ACTION 3D HORROR GAME

Venkata Narasimha Aditya Nyayapati - 11553225
Naveed Aqeel - 11549168
Geethika Parvataneni - 11479145

Course Title: SPECIAL PROBLEMS

Course Number: INFO 5900

Faculty: Prof. Sharad Sharma

Abstract: The Action 3D Horror Game is a virtual reality game that showcases a haunted house environment where the player must survive and escape from the horror creatures. The game is designed for horror game enthusiasts who enjoy experiencing the thrill of the horror genre. The application is intended to provide a unique and immersive experience to the users by allowing them to explore the haunted house and interact with the environment. The project is important because it provides a new experience in gaming and explores the capabilities of virtual reality technology.

Introduction:

Goal:

The goal of the project is to create an action horror 3D game that immerses the player in a scary and intense experience. The environment is a haunted house that is designed to be eerie and suspenseful. The game allows the player to navigate through the environment while encountering horror creatures and obstacles. The different aspects of the environment include the haunted house building, trees, furniture, landscaping elements, and interactive objects.

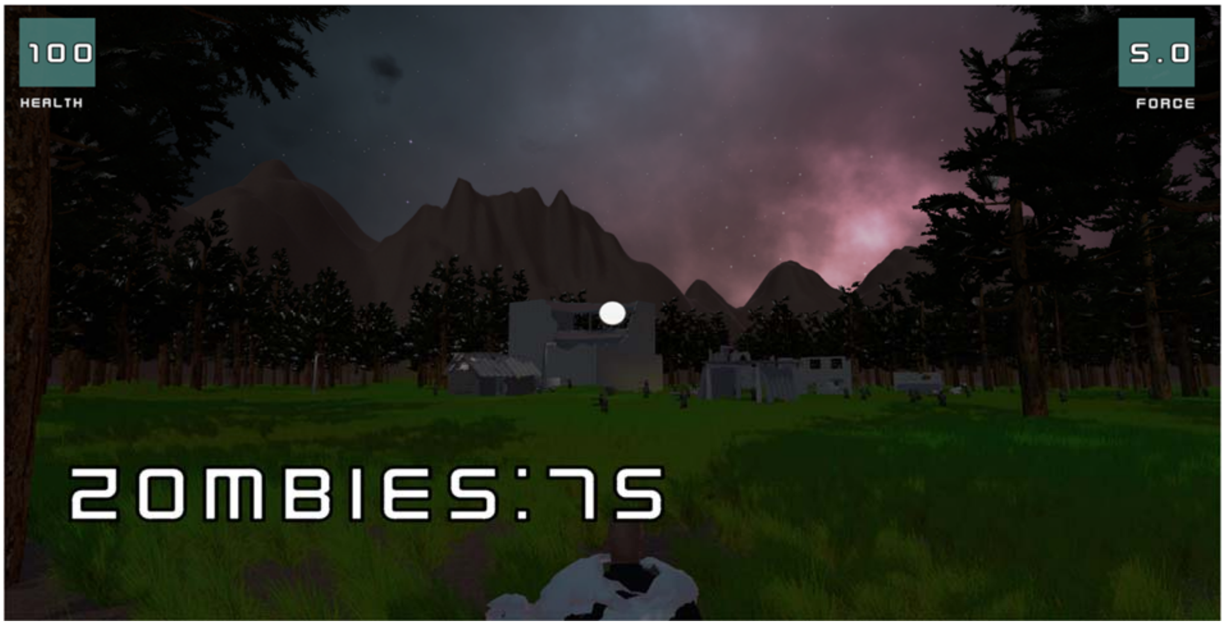
Objectives:

The objectives are to:

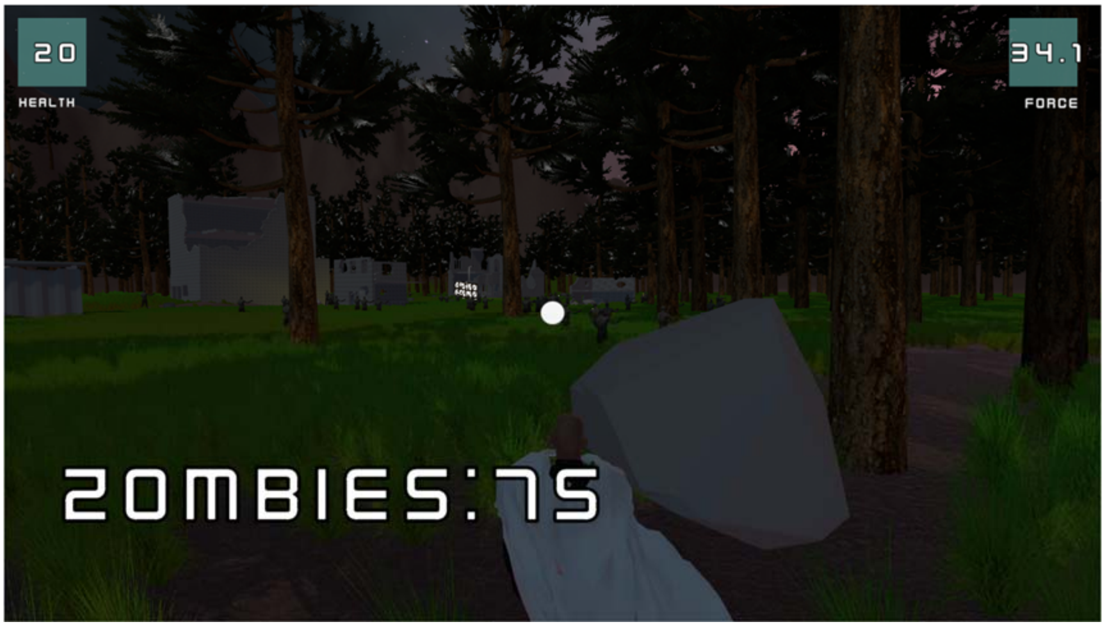
1. Create a detailed and immersive virtual environment that represents a dark and spooky location, such as a haunted house or abandoned asylum.
2. Implement a variety of interactive elements, such as weapons (boxes, stones, and barrels), and enemies, that challenge the player and increase the tension and fear using mouse and keyboard press.
3. Implement high-quality graphics, sound effects, and music to create a realistic and frightening atmosphere.

