ISSN: 2584-0495 Vol. 3, Issue 1, pp. 1499-1506

International Journal of Microsystems and IoT

ISSN: (Online) Journal homepage: https://www.ijmit.org

# A Comprehensive Investigation of 3D Game Development with Unreal Engine

**Nitesh Ghodichor, Vaishali Patil, Yash Nagpure and Yashavi Viragade**

**Cite as:** Ghodichor, N., Patil, V., Nagpure, Y., & Viragade, Y. (2025). A Comprehensive Investigation of 3D Game Development with Unreal Engine. International Journal of Microsystems and IoT, 3(1), 1499–1506. https://doi.org/10.5281/zenodo.15473298



Published online: 20 January 2025



DOI: https://doi.org/10.5281/zenodo.15473298

International Journal of Microsystems and IoT
Vol.3, Issue 1, pp.1499-1506; DOI: https://doi.org/10.5281/zenodo.15473298



# A Comprehensive Investigation of 3D Game Development with Unreal Engine

Nitesh Ghodichor[1], Vaishali Patil[2], Yash Nagpure [1] and Yashavi Viragade[1]

[1] Department of Computer Technology, Priyadarshini Collge of Engineering, Nagpur,Maharshtra, India
[2] Department of Artificial Intelligence, G H Raisoni College of Engineering & Management, Nagpur, Maharashtra, India

**ABSTRACT**
This study represents a comprehensive investigation into the creation of a three-dimensional (3D) game utilizing the features of the Unreal Engine, a leading resource in the gaming sector. The study carefully analyzes every phase of game development, from initial design to thorough testing, highlighting the distinctive features of the Unreal Engine that streamline this process. The game, created within the framework of this investigation, is designed to be captivating, user-focused, and visually outstanding, leveraging the advanced graphics rendering and physics simulation features of the Unreal Engine to their fullest potential. This study's outcomes offer valuable insights into 3D game development with the Unreal Engine, thus enhancing the growing body of knowledge in the field of game development. The results of this study provide a significant resource for upcoming game developers, enabling them to effectively utilize the Unreal Engine in crafting immersive and high-quality 3D games.

**KEYWORDS**
Unreal Engine; Videogame: Engine 3D Game Development; Video Game; Simulators

## 1. INTRODUCTION

The gaming business has changed dramatically with the introduction of cutting-edge technology, and three-dimensional (3D) games have become a major component of this development. These games have enthralled audiences all around the world thanks to their realistic graphics, engaging gameplay, and dynamic worlds, which have helped the gaming industry grow rapidly. However, creating such complicated games is difficult and involves a thorough comprehension of a number of areas, including as user experience, programming, graphics rendering, and game design.

The Unreal Engine is one of the most widely used programs for creating 3D games. The Unreal Engine is a full suite of creation tools created by Epic Games that are intended to fulfill ambitious artistic ambitions while being adaptable enough to guarantee success for teams of all sizes. The Unreal Engine is a well-known and adaptable game production tool that has been utilized to produce some of the most well-known and aesthetically spectacular video games available, such as the Borderlands series, Street Fighter V, and Fortnight.

This research seeks to explore the methodology of developing a 3D game utilizing Unreal Engine. The several phases of game production, encompassing the initial concept and design through to the concluding stages of testing and user input, are examined. This research emphasizes the distinctive attributes and functionalities of the Unreal Engine, including its sophisticated graphics rendering, physics simulation, and intuitive interface, rendering it a favored option for game creators.

The game created for this study will exemplify the utilization of Unreal Engine elements in the development of a 3D game.



The game is crafted to be captivating, intuitive, and visually elaborate, fully leveraging the sophisticated graphics rendering and physics simulation features of the Unreal Engine. This research will yield significant insights into the practical dimensions of game production, enhancing the broader discipline of game development studies.

This document is organized as follows: Following this introduction, a literature analysis is provided, examining prior research pertinent to 3D game production and the Unreal Engine. This is succeeded by a comprehensive elucidation of the game mechanics employed in this study, encompassing the design and development process of the 3D game. The results section delineates the study's conclusions, encompassing game performance and user input. The discussion section analyses these findings within the framework of existing literature, offering a thorough examination of the data. The report ultimately finishes with a synthesis of the findings and its implications for subsequent studies.

## 2. LITERATURE REVIEW

For the purpose of this research paper, the literature review covers two primary topics: the creation of three-dimensional games and the application of the Unreal Engine in the process of doing so.

### 2.1 The Development of 3D Games:

The creation of three-dimensional games has been a subject of interest for a great number of researchers due to the intricacy of the process and the extensive skill set that is required. Numerous studies have been conducted to investigate various

areas of the production of three-dimensional games, such as game design, programming, graphics rendering, and interaction with the player.

