Shadowborn Hunter – Enemy AI Implementation Using Behaviour Trees

CS4096 - ARTIFICIAL INTELLIGENCE FOR GAMES 2025/6 SEM1

Student Name: KALYAAN KOTHANDA RAM
Student ID: 25012797
Word Count: 1735

Introduction

This report presents the design and implementation of an artificial intelligence (AI) system for the individual project Shadowborn Hunter, that I developed as part of the CS4096 – Artificial Intelligence for Games module. This game is a 2D platformer where the player controls the human character who has to jump between platforms, collect coins simultaneously should avoid falling, and reach the end portal to complete the level. The platforms are stable brick platforms which doesn't collapse and magical platforms that collapse a few seconds after stepping on it, so it required constant movement and proper timing by the player. So the player must run away and survive from an enemy monster who actively chases the player and attacks him using fireball projectiles. The player begins the game with 100 health points, when he is hit from a fireball it reduces his health by 10 hp. So basically if the player falls down or his health reduces to zero, you lose the game.

So the problem that is addressed in this project was creating an enemy that behaves intelligently rather than following a fixed movement pattern. Many traditional platformer games have had enemies that simply move back and forth along fixed paths, which makes them predictable and offers the players limited challenge. So here it makes sure, the enemy in Shadowborn Hunter actively detects the player, updates internal memory, and adjusts its behaviour accordingly. This is what makes the gameplay more engaging and creates a sense of pressure by forcing the player to react rather than to memorize the enemy's pattern.

Research shows that "behaviour trees became widely used in game AI because they handle complexity better than finite state machines" (Isla, 2005, GDC Talk Handling Complexity in Halo 2 AI). This supports the decision to use a behaviour tree instead of a simple state machine. Similarly, Blackboard architectures give

all the AI parts one shared place to store and read information, so they don't have to depend on each other directly, because of which the system becomes easier to update and add new features to.

This report mainly presents an analysis of the AI techniques used, explains how they were implemented, evaluates the resulting behaviour, and reflects on limitations and potential improvements. The aim is to demonstrate understanding of game AI concepts and justify the choices made during development.

Analysis

The core AI system in my game is structured around three key elements i.e. a behaviour tree, a blackboard memory component, and a perception system. All these elements that I implemented, work together to allow the enemy (monster) to observe the player, make decisions based on the updated information, and execute appropriate actions such as chasing or shooting.

- **Behaviour Tree**

I chose the behaviour tree method for this project because it is easy to understand, and it lets me break the AI's actions into small and manageable parts. "Behaviour trees scale more effectively than finite state machines as complexity increases" (Isla, 2005, GDC Talk Handling Complexity in Halo 2 AI). This makes it easier to expand when compared of using simple state machines, which may become difficult to manage as more states are added.

In this game, the behaviour tree consists of a root selector node that has two main sequences, one is the engage sequence and the other is the search sequence. The engage sequence is the one that's always checked first because it has higher priority. So It begins by checking whether the player is visible or not. If the player

is visible, the enemy moves towards the player and turns on the shooting behaviour, this makes the enemy actively chase and attack the player.

If the player runs too far away from the enemy and goes out of the vision range of the enemy, the engage sequence fails and when this happens, the selector activates the search sequence. The search sequence stops the shooting and makes the enemy go to the last position where the player was seen. Even though the game has no walls to hide behind, this still works because the enemy loses the player only when they go outside the vision radius.

This way it is structured and it ensures that chasing and shooting always have higher priority, and searching only happens when the player moves too far away.

Blackboard System

The blackboard system acts as shared memory between different AI components. It stores data such as the player reference, visibility status, last seen position, last seen time, and estimated velocity. Components such as the vision script and shooting script reads from and write to the blackboard, ensuring that all AI behaviours operate using proper information.

Research shows that "blackboard architectures reduce coupling between AI components" (Blackboard System Overview, 2025, Wikipedia), which simplifies communication and makes the system easier to extend. Using a blackboard made it easier for the movement part and the shooting part to work on their own. They did not have the need to depend on each other, but they could still change their actions because they were both looking at the same player information on the blackboard.

Perception System

The perception system checks whether the enemy can detect the player. Then it starts to calculate the distance between the enemy and the player, and if the player is found to be within a specific range, the system updates the blackboard with the player's position and the predicted velocity. If the player leaves this range, the system marks the player as not visible.

This design felt very simple but effective for a 2D platformer. It gives the AI a steady signal for when to change what it's doing, helping it switch smoothly between chasing the player and looking for them.

Predictive Shooting

One of the most important gameplay affecting AI features is predictive shooting. Research shows that " You have to take into account the speed of the bullet when predicting where the target will be" (Unity Discussions – 2D AI: How to get the AI to aim ahead of its target). So shooting script creates a fireball and assigns it a velocity that is directed towards the predicted position, increasing the likelihood that the fireball will hit a moving target.

This makes the game combat more challenging and encourages the player to do different movement constantly.