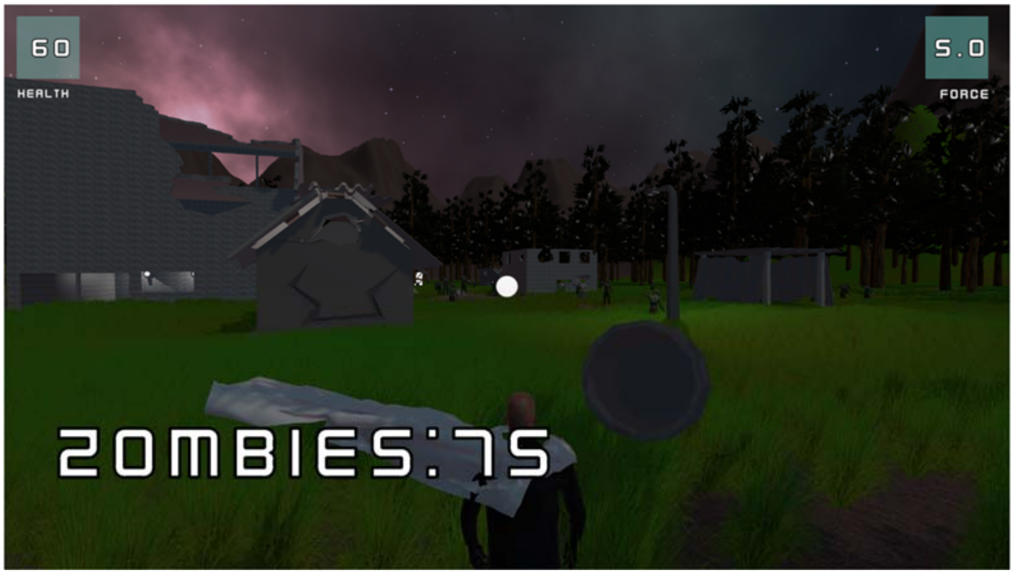
Implementation: The game is developed using Unity 3D with C# programming. The following functionalities were implemented:

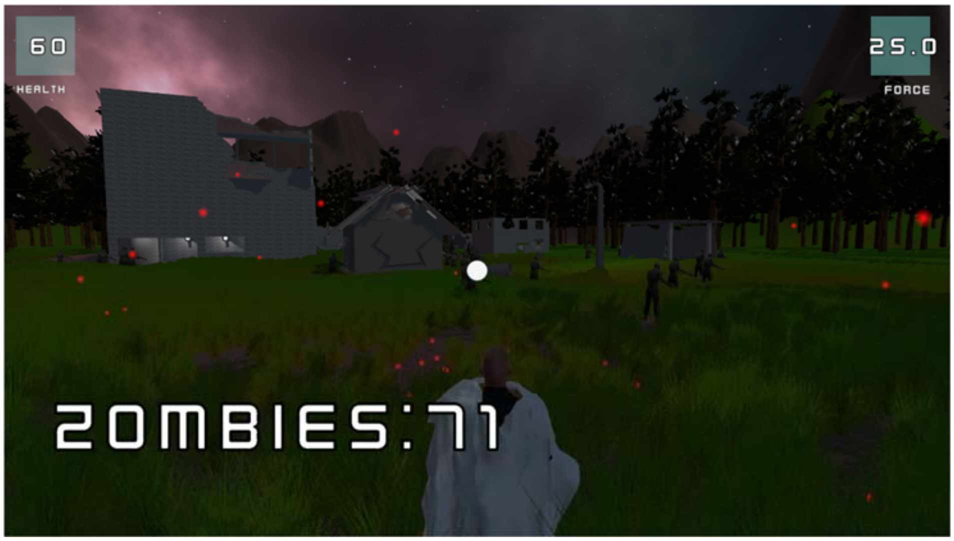
- Vision: We used high-quality textures and 3D models to create a detailed and immersive environment for our action horror game. We also created realistic lighting effects.



- Sound: We used a combination of ambient sounds, speech, and music to create a terrifying atmosphere for the players. The sounds were carefully chosen to enhance the horror experience and to provide important information about the environment.
- Animation: Three animated objects were implemented in the game, including animated enemies (zombies), animated weapons (guns), and a player controller. The animations were created using keyframe animation techniques in Unity and were triggered based on specific events in the game.
- Interactivity: Five user-triggered events were implemented in the game, including the ability to shoot enemies, open doors, pick up weapons, interact with objects, and reload weapons. These events were triggered by using input devices such as the mouse and keyboard.

