Tower Defense 1 Project Plan

ELEC-A7151

Joel Toppinen, Miikka Åsnabrygg, Petrus Nikoskinen, Petteri Kippo

---

1. Basic description

Our project is going to be a classic tower defence game, in which the core gameplay consists of building towers in order to stop incoming waves of enemies. The theme of the game revolves around board games, with the towers being different chess pieces and enemies potentially resembling pieces from other board games.

The basic features of the game include different maps with unique enemy paths and wave compositions, being mixtures of different kinds of enemies. Some enemies can take more damage before dying whereas some split into multiple weaker enemies when going down. Map difficulty is adjusted through the path length as well as the speed and number of enemies. Shorter paths and faster enemies give the player less time to take them down.

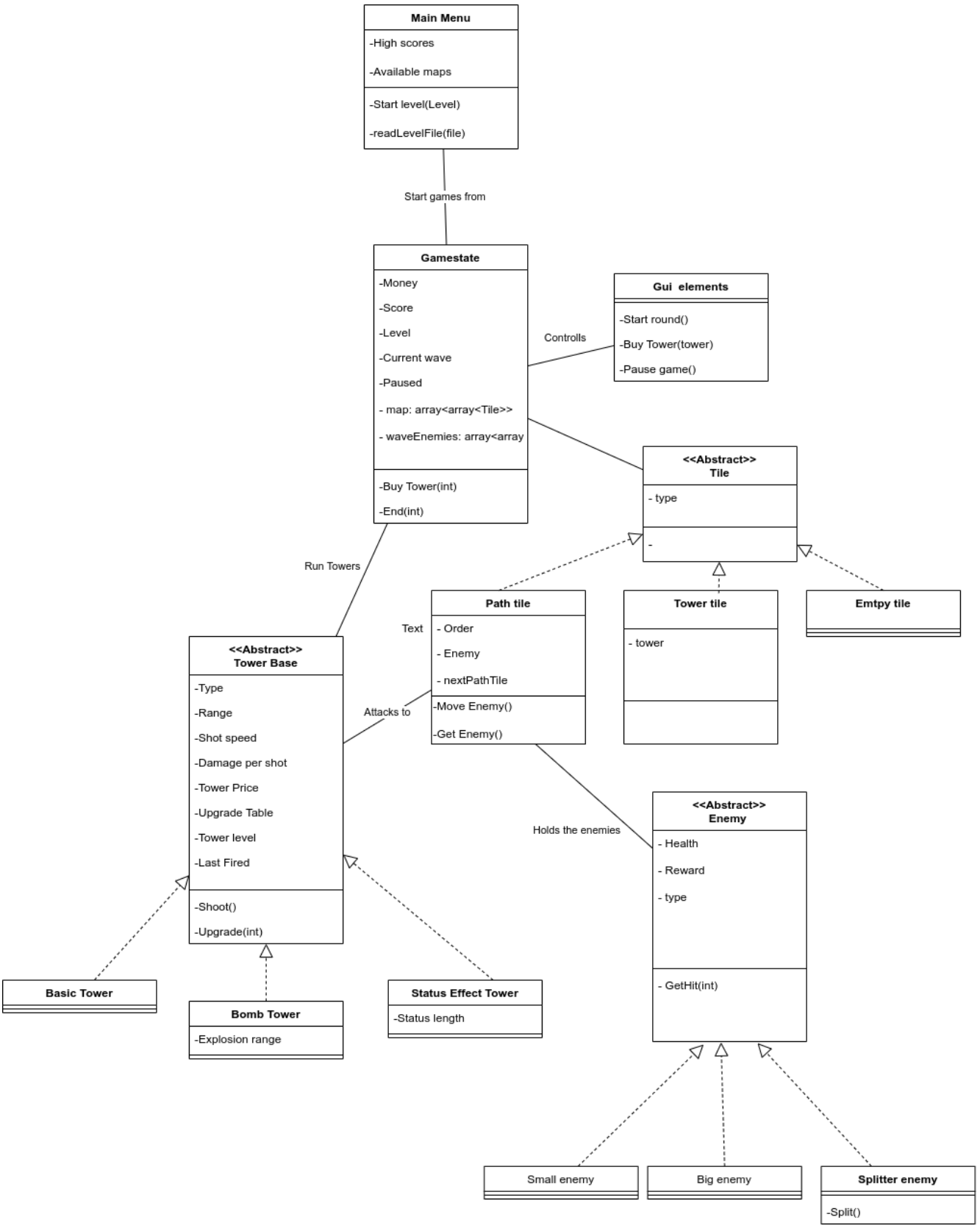
The gameplay loop consists of destroying enemies with towers, earning money doing so and then spending the money to build different kinds of towers to destroy even more enemies. Stronger enemies award more money to compensate for the additional effort and risk required to take them down. Money can also be used to upgrade existing towers to make them stronger, e.g. have more range or deal more damage.

