UNIVERSITI
TEKNOLOGI
PETRONAS

FACULTY OF SCIENCE AND INFORMATION TECHNOLOGY

TEB1043/TFB1033: OBJECT-ORIENTED PROGRAMMING

PROJECT REPORT

Prepared By:

NAME	STUDENT ID	COURSE
Putriani binti Azlar	22010298	Information Technology
Nur Huda Adila binti Fathul Hanif	22010210	
Nurkamilia Hadirah binti Mohamad Suhaimi	22010014	
Muhammad Iman Hafizi bin Muhammad Zailan	2210144	
Puteri Natasha binti Hairul Nizam	22010207	Computer Engineering
Nor Diana binti Aziz	22010477	

Submitted To:
Dr. M Nordin B Zakaria,
Senior Lecturer,
Computer & Information Science.

On 21st November 2024

Table of Contents

1.0 INTRODUCTION

The gaming industry has seen a significant rise in the popularity of survival horror games, driven by their ability to provide players with intense, immersive experiences. ***Ultimate Zombie School Survival*** is a thrilling horror game designed to captivate players through its unique setting and challenging gameplay.

Set in a university overtaken by a zombie apocalypse, this game pits players against relentless swarms of zombies in a desperate fight for survival. The gameplay is designed to immerse players in a tense and gripping atmosphere where strategic thinking, quick reflexes, and resourcefulness are essential for survival.

The core gameplay revolves around swarm attacks, where waves of zombies unite to overpower the lone player. To support the player's survival, a health system is implemented as a key feature. Players can collect health items scattered throughout the environment to restore and increase their health, adding an extra layer of strategy to the game.

To enhance replayability and cater to players of varying skill levels, ***Ultimate Zombie School Survival*** features three difficulty modes: easy, medium, and hard. Each level presents escalating challenges, ensuring that players remain engaged and motivated to push their limits.

By integrating atmospheric graphics, dynamic zombie AI, level progression, and health regeneration mechanics, this project aims to deliver an exhilarating gaming experience. This report outlines the development process, from concept to implementation, and evaluates the chosen platform's suitability in creating an engaging and seamless game.

2.0 PROJECT DESCRIPTION

Ultimate Zombie School Survival is a survival horror game that thrusts players into the chilling scenario of a university overtaken by a zombie apocalypse. Designed as a single-player experience, the game challenges players to survive relentless swarm attacks from waves of zombies. The game's core mechanics and immersive features are crafted to keep players engaged and on edge throughout their journey.

1. <u>Swarm Attach Gameplay</u>
   Players face waves of zombies that work together in coordinated swarm attacks. The intensity increases as players progress through difference levels. The game introduces three distinct types of zombies, each with unique characteristics and difficulty levels:

   - ***Normal Zombie*** is the most common type of zombie, requiring 1 dart shot to eliminate
   - ***Speed Zombie*** is faster and more agile than zombie, requiring 2 dart shots to defeat
   - ***Big Zombie*** is a tougher and more resilient enemy, requiring 3 dart shots to take down.

2. Difficulty Levels
   The game includes three distinct difficulty modes:
   - Easy is designed for casual players to enjoy a less stressful experience. Features only Normal Zombies
   - Medium offers a balanced challenge for experienced players. Introduces both Normal Zombies and Speed Zombies.
   - Hard tests even the most skilled players with relentless zombie attacks. Includes all three zombie which are Normal Zombies, Speed Zombies, and Big Zombies.

3. Health System
   Players can collect health items to increase their health. This feature introduces an element of resource management and survival strategy. Each health item increases the player's health by *5 points*, encouraging exploration and resource management.

   Players start with a finite health pool that decreases when attacked by zombies based on their type:

   - ***Normal Zombie***: Deals 10 damage per attack
   - ***Speed Zombie***: Deals 20 damage per attack
   - ***Big Zombie***: Deals 30 damage per attack

4. Immersive Environment
   The university setting is designed to enhance the horror experience, featuring dimly lit corridors, eerie sound effects and unpredictable zombie encounters.

3.0 PLATFORM SELECTION AND EVALUATION

Our group choose Unity for our game project because of its versatility and cross-platform capabilities. With Unity, developers can create games that run seamlessly on multiple platforms, including PCs, consoles, mobile devices, and VR/AR systems, without requiring significant adjustments to the code but in this case, we make the game for PC platform. This saves time and effort while ensuring broad accessibility for our games. Additionally, Unity's user-friendly interface and visual editor make it easy for a newbie developers like us to design scenes, import assets, and manipulate objects with minimal technical barriers. These features allow us to focus on refining the gameplay and storytelling instead of struggling with complex tools.