In the field of game design, academics have investigated a variety of design ideas and methodologies that contribute to the production of 3D games that are both engaging and immersive. For example, Adams and Rollings (2007) offer a detailed review of the process of game design. They examine important aspects of the game design process, including gameplay, narrative, characters, and world design.

When it comes to the technical aspects of developing a 3D game, the most important components are the programming and graphics rendering process. Insights into the programming languages, algorithms, and graphics application programming interfaces (APIs) that are typically utilized in game development can be gained from studies such as those conducted by Gregory (2014) and Rabin (2010).

Additionally, the user experience is an essential component of the production of 3D games. This area of research has been centered on gaining an understanding of how players engage with games and how these interactions might be modified to increase the level of satisfaction experienced by players. In this particular area of study, the work that Isbister and Schaffer (2008) have done is a significant contribution.

### 2.2 Utilizing Unreal Engine for the Development of 3D Games:

The Unreal Engine, which was created by Epic Games, is currently one of the most widely used tools for the development of three-dimensional games. Numerous research have been conducted on them because of the sophisticated capabilities and traits that they possess.

The architecture of the Unreal Engine has been investigated by a number of scholars, who have discussed the various components of the engine and how they contribute to the process of game production. An in-depth analysis of the engine architecture was presented in the Keler (2004) investigation. The application of Unreal Engine to the creation of particular categories of video games has also been the subject of research. The work of Lewis and Jacobson (2002), for instance, explores the application of the Unreal Engine in the process of developing first-person shooter video games.

In addition, a number of scientific investigations have concentrated on the ability of Unreal Engine to produce graphics. Several studies, such as the one conducted by Engel (2008), offer valuable insights into the manner in which the Unreal Engine manages the rendering of images. These studies examine aspects such as the engine's lighting model, shadowing approaches, and shade programming. In summing up, the body of literature concerning the production of three-dimensional games with the Unreal Engine is broad and varied. The purpose of this study is to make a contribution to the existing body of knowledge by presenting a real-world example of the production of a three-dimensional game utilizing the Unreal Engine.

## 3. GAME MECHANICS

### 3.1 World design and environment creation

*Conceptualization*: Start by conceptualizing the game world. This could involve sketching out ideas, creating mood boards, and writing descriptions of different areas within the game world.

*Design*: Use the landscape and foliage systems of Unreal Engine to design the game world. This includes creating terrain, adding trees and other vegetation, and designing bodies of water.

*Weather and Lighting Effects*: Implement different weather systems and lighting effects to increase the realism of the game world. This could involve creating a day/night cycle, adding rain or snow effects, using Unreal Engine's lighting system to create the desired atmosphere.

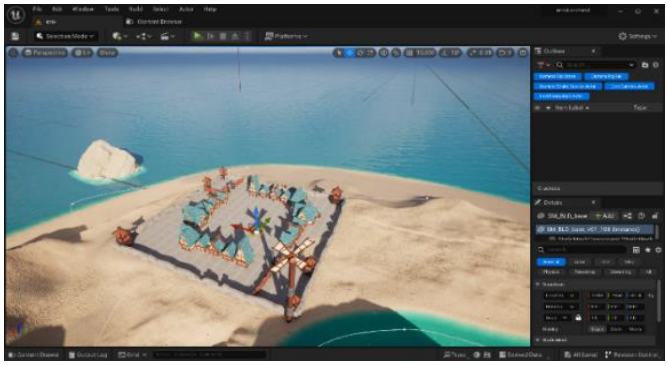

**Fig. 1** Desired atmosphere

### 3.2 Character Design and Animation

*Character Design*: Sketch and model characters were created using 3D modeling software such as Blender or Maya. This includes creating the character's physical appearance, clothing, and any accessories they might have.

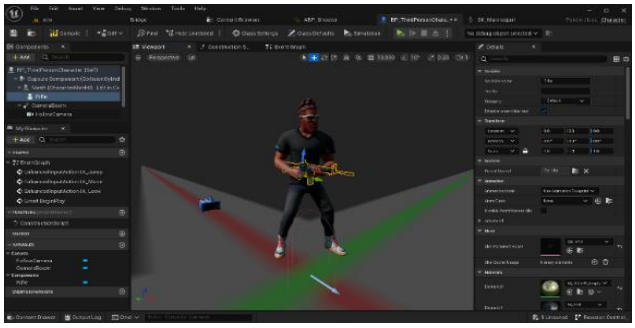

**Fig. 2 Creating the character's physical appearance**

*Character Animation*: Import the character models into Unreal Engine and create character animations using Unreal Engine's animation system. This includes walking, running, jumping, and any other movements the character will need to make in the game

