Project Presentation of Industry Oriented Hands on Experience (CS-253)

On

Zombie Survival: Last Stand

Harkabeer Singh
2110991933

Supervised By
Mr. Naveen Kumar

Department of Computer Science and Engineering,
Chitkara University, Punjab

Introduction

Zombie Survival: Last Stand is a single-player survival game where the player must fight off waves of zombies using ranged weapons. The game **features realistic enemy AI**, health mechanics, and **combat physics** to make the survival experience engaging and challenging.

The game is designed to demonstrate **AI-driven character** behavior without using Unity's NavMesh, instead relying on **physics-based AI movement.** This ensures **smoother performance and more dynamic zombie behavior.** The primary goal of this project is to implement real-time enemy AI, functional shooting mechanics, health systems, and interactive animations, all while maintaining optimal performance and code efficiency.

Additionally, this project serves as a learning experience in game programming, animation blending, and AI behavior development, which are crucial skills in professional game development.

Methodology & Technology used for implementation (Front End and Back End Utilization)

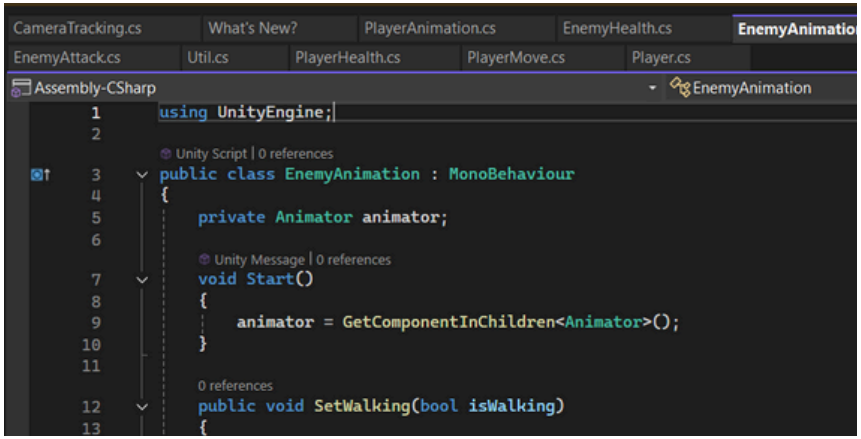
1. Front-End (VR Interaction & UI Implementation)

The front-end of the game is developed using the **Unity Engine,** which handles **game physics, animations, and player interactions**. The **C# programming language** is used to write scripts for player movement, enemy behavior, health management, and combat interactions. The user experience is enhanced by smooth animations, camera effects, and weapon mechanics to create an immersive combat system.

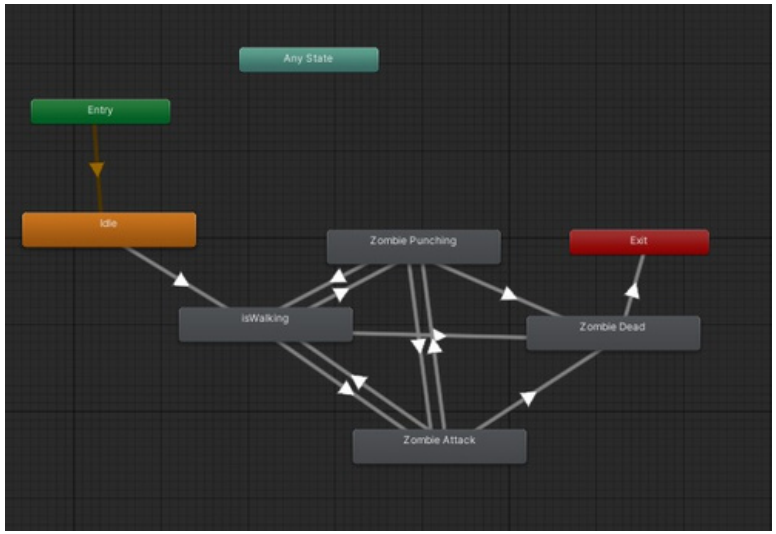
Methodology & Technology used for implementation (Front End and Back End Utilization)