Another reason Unity is a popular choice is its rich Asset Store and powerful engine. The Asset Store provides access to a vast library of pre-made assets, plugins, and tools, which helps speed up the development process and reduces the need to build everything from scratch. Unity's robust engine supports both 2D and 3D development, offering advanced rendering, physics, and animation tools to create visually stunning and dynamic games. Furthermore, Unity uses C# for scripting, a widely used and well-supported programming language, making it easier for us to write and debug code, as example it really helps us to code for the zombies and the shooter like the health bar and zombie damage.

` Other than that, our group also used Gamemaker applications to make the background of our game. It may be out of the syllabus, but we learn it on the internet, and we just use this application to make the background of our game. Gamemaker it gives simplicity, efficiency, and focus on 2D games. We search for tileset on the internet as the base for our background and implement it on the GameMaker. The interface of the application is also user-friendly and easy for the new user to try using the application. It also has a lot of active community and wealth of tutorials provide valuable resources for both new and experienced users which really help us while using this application.

4.0 SYSTEM DESIGN

Here's a brief explanation of the classes and their functionalities:

1. HealthCollectable

Inherits: MonoBehaviour

Fields: _healthAmount

Methods: OnCollected: Handles behavior when the health collectible is picked up.

2. MonoBehaviour (Unity Base Class)

Provides base functionality for Unity scripts.

Includes properties like runInEditMode, useGUILayout.

Common methods: StartCoroutine, Invoke, CancelInvoke, and more.

3. Behaviour (Base Class)

Properties: enabled, isActiveAndEnabled.

Represents a component that can be enabled or disabled in Unity.

4. Component

Properties: Access to Unity components (e.g., transform, collider, rigidbody).

Methods: Operations to manage or retrieve components (e.g., GetComponent, BroadcastMessage).

5. Object

Represents base Unity object functionality.

Methods include Destroy, Instantiate, and equality operations (operator ==).

<u>Game-Specific Classes:</u>

6. CollectableSpawner

Fields: _collectablePrefabs

Methods: SpawnCollectible: Handles spawning collectible items.

7. Collectable

Fields: _collectableBehaviour

Methods:

Awake: Initialization logic.

OnTriggerEnter2D: Handles collision or trigger interactions.

8. EnemyAttack

Fields: _damageAmount

Methods: OnCollisionStay2D: Handles collision-based attacks.

9. EnemyCollectableDrop

Fields: _chanceOfCollectableDrop, _collectableSpawner.

Methods: Logic for dropping collectibles when an enemy is defeated.

10. EnemyDestroyController

Methods: DestroyEnemy: Handles enemy destruction.

11. EnemyMovement

Fields: Handles various parameters like speed, direction, and obstacle detection.

Methods: Includes AI-like behavior for movement, obstacle avoidance, and targeting.

12. EnemyScoreAllocator

Fields: _killScore, _scoreController.

Methods: AllocateScore, Awake.

13. EnemySpawner

Fields: Spawn configuration like spawn times and prefab references.

Methods: Awake, Update, and setting spawn timings.

14. PlayerAwarenessController

Fields: Awareness range and references to player objects.

Properties: AwareOfPlayer, DirectionToPlayer.

15. HealthBarUI

Fields: UI element for health bar.

Methods: UpdateHealthBar: Adjusts the UI based on health.

16. HealthController

Fields: Tracks current and maximum health.

Properties: Health state and percentage.

Methods: Healing and damage application logic.

17. InvisibilityController

Fields: Related to invincibility effects.

Methods: Controls temporary invincibility states.

18. Bullet

Fields: Handles bullet logic and destruction off-screen.

Methods: Includes collision handling.

19. PlayerDamagedInvincibility

Fields: Visual and duration configurations for invincibility.

Methods: StartInvincibility.

20. PlayerMovement

Fields: Handles input, animation, and movement smoothing.

Methods: Includes logic for player movement and boundary restrictions.

21. PlayerShooting

Fields: Bullet configurations (speed, prefab, fire rate).

Methods: FireBullet, Update.

22. ScoreController

Fields: Event for score changes.

Methods: Logic to add scores and track them.

23. ScoreUI

Fields: Text field for score display.

Methods: Updates UI with the latest score.

24. GameManager

Fields: Manages game state, such as ending the game.

Methods: Handles player death and game over logic.

25. SpriteFlash

Fields: Renderer reference.

Methods: Visual flash effects (e.g., damage feedback).

26. LevelSelection

Fields: References for scene transitions.

Methods: Level difficulty setup.

27. MainMenu

Fields: Scene controller reference.

