to its original position. Overall, the creation of the ENEMY CLASS is a critical step in creating an Unreal Engine-based AI shooting game. It involves defining the enemy's visual perception, movement and behaviour in response to detecting the target (USER) and setting up the field of vision and behaviour for when the target is out of sight.

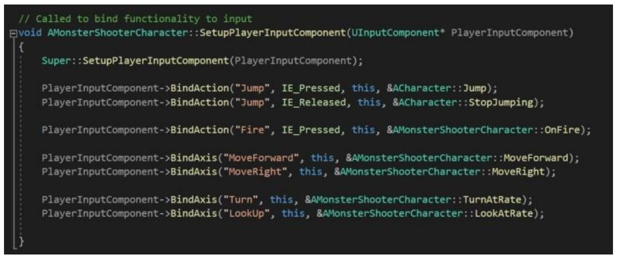


```
// Called to bind functionality to input
virtual void SetupPlayerInputComponent(class UInputComponent* PlayerInputComponent) override;

UPROPERTY(EditAnywhere)
    class UBoxComponent* DamageCollision;

UFUNCTION()
    void OnHit(UPrimitiveComponent* HitComp, AActor* otherActor,
        UPrimitiveComponent* OtherComp, int32 OtherBodyIndex,
        bool bFromSweep, const FHitResult& Hit);


UPROPERTY(VisibleDefaultOnly, Category = Enemy)
    class UAIPerceptionComponent* AIPerComp;

UPROPERTY(VisibleDefaultOnly, Category = Enemy)
    class UAISenseConfig_Sight* SightConfig;

UFUNCTION()
    void OnSensed(const TArray<AActor*>& UpdatedActors);

UPROPERTY(VisibleAnywhere, Category = Movement)
    FRotator EnemyRotation;

UPROPERTY(VisibleAnywhere, Category = Movement)
    FVector BaseLocation;
```

Fig 3:AI perception integration

The SHOOTERCHARACTER class would define the attributes and actions of the player-controlled character, such as its movement, aiming, and shooting capabilities.

The SHOOTER CHARACTER module is the user-controlled module. As it is an FPS (First Person Shooting) game, only the hand's mesh is visible in the POV of the user. The user's movement is controlled by the player input component class, which has bindings in C++ code for left, right, front, back, and jump movements. The C++ code controls the movement of the user.Additionally, the SHOOTER CHARACTER is also set to have a health bar that reduces when the enemy AI attacks the user. The level of damage is set in the code. This allows the user to keep track of their health status and strategise accordingly.The health bar is updated through the damage system. When the user gets hit by an enemy AI, the damage system reduces the health bar's value by the damage level specified in the code. The health bar's value is then updated on the UI screen, allowing users to track their health status.



```
// Called to bind functionality to input
void AMonsterShooterCharacter::SetupPlayerInputComponent(UInputComponent* PlayerInputComponent)
{
    Super::SetupPlayerInputComponent(PlayerInputComponent);

    PlayerInputComponent->BindAction("Jump", IE_Pressed, this, &ACharacter::Jump);
    PlayerInputComponent->BindAction("Jump", IE_Released, this, &ACharacter::StopJumping);

    PlayerInputComponent->BindAction("Fire", IE_Pressed, this, &AMonsterShooterCharacter::OnFire);

    PlayerInputComponent->BindAxis("MoveForward", this, &AMonsterShooterCharacter::MoveForward);
    PlayerInputComponent->BindAxis("MoveRight", this, &AMonsterShooterCharacter::MoveRight);

    PlayerInputComponent->BindAxis("Turn", this, &AMonsterShooterCharacter::TurnAtRate);
    PlayerInputComponent->BindAxis("LookUp", this, &AMonsterShooterCharacter::LookAtRate);

}
```

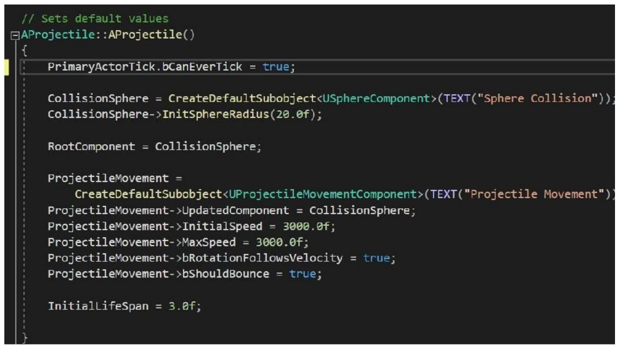
Fig 4: Character Key-bindings

Overall, the SHOOTER CHARACTER module is a crucial component of an Unreal Engine-based AI shooting game. It controls the user's movement and health status, allowing them to navigate the game environment and take appropriate actions based on their health status. The health bar system and damage system ensure that the user remains engaged in the game and that their actions have consequences.

