Tower Defence: Alien Enigma

Project Proposal and Requirements Specification

Keerthish Suresh

**Contents**

[**1.** **Overview** 2](#_Toc191158423)

[**2.** **Requirements** 3](#_Toc191158424)

[**3.** **Game outline and Features** 5](#_Toc191158425)

[**4.** **Initial Design Sketches** 6](#_Toc191158426)

[**5.** **OOP concepts and Class Diagram** 8](#_Toc191158427)

[**6.** **Technical Requirements and Implementation Plan** 10](#_Toc191158428)

[**7.** **Challenges and Future Scope** 12](#_Toc191158429)

[**8.** **GitHub Repository and Conclusion** 13](#_Toc191158430)

[**9.** **Appendix** 14](#_Toc191158431)

## **Overview**

In this game of Tower Defence: Alien Enigma, the player must place their Watchtowers strategically to kill the enemies from reaching the target. By killing the enemies, player can earn resources that can be used to build and upgrade their towers. Each tower has its own unique range, strength, and attack style. Enemies have varying speed, health, and the reward capabilities. As the game goes on, the level gets more challenging with more waves and tougher enemies introduced to challenge the player.

**Justification:**

Tower Defence: Alien Enigma is chosen to challenge players with complicated improvised Watchtowers that require strategic thinking and problem-solving skills for better placements. The game integrates watchtower - defence mechanics, which adds complexity that makes it more than just a defence from the nemesis. My game selection will engage a broad audience.

This game will apply the fundamental computer science concepts such as pathfinding algorithms for enemy moved and tower behaviour, game state management and user interaction design.

Additionally, the game will be developed using Object-Oriented Programming concepts such as modular coding, inheritance, polymorphism and encapsulation.

## **Requirements**

**Objectives**

- Develop a challenging and problem solving Tower Defence game with increasing difficulty and boosting the interest of the player.

- Implement OOP principles to ensure flexibility and efficiency.

- Create a visually engaging technophile with distinct nemeses and watchtower designs.

- Provide multiple difficulty levels, ranging from beginner to expert.

- Ensuring the game is up scalable and open to future enhancements.

**Problem statement**

Most of the Tower defence games mainly focus on killing the rivals, they lack in keeping the players engaged. However my game aims to solve these problems by introducing the below features,

1. Creating dynamic tower structures that can be paced at a different location and different direction.

2. Tactical watchtower placement, requiring players to think ahead.

3. Adaptive enemy behaviour, ensuring progressing challenges.

4. An evolving system with increasing difficulty levels.

By combining these elements, the game provides a progressing challenge that keeps players involved.

**Target Audience**

- Casual, professional and committed gamers looking for immersive problem solving experiences.

- Technophiles who enjoy futuristic game aesthetics.

- Suitable for all age groups above 12, medium complexity will suite a broader audience.

- Entry level game developers can analyse and modify the game’s OOP structure.

## **Game outline and Features**

**Basic Features**

1. Tower Generation: Procedurally generated paths ensuring fresh challenges.

2. Watchtower: Players can deploy different types of watchtower to slow, damage, or slow nemeses.

3. Nemeses AI: Increasingly intelligent opponents adapt to player strategies.

4. Power-Ups: Items that enhance player movement or watchtower efficiency.

5. Heart points: Performance tracking for competition and progression.

6. Credit scores: Allow players to buy watchtowers to destroy the Nemesis.

**Game Progression**

- Level-based advancement: Players must clear multiple levels, each harder than the previous one.

- Upgrades system: Players can improve watchtowers and abilities over time by gaining more heart points.

- Storyline elements: Background knowledge connecting the towers to a larger skiffy narrative.

## **Initial Design Sketches**

**Design Sketches**

1. A blank play map

A maze with circles on it

AI-generated content may be incorrect.

1. Initial screen with points and player life

**A screenshot of a game

AI-generated content may be incorrect.**

‘

1. Route map for the enemies

A screenshot of a game

AI-generated content may be incorrect.

## **OOP concepts and Class Diagram**

**Base Classes:**

* GameObject (This will be the base class for all objects in the game, including watchtower and nemeses.)

**Subclasses:**

* WatchTower (Derived from GameObject, featuring different tower types: laser, plasma, electric.)
* Nemesis (Derived from GameObject, including different alien enemy types.)
* Projectile (Handles movement and interaction with game objects.)
* Heart point (Special items granting temporary benefits.)