The player has different types of towers to choose from. Towers have different ranges, damage values as well as attack types. These attributes also affect the purchase and upgrade prices of the tower. For example a tower that deals more damage or can attack multiple opponents at a time is going to cost more than an inferior tower.

The game is going to have a graphical user interface. It contains initially a menu through which the user can choose a level to play. The gameplay is going to have two distinct phases: placing and upgrading towers between waves, and then

witnessing them holding the wave off. The player is able to place towers by dragging them into position using the mouse. When placing a tower its range is displayed so that the player can assess a good position for it.

2. Internal program structure



This is a basic UML of our programs class hierarchy. The levels are stored in files containing all the relevant information . When selecting the level, Main Menu generates the Gamestate object from that information.
Gamestate stores the map as 2D Array with Tile objects. PathTile manages the Enemy objects that are on that specific tile. PathTiles are linked to each other like linkedlist, so the tile can move enemies to next tile.
When Tower is attacking to enemy, it will scan all tiles that are on its range, and it will attack to enemy which is on highest ordered PathTile (nearest tile to the last tile).
Currently we don't have any insight on how the GUI class will be implemented as it is dependant on the library no of us have any previous experience from. It should simply depict the current gamestate and allow us to make changes through it.

3. External libraries

For now the only major external library we need is the SFML that we will use to build the visual elements on the graphical user interface, as well as potential audio elements. Further tools will be considered if a significant need suddenly rises.

4. Division of work and responsibilities between the group members

We are going to partially specialise so that e.g. not everyone has to learn the gimmicks of an external library. Some are going to focus on the GUI and its functionalities while others work on the engine behind the actual game process. However, everyone is responsible for keeping up with all of the functionalities on a general level, and can ask help from the corresponding person if deeper understanding is required.

5. Planned schedule and project milestones

As a group, we will have scheduled meetings two times a week either on campus or remotely depending on the circumstances. The first meeting of the week will tentatively take place on Monday, with the second meeting occurring later during the week when group members have time, most likely either Wednesday or Thursday. During the meetings, we'll go over our short-term goals of the current week and evaluate our progress as a group. Between the meetings, communication will take place mainly in Teams and Telegram.

Our first main milestone is the minimum viable product, on top of which additional features may be added. This version should have at least one working type of tower, enemy, and a map to play on, as well as a basic user interface so that the main gameplay loop can be roughly demonstrated. These features should be implemented *at least* before the project demo day, which take place starting Monday, 5.12.

However, considering that the project demo day is right around the final commit deadline of 9.12., it would be wise to have these basic features ready much earlier, perhaps even two weeks before the deadline. This way the next week would be reserved for adding the rest of the required game object types and all required functionality for a passing grade, as outlined on the Tower Defense project page. The final week would then be reserved for implementing purely additional features, with the final milestone being the finished product at the final commit deadline.

Tentative schedule outline:

Week 1 (14.11 - 18.11): Setting up, game engine, basic menu GUI

Week 2 (21.11 - 25.11): Minimum viable product (basic functionality)

Week 3 (28.11 - 2.12): Finishing up mandatory features (for a passing grade)

Week 4 (5.12 - 9.12): Additional features, documentation, finished product