2. Back-End (Data Storage & Authentication)

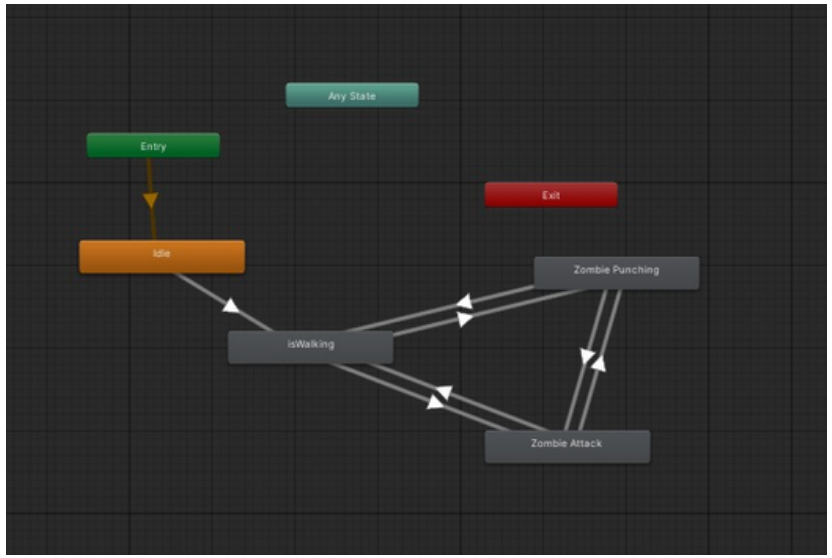
The **back-end** consists of **AI-driven enemy movement**, which is managed using a state machine (Idle → Walk → Attack → Die). Instead of relying on Unity's NavMesh, the zombies use velocity-based movement with physics interactions, ensuring more natural and unpredictable behavior. The enemy's attack patterns, detection radius, and animations are handled via script-based logic, making them adaptable to different game scenarios.



Unity Animator System



Zombie Animation System



Player Animation System

Project Functionality (Database Design, Authentication and Authorization, Framework Implementation)

1. Database Design & Management

- Currently, the project does not use a database, but future iterations may integrate Firebase for storing user data.
- A planned leaderboard system will track player scores and session data in real-time.

2. Authentication & Authorization

- If multiplayer features are added, Firebase Authentication will be implemented for secure user login.
- Future plans include role-based access for different player profiles.

3. Frameworks & Libraries Used

- Unity Engine, C# Programming, Unity Animator System, Unity Physics engine.

Project Functionality

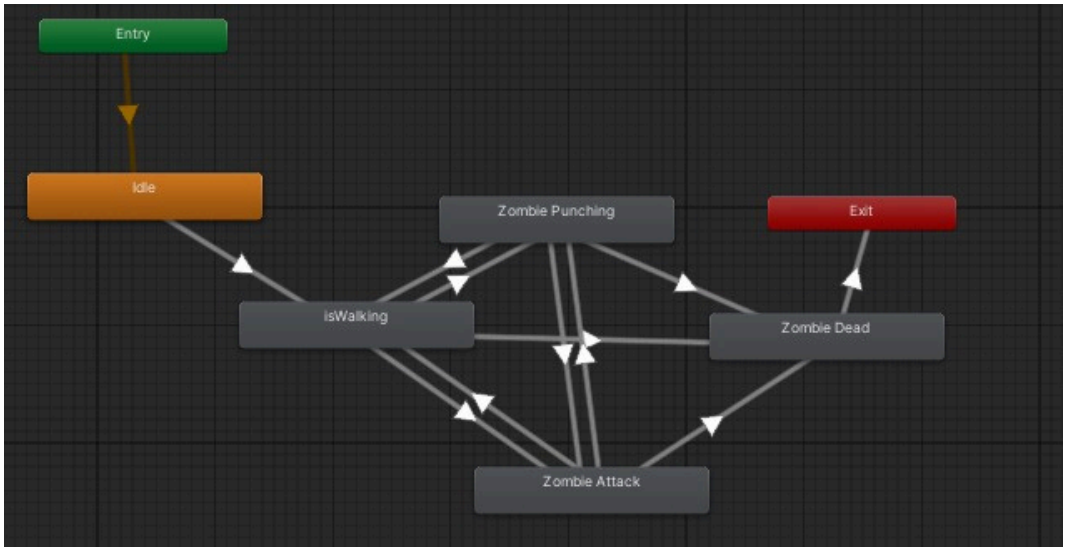
Game Mechanics Overview:

1. Player Movement & Controls: WASD controls, aiming with the mouse, shooting mechanics.
2. Zombie AI & Pathfinding: Enemies detect the player and chase them, switching between states dynamically.
3. Health System: Both player and zombies have health values that decrease upon taking damage.
4. Weapon System: Player can shoot bullets, dealing damage and triggering animations.