The PROJECTILE class would define the properties of the bullets or projectiles fired by the SHOOTERCHARACTER. The creation of the PROJECTILE CLASS involves designing the bullets to be used as projectiles from the user's gun. Initially, the projectile is set with a 3D circular mesh with size reference and physical material.In the backend blueprint coding, the projectile's characteristics are defined, such as its speed of movement, general physics, and bounce back. Additionally, the code is set up to reduce the enemy AI's health when it gets hit by the projectile. This is a crucial component of the game, as it allows the user to eliminate the enemy AI and progress through the game.The projectile class also handles collision detection between the projectile and the enemy AI. When the projectile collides with the enemy AI, the code reduces the AI's health by the damage level specified in the code. This damage system is designed to ensure that the game remains challenging and that the user needs to use strategy and precision to progress.

Overall, the creation of the PROJECTILE CLASS is an important component of an Unreal Engine-based AI shooting game. It enables the user to shoot and eliminate the enemy AI, creating an engaging and challenging gameplay experience. The projectile's speed, physics, and damage level are all defined in the code, ensuring that the game mechanics remain consistent and fair for the user.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 8, April 2024



```
// Sets default values
AProjectile::AProjectile()
{
    PrimaryActorTick.bCanEverTick = true;

    CollisionSphere = CreateDefaultSubobject<USphereComponent>(TEXT("Sphere Collision"));
    CollisionSphere->InitSphereRadius(20.0f);

    RootComponent = CollisionSphere;

    ProjectileMovement =
        CreateDefaultSubobject<UProjectileMovementComponent>(TEXT("Projectile Movement"))
    ProjectileMovement->UpdatedComponent = CollisionSphere;
    ProjectileMovement->InitialSpeed = 3000.0f;
    ProjectileMovement->MaxSpeed = 3000.0f;
    ProjectileMovement->bRotationFollowsVelocity = true;
    ProjectileMovement->bShouldBounce = true;

    InitialLifeSpan = 3.0f;

}
```

Fig 5: Projectile's Movement Components

The SHOOTERGAMEMODE class would handle the overall game logic, such as initialising the game, handling player input, updating the game state, and determining the win/lose conditions. the creation of the SHOOTERGAMEMODE involves designing the general game environment. This includes setting up the playable area by defining a boundary for the game, initialising the game with the initial positions of the characters, implementing the general FPS shooter game logic, managing the health bars of the characters, and resetting the game when the level is over.The SHOOTERGAMEMODE class is responsible for managing the game environment and ensuring that the game mechanics work as intended. It defines the boundaries of the game world, such as the playable area and the spawn points for the enemy AI and the user's character. The SHOOTERGAMEMODE class also sets up the general FPS shooter game logic, such as the conditions for winning or losing the game.In addition, the SHOOTERGAMEMODE class handles the health bars of the characters. It tracks the user's and enemy AI's health levels and updates them as needed when the characters take damage. The SHOOTERGAMEMODE class also defines the conditions for resetting the game when the level is over, such as displaying the game over the screen and allowing the user to restart the level. Overall, the creation of the SHOOTERGAMEMODE class is a crucial component of an Unreal Engine-based AI shooting game. It defines the game environment and ensures the game mechanics work as intended. The SHOOTERGAMEMODE class manages the health bars of the characters and defines the conditions for resetting the game when the level is over, creating a challenging and engaging gameplay experience for the user.

