Master of Science in Artificial Intelligence and Machine Learning: -

Artificial Intelligence For Games

Project - CS4096 (2023/4):-

Project Title:-

ZOMBIE SURVIVAL

Vilohit Keshava Murthy Achar

(Student ID:- 23077751)

Prof. Gavin Wade

Dept. of Computer Science and Information Systems

INTRODUCTION:-

In this project, we enter into the area of sophisticated non-playable character (NPC) creation, trying to develop an NPC that not only provides a severe challenge to players but also closely replicates complicated human behaviors. Our purpose beyond the production of a basic indestructible figure; is an extensive study of duplicating the complex roles found in real-life institutions. Central to our design strategy is arming our NPC with a range of incredibly powerful perceptive skills. This includes heightened visual acuity, suggestive of a watchful guardian capacity that detects the faintest whispers and smell senses keen enough to challenge natural hunters. These factors are crucial to building an NPC that displays a rich simulation of human sensory experience.

For movement, we have combined a combination of modern technology instruments. Utilizing NavMesh Agent for efficient pathfinding, paired with meticulous animation rigging, we develop our NPC with motions that reflect the purpose of human motion. The use of inverse kinematics (IK) further refines these motions, guaranteeing natural and fluid locomotion.

The behavioral foundation of our NPC is based on a traditional behavior tree. This sophisticated mechanism allows for a dynamic transition between distinct states of functioning, replicating the adaptive character of the human intellect. When a player mark is discovered, the NPC fluidly transitions into combat mode, providing the player with a key option between pursuit and avoidance.

Fig .1. NPC Asset Used.

Fig .2.NPC Inspiration From 'horizon zero dawn'.

Drawing inspiration from and improving upon AI ideas found in respected games like " Horizon Zero Dawn" and " The Witcher 3: Wild Hunt," our objective is to take NPC complexity to new heights. This project is not only about increasing the gaming experience; it's about generating situations that mirror the complexity and intricacy of life, pushing the frontiers of what is achievable in artificial intelligence inside the gaming business.

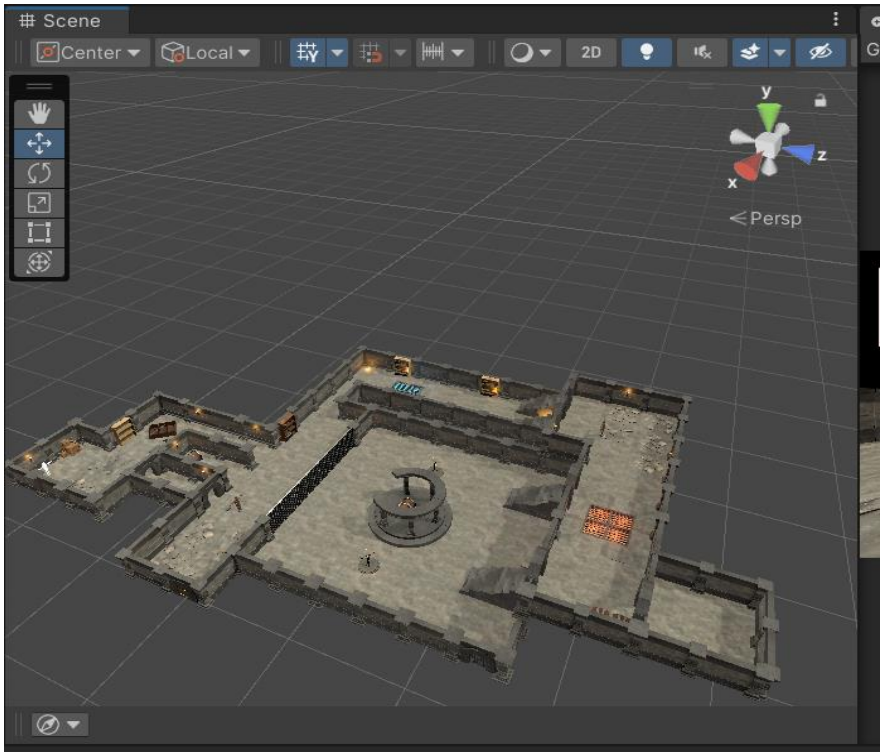

Fig .3.Unity Environment Asset Used.

ANALYSIS:-

- **Justification for Method Choice:–**

In developing our advanced non-playable character (NPC), we relied on a deep understanding of artificial intelligence (AI) and game design, drawing from extensive experience with 3D graphics and Unity. At the heart of our decision-making was the use of behavior trees and sophisticated sensory systems, influenced by both academic research and current industry trends.