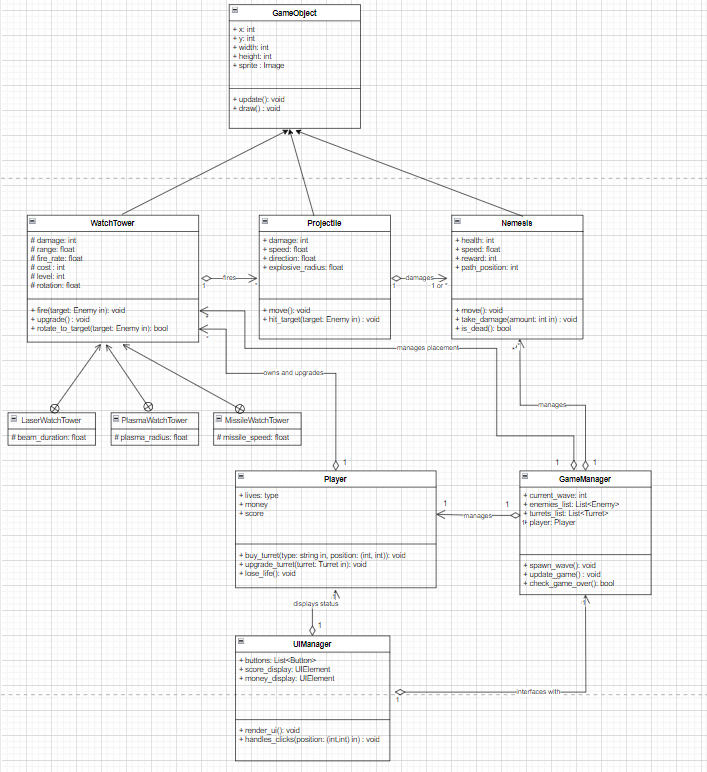
**Other Classes:**

* Player (Class used to manage the player state and owns and upgrade WatchTowers)
* GameManager (Class which manages all Player, Nemesis and WatchTower objects)
* UIManager (Interacts with the GameManager object and displays status of the Player object)

**OOP Concepts Used:**

* **Inheritance**: The WatchTower class will be a base class for the specific tower types like SniperTower and SplashTower inheriting from it. This allows for shared functionality and allowing us to customise different tower behaviours.
* **Encapsulation**: Attributes like health, damage, and range will be encapsulated within each class. Each class hides its internal state, making it easier to maintain and debug.
* **Polymorphism**: Specific tower types override the attack() method to define their unique behaviour, demonstrating polymorphism where a method behaves differently depending on the object calling it.
* **Abstraction**: Complex game elements like the interaction between enemies and towers are abstracted into classes, reducing complexity and allowing for easier management.

**Class Diagram:**



Original is attached in the Appendix of this document.

## **Technical Requirements and Implementation Plan**

**Software Requirements**

- Programming Language:

* Python v3.12.0, pygame v2.5.2
* Visual Studio Code

- Version Control:

* GitHub for source control
* Git Bash and Source tree for committing the code and pulling from the source control.

- Graphics & Assets:

* Custom-designed images for gameplay.

- Documentation and designing/sketching:

* Microsoft Word for documentation
* Draw.io for UML Designs

**Hardware Requirements**

- System Requirements: A PC with any operating system.

- Storage: At least 500MB for assets and game files.

**Implementation Plan**

1. Stage 1 (Week 1-2): Research & Initial Game Design Sketches.

Week 3 - Deliverable 1 – Project proposal and equipment specification.

2. Stage 2 (Week 3-4): Develop the Game Engine and Player Movement.

3. Stage 3 (Week 5-6): Implement WatchTower and Nemesis AI Mechanics.

Week 5 - Deliverable 2 – Detailed project design document.

4. Stage 4 (Week 7-8): Refine Game Levels and Introduce Power-Ups.

Week 7 - Deliverable 3 – Initial Prototype / Development milestone.

5. Stage 5 (Week 9-10): Testing, Debugging, and Performance Optimization.

Week 9 - Deliverable 4 – Intermediate project update.

Week 11 – Final Delivery – Final Project Submission.

## **Challenges and Future Scope**

**Potential Challenges**

- Balancing difficulty levels, to ensure impartiality and engagement.

- Optimizing AI behaviour for smooth performance and pragmatism.

- Using high quality 3D images without affecting game performance.

**Scope for enhancement:**

- Enhancement to include multiplayer features.

- Additional game modes, like time-based challenges and persistence modes.

## **GitHub Repository and Conclusion**

For ongoing development, access the source code and latest updates here:

GitHub Repository: Alien Enigma https://github.com/skeerthish/AlienEnigma

**Conclusion**

Tower Defence: The Alien Enigma is designed to push players’ strategic thinking while providing an immersive experience. With structured OOP design and sensibly created game mechanics, this game will provide both technical and entertainment value.

This document serves as a broad project proposal and requirements specification, guiding the structured development of the game. Additional enhancements will be made based on game development, iterative testing and feedback.

## **Appendix**

1. Class diagrams