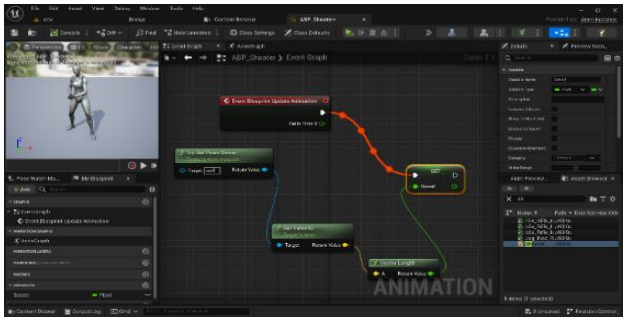


Fig. 3 Movement of characters

### 3.3 Multiplayer Networking

*Network Basics*: Learn about the networking and multiplayer features of Unreal Engine. Understanding how to manage latency and synchronize game states across several clients are part of this.
*Character Synchronization*: Make sure that characters on different clients are in sync. This guarantees that every player will witness the same motions and behaviours from every character.
*Gameplay Mechanics*: Creating server-authoritative gaming mechanics is the goal. Because the server has the last word over how any action in the game turns out, cheating may be less likely to occur.

### 3.4 Combat System Development

**Combat Mechanics**: Design the combat mechanics and systems. This includes deciding how players will attack and defend themselves, what kinds of abilities they will have, and how health and damage will be calculated.

Fig. 4 Character combat animation

Combat Animations: Implement combat animations and character abilities. This includes creating animations for different attacks and abilities, and programming these abilities into the game.

### 3.5 Opponent AI Implementation

AI Design: the AI is Designed for the opponents in the game. This includes deciding how the AI will behave, what tactics it will use, and how it will react to the player's actions.

AI Programming: Implement the AI using Unreal Engine's AI tools. This includes programming the AI's behavior, reactions, and tactics.

### 3.6 Quest System Integration

Quest Design: Design the quest system for the game. This includes creating different quests for the player to complete, deciding what rewards the player will receive, and writing a dialogue for quest-related NPCs.

Quest Implementation: Implement the quest system in the game. This includes programming the quests into the game, creating quest markers and objectives, and testing the quests to ensure that they work correctly.

### 3.7 User Interface (UI) Design

UI Design: Design the user interface for the game. This includes creating menus, buttons, health bars, and other UI elements.

UI Implementation: Implement the UI in the game. This includes programming the UI elements to work correctly and testing the UI to ensure that it is intuitive and user friendly.

### 3.8 Sound and Music Integration

Sound Design: Design the sound effects and music for the game. This includes creating sound effects for different actions and events in the game, and composing or selecting music for different areas and situations in the game.

Sound Implementation: The sound effects and music implemented in the game. This includes programming the sounds to play at the correct times and testing the sound to ensure that it enhances the game experience.

### 3.9 Testing and Debugging

Testing: Test the game thoroughly to find any bugs or issues. This includes testing all aspects of the game, from the gameplay mechanics to the UI.

Debugging: Fix any bugs or issues found during testing. This includes debugging the game code, fixing any issues with the game art or sound, and retesting to ensure that the issues have been resolved.

### 3.10 Optimization for Performance

Performance analysis: Analyze the game's performance to find any areas that could be optimized. This includes checking the game's frame rate, load times, and memory usage.

Optimization: Optimize the game to improve performance. This includes optimizing the game code, art assets, and sound files, and retesting to ensure that the optimizations have improved the game's performance

## 4. A UNREAL ENGINE 5 (DEVELOPMENT TOOL

Epic Games is responsible for the development of the Unreal Engine, which is a powerful real-time 3D creative tool that is

utilized extensively across a variety of industries1. The high-fidelity graphics and efficient blueprint visual scripting mechanism that it possesses have garnered a lot of praise. Listed below are some of the most important aspects of it:

- **Incomparable Graphical Capabilities**: An Unreal Engine is renowned for its exceptional graphical proof. Developers are able to create worlds that are both visually attractive and photorealistic because to its cutting-edge rendering capabilities, which are represented by the powerful Unreal Engine 5. Players are plunged into intriguing surroundings because to the engine's high-fidelity graphics and dynamic lighting technologies, which provide an unrivalled level of realism and aesthetic appeal.
- **Blueprint Visual Scripting System**: The Blueprint visual scripting system makes it easier for designers and artists to write code, which in turn encourages collaboration and streamlines the development process1. The use of this feature can be advantageous in every facet of the game development process.
- **The capacity of an Unreal Engine**: to scale to a larger number of users is yet another significant advantage. It is designed to work with a wide variety of platforms, ranging from personal computers and gaming consoles to mobile devices and virtual reality systems, which ensures that games may be played by a large number of people. Because of this versatility, developers are able to simply generate and deploy a single project across numerous platforms1, which reduces the amount of time and work the development process takes.
- **Community and Documentation**: The Unreal Engine provides customers with a community that is both active and encouraging. The extensive documentation, tutorials, and active online community that the engine offers are extremely helpful resources for developers of all skill levels. These resources enable developers to solve problems and share their knowledge with one another.
- When it comes to cost-effectiveness, the development of video games employing an Unreal Engine is a good choice1. In spite of the fact that it provides a robust free version, creators are only required to pay royalties on their game's gross income if it reaches a certain threshold. This makes it an appealing choice for both indie and established companies.

In the world of video game production, Unreal Engine has earned a stellar reputation, and there are a multitude of reasons why developers should consider using it. Known for its wide range of capabilities that enable developers to create gaming experiences that are both immersive and visually captivating, it is a formidable force in the field of game creation. With Unreal Engine, you will be able to bring your game development projects to life, regardless of whether you are an experienced developer or a novice to the engine. You will get helpful insights and techniques that you can put into practice.

## 5. Result and Discussion

Blueprints is the name of the visual programming language that is used to run a three-dimensional game on "Arena Island."

- Blueprints are visual scripting tools that are included in the Unreal Engine. These tools enable developers to design game logic without having to write individual lines of code. There is a significant emphasis placed on blueprinting on Arena Island.
- *The mechanics of the game are as follows*: The behavior of the game is determined by blueprints, which include everything from the movement of characters and combat systems to interactive objects and riddles. The use of blueprints is what makes this happen, whether it be a complicated AI behavior or a straightforward trigger event.
- *In terms of level design*, the landscapes, architecture, and environmental interactions of Arena Island are all organized through the use of blueprints. Want a secret cave to be revealed to the player after they have solved a puzzle? It's a blueprint!!
- *Input from the User*: The use of blueprints allows for the construction of menus, aspects of the HUD, and interactive user interface components. Players are able to access their settings, missions, and inventory without any difficulty.

The nodes shown in Fig.6 are defined as follows: Event BeginPlay: This event is triggered when an actor spawns into the game world or when the game launches. It is frequently used to carry out setup chores such as setting timers and initializing variables. This serves as the script's beginning point in this plan.
Get All Actors of Class: This node obtains every actor that belongs to a given class in the game universe. Here, searched for every actor in the class "BP_ThirdPersonCharacter." This could be used, for instance, to count the number of characters in the game or to apply an effect to every character.

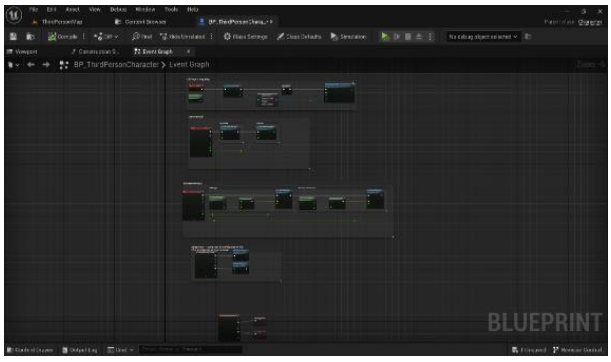


**Fig. 5** Basic movements

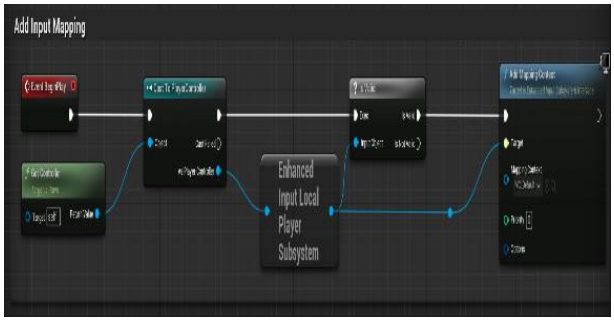


**Fig. 6** Input Mapping

*For Each Loop*: Every element in an array is iterated

over by this node. Here, the "Get All Actors of Class" node is used to retrieve each actor one at a time. It carries out the activities associated with the loop body for every actor.