- **Justification of Combined AI Techniques**

So all these systems work together using the behaviour tree. When the player is visible, the tree makes sure it enable both movement toward the player and shooting and when the player is not visible, the tree disables shooting and triggers movement toward the last seen location of the player . Many sources highlight

that "AI enhances realism by creating intelligent game characters and opponents that mimic human-like behaviour" (Nadiy, 2024, Lizard Global Blog). This supports the goal of Shadowborn Hunter, where enemy behaviour has to feel reactive and engaging.

Reflection

Because of the implemented AI techniques the game achieved the desired outcome of creating a reactive and challenging enemy. The enemy (monster) successfully detects the player, moves towards him, and attacks using predictive shooting. The behaviour tree structure made the AI easier to understand and expand during development.

One of the main strengths of the AI in the game was the interaction between perception and memory. When the player suddenly moves out of range, the enemy does not stop immediately. Instead, it attempts to search for the player by moving to the last known position stored in the blackboard. As developers commonly implement memory based search behaviours using the last known player position. Unity community discussions note that "AI systems track the last player sighting and continue navigating toward that location" (Fox-Handler & TonyLi, 2014, Unity Discussions).

However, several issues and limitations were identified. One of the most notable limitation is the lack of line-of-sight detection. Because the perception system only checks distance, the enemy can detect and shoot the player through walls or platforms. This reduces realism and can feel unfair to the player. A potential improvement would be implementing a raycast to check whether obstacles block the enemy’s view. Research from Unity developers shows that line-of-sight checks using raycasts are important for fair and realistic enemy perception, as

distance-only detection can allow AI to see through obstacles (Unity AI Sight Thread, 2014, Unity Forum).

Another issue involves predictive shooting accuracy. While predictive shooting improves challenge, it can sometimes appear too accurate, especially when the player is moving in a straight line. This can make the game feel unbalanced and reduce the player's sense of control.

During development, several difficulties were encountered. Integrating multiple AI techniques required careful coordination, particularly ensuring that movement and shooting did not conflict. Debugging perception and predicting player velocity also required experimentation, especially when the player moved quickly between platforms. Another challenge was balancing difficulty, as small adjustments to movement speed or shooting frequency significantly changed gameplay.

This project also improved my confidence in designing and debugging AI behaviour, especially in integrating multiple systems such as perception, memory, and decision making. Understanding how these systems interact gave me practical insight into game AI development that I did not have before.

Future improvements could include adding more behaviour branches, such as patrolling when the player has not been seen for a long time, retreating when low on health, or attacking in different patterns. Implementing obstacle-based perception and more advanced movement logic would make the enemy feel more realistic. Multiple enemies with different AI behaviours could also increase gameplay depth. Industry research highlights that "predictive gameplay systems adapt to player behaviour in real-time to maintain engagement and retention" (Smartico, 2025, Smartico.ai).

Conclusion

This report presents the AI implementation for my game Shadowborn Hunter developed for the CS4096 – Artificial Intelligence for Games module. The project successfully applied behaviour trees, blackboard memory, perception, and predictive shooting to create a reactive enemy that challenges the player.

The AI demonstrated the ability to detect the player, pursue them, and attack effectively. The integration of perception and memory contributed to a more engaging experience. Several limitations were identified, particularly the lack of obstacle-based vision and sometimes overly accurate shooting. Future improvements could address these issues and expand the behaviour set.

Overall, this project strengthened my understanding of AI techniques used in games and provided valuable experience in applying them to a real-time interactive system. This experience also increased my confidence in developing AI behaviour and solving practical implementation challenges.

References

- (Isla, 2005, GDC Talk Handling Complexity in Halo 2 AI)

https://www.gamedeveloper.com/programming/gdc-2005-proceeding-handling-complexity-in-the-i-halo-2-i-ai

- (Blackboard System Overview, 2025, *Wikipedia*)

https://en.wikipedia.org/wiki/Blackboard_system

- (Unity Discussions – 2D AI: How to get the AI to aim ahead of its target)

https://discussions.unity.com/t/2d-ai-how-to-get-the-ai-to-aim-ahead-of-its-target/137441

- (Nadiy, 2024, Lizard Global Blog)

https://www.lizard.global/en/blog/ai-in-gaming-how-ai-is-used-to-create-intelligent-game-characters-opponents

- (Fox-Handler & TonyLi, 2014, Unity Discussions).

https://discussions.unity.com/t/smarter-ai-searching-for-the-player/546179

- (Unity AI Sight Thread, 2014, Unity Forum)

https://discussions.unity.com/t/ai-sight-go-away-from-raycasting-line-of-sight/525676

- (Smartico, 2025, Smartico.ai)

https://www.smartico.ai/blog-post/predictive-gameplay-mechanics-ai-that-adapts-games-to-player-behavior-in-real-time

Link For The Source Code :

https://drive.google.com/file/d/1BQDyIbWmeIQDmC1ChLcxAipeGKsLFjGH/view?usp=sharing

Link For The Video:

https://youtu.be/duR0QBs0w_Q