Aalto University
ELEC-A7151 Object oriented programming with C++ autumn 2022

Tower Defense 2

Project plan

Peltokangas Elias, Kabir Bissessar, Juho Poteri, Antti Pekkanen
8-11-2022

1. Scope of the work

WHAT FEATURES AND FUNCTIONALITIES WILL BE IMPLEMENTED, HOW IS THE PROGRAM USED, HOW DOES IT WORK

The project aims to create a university / schoolwork themed tower defense game. The game will be completed as a group project during November and December 2022.

Tower defense games in general

"Tower defense (or informally TD) is a subgenre of strategy video game where the goal is to defend a player's territories or possessions by obstructing enemy attackers, usually achieved by placing defensive structures on or along their path of attack." (Wikipedia, 2022)

In a tower defense game, the enemies move in waves from some position of the map to another. The goal of the player is to place towers on their path to block, impede, attack, or destroy the enemies before they can reach their goal. The primary object is the survival of the base. (Aalto C++ course, 2022)

An example tower defense GUI (gamedev.stackexchange.com, 2022)

Features and functionalities

The game will start by loading a map from a text file using the game's graphical user interface. These maps can be created by the player, but ready-made maps will also be included within the game files. All rounds of a single game instance will be run on the same map. The maps will be built with a grid system.

The game itself will run by alternating between two states: building and waves. During the building state, the player can purchase and place towers. The player also has the option to buy different kinds of boosts / upgrades during that time.

When the player toggles the wave state, different kinds of enemies will run through the road/track on the map. Towers will try to attack and destroy these enemies. If an enemy reaches the end of the track, healthpoints of the player will be deducted. Each round the waves get progressively harder with different kinds and amounts of enemies, while the player will be able to build more and better defenses. When the HP reaches zero, the game is lost. The high scores (number of cleared rounds) will be stored locally and can be viewed.

Planned feature list

- GUI
  - Control the game with your mouse
  - Sound effects
- Two state system:
  - Building state
  - Unlimited waves, progressively harder
- Maps:
  - Create and load maps as text files (ad-hoc syntax)
  - Single path maps
- Money system and shop with towers and boosts
- Towers:
  - Upgradable
  - 6+ with different features (range, health, abilities)
- Enemies
  - 6+ with different features (attack, health, splitting)
- Player healthpoint system
  - Game is lost when HP = 0
- Locally saved high scores
- More additional features may be added

2. High-level structure of the software

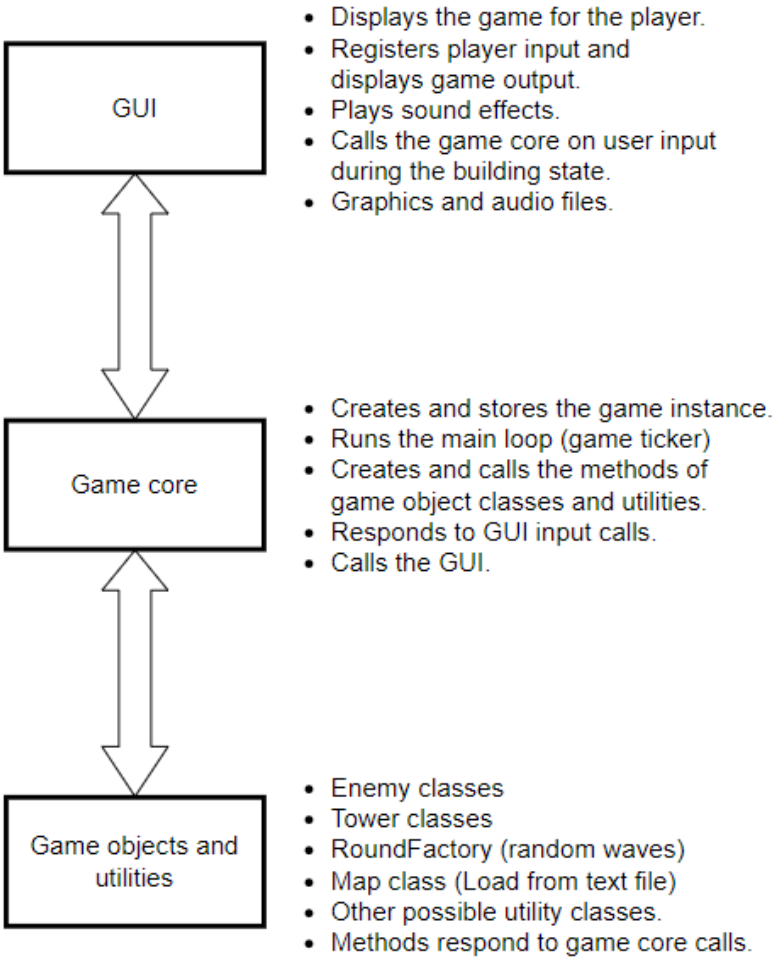
MAIN MODULES, MAIN CLASSES (ACCORDING TO CURRENT UNDERSTANDING)

The high-level structure of the game consists of three different modules that communicate with each other. GUI module is responsible for the displaying the game and sounds as well as user input/output. GUI will call the Game core on user input.