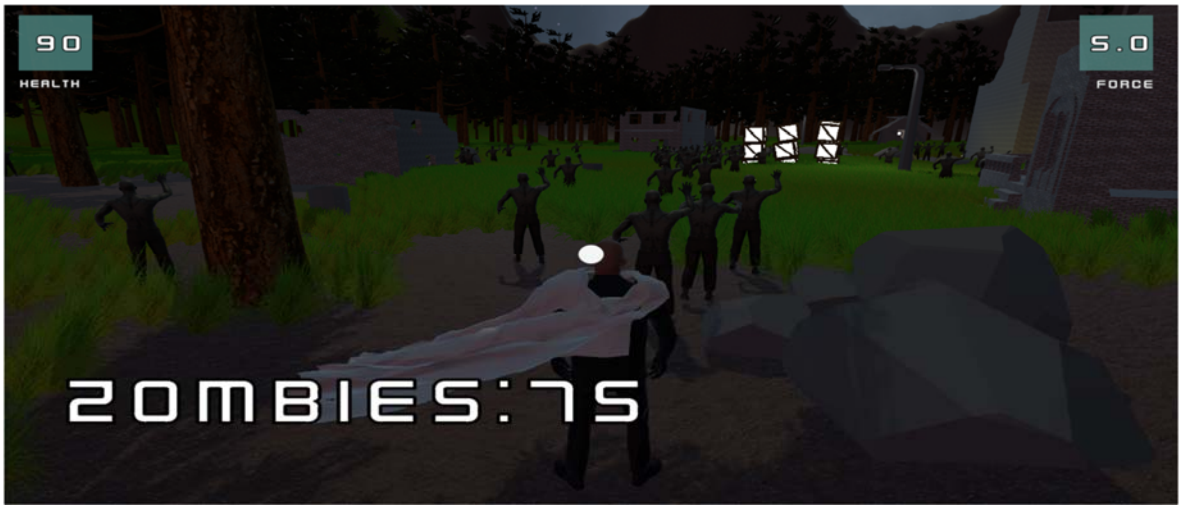

20
HEALTH
34.1
FORCE
ZOMBIES:75


60
HEALTH
5.0
FORCE
ZOMBIES:75

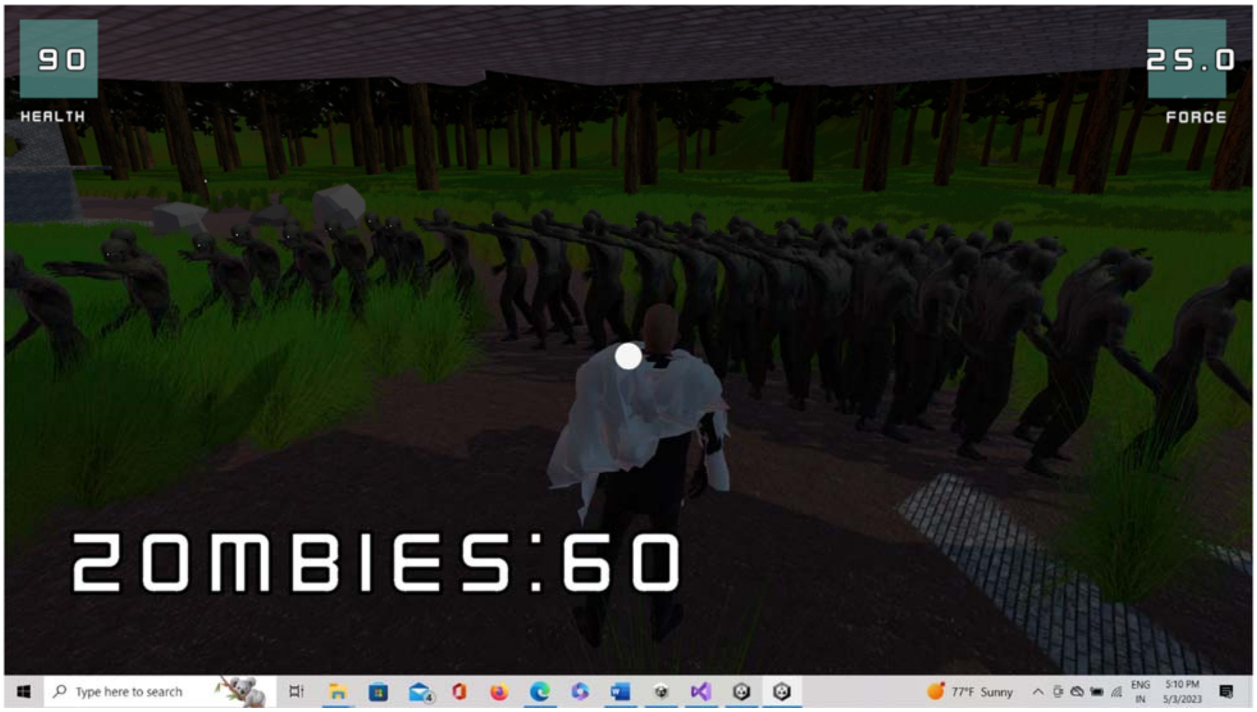


In the above pictures, we have 3 different throwables used as weapons(boxes, barrels, and stones) to attack and escape from the enemies. On the left mouse click, weapons are picked up. On pressing the key F, the player drops the weapon. On long press of left mouse button, the player generates more force. The more the force the more number of enemies are killed.

