# Tower Defense Game

## 1. Project Overview

This project aims to develop a **Tower Defense Game** using **Pygame**. The game will feature a strategic gameplay where players place defensive towers to stop waves of enemies from reaching their base. The core mechanics include placing towers, upgrading them, managing in-game currency, and balancing different enemy types to create a challenging experience.

## 2. Project Review

Existing tower defense games like **Bloons TD, Plants vs Zombies** serve as inspiration.

This project will introduce unique elements such as custom AI for enemies, upgradable towers with different abilities, and dynamic difficulty scaling.

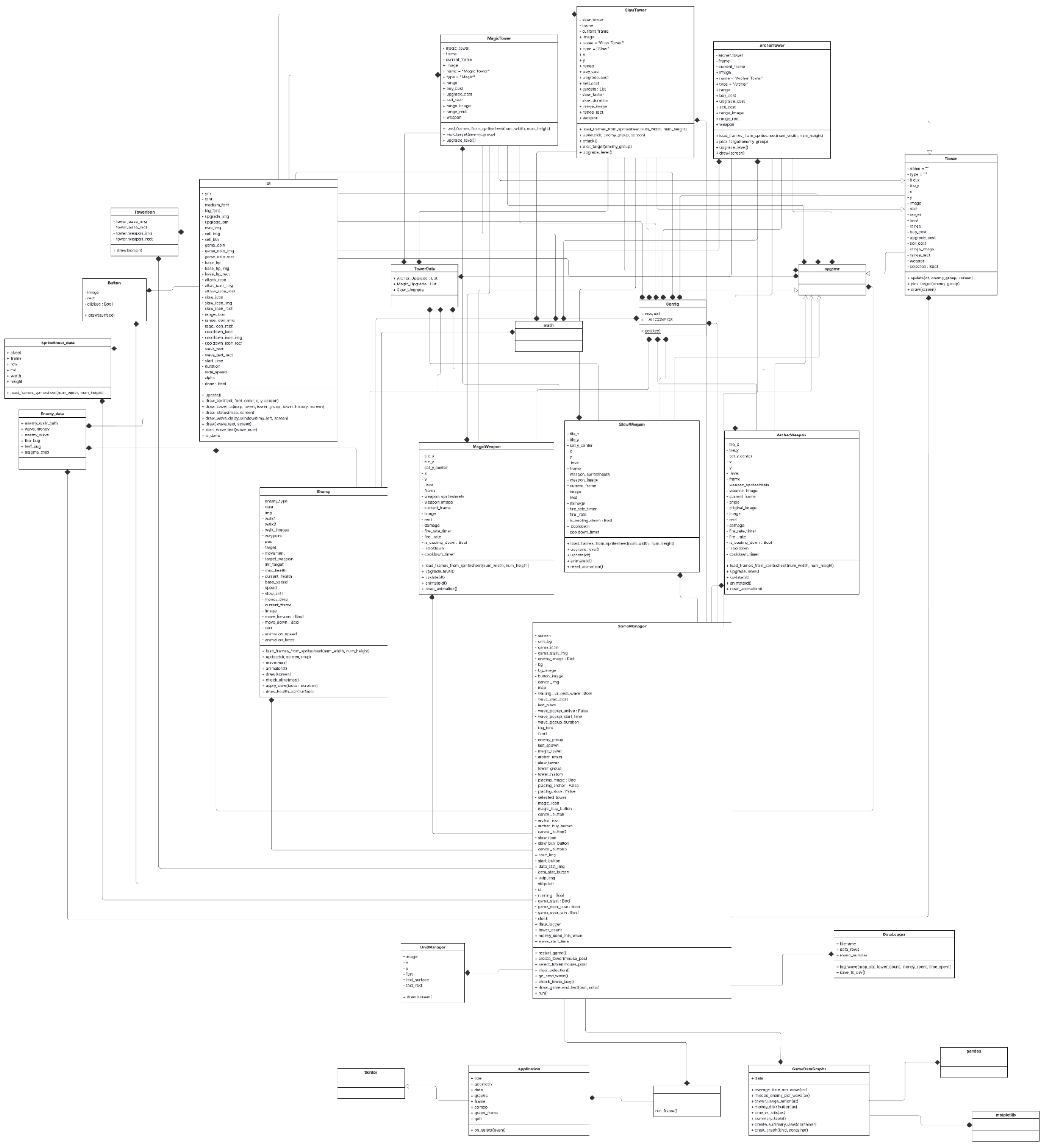
## 3. Programming Development

### 3.1 Game Concept

- Players will place towers along a predefined enemy path.
- Enemies will move towards the base, and towers will automatically attack them.
- Defeating enemies grants in-game currency to buy or upgrade towers.
- If too many enemies reach the base, the player loses.

