DEFEND OR DIE

MINI PROJECT REPORT

Submitted By

Karthigeyan R(22C061)

Mugil Varnan M(22C076)

Ravi Prasath S(22C098)

Suriya Navaneetha Krishnan K(22C125)

in partial fulfillment for the award of the

degree of

Bachelor of

Engineering in

Computer Science and Engineering

Thiagarajar College of Engineering, Madurai - 625015
(A Govt. Aided ISO 9001:2008 certified Autonomous institution Affiliated to Anna University)

APRIL 2025

Thiagarajar College of Engineering, Madurai – 625015

(A Govt. Aided ISO 9001:2008 certified Autonomous institution Affiliated to Anna University)

BONAFIDE CERTIFICATE

Certified that this mini project report, “Solar System Visualizer” is the bonafide work of

Karthigeyan R(22C061)
Mugil Varnan M(22C076)
Ravi Prasath S(22C098)
Suriya Navaneetha Krishnan K(22C125)

who carried out the project work under my supervision.

Signature

Course Instructor

Dr Raja Lavanya,

Assistant Professor,

Dept of Computer Science and Engineering,

Thiagarajar college of Engineering, Madurai

ABSTRACT

"Defend or Die" is a 2D tower defense strategy game developed using Unity, designed to challenge players' strategic thinking and quick reflexes. The primary objective of the game is to prevent waves of enemies from reaching a designated target by strategically placing various types of defenses. This project serves as a practical demonstration of key Unity development techniques, including procedural enemy spawning, user interface design, gameplay loop creation, and modular game development.

The game leverages several core Unity components, such as tilemaps, canvases, event systems, and prefabs, to create a seamless and engaging player experience. Key features include player controls, enemy management, shooting mechanics, and dynamic UI systems, all of which are controlled through well-structured scripts. The use of Unity’s extensibility through packages like TextMesh Pro for enhanced text rendering and the Analytics Library for tracking player behavior further contributes to the game’s quality and depth.

The development process emphasizes efficient asset organization, scene management, and script optimization, ensuring that the game runs smoothly across different platforms. "Defend or Die" not only demonstrates best practices in modern game design but also provides an immersive and interactive experience that highlights the essentials of Unity development, from initial concept to final implementation. This report outlines the methodologies, tools, and techniques employed in the creation of the game, offering a comprehensive overview of the development cycle.

CONTENT

1. Introduction

The gaming industry has witnessed a tremendous transformation with the advent of powerful development platforms like Unity. "Defend or Die" is a 2D defense strategy game created in Unity, where players must tactically place defenses and prevent enemies from reaching a specified target. This project aims to demonstrate skills in modular game development, procedural spawning of enemies, user interface design, and gameplay loop creation. Using components such as scripts, tilemaps, canvases, and event systems, "Defend or Die" offers an engaging and challenging experience that tests players' strategic thinking and reflexes. This project showcases the essential practices of Unity development, including scene management, script optimization, prefab usage, and efficient asset organization, ensuring an immersive player experience.

Unity’s extensibility through packages like TextMesh Pro and Analytics Library has been fully utilized. The scripts developed include player controls, enemy management, shooting mechanics, and UI management. With well-structured scenes and carefully arranged assets, the project provides a holistic example of modern game design. This report details the entire development cycle, from problem definition to experimental results, with a comprehensive exploration of the techniques and methodologies applied to bring "Defend or Die" to life.

2. Literature Survey

1. "The Role of Game Mechanics in Enhancing User Engagement in Mobile Games" — International Journal of Computer Games Technology, 2018.

This paper discusses how well-designed game mechanics, like strategic defense elements and dynamic progression, significantly increase user retention and satisfaction across different genres, including tower defense games.

2. "Procedural Content Generation in Games: A Survey" — IEEE Transactions on Computational Intelligence and AI in Games, 2011.

The study explores procedural content generation for environments and enemy waves, helping developers create infinite game variations, improving replay value and player immersion within defense-based games.

3. "Unity Game Engine: Revolutionizing Indie Game Development" — International Journal of Interactive Multimedia and Artificial Intelligence, 2017.

This paper analyzes how Unity's modular architecture and asset store empower developers to build scalable games efficiently, making complex features like tilemaps and monetization accessible to all.