*Cast to* ***ThirdPersonCharacter***: This node aims to handle a generic actor as an instance of a particular class of actor, a ThirdPersonCharacter in this case. The cast succeeds, and the screenplay continue if the actor is, in fact, a Third-Person Character. Otherwise, the script ends, and the cast fails. This is how you get at methods or properties unique to the ThirdPersonCharacter class.

***Print String***: Text is displayed on the screen using this node. It is frequently used to confirm that the script is operating as intended during troubleshooting. In this instance, it prints "hello" each time the loop executes.

***Add Movement Input***: This node moves a character by applying a movement input to it. The inputs for the World Direction and Axis Value define the movement's power and direction. In this instance, the image does not indicate the World Direction, but the Axis Value is set to 1. The Enhanced Input Local Player Subsystem, comprising this node, offers more versatile and potent input management than does the conventional input system. It is difficult to pinpoint its particular function without understanding how it fits into the overall plot.

***Input action enhancement input***: This is where the blueprint begins. It simulates a player-inputted action, such as pushing a button or navigating a joystick. The node has multiple outputs, each of which represents a distinct input action state:

***Started*****:** When the player first starts the input action, this output is set off.

***Continuous***: As long as the player is pressing a button or navigating with a joystick, this output will be constantly activated.

***Cancelled***: If the input operation is canceled before it is finished, this output is generated.

***Completed*****:** When the player releases the button or stops moving the joystick, the input operation is considered complete, and this output is triggered.

***Add Controller Yaw Input***: This node adds a yaw (vertical) rotational input. Usually, it is used to move the player's character or camera left and right. The amount of rotation is determined by the "Val" input, which is linked to a value that is altered in response to player input.

***Add Controller Pitch Input***: Add Controller Pitch Input, and add rotational input around the pitch (horizontal) axis. Usually, is utilized to enable up and down viewing of the player's camera or character. The amount of rotation is also set by the "Val" input, which is linked to a value that is affected by the player input.

The green line connecting the yaw and pitch nodes' "Val" inputs to the EnhancedInputAction Input node's "Ongoing" output implies that the player can smoothly control the camera by holding down an input, such as a key or joystick, which will continually feed into the rotation values. With the help of this configuration, responsive camera controls that respond to player actions can be created, providing players with an easy method for exploring the game environment. Recall that the precise actions you do will be determined by the connections and values you provide in your design.

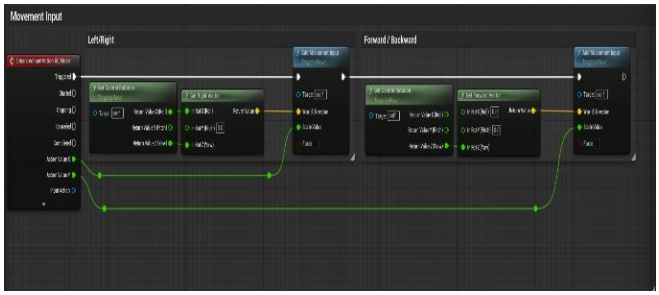


**Fig. 8** Movement Input

***Branch Nodes***: These nodes serve as decision points in the blueprint's execution flow, comparable to an "if" statement in traditional programming. They use three inputs:

***Condition***: The Boolean input determines the execution path. If the condition is true, the execution will proceed from the "true" pin. If the condition is false, the execution will continue at the "False" pin. True (Execution Pin): Indicates the execution path when the condition is true. If the condition is true, all nodes related to this pin will be executed.

***False (Execution Pin)***: In the event that the condition is false, this is the execution path that is taken. When the condition is false, any nodes attached to this pin will be put into action.

***Input Axis MoveForward/MoveRight Nodes*****:** The player can move forward, backward, right or left by providing input to these nodes. These nodes produce a float value as an output, which is the input's axis value. This value has a range of -1 to 1, where 0 denotes no input, -1 denotes full forward/left input, and 1 denotes complete backward/left input.

***Add Movement Input Nodes*****:** The character may move thanks to these nodes. Three inputs are available to them: Target (Self): This is the persona to whom the motion will be directed. This would typically be the character that the player controls. The direction in which the character moves is indicated by the vector called *World Direction*: This is a variable that varies according to the game's circumstances, such as moving the character toward a target, or it possibly a constant value that moves the character ahead always. ***Scale Value*****:** A float value that indicates the magnitude of the movement. It is frequently linked to the axis value of the input, which implies that the character's speed changes depending on how much input the user provides.

The connections between these nodes represent the flow of data and the sequence of processes. For example, InputAxis nodes are linked to Branch nodes. This signifies that the branch will verify the value of the player's input. If the input value is not zero (i.e., the player is providing input), the "true" execution pin will be activated, and the character will move in the appropriate direction.