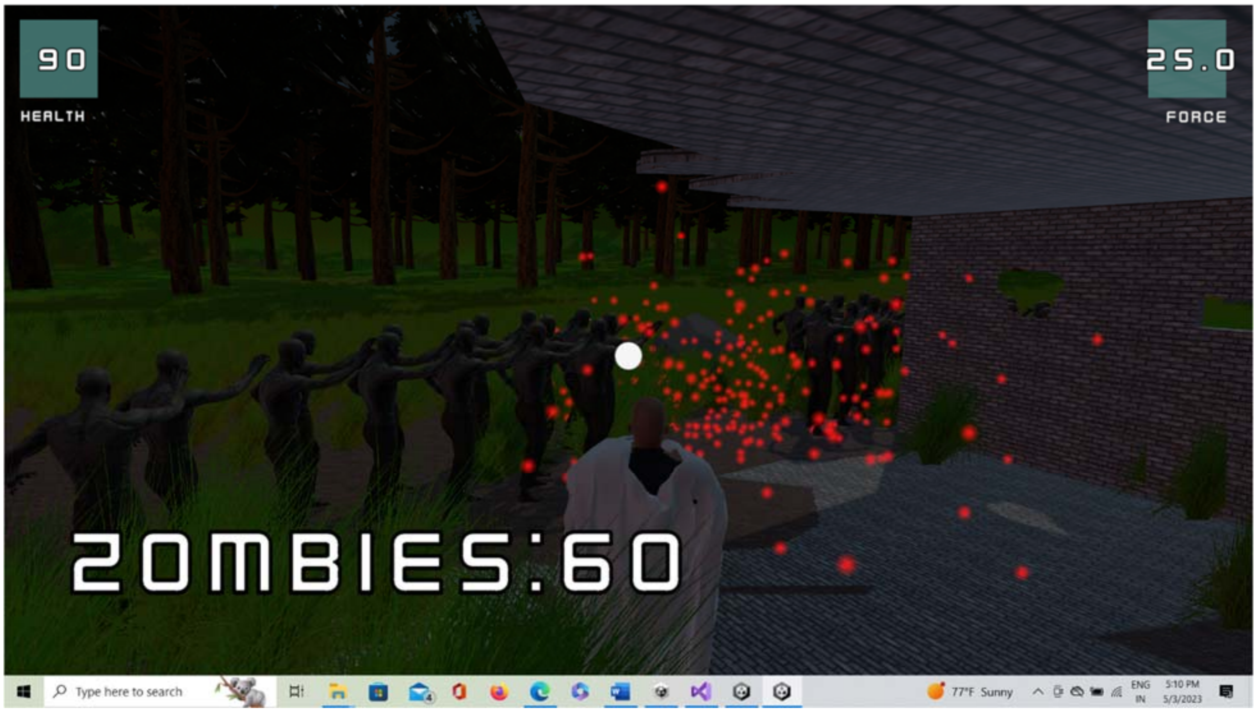
- Characters/Avatars: Animated agents with path-following behavior were implemented in the game as enemies (zombies). They were programmed to follow the player using the A* pathfinding algorithm and would attack the player when they were close enough. The player could also interact with the agents using the keyboard to shoot or avoid them.



- Sensors: Three different types of sensors were implemented in the game, including proximity sensors for enemies and objects, touch sensors for doors and buttons, and time sensors for time-based events. These sensors were used to trigger specific events in the game and were implemented using Unity's built-in physics and scripting tools.



In the above picture, the player is inside a building. We can also see that the zombies cannot get closer to the player while he is inside the building and go into an idle state or behavior. This means that zombies cannot enter certain proximities of the entire area.