4. "Optimizing Performance in Mobile Tower Defense Games" — ACM Symposium on Computer-Human Interaction in Play, 2019.

The authors focus on optimization techniques such as object pooling, lightweight UI, and selective rendering, critical to delivering smooth gameplay experiences on resource-constrained mobile devices.

5. "Gamification and Motivation: How Player Rewards Impact Behavior" — Games and Culture Journal, 2015.

This paper presents evidence that reward structures, such as coins and upgrades in tower defense games, enhance player motivation and increase the time users spend engaging with the app.

6. "Ads Versus In-App Purchases: Monetization Strategies for Free-to-Play Games" — Entertainment Computing, Elsevier, 2020.

The research compares the effectiveness of advertisements and in-app purchases in mobile games, suggesting that integrating rewarded ads carefully maximizes user acceptance and boosts developer revenue.

3. Problem Definition

3.1 Problem Statement

In modern mobile and PC gaming, many players seek short, strategic, and engaging experiences. However, there exists a gap in simple, lightweight tower defense games that combine modular design, smooth animations, and strategic depth without overburdening device resources.

3.2 Problem Description

"Defend or Die" addresses this gap by providing an easily accessible, low-resource, but highly engaging defense game built with Unity. It features simple, modular levels, dynamic enemy waves, and interactive UI elements that maintain player interest and ensure replayability.

4. Requirements

4.1 Hardware Requirements

- Processor: Intel Core i5 or equivalent
- RAM: 8 GB minimum
- Graphics: Integrated GPU sufficient; dedicated GPU recommended
- Storage: 2 GB free disk space
- Operating System: Windows 10/11 or macOS 10.14+

4.2 Software Requirements

- Unity Engine 2021.3.0f1 or higher
- Visual Studio Community 2022
- .NET Framework 4.x (supported by Unity)
- Photoshop or GIMP (optional for asset editing)

4.3 Technologies Used

- Unity Game Engine (Primary development platform)
- C# Programming Language (Script development)
- Tilemap Editor (Level design)
- TextMesh Pro (UI text rendering)
- Unity Analytics and In-App Purchasing packages
- Photoshop (Sprites editing)

5. Proposed Method

5.1 Existing Solution

Numerous tower defense games exist, but many either compromise on performance or require significant resources. Additionally, many solutions lack clean modularity, making scalability difficult.

5.2 Proposed Methodology

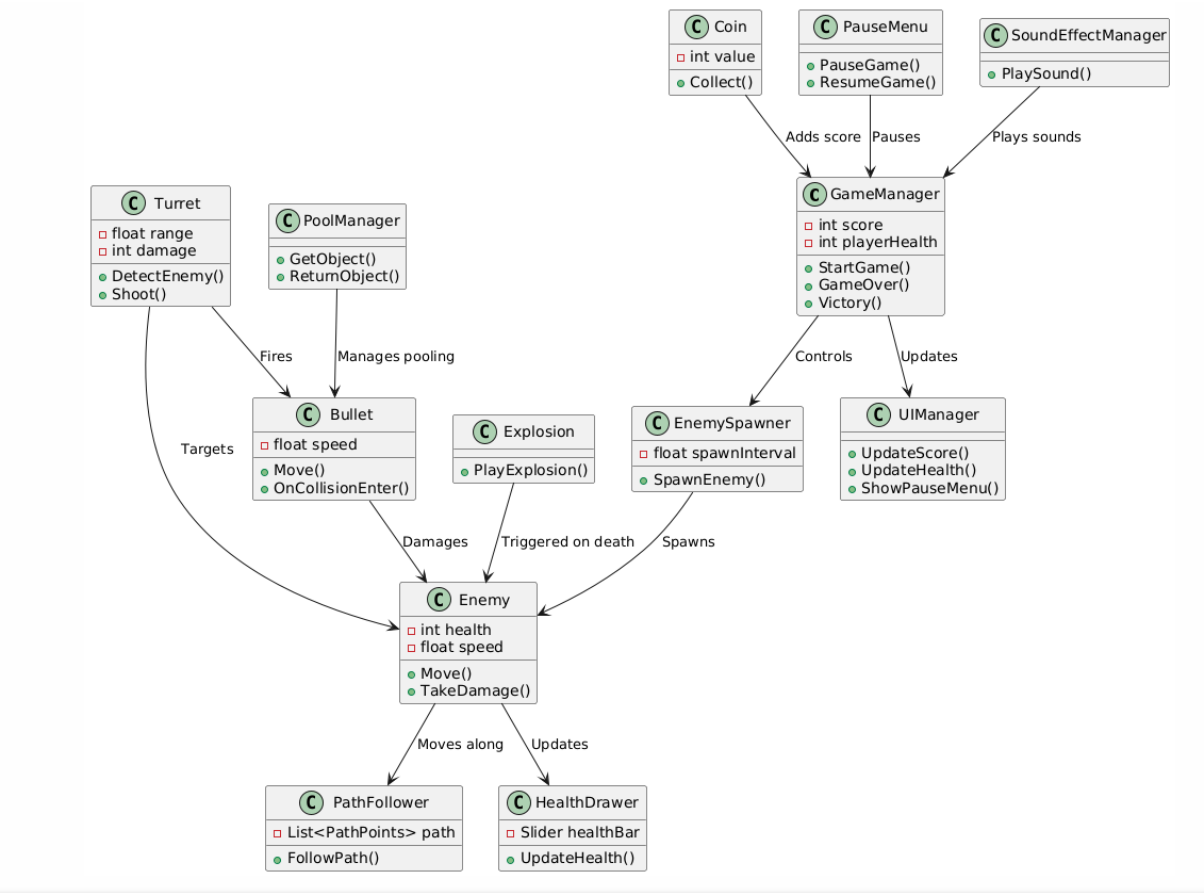
"Defend or Die" uses a modular system where each game mechanic (enemy, pathfinding, player controls) is separated into different scripts. Prefabs ensure efficient asset management. Object pooling reduces resource load, enhancing performance even on low-end devices. A clean event-driven model facilitates game state changes like pausing or victory transitions.