***EnhancedInputAction IA_Jump***: The jump action's

input node is located here. For various input action stages, it has multiple outputs: occur. Triggered: This mode becomes active at the initial trigger of the jump input. Started: Turns on as soon as the jump input action starts. Ongoing: Stays active for as long as there is continuous jump input. Canceled: Turns on in the event that the leap input action is stopped. Completed: Becoming active after the leap input action is finished.

***Jump*:** This node is linked to the EnhancedInputAction IA_Jump node's "Triggered" output. When the input is triggered, the character's jumping action starts. The fact that the input pin labeled "Target" is back linked to itself suggests that the action has an impact on the character receiving it.

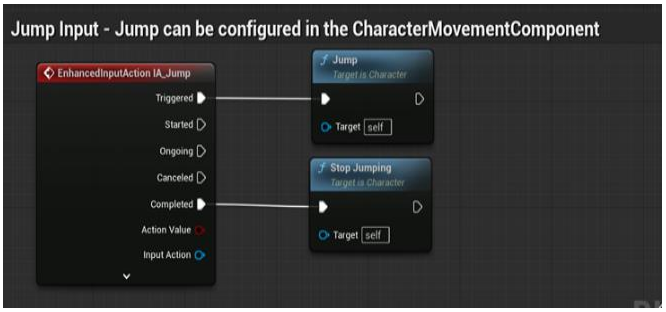


Fig. 9 Jump Input

***Stop Jumping***: This node is linked to the EnhancedInputAction IA_Jump node's "Canceled" output. If the input is canceled, the character's jump action is stopped. The "Target" input pin is connected back to itself, influencing the character that receives the input, much like the Jump node.

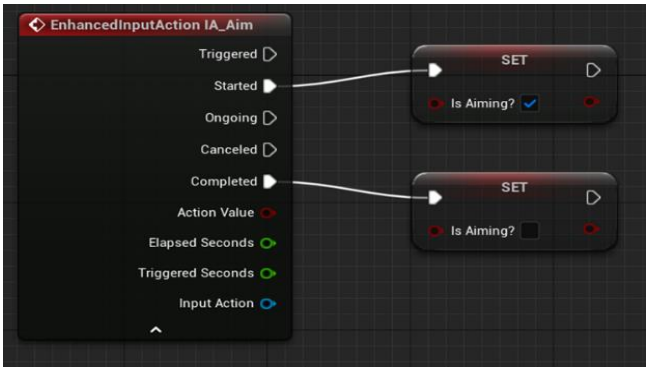


Fig. 10 Aiming

*Input Action Enhancement* ***IA_Aim***: An input action for aiming is represented by this node. It functions similarly to a listener, waiting for input from the player regarding the aim. The node has multiple outputs, each of which represents a distinct input action stage:

***Triggered***: When the aim input is first triggered, this output is turned on. In the event that the aim action is assigned to a mouse button, for instance, this output would become active at the player's initial button press.

***Started*:** As soon as the aim input action starts, this output is triggered. In certain scenarios, this could be the same as the triggered output; however, in other scenarios, the action's official start time could be determined by a delay or another condition.

***Ongoing***: As long as the aim input is active, this output will always be active. The player can continue to activate this output by holding down the button or key linked to the aim action.

***Canceled***: When the aim input action is canceled, this output is triggered. This could occur if the player leaves the button before the action is finished or if the action is interrupted by another game event.

***Completed***: Upon completion of the goal input action, this output becomes active. Usually, this occurs when the player leaves the button following a successful aim action.

***SET Nodes***: These nodes are employed to establish a variable's state. Here, they use the state of the input action to determine whether to set the value of "Is Aiming?" to true or false: The Enhanced Input Action IA_ Aim node's "Triggered" output is linked to the first SET node. This indicates that the "Is Aiming?" variable is set to true when the aim input is triggered. The Enhanced Input Action IA_ Aim node's "Canceled" and "Completed" outputs are connected to the second SET node. This indicates that "Is Aiming?" variable is set to false when the aim input is either canceled or finished.

***Elapsed Seconds and Triggered Seconds***: These outputs provide the temporal context of the input action. "Triggered Seconds" might indicate the moment the input action was initiated, while "Elapsed Seconds" could indicate the entire amount of time that has elapsed since the input action was initially triggered. Because of this configuration, the game can react to user inputs in real time and alter its state in response to what the user does. For instance, the game may zoom in on the camera to provide a closer look at the target if the user initiates the aim action. The game would zoom the camera back out if the player then aborted or finished the aim action.