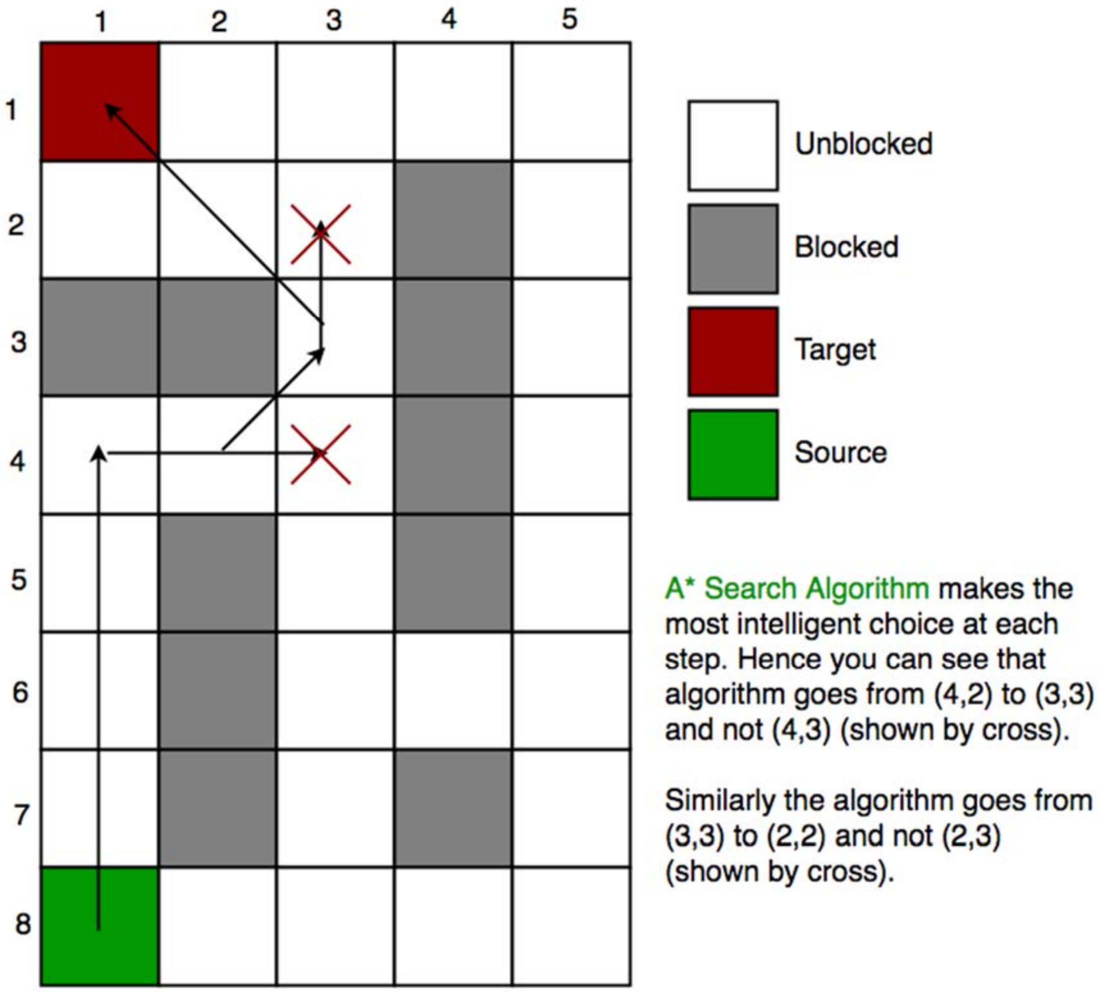


In the above picture, as soon as the zombies touch the player, the player's health decreases, and if the player throws an object at the enemies they die and their blood splashes the moment the object strikes them.

- Player: A Third-Person Controller was added to the scene to control the player's movement and view. The controller allowed the player to move around in the game world using the keyboard and mouse, and to interact with objects and enemies using input devices.
- AI Implementation: AI functionality was implemented in the game using the A* pathfinding algorithm and a state machine system. The enemies were programmed to follow the player and to attack when they were close enough. The state machine system allowed the enemies to switch between different behaviors, such as wandering or chasing the player, depending on specific game events. The AI system was also customizable through a user menu that allowed the player to adjust the number of enemies and their behavior.

- A* Search algorithm is one of the best algorithms in pathfinding and graph traversals.

- A* Search algorithms, unlike other traversal techniques, it has a "brain". It means that the algorithm finds the shortest path and teaches the zombies to follow the shortest path.



A* Search Algorithm makes the most intelligent choice at each step. Hence you can see that algorithm goes from (4,2) to (3,3) and not (4,3) (shown by cross).

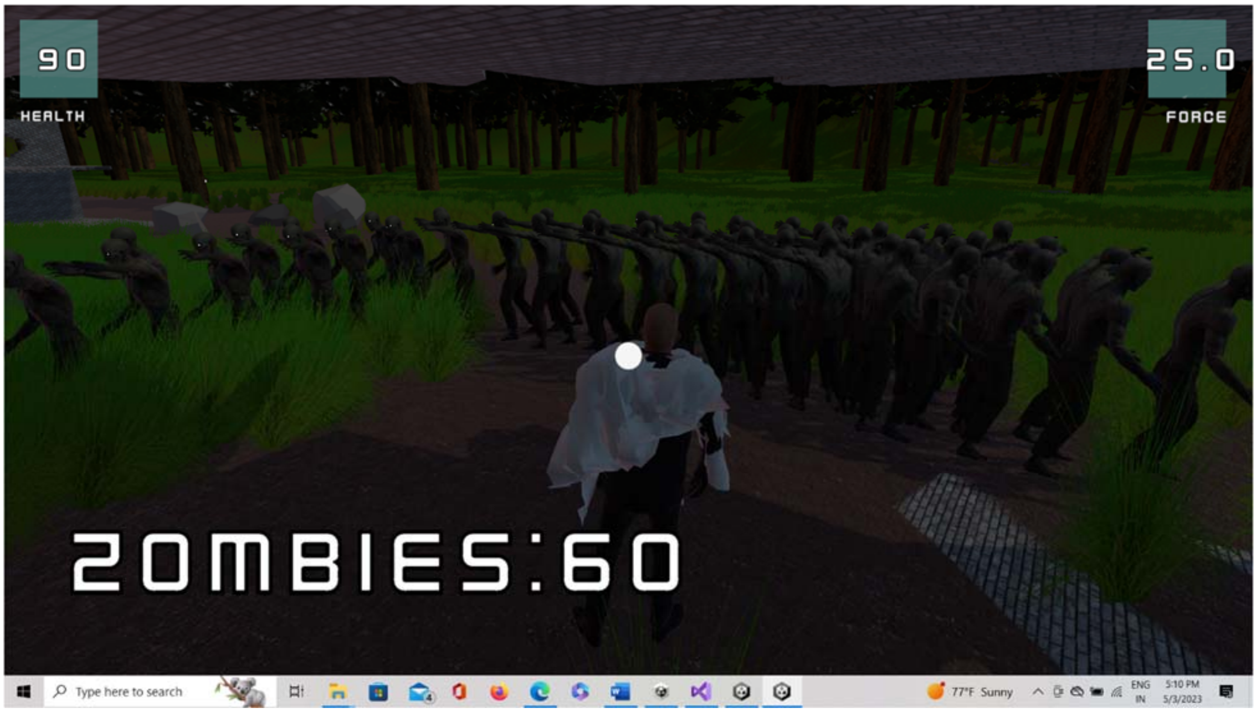
Similarly the algorithm goes from (3,3) to (2,2) and not (2,3) (shown by cross).