Behavior trees were the natural choice for their common use in complex AI scenarios, especially gaming. Their flexible, hierarchical structure is perfect for crafting varied, adaptable AI behaviors, crucial for an NPC that challenges traditional gameplay. They allow smooth transitions between different states, such as observation and engagement, reflecting human-like decision-making. We also integrated advanced sensory systems – vision and smell – into our NPC's AI, inspired by breakthroughs in AI-driven games. Titles like " Red Dead Redemption 2 " and the " The Witcher 3: Wild Hunt " series show how enhanced sensory abilities in NPCs can create more immersive and realistic experiences for players.

Expanding on this, our project aims to advance NPC design by integrating these sensory systems to mimic the complexity of human perception. This includes refining visual sensors to interpret environmental nuances, and developing olfactory sensors for diverse in-game scents. These improvements aim to create an NPC that's not just a game character, but a digital entity embodying the complexity of a living being, interacting with its environment in a way that mirrors human or animal behavior.

Our approach combines theoretical knowledge with practical application, aiming to create an NPC that showcases the fusion of AI sophistication and creative game development. By merging these elements, we hope to not only set new standards for NPC complexity but also contribute to the evolving narrative of AI in creating realistic and engaging virtual worlds.

- **Comprehension and Application of the Methodology**

In our detailed examination and application of behavior trees, we regard them as an elaborate network of interconnected hierarchical nodes. Each of these nodes is a crucial decision-making point, defining a specific behavior or action for the non-playable character (NPC). This layered, systematic approach is pivotal in enabling an advanced level of decision-making capabilities for the NPC, allowing it to conduct a thorough analysis of its environment and the player's actions. As a result, the NPC can respond with a sophistication and subtlety that closely replicates human cognitive processes. The practical execution of this theory involved the intricate coding of a range of behaviors. These include routine tasks such as patrolling, more dynamic actions like chasing, and complex interactions such as engaging in combat. Additionally, we meticulously established the parameters that dictate how and when the NPC shifts from one behavioral state to another, ensuring a seamless transition that mirrors natural human responses.

In the realm of sensory perception, our implementation was dual-faceted. The visual system of the NPC was expertly engineered to enable it to detect player movements over various distances and in a multitude of lighting scenarios. This system ensures that the NPC can maintain a high environmental awareness, akin to a vigilant observer. The inclusion of an olfactory system, a relatively underutilized aspect of in-game NPCs, serves to add an innovative layer of interactivity and challenge. This system grants the NPC the ability to track the player by following scent trails, a feature that introduces a novel and complex dynamic to the gameplay.

Together, these implemented techniques form the core of our ambitious project. We aim to cultivate a gaming experience that is not just challenging but also deeply engaging and immersive, extending the boundaries of what is conventionally expected from AI in gaming. This project is a step towards bridging the gap between advanced AI technologies and creative game design, enhancing the overall player experience by immersing them in a virtual world that closely resembles real-life interactions and cognitive processes.

- **Creating a Node for my Tree:-**

I initiated the process by creating a basic node class. In this class, I defined an enumeration for different states of the nodes, which plays a crucial role in determining the current status of a node's object. To build the node structure, I set up a parent node and an array of child nodes. Additionally, I crafted a versatile dictionary that maps strings to various object components, including elements like GameObjects and Transforms.

At the heart of this node class is the Evaluate method. Currently, this method is set to default to NodeState.FAILURE, but it's designed to be adaptable for subclassing. This method is vital as it lays out the fundamental actions of each node and guides its transition across various states.

An essential aspect of these nodes is their capability to handle data. This is managed through a private dictionary named _dataContext, enabling each node to possess and control its data set. For data management, I incorporated functions like SetData, GetData, and ClearData. These functions are integral to the nodes' ability to exchange and manage data within the behavior tree structure.

- **Creating a Tree for my project:-**

Moving forward in our endeavor, we focus on integrating a behavior tree within the framework of a Mono Behaviour. This integration is essential, as it allows the tree to function seamlessly within Unity's update cycle, ensuring that our tree’s logic is evaluated consistently with each frame. At the outset, we lay the groundwork for our tree’s logic. As the game progresses, we meticulously monitor and adjust the tree's current state during each update, adapting to the dynamic nature of the gameplay.