### 3.2 Object-Oriented Programming Implementation

- UML Diagram



## 3.3 Algorithms Involved

- **Waypoint Navigation**: Enemies follow a predefined path.
- **Collision Detection**: Towers' projectiles detect enemy hits.
- **Wave System**: Dynamically increases difficulty over time.

## 4. Statistical Data (Prop Stats)

### 4.1 Data Features

1.Enemy Defeated
2.Money spent
3.Wave Reached
4.Tower Types Used
5.Enemies Reached the base
6.Time Per Wave

## 4.2 Data Recording Method

The game will store statistical data using **CSV files** to track player performance and game events.

## 4.3 Data Analysis Report

| | Why is it good to have this data? What can it be used for? | How will you obtain this feature data? | Which variable (and which class will you collect this from?) | How will you display this feature data (via summarization statistics or via graph)? |
|---|---|---|---|---|
| Enemy Defeated | Tracks how many enemies are killed by the player and helps analyze game difficulty. | Record the number of enemies defeated in 50 waves. | killed_enemise in Map class | Scatter plot |
| Money Spent | Measures player resource usage and economic behavior. | Track how much money is spent in each wave in 50 waves. | money_used_this_wave from GameManager class | Box plot |
| Wave Reached | Shows how far the player can progress; useful to measure difficulty and progression. | Record the maximum wave reached per game | wave from Map class | Table |
| Tower Types Used | Analyzes which towers are most used and helps balance the game. | Count how many times each type of tower is used in 50 waves. | mower_count from GameManager class | Pie chart |
| Enemies Reached the Base | Tracks how many enemies escape to adjust difficulty balancing. | Record the number of enemies that reach the base per wave across 50 waves. | missed_enemies from Map class | Bar graph |

| | | | | |
|---|---|---|---|---|
| Time per Wave | Helps assess how challenging each wave is by looking at time spent. | Measure and record the time spent per wave | time_spent in run method from GameManager class | Line graph |

## Table

| wave | Avg Killed | Avg Missed | Avg Spent | Avg Archer | Avg Magic | Avg Slow | Avg Time(s) |
|---|---|---|---|---|---|---|---|
| 1 | | | | | | | |

## Graphs

| Feature | Graph objective | Graph type | X-axis/ Genre | Y-axis |
|---|---|---|---|---|
| Average Time per Wave | Helps analyze how long players spend on each wave | Line graph | Wave | Time(s) |
| Missed Enemy per Wave | Indicates which wave caused the most enemy leaks | Bar graph | Wave | Missed Enemies |
| Tower Usage Ratio | Analyzes which tower types are most used by players | Pie chart | Archer, Magic, Slow | None |
| Distribution of Money | Helps analyze how players spend their money | Box plot | Money Spent | None |
| Time vs Kills | Explores the correlation between time spent and number of enemies defeated | Scatter plot | Time(s) | Killed Enemies |

## 5. Project Timeline

| Week | Task |
|---|---|
| 1 (10 March) | Proposal submission / Project initiation |
| 2 (17 March) | Full proposal submission |
| 3 (24 March) | Develop tower placement and shooting mechanics |
| 4 (31 March) | Add UI, upgrade system and in-game economy |
| 5 (7 April) | Final testing balancing, and bug fixes |
| 6 (14 April) | Submission week (Draft) |

### Weekly Goal

| Week | Task |
|---|---|
| 26 Mar - 2 Apr | - Completed Enemy class and Map class (Add more attributes and method)<br>- Maybe have more class added |
| 3 Apr - 9 Apr | - Start writing the code for remaining class<br>- Add data logging system for 5 features |
| 10 Apr - 16 Apr | - Store and log data from multiple game sessions<br>- Create statistic table<br>- Create graph |
| 17 Apr - 23 Apr | - Balance the game from data |
| 24 Apr - 11 May | - Debug and clean up code |

50% : Coding, Data collection + Visualization
75% : Game balancing
100% : Test + Debugging code

## 6. Document version
Version: 5.0
Date: 11 May 2025