The above picture illustrates the A* search algorithm. The algorithm finds the shortest possible path by summing up two parameters g and h.

g: the movement cost to move from starting point to a given square in the grid.

h: the estimated cost to move from the starting point to the destination point.

h is called heuristics. We have to either calculate the exact value of h or the approximate value of h. Manhattan distance, diagonal distance, and Euclidean distance are used to calculate the approximate value of h. Finding the exact value of h is not easy and it should be calculated before starting the algorithm.



- Interface elements: The game interface included menu items such as buttons for starting and pausing the game, a health bar, an ammo count, and a weapon selection menu. These elements were implemented using Unity's UI tools and were designed to be intuitive and easy to use.

Scripts:

EnemySpawner - This script is responsible for spawning enemies at specific locations in the game world. It may include settings such as the number of enemies to spawn. We used 75 enemies for the game. We can change the number of enemies up to 100 as per the game requirement.

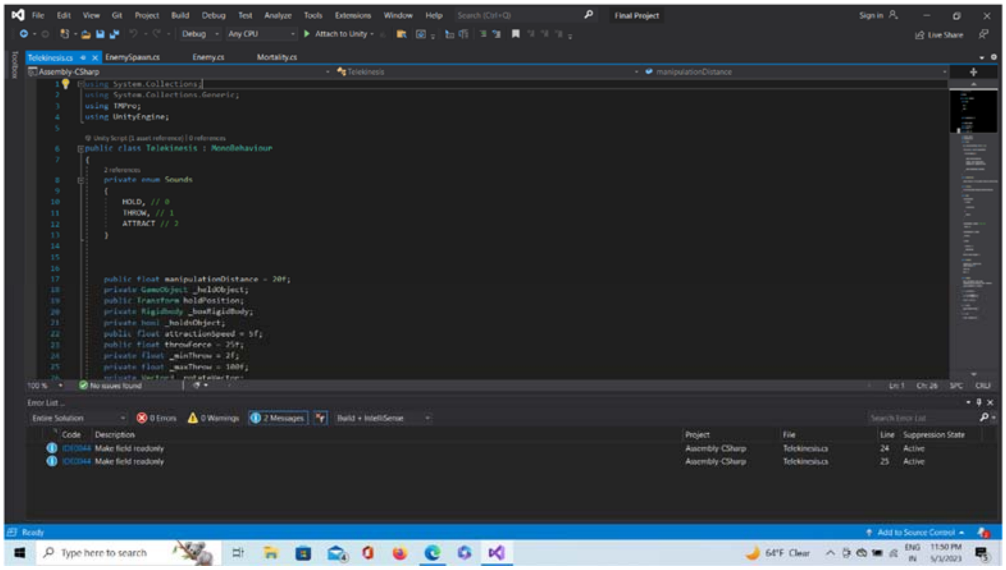
Telekinesis - This script may have been used to allow the player character to interact with objects in the game world using telekinetic powers. The script may include settings such as the range of the telekinetic power, the types of objects that can be moved, and the strength of the telekinetic force.

Enemy - This script defines the behavior of the enemy characters in the game. It may include settings such as the enemy's movement patterns, attack patterns, health and number of enemies alive.

Player Controller - This script defines the behavior of the player character in the game. It may include settings such as the player's movement speed, jumping ability, and health.

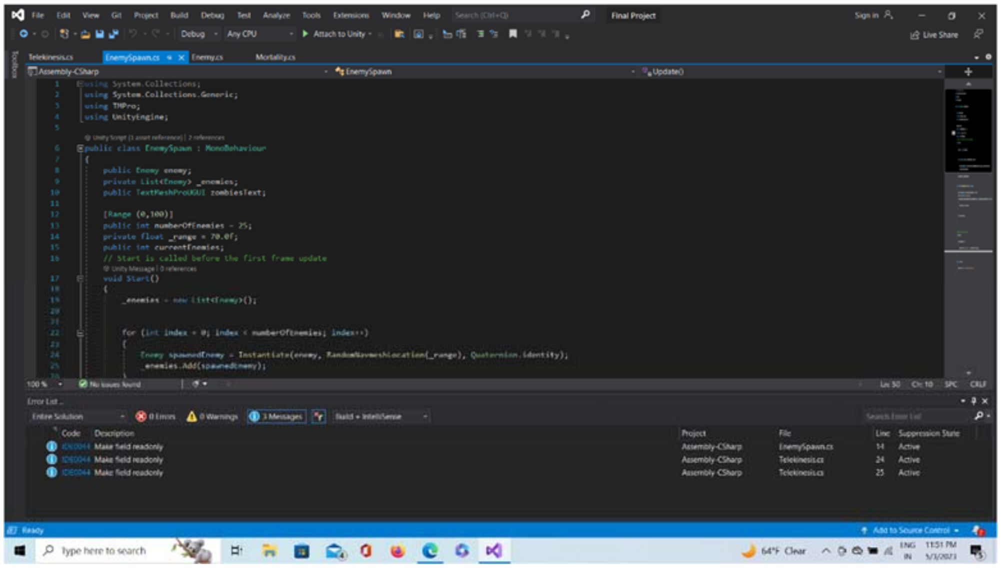
Camera Controller - This script defines the behavior of the camera in the game. It may include settings such as the camera's position, rotation, and movement speed.