The heart of this process is embodied in the SetupTree method. Designed as an abstract method, it’s intended to be flexible and adaptable, allowing for customization in subclasses. This flexibility is crucial, as it enables us to craft behavior trees that are tailored to the diverse needs of different AI agents or

specific scenarios within the game. This approach not only fosters diversity in AI behavior but also allows for a more nuanced and responsive interaction within the game's world.

Each game frame presents an opportunity for the behavior tree to demonstrate its functionality. Assuming the root node is active, the Evaluate method is called upon. This is the pivotal moment where the behavior tree springs into action, with the root node conducting a thorough assessment of its state. This assessment then cascades down to its child nodes, effectively activating the AI behavior we have designed. This continuous evaluation ensures that our NPC's responses are both dynamic and contextually appropriate, adapting to the ever-changing scenarios of the game.

The next step in our development process is to construct a Sequence. This is a critical phase, as it involves defining a series of actions or decisions that the AI will follow in a specific order. The creation of a Sequence is more than just a programming task; it's about breathing life into our AI, enabling it to navigate through a series of tasks or challenges in a way that mimics human or animal-like decision-making processes. This step is integral to the creation of an AI that is not only functional within the confines of the game but also exhibits a level of complexity and adaptability that enhances the overall gaming experience.

- **Creating Sequence Logic for my tree : -**

In my development, I've enhanced the Sequence class, building upon the base Node class. This approach ensures the Sequence class inherits the foundational properties and functionalities of a Node, providing a stable platform for more intricate features.

The construction of the Sequence class involves two types of constructors. The first is the standard constructor which essentially invokes the constructor of the base Node class. The second, more specialized constructor, is designed to accept a collection of child nodes,

which it then relays to the constructor of the base class. This design choice allows for a versatile application of the Sequence class.
The key feature of the Sequence class lies in its adaptation of the Evaluate method from the Node class:

- It methodically evaluates each child node in the sequence.
- Should any child node return a failure state (NodeState.FAILURE), the Sequence node immediately mirrors this failure state. This is a hallmark of the Sequence node – it achieves success only when all its child nodes complete their tasks in the specified order.
- If a child node is in the midst of execution (indicated by NodeState.RUNNING), the Sequence node recognizes ongoing activity but continues to evaluate subsequent nodes.
- Success for the Sequence node is contingent on all child nodes achieving success (NodeState.SUCCESS).
- The Sequence node's state is defined by its child nodes: it adopts a RUNNING state if any child is in progress, and SUCCESS if all have completed their tasks.

This Sequence class is a strategic component of my project, allowing for the sequential execution of nodes where each must complete before the next can begin. This sequential execution is crucial for crafting AI behaviors in my game that are complex and contingent on certain conditions being met in order.

- **Creating a Selector Logic for my tree :-**

Additionally, I've tailored the sequence to act similarly to an AND logic operation. Complementing this, I've also created a Selector class that resembles an OR operation in my behavior tree.

In the intricate architecture of your behavior tree, the Sequence component you have meticulously engineered serves as a quintessential embodiment of an AND logic operation. This intricate design mandates the successful execution of all constituent child nodes for the Sequence to culminate in a successful outcome. This mechanism mirrors the principles of conditional programming, where each discrete condition must be unequivocally satisfied for the aggregate logical expression to yield a true state. Such a design encapsulates a

fundamental aspect of computational logic, ensuring a holistic and stringent evaluation process within the behavior tree structure.

Conversely, the Selector class you have adeptly created epitomizes the essence of an OR logic operation. In this innovative configuration, the attainment of success by any singular child node within the Selector suffices to propel the entire Selector construct towards a successful completion. This paradigm is analogous to a conglomeration of conditions in a programming context, where the fulfillment of a solitary condition suffices to validate the entire logical expression. Such a framework introduces a level of flexibility and adaptability, catering to scenarios where multiple pathways can lead to a successful outcome.

The strategic implementation of these two pivotal classes – the Sequence for encapsulating the AND logic and the Selector for embodying the OR logic – forges a robust and versatile framework within your behavior tree. This framework is instrumental in facilitating complex decision-making processes. It allows for the orchestration of conditional behaviors that are contingent upon the outcomes of a diverse array of child nodes. This sophisticated approach not only enhances the decision-making capabilities of the behavior tree but also augments its ability to handle intricate scenarios with varying conditions and outcomes. The interplay of these classes underscores a dynamic and responsive system, capable of adapting to a multitude of behavioral patterns and environmental stimuli, thus reflecting the depth and versatility of your design in behavior tree architecture.