Framework Implementation

This project heavily utilizes state machines and Unity's physics system for AI behavior instead of traditional pathfinding methods. Key framework elements include:

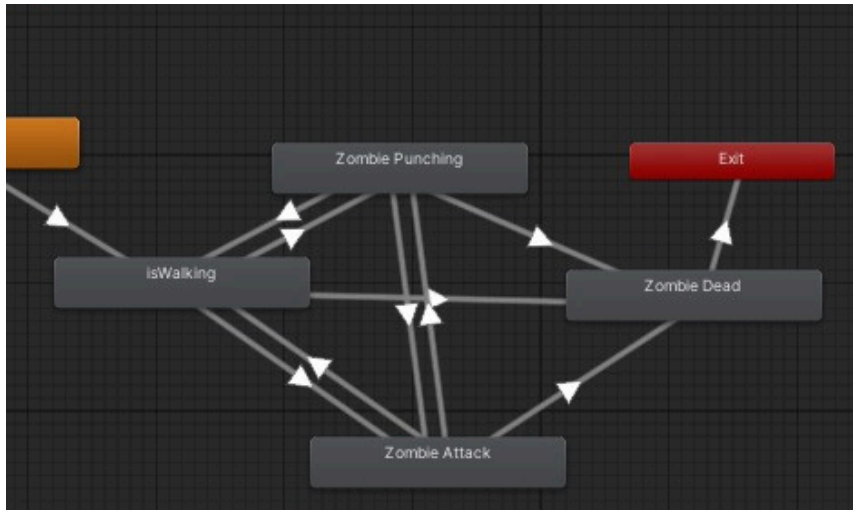
1. **Animation System:** The Unity Animator controls smooth transitions between Idle, Walking, Shooting, and Dying animations.



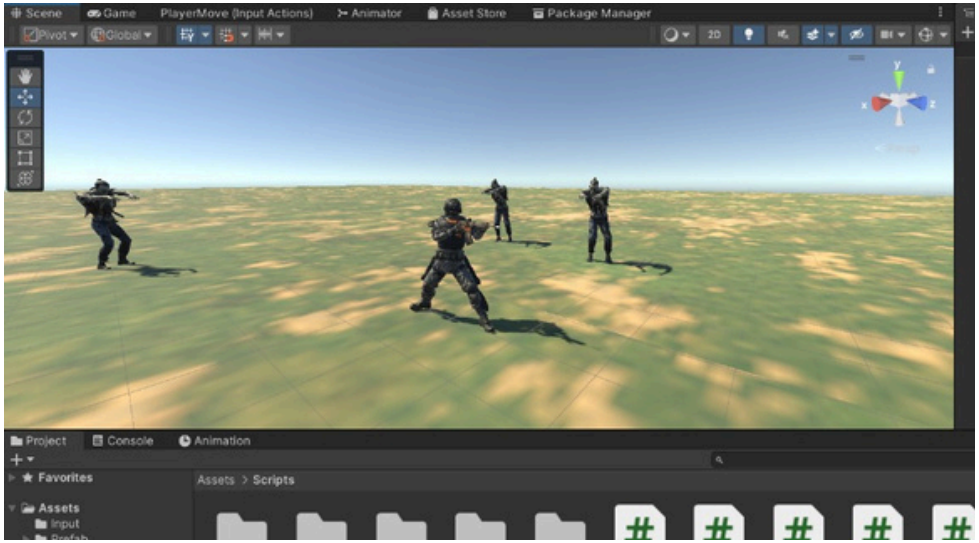
Framework Implementation

2. State Machine for Enemy AI: The zombies switch between different states dynamically:

- Walking State: Moves toward the player.
- Attacking State: Stops and attacks when in range.
- Dead State: Plays the death animation, then removes the zombie from the game.