## 6. CONCLUSION

The process of developing a 3D game with Unreal Engine is summarized in the research paper's conclusion, which also highlights the main conclusions and their implications for further study.

This study set out to investigate the potential of Unreal Engine in the context of creating 3D video games. The game created during this study demonstrated the Unreal Engine's strength and adaptability. World design, character animation, multiplayer networking, combat system development, opponent AI implementation, quest system integration, user interface (UI) design, sound and music integration, testing and debugging, and performance optimization are just a few of the game development elements it encompasses.

The developed game's performance was encouraging, proving how well the Unreal Engine handles challenging 3D game development requirements. The success of the game was further confirmed by user feedback, which showed that the functionality and design were well received.

The study's conclusions highlight how reliable the Unreal Engine is as a tool for creating video games. Its many

features and capabilities make it easier to create intricate and captivating 3D games. Furthermore, both inexperienced and seasoned developers can utilize the engine thanks to its comprehensive documentation and user-friendly UI.
There is a great deal of room for more research in this area in the future. One might investigate more complex features of Unreal Engine, like physics simulations, ray tracing, and VR integration. Studies comparing Unreal Engine to other game development engines could be carried out to determine their relative advantages and disadvantages.
To sum up, this study makes a substantial contribution to the game development industry. This opens the door for further study in addition to showcasing the Unreal Engine's usefulness in producing a 3D game. It promotes further research and creativity in the constantly changing field of game creation by showcasing Unreal Engine's capabilities. Thus, this study represents a first step in the direction of the gaming industry's development.

## 7. FUTURE SCOPE

Several new trends and technology will transform game creation in the future: Advances in technology The gaming scene is redefined by combining AI, blockchain, and AR with game production. These technologies represent fundamental alterations that shape game design, development, and experience.
Acquisition and Retention of Talent: In game development, talent recruiting and retention are major issues. A global labor shortage affects all industries, including game creation. Future efforts will focus on this issue. Changes in consumer tastes Game creation evolves with user tastes. Developers must adjust their games to these developments. Innovative business models: The free-to-play model and NFT integration are predicted to shape game development in the future.
Gaming Industry Inclusivity: Inclusivity is becoming more important. Future games may have more varied characters and storylines and be more accessible to all skills.
Virtual and Augmented Reality: VR and AR games will grow. This technology provides immersive experiences that traditional video games cannot equal. Mobile Gaming: This tendency is predicted to continue as mobile gaming becomes more popular. To expand their audience, developers must optimize games for mobile. The future of game creation is full of technical advancement, player-centric design, and industry change. From AI and AR integration to novel business models and inclusion, the gaming industry will change how we play, develop, and experience games.

## 8. ACKNOWLEDGMENTS

I would like to thank all the authors for their contributions and for the success of this manuscript and all the editors and anonymous reviewers of this manuscript.

## 9. CONFLICTS OF INTEREST

The authors declare no conflicts of interest.