Methods: Navigation and game start/exit logic.

28. SceneController

Fields: Scene transition properties.

Methods: Handles scene loading and fading.

29. SceneFade

Fields: Image for fade effect.

Methods: Fade-in and fade-out effects.

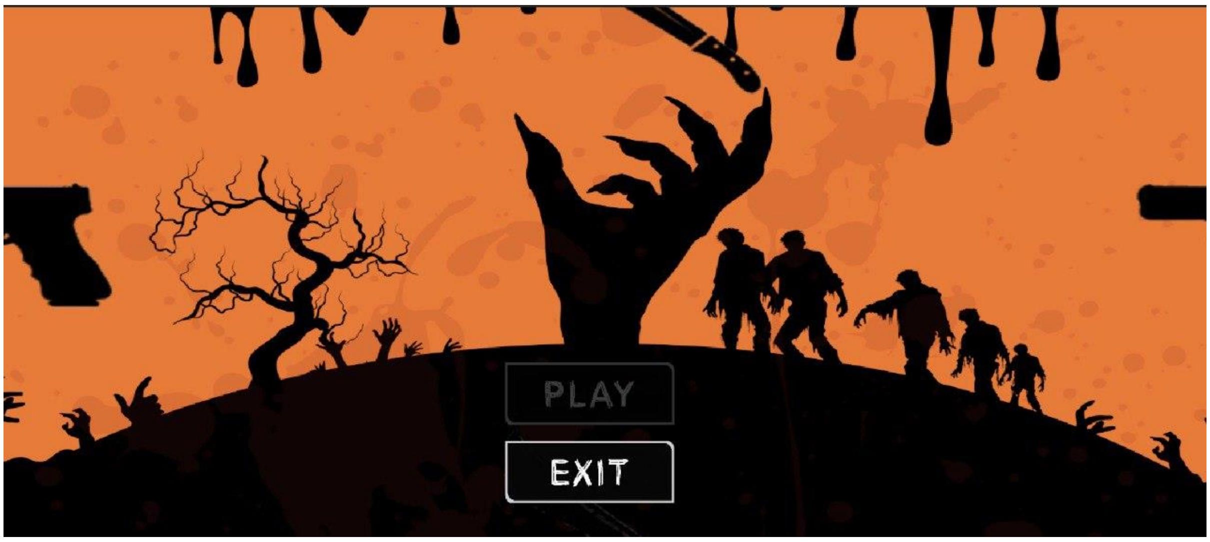
Interfaces:

ICollectableBehaviour

Methods: OnCollected: Ensures collectable items implement specific behaviors when collected.