Framework Implementation

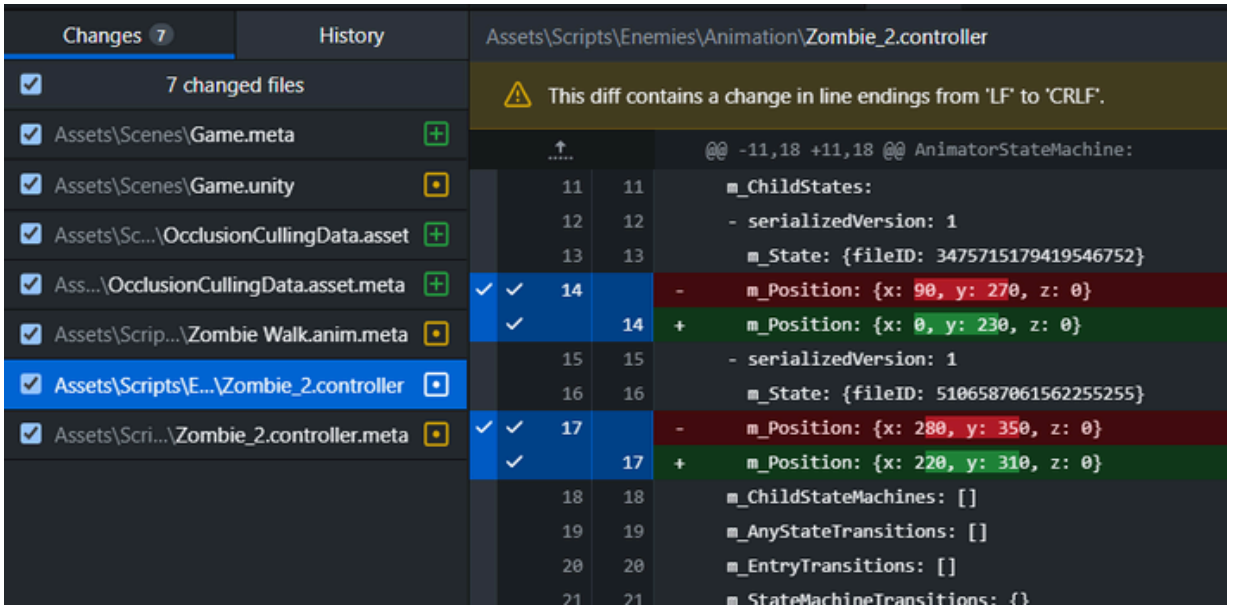


3. Physics-Based Enemy Movement: Instead of using NavMesh, the enemy moves based on calculated velocity and direction, making behavior less predictable and more engaging.

Project Utility (Unit Testing, GitHub & Debugging)

1. GitHub for Collaboration

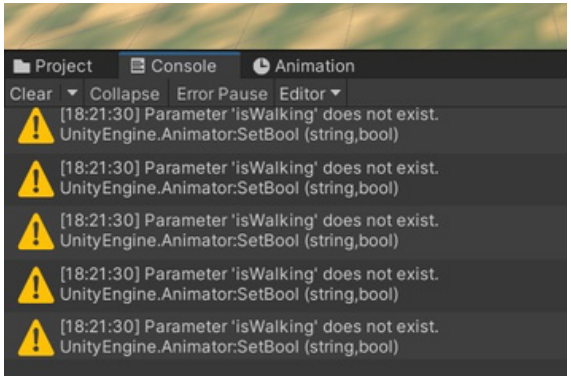
- Used for version control, feature tracking, and bug fixes.
- Utilizing Github features for Unity collaboration.



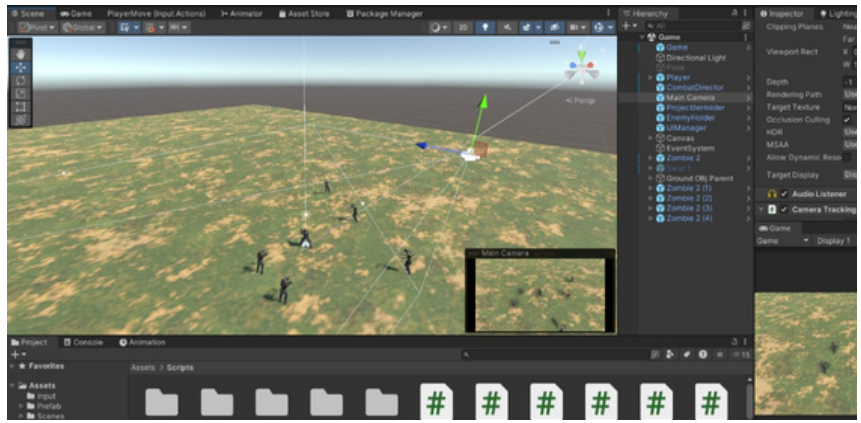
2. Testing & Debugging

- Unit tests are essential to ensure smooth game functionality. In this project, testing includes:

- **AI Behavior:** Ensuring zombies properly switch states (Idle → Walking → Attacking).
- **Player Combat:** Making sure bullets correctly reduce enemy health.
- **Animation Testing:** Ensuring correct animation transitions occur when needed.