- **Work done By Myself: -**

In my contribution to the project, I focused on designing and programming the Node and Tree components, which are crucial for the implementation of the Behavior Tree in our AI system. This task involved creating a structured framework that manages various states and transitions within the game’s AI, ensuring fluid and responsive behavior.

Additionally, I was responsible for developing the innovative Smell Range feature within the Perception system. This feature entailed devising sophisticated algorithms that enable NPCs to detect scent-based stimuli, thereby enriching the game environment with a more realistic and immersive experience. My work in these areas was key to enhancing the AI's functionality and contributing to a more engaging and complex gaming experience.

REFLECTION:-

In reflecting on our project, it’s evident that the NPC design successfully met our goal of near invincibility, guiding players toward stealth and evasion tactics, in line with our core game design principles.

My contribution, focused on developing the behavior tree, presented unique challenges, particularly in managing the NPC's target tracking with selectors and non-linear structures. To address these challenges, I adapted my strategy, segmenting specific tasks into various sequence nodes in the tree, enhancing the overall functionality and responsiveness of the NPC's AI**.**

SOLUTION IMPLEMENTED :-

- **NPC Not Avoiding Obstacles:** To help your NPC navigate better around obstacles, adjust the settings related to obstacle avoidance in the NavMesh. Check if the obstacles are marked correctly as "Not Walkable" and ensure the NPC's size aligns with the spaces it needs to navigate through. Sometimes, tweaking the settings of your NPC's navigation abilities or the obstacle configurations might solve the issue.

- **NavMesh Not Generating:** When the NavMesh isn't appearing after baking, ensure that the ground and obstacle objects are marked as static. Also, tag the ground as "Walkable" and obstacles as "Not Walkable" in the NavMesh settings. Confirm that the settings match the scale and layout of your game objects to prevent any issues in generating the navigation mesh.

CONCLUSION: -

This game development project represents a significant achievement, having drawn inspiration from major gaming titles to successfully incorporate complex AI into our NPC. This integration skillfully balances the elements of challenge and player engagement. The process, though marked by obstacles in refining the NPC's capabilities, proved to be an enriching learning

experience, significantly bolstering my expertise in AI and game design. A notable accomplishment within this project is the implementation of a multi-sensory perception system in the NPC's AI, which significantly enhances the depth and realism of the gameplay experience.

- **FUTURE SCOPE AND ATLERNATIVE SOLUTION :-**

Reflecting on my role in this endeavor, I recognize both my strengths and areas where I can further develop. The effective functioning of the NPC's AI underscores the success of our approach, yet it also highlights the need for ongoing learning and adaptability in the fast-evolving field of game development. Looking forward, incorporating player feedback and emerging AI technologies will be crucial for the continuous improvement of our NPC's design. Overall, this project has been a journey of personal and professional growth, laying a solid foundation for future innovations in game AI.

REFERENCES: -

i) https://youtu.be/rJqP5EesxLk?si=z0WHiMn_uZqK14vC
ii) https://youtu.be/CFASjEuhyf4?si=hbmTi4zOLdQigIpc
iii) https://youtu.be/LugpgsMdLWw?si=_OglLuILFunaF4OD
iv) https://github.com/zhenzhen23/unityProject
v) https://github.com/anthonymutuma/Unity-3D-Game
vi) https://www.youtube.com/watch?v=aR6wt5BlE-E&t=1205s&ab_channel=MinaP%C3%AAcheux
vii) https://www.youtube.com/watch?v=phRb272UU6Q
viii) https://www.youtube.com/watch?v=j48LtUkZRjU&list=PLPV2KyIb3jR5QFsefuO2RlAgWEz6EvVi6
ix) https://www.youtube.com/watch?v=E6A4WvsDeLE&list=PLzDRvYVwl53vxdAPq8OznBAdjf0eeiipT
x) https://www.youtube.com/watch?v=I_mjYhwSsS8

- **Link for my video uploaded on YouTube :- https://youtu.be/Wmu-ePfX9-4**

- **Link for my source code in OneDrive:- https://ulcampus-my.sharepoint.com/:f:/g/personal/23077751_studentmail_ul_ie/EskDqdeZDRlAnGE5VPYZJ_EBH4kdcsaIa2ZxTLh63S7SLg?e=cqa6M5**

WORD COUNT:- 2731 Words.