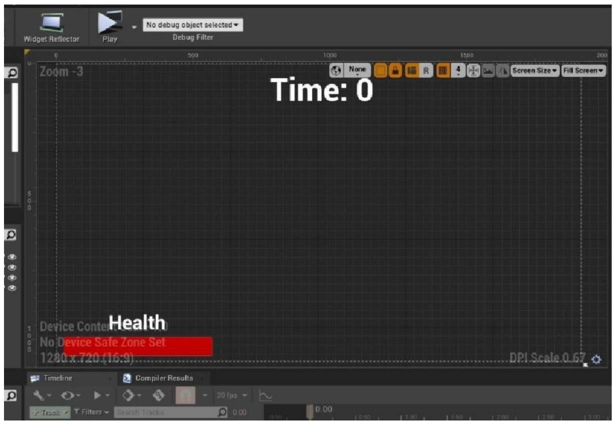


Fig 6: Health Bar HUD

Finally, the UI HUD FILES would define the layout and behaviour of the user interface, such as displaying the player's health, ammunition count, and other game statistics.The creation of the UI HUD FILES is the final step in designing the visual user interface. This involves adding several elements to the user interface, including a timer that tracks the duration of the game, a health bar that displays the remaining health of both the enemy AI and the user, and a game reset option when the level is over.The UI HUD FILES are an essential part of the game, as they provide the user with important information about the game's progress and the status of their character. The timer allows the user to keep track of the game's duration, and the health bars indicate the remaining health of both the enemy AI and the user.In addition, the UI HUD FILES allow the user to reset the game when the level is over. This provides a way for the user to start a new game once they have completed the previous level. The game reset option may also include the ability to save progress, load saved games, or adjust game settings.

Overall, the creation of the UI HUD FILES is an important part of an Unreal Engine-based AI shooting game. It adds crucial information to the game's visual interface and provides the user with a way to reset the game once the level is over.

In general, designing C++ classes for an Unreal Engine AI FPS shooting game entails defining data sets and implementing the behaviours and actions of each game character type. These classes are then compiled into a module that can be added to the game project, allowing for more complicated and optimal gameplay features.

VIII. FINAL RENDERING & VISUALIZATION

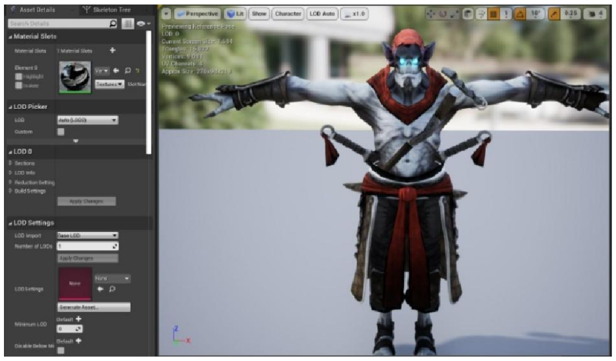
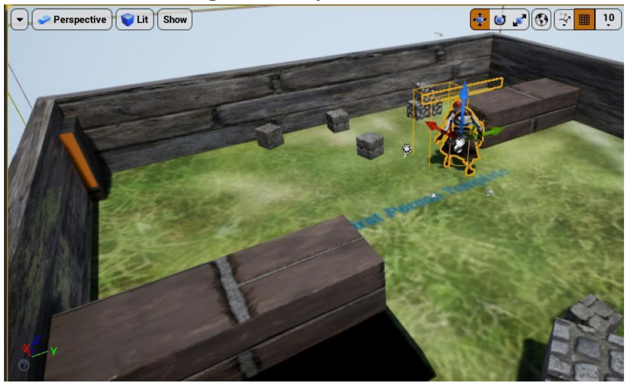
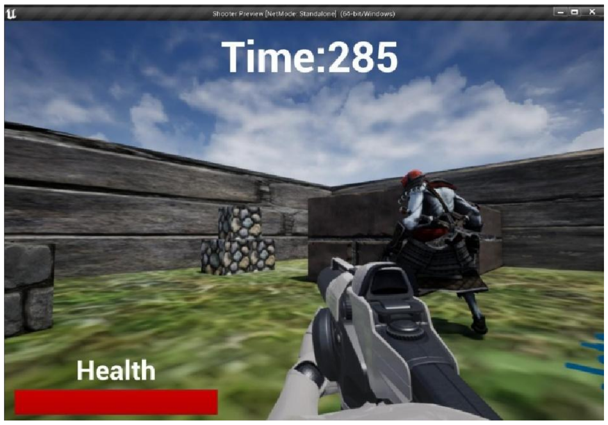