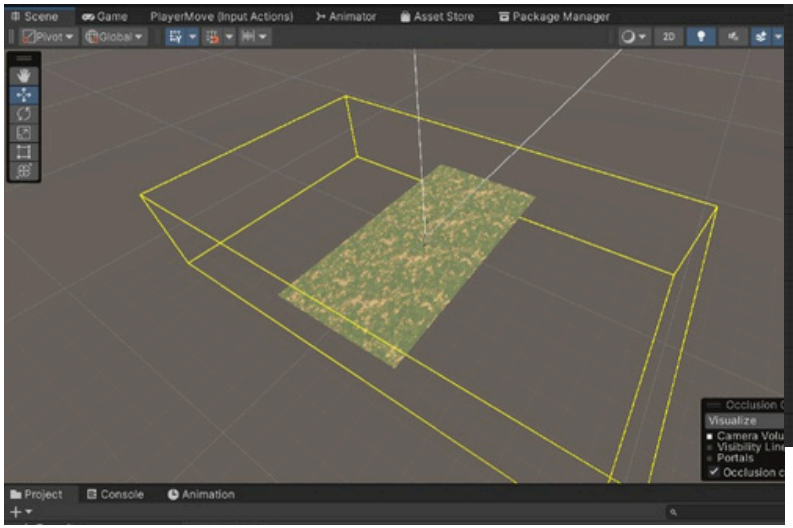
Unity Debug Console

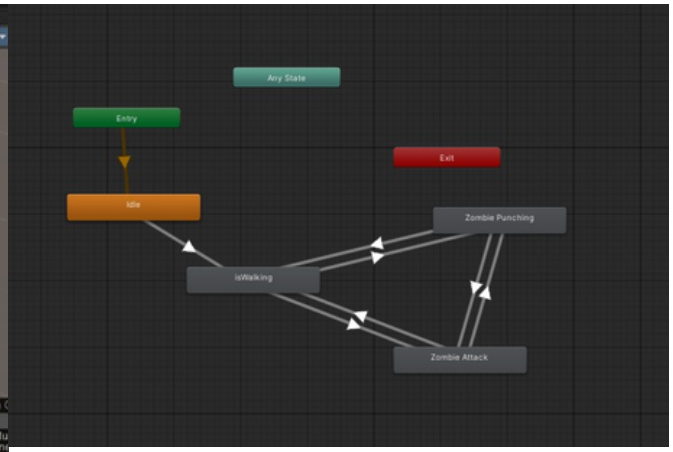


Play Mode Testing

Future Enhancements

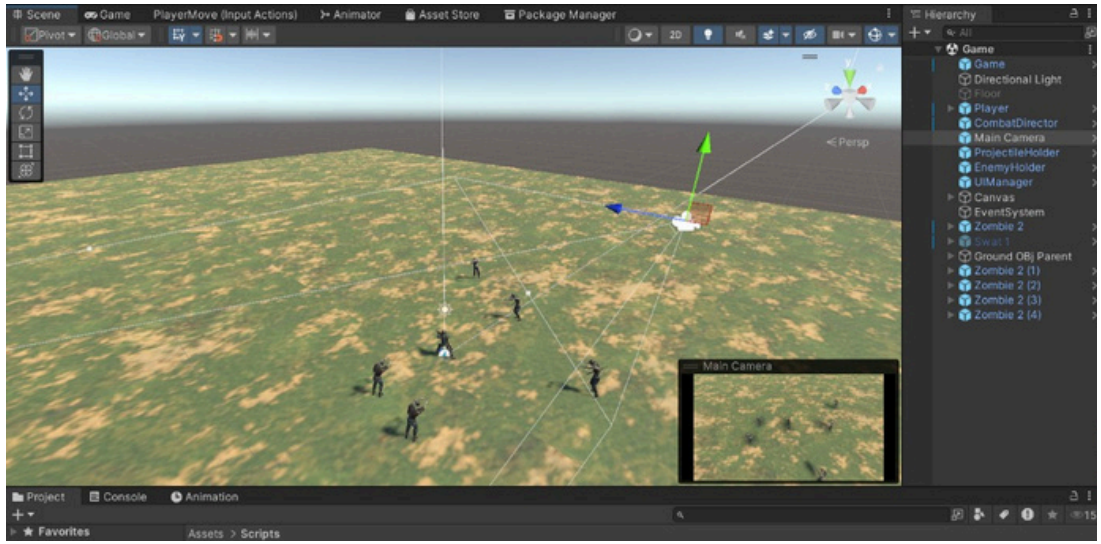
- NavMesh AI for Smarter Enemy Movement.
- More Zombie Variants with Different Behaviors.
- Multiplayer Mode with Online Leaderboards.
- Weapon Upgrades & Inventory System.
- Optimizing for larger maps.





Demonstration skill & Visual Aids

- More of the game



Conclusion

The project successfully demonstrates AI-driven enemy behavior, smooth animations, shooting mechanics, and physics-based interactions. Future improvements will expand on game mechanics, AI complexity, and multiplayer functionality. The project's success highlights the power of Unity and C# in creating engaging and interactive game experiences.

CHITKARA UNIVERSITY

Thank You

by

Harkabeer Singh
2110991933