5.3 Module Descriptions

- GameManager: Controls the game lifecycle, including game start, game over, and victory conditions.
- Ground: Contains environment layers like background, airpath, and click detectors for dynamic interaction.
- PauseMenu: Allows players to pause the game and resume without disrupting gameplay states.
- EnemySpawner: Dynamically generates enemy waves based on difficulty levels.
- PathFollower: Manages enemy movement along predefined ground paths to simulate strategic attacks.
- UI Canvas: Displays player stats, health, score, victory, and defeat screens.

5.4 Class Diagram

The Class Diagram features primary classes like GameManager, EnemySpawner, PathFollower, and PlayerController interacting through clearly defined methods and variables.



6. Implementation

Script Details:

- AutoDisableScript: Disables objects (like bullets) after a set time to save resources.
- BuildLocationScript: Manages build points where players can place turrets.
- BulletScript: Controls bullet behaviors, including speed, collision detection, and damage.
- CameraShaker: Shakes the camera when major events happen (enemy explosion, tower destruction).
- CannonScript: Manages cannon towers' shooting behavior against incoming enemies.

- ClickOutsideScript: Detects clicks outside UI menus to close unwanted panels.
- CoinScript: Handles coin drops from defeated enemies.
- EnemyManagerScript: Oversees enemy attributes like speed, health, and damage.
- EnemySpawnerScript: Spawns enemies at designated spawn points with timed waves.
- ExplosionScript: Handles enemy explosion animation upon destruction.
- FlyingShotsScript: Manages visual flying projectiles for aesthetic effects.
- GameManager: Central controller script managing gameplay states, health, score, and level progression.
- HealthDrawerScript: Dynamically updates enemy and player health bars on UI.
- MathHelpers: Provides reusable mathematical functions like enemy targeting or projectile angle calculation.
- MoneyDrawer: Updates player's available money based on actions like turret building or selling.
- PathFollower: Ensures enemies follow the designed path toward the player’s base.
- PauseButtonScript: Pauses and resumes gameplay without breaking the flow.
- PlaneScript: Additional flying enemy logic.
- Pool: Implements object pooling for bullets and effects.
- RocketScript: Special turret shooting heavy projectiles at high damage.
- SoundEffectScript: Plays various sound effects for game interactions.
- StartButtonScript: Manages UI button for starting levels.
- TurretScript: Controls turret firing, enemy targeting, and rotation.
- WalkingRotator: Animates walking enemies for visual richness.