Mortality - This script may have been used to handle the death and respawning of the player character and enemies in the game. It may include settings such as the time delay before respawning and the position of the respawn point.



```
using System.Collections;
using System.Collections.Generic;
using TMPro;
using UnityEngine;

public class Telekinesis : MonoBehaviour
{
    private enum Sounds
    {
        HOLD, // 0
        THROW, // 1
        ATTRACT // 2
    }



    public float manipulationDistance = 20f;
    private GameObject _heldObject;
    public Transform holdPosition;
    private Rigidbody _boxRigidBody;
    private bool _holdsObject;
    public float attractionSpeed = 5f;
    public float throwForce = 25f;
    private float _minThrow = 2f;
    private float _maxThrow = 100f;
    private Vector3 _rotateVector;
```

Code	Description	Project	File	Line	Suppression State
IDE0044	Make field readonly	Assembly-CSharp	Telekinesis.cs	24	Active
IDE0044	Make field readonly	Assembly-CSharp	Telekinesis.cs	25	Active



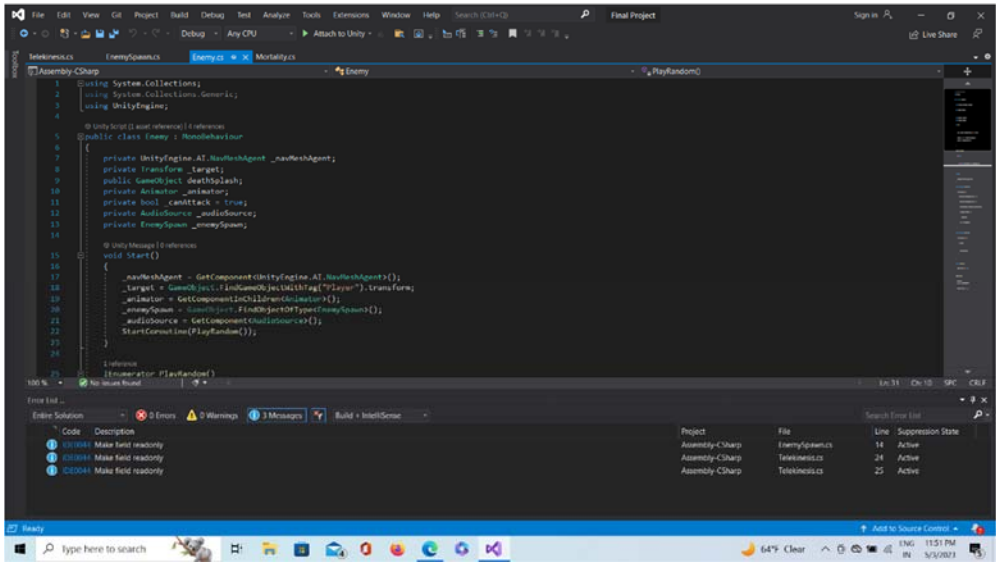
```
using System.Collections;
using System.Collections.Generic;
using TMPro;
using UnityEngine;

public class EnemySpawn : MonoBehaviour
{
    public Enemy enemy;
    private List<Enemy> _enemies;
    public TextMeshProUGUI zombiesText;

    [Range (0,100)]
    public int numberOfEnemies = 25;
    private float _range = 70.0f;
    public int currentEnemies;
    // Start is called before the first frame update
    void Start()
    {
        _enemies = new List<Enemy>();


        for (int index = 0; index < numberOfEnemies; index++)
        {
            Enemy spawnedEnemy = Instantiate(enemy, RandomNavmeshLocation(_range), Quaternion.identity);
            _enemies.Add(spawnedEnemy);
```

Code	Description	Project	File	Line	Suppression State
IDE0044	Make field readonly	Assembly-CSharp	EnemySpawn.cs	14	Active
IDE0044	Make field readonly	Assembly-CSharp	Telekinesis.cs	24	Active
IDE0044	Make field readonly	Assembly-CSharp	Telekinesis.cs	25	Active