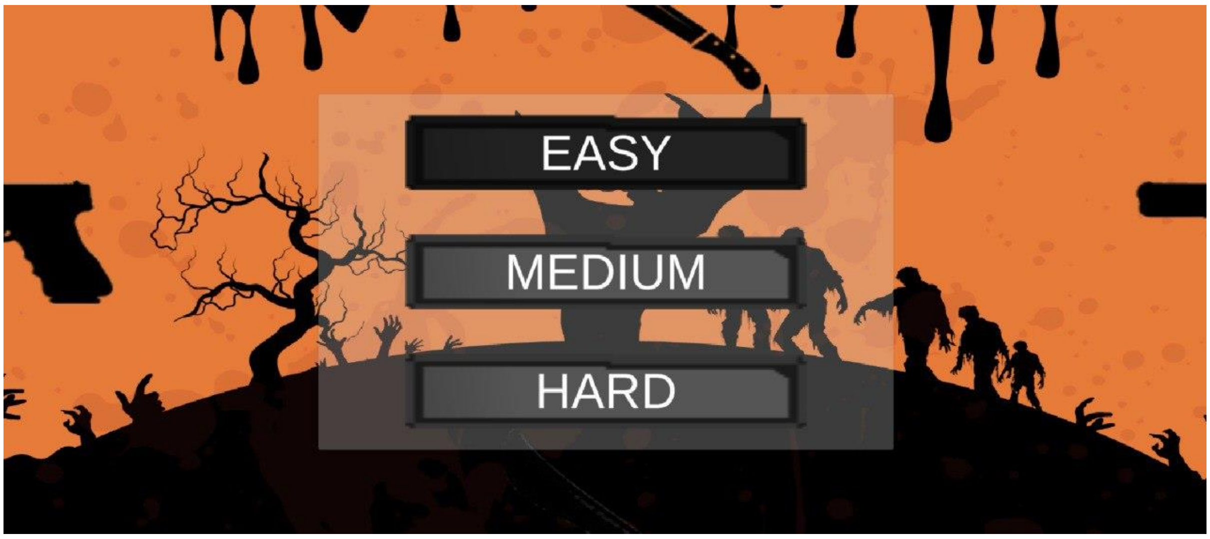
5.0 SCREENSHOTS AND DEMONTRATIONS

<u>Play/Exit Menu Page</u>



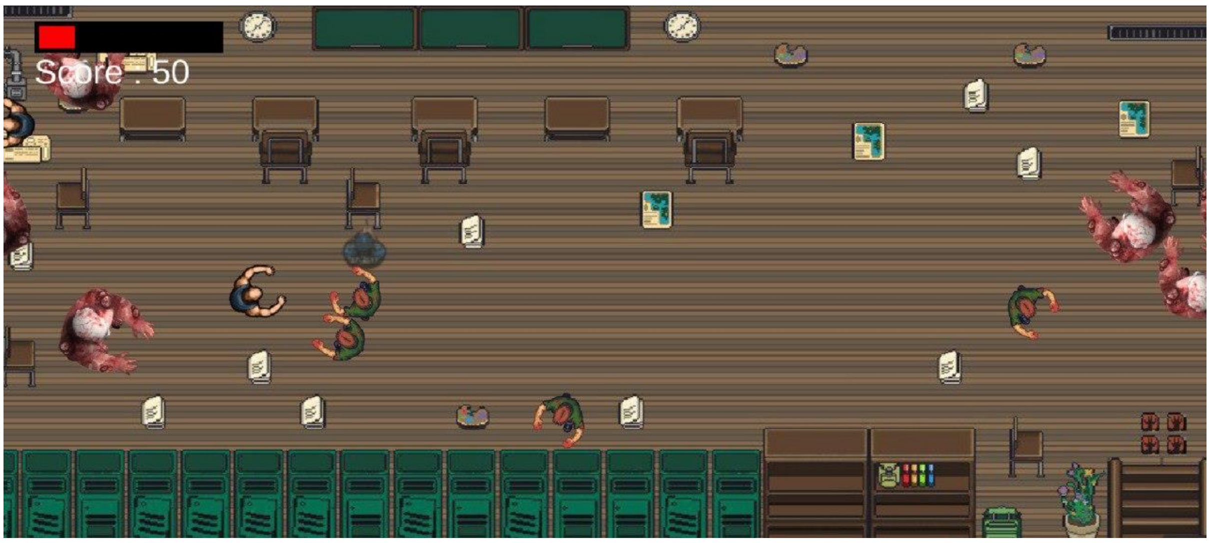
This menu page shows the starting of our game which have two buttons, which are play and exit buttons. We also have a background music playing in this page.

Stage Difficulty Menu Page



This page shows stage difficulties that our game has which are easy, medium, hard. Every stage has its own difficulty, and a different type of zombie will spawn for the harder stage.

During the Game



To control the shooter, the user can use arrows key to move the shooter or can use WSAD key to move, W key is used to move upward, S key to move downward, A key is used to move left and D key to move to the right. The user needs to use spacebar to shoot at the zombies. The health bar of the zombie has been explained in the project description.

6.0 CHALLENGERS AND LIMITATIONS

One significant challenge is repetitiveness, as players may find the gameplay less engaging over time due to the limited variety of enemies and environmental features. Expanding enemy types or dynamic hazards was constrained by the project timeline.

Another issue is the player strategy limitation, as the linear map design restricts the freedom to explore alternative paths or create defensive strategies. This reduces opportunities for players to experiment with different approaches to survival.

Lastly, environment interaction is minimal, as the game lacks advanced features like traps, destructible objects, or interactive escape routes. Technical constraints limited the incorporation of these elements, which could have enhanced immersion and gameplay depth.

7.0 CONCLUSION

In conclusion, *Ultimate Zombie School Survival* successfully delivers an engaging and challenging survival horror experience set in a university during a zombie apocalypse. The game features a dynamic difficulty system, diverse zombie types, and an immersive environment that adapts to the player's skill level, ensuring a thrilling experience for a wide range of players. While there were challenges in balancing difficulty, optimizing AI, and designing interactive environments, these obstacles were overcome within the project's scope, creating a solid gameplay foundation.

The limitations, such as repetitiveness, restricted player strategy, and minimal environmental interaction, provide areas for future improvements and expansions. Despite these constraints, the game offers an enjoyable survival horror experience, with room for future updates to further enhance gameplay depth and player engagement. This project showcases the successful application of game development principles, and with additional resources and time, the concept could be expanded into a more complex and immersive experience.

8.0 APPENDICES

GitHub Link.