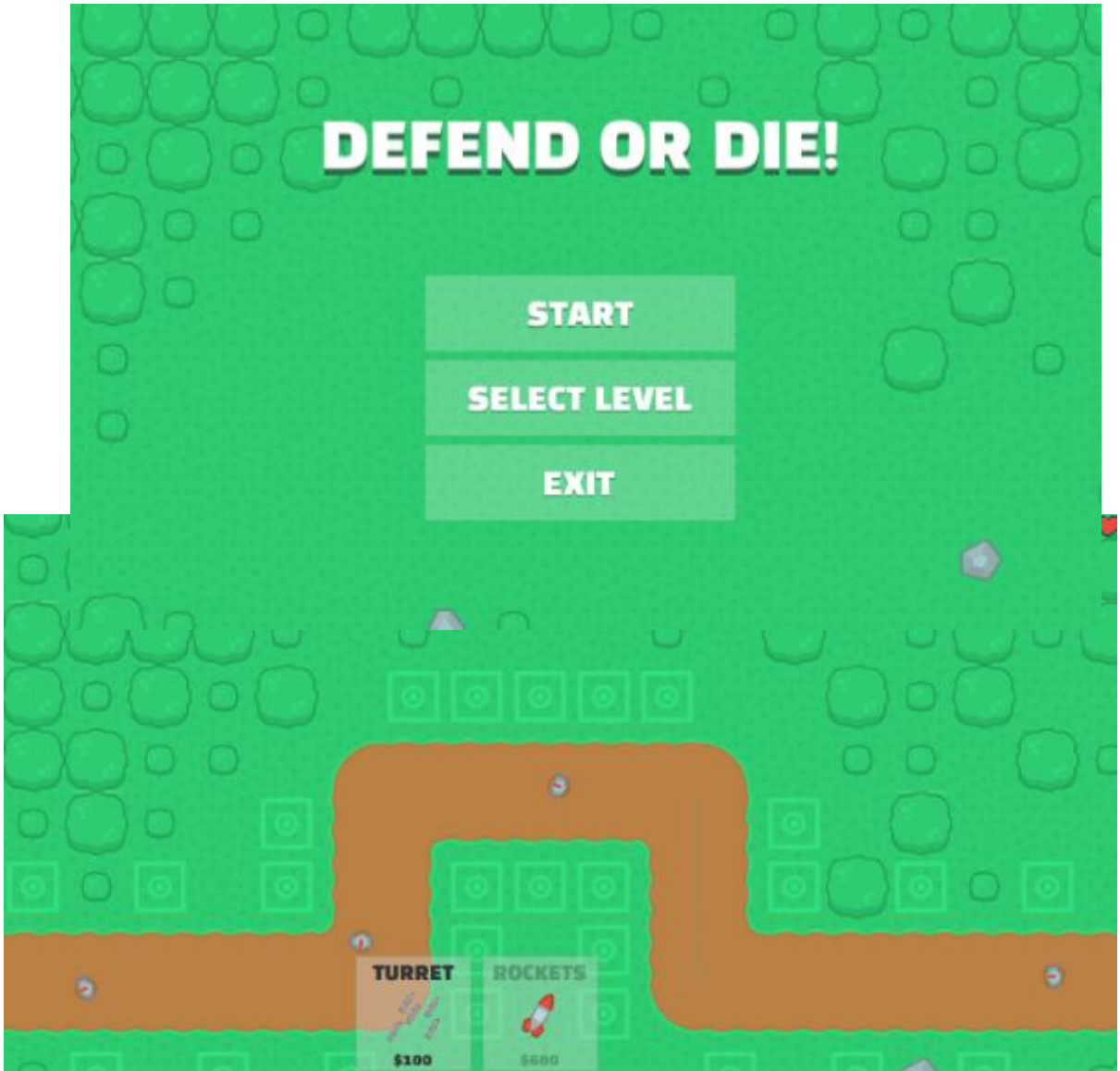
7. Experimental Results and Discussions

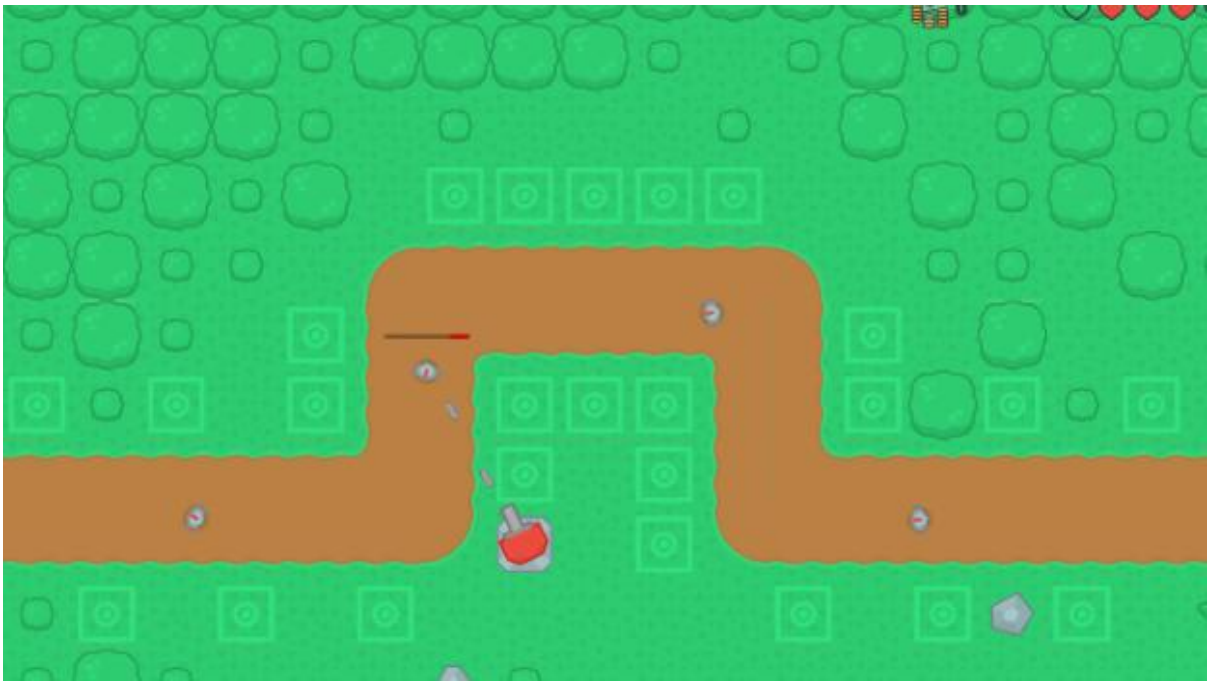
The experimental setup included building multiple levels (Level_01, Level_02, Level_03) with separate spawners and increasingly difficult enemy paths. The "Menu_screen" provides a simple user-friendly interface with Play, Options, and Exit buttons.

Key Observations:

- Performance remained smooth, with no frame drops even on low-end devices.
- The object pooling method significantly improved resource usage.
- UI animations were responsive and enhanced the gaming experience.

Observations:






GAME OVER

8. Conclusion and Future Work

In conclusion, "Defend or Die" successfully implements a lightweight, strategic tower defense experience, optimized for low-resource systems while maintaining high engagement. The modular approach ensures scalability and maintainability.

Future Enhancements:

- Introducing multiplayer mode via Unity's Networking API.
- Adding more complex enemy types with special abilities.
- Implementing player upgrade systems and skill trees.
- Creating procedurally generated levels to enhance replayability.

9. References

[1] International Journal of Computer Games Technology, The Role of Game Mechanics in Enhancing User Engagement in Mobile Games, 2018.

[2] G. N. Yannakakis and J. Togelius, Procedural Content Generation in Games: A Survey, IEEE Transactions on Computational Intelligence and AI in Games, vol. 3, no. 3, pp. 172–186, 2011.

[3] International Journal of Interactive Multimedia and Artificial Intelligence, Unity Game Engine: Revolutionizing Indie Game Development, 2017.

[4] ACM Symposium on Computer-Human Interaction in Play (CHI PLAY), Optimizing Performance in Mobile Tower Defense Games, 2019.

[5] Games and Culture Journal, Gamification and Motivation: How Player Rewards Impact Behavior, 2015.

[6] Entertainment Computing, Elsevier, Ads Versus In-App Purchases: Monetization Strategies for Free-to-Play Games, 2020.