Fig 7: Enemy Character





IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 8, April 2024

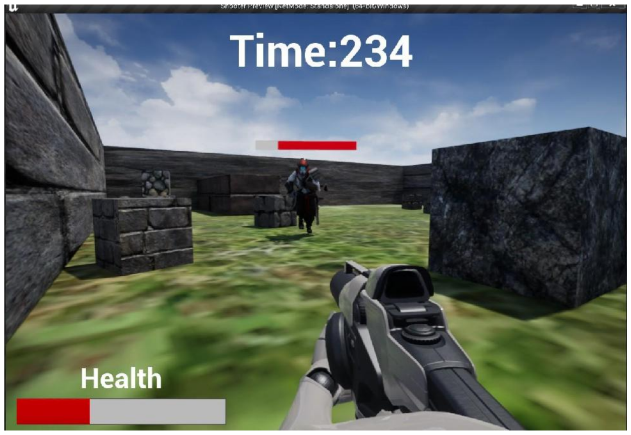


Fig 8: AI tracking in Action Gameplay

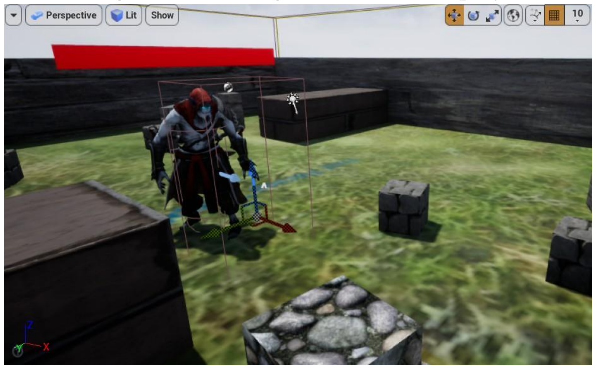


Fig 9: Game & Character Environment

IX. CONCLUSION

The utilization of Artificial Intelligence (AI) has proven to enhance the gaming experience by creating a more realistic and impactful game. The prototype First-Person-Shooter (FPS) game implemented AI Perception within Unreal Engine 4, which simulated combat from a first-person perspective. By encountering opponent waves that needed to is defeated to progress, the game successfully demonstrated how AI could revolutionize the gaming industry. The project serves as an example of the fundamental use of AI in a game, highlighting the vast potential for advanced AI implementations within Unreal Engine 4. With the inclusion of various functions within Unreal Engine 4, the integration of advanced AI could be used to further enhance future gaming experiences

REFERENCES

[1]. "Grid-based path-finding." In Conference of the Canadian Society for Computational Studies of Intelligence, Springer, Berlin, Heidelberg, 2002.

[2]. “Game Development using Artificial Intelligence in Unreal Engine”" published in the Journal of International Research Journal of Engineering and Technology in 2012

[3]. "A* Search Algorithm in Game AI," published in the Journal of Information Science and Engineering in 2014, by Liu and Zhang

[4]. Immanuel, R. R., & Sangeetha, S. K. B. (2022, August). Analysis of EEG Signal with Feature and Feature Extraction Techniques for Emotion Recognition Using Deep Learning Techniques. In International Conference on Computational Intelligence and Data Engineering (pp. 141-154). Singapore: Springer Nature Singa

[5]. "Imitation of Human Behavior in 3D-Shooter Game” Makarov Ilya, Tokmakov Mikhail, Tokmakova Lada published in the Journal of International Research Journal of Engineering and Technology in 2015

IJARSCT

ISSN (Online) 2581-9429

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Impact Factor: 7.53

Volume 4, Issue 8, April 2024

[6]. Lara-Cabrera, R., Nogueira-Collazo, M., Cotta, C., & Fernández-Leiva, A. J. (2015). Game artificial intelligence: challenges for the scientific community

[7]. "Reinforcement Learning for Autonomous Agents in Dynamic and Uncertain Environments," published in the Journal of Artificial Intelligence Research in 2017, Li and Malik

[8]. "Game AI Perception for NPCs in First-Person Shooter Games," published in the Journal of Computer Science and Technology in 2018, Liu and Wu

[9]. Immanuel, R. R., & Sangeetha, S. (2023). Identifying Different Emotions Of Human Using Eeg Signals Using Deep Learning Techniques. Journal of Theoretical and Applied Information Technology, 101(18).

[10]. "Players Versus Bots: The Perception of Artificial Intelligence in League of Legends" by Kayan Terrence Wu published in the Journal of Education, Humanities and Social Sciences in 2022.