Course Title: SPECIAL PROBLEMS

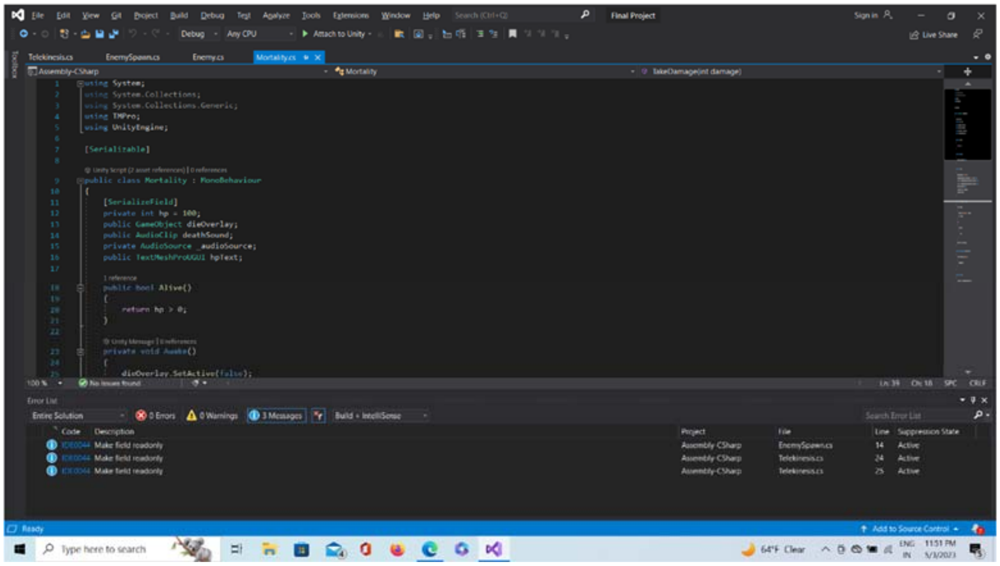
Course Number: INFO 5900

Faculty: Prof. Sharad Sharma



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Enemy : MonoBehaviour
{
    private UnityEngine.AI.NavMeshAgent _navMeshAgent;
    private Transform _target;
    public GameObject deathSplash;
    private Animator _animator;
    private bool _canAttack = true;
    private AudioSource _audioSource;
    private EnemySpawn _enemySpawn;

    void Start()
    {
        _navMeshAgent = GetComponent<UnityEngine.AI.NavMeshAgent>();
        _target = GameObject.FindGameObjectWithTag("Player").transform;
        _animator = GetComponentInChildren<Animator>();
        _enemySpawn = GameObject.FindObjectOfType<EnemySpawn>();
        _audioSource = GetComponent<AudioSource>();
        StartCoroutine(PlayRandom());
    }

    IEnumerator PlayRandom()
```

Code	Description	Project	File	Line	Suppression State
IDE0044	Make field readonly	Assembly-CSharp	EnemySpawn.cs	14	Active
IDE0044	Make field readonly	Assembly-CSharp	Telekinesis.cs	24	Active
IDE0044	Make field readonly	Assembly-CSharp	Telekinesis.cs	25	Active



```
using System;
using System.Collections;
using System.Collections.Generic;
using TMPro;
using UnityEngine;

[Serializable]

public class Mortality : MonoBehaviour
{
    [SerializeField]
    private int hp = 100;
    public GameObject dieOverlay;
    public AudioClip deathSound;
    private AudioSource _audioSource;
    public TextMeshProUGUI hpText;

    public bool Alive()
    {
        return hp > 0;
    }

    private void Awake()
    {
        dieOverlay.SetActive(false);
```

Code	Description	Project	File	Line	Suppression State
IDE0044	Make field readonly	Assembly-CSharp	EnemySpawn.cs	14	Active
IDE0044	Make field readonly	Assembly-CSharp	Telekinesis.cs	24	Active
IDE0044	Make field readonly	Assembly-CSharp	Telekinesis.cs	25	Active

Behaviors Implemented:

- We implemented different behaviors for the AI agents in the game. The zombies have path-following behavior, which allows them to follow the player around the haunted house.
- The zombie's behavior is selfish, as it will keep on wandering behind the player and attack the player if it comes too close.
- If the player enters the haunted houses present in the forest, the zombies follow a different behavior. They change their behavior from wandering to idle, waiting outside in the vicinity of the building for the player to come out.

Importance of application and role of Virtual Reality:

This application is useful because it provides a new level of immersion and realism, which is not possible with traditional gaming platforms. By using virtual reality, we can create a sense of presence and put the player in a terrifying environment, which can be very exciting for horror game enthusiasts. Virtual reality technology also allows us to provide a high level of interactivity.

Problems Encountered and Remaining Shortcomings:

We encountered several coding issues during the development of this project and encountered issues with optimization. Future work could focus on improving the code and optimization of the game and adding more complex AI behaviors to create an even more challenging experience for the player.

Conclusion:

- In conclusion, our team designed the Action 3D horror game successfully and implemented various functionalities, such as textures and 3D models, sound effects, animations, interactivity, avatars, sensors, player controller, AI implementation, and interface elements. These functionalities were implemented to create an immersive horror gaming experience for the players.
- Through the implementation of various features and functionalities, this game provides a unique and engaging experience for players. The avatars with different behaviors and path-following abilities add to the thrill of the game. The use of sensors such as proximity, touch, and time sensors creates an interactive environment that adds to the immersive experience.
- Despite the successful implementation of various features, some problems were encountered during this game's development. These issues include the optimization of the game's performance and the improvement of the AI implementation. Future work could address these issues and add more features to enhance the gaming experience.
- Overall, the Action 3D horror game designed by our team showcases the capabilities of virtual reality technology in creating engaging and immersive gaming environments. It provides a unique and thrilling experience for players and serves as an example of the potential of VR technology in the gaming industry.

References:

- Sketchup for modeling
- Unity 3d for game designing

- MS Studio 2019 for coding