## REFERENCES

1. Kensuke Yasufuku, Ginga Katou, & Sota Shoman. (2017). Game Engine (Unity, Unreal Engine). eizō jōhō media gakkaishi, 71(5), 353–357. doi:10.3169/itej.71.353
2. Andrew Rollings and Ernest Adams on Game Design, Publisher: New Riders Games, ISBN:978-1-59273-001-8, Published:01 May 2003, https://dl.acm.org/doi/book/10.5555/1213088
3. Anderson, E. F., Engel, S., Comninos, P., & McLoughlin, L. (2008, November). The case for research in game engine architecture. In Proceedings of the 2008 Conference on Future Play: Research, Play, Share (pp. 228-231).
4. Roger Eastman, “CMSC425 Game Programming Fall 2019” Course online, www.cs.umd.edu/class/fall2019/cmsc425/syllabus.shtml
5. Manojlovich, Joseph & Keeratiwintakorn, Phongsak & Hughes, Stephen & Chen, Jinlin & Lewis, Michael. (2003). UTSAF: Getting The Best of Consumer Graphics into Military Simulations. Proceedings of the Human Factors and Ergonomics Society Annual Meeting. 47. 10.1177/154193120304702010.
6. Torres-Ferreyros, Carlos & Festini-Wendorff, Matthew & Shiguihara, Pedro. (2016). Developing a videogame using unreal engine based on a four stages methodology. 14.10.1109/ANDESCON.2016.7836249.
7. Christos Gatzidis, Stuart Baker, A Review Of First-Person Shooter Game Engines And Their Use In Researching Scientific Disciplines, IADIS International Conference Gaming 2008, 2008, pp 67-72
8. X. Chen, M. Wang and Q. Wu, "Research and development of virtual reality game based on unreal engine 4," 2017 4th International Conference on Systems and Informatics (ICSAI), Hangzhou, China, 2017, pp. 1388-1393, doi: 10.1109/ICSAI.2017.8248503.
9. Al Lawati, H. A. J. (2020). The Path of UNITY or the Path of UNREAL? A Comparative Study on Suitability for Game Development. Journal of Student Research. https://doi.org/10.47611/jsr.vi.976
10. Telang, S., Patki, A., Kale, A., Jadhav, P., & Mujumdar, G. (IJRASAT). "Shooting Game Using Unreal Engine." Department of Computer Engineering, Pimpri Chinchwad Polytechnic, Akurdi, Pune-411044.
11. C. Vohera, H. Chheda, D. Chouhan, A. Desai and V. Jain, "Game Engine Architecture and Comparative Study of Different Game Engines," 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT), Kharagpur, India, 2021,pp. 1-6, doi: 10.1109/ICCCNT51525.2021.9579618.
12. Qiu, W., Yuille, A. (2016). Unreal CV: Connecting Computer Vision to Unreal Engine. In: Hua, G., Jégou, H. (eds) Computer Vision – ECCV 2016 Workshops. ECCV 2016. Lecture Notes in Computer Science(), vol 9915. Springer, Cham. https://doi.org/10.1007/978-3-319-49409 8_75
13. R. A. Boyd and S. E. Barbosa, "Reinforcement Learning for All: An Implementation Using Unreal Engine Blueprint," 2017 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 2017, pp. 787-792, doi: 10.1109/CSCI.2017.136.
14. Šmíd A. (2017), Comparison of Unity and Unreal Engine.
15. B. J. Geisler, F. J. Mitropoulos and S. Kavage, "GAMESPECT: Aspect Oriented Programming for a Video Game Engine using Meta-languages," 2019 SoutheastCon, Huntsville, AL, USA, 2019, pp. 1-8, doi: 10.1109/SoutheastCon42311.2019.9020369.
16. D. Ginchev and S. Stavrev, "A low-cost battle tank simulator using Unreal Engine 4 and open-hardware microcontrollers," 2020 XXIX International Scientific Conference Electronics (ET), Sozopol, Bulgaria, 2020, pp. 1-4, doi: 10.1109/ET50336.2020.9238266.

## AUTHORS:

Assistant Professor in Computer Technology **Dr. Nitesh Ghodichor** works for Priyadarshini College of Engineering, RTM Nagpur University, Nagpur. With a Ph.D. in Computer Science Engineering from Sarvepalli Radhakrishnan University,

Bhopal, India Dr. Ghodichor offers MANET Security and Blockchain knowledge abound. MANET Security investigating Blockchain and Cloud Computing technology to design a rules of Security interests their research. Before starting academics, Dr. Ghodichor worked as a Linux (RHCE) Network Administrator for some time. Their commitment lies in creating a dynamic classroom where they involve students in research projects and critical thinking exercises. Dr. Ghodichor, an emerging researcher, is dedicated to furthering Blockchain and Cloud Computing technology knowledge by means of publications and joint projects, thereby augmenting scholarly debate.

Corresponding Author:

E-mail: niteshgho@gmail.com

**Vaishali Ghodichor (Patil)** serves as an Assistant Professor in the Computer Science and Engineering department at G H Raisoni College of Engineering and Management in Nagpur, Maharashtra, India, while concurrently pursuing a Ph.D. in Computer Science Engineering at GH Raisoni Amaravati University, Amravati, Maharashtra, India. Her study focuses on Blockchain and Mobile Ad Hoc Network (MANET) Security. Before entering academics, Ms. Ghodichor acquired expertise as a Marketing Executive and Trainer at SCADA Training Institute. She is committed to cultivating an engaging educational atmosphere. Ms. Ghodichor, as an emerging scholar, is dedicated to finalizing her research in Computer Science and Engineering and enhancing academic discourse through publications and partnerships.

E-mail: vaishup2004@gmail.com

**Yash Nagpur** is received his Engineering Graduate (B.Tech) degree in Computer Technology from RTMNU Univessity Maharshtra in 2023. His areas of interest are cloud computing, and 3D game design on game platforms, and artificial intelligence & machine learning,

E-mail: yashnagpure6876@gmail.com

**Yashasvi Vairagade** Nagpur is currently engineering students of computer Tecchnology Department Third Year B. Tech. from RTMNU University, Maharshtra. I am active member of PCE Incubation center, Nagpur and also member of ACM students Chapter Nagpur Branch.

E-mail: yashasvivairagade6@gmail.com