The game core runs and stores the game instance (map, enemies, towers, player information, wave information etc.). During the wave state it runs the gametime ticker, calls the methods of game objects and utilities, and talks to the GUI to display all the information. During the building state, it operates the store etc.

Thirdly, the Game objects and utilities module consists of the necessary classes that represent all the functioning parts of the game. These include for example the abstract tower and enemy classes, all their subclasses and the map class.

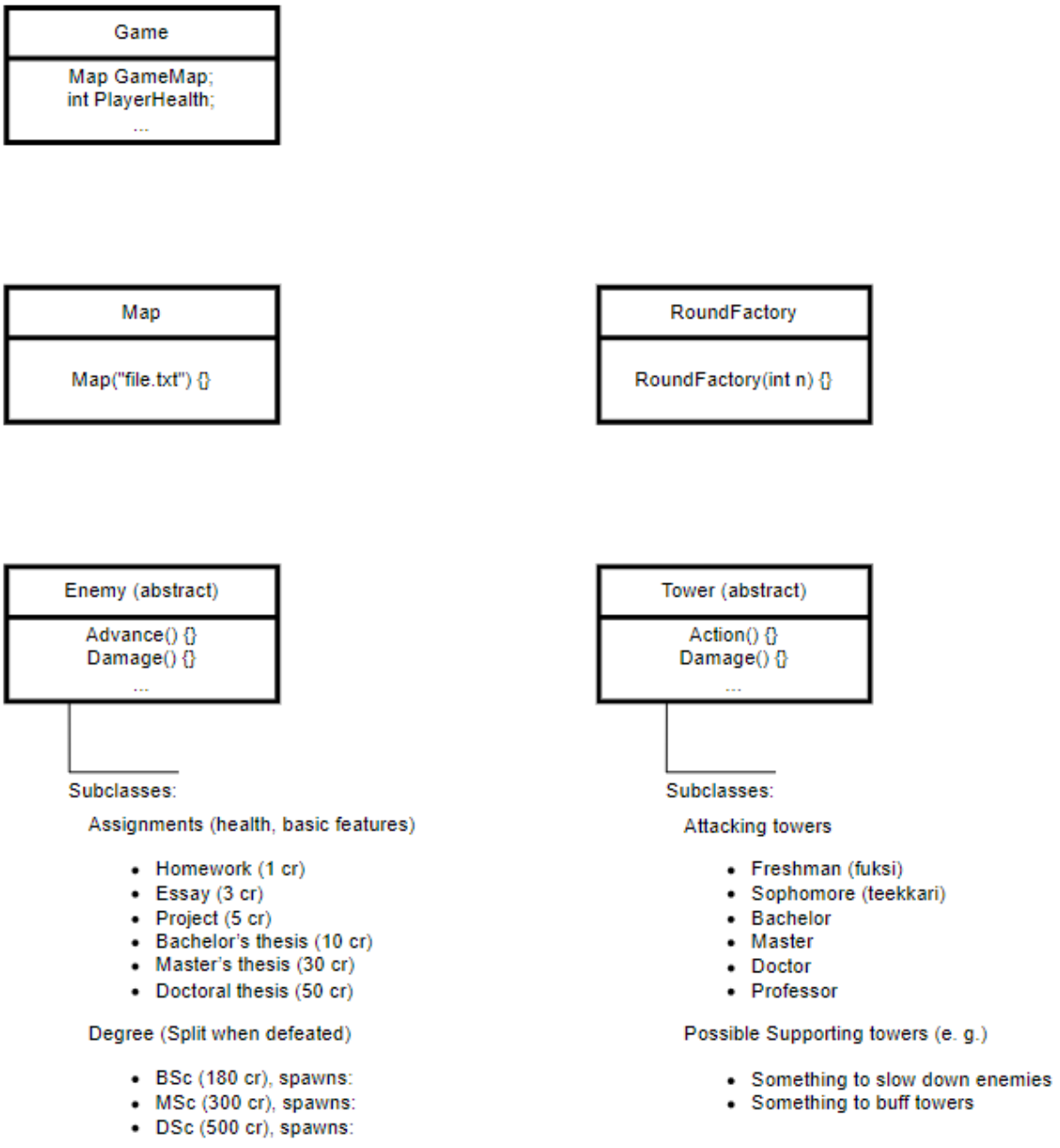
Program modules and their main functions



Planned main classes

The planned main classes are Game (the game instance), Map (the map which can be instantiated from a text file using an ad-hoc syntax), RoundFactory (random wave of enemies, difficulty based on the round number), Enemy (abstract class) and its subclasses (different kinds of enemies with different properties and features) and Tower (abstract class) and its subclasses (different kinds of towers with different properties and abilities).

The necessary GUI classes will be determined when that module is designed in more detail as we have slim previous experience on that front. Also, other utility classes and/or classes needed for additional features can be created if necessary. For example, the need for utility classes for the game class (class player etc.) will be decided later.



3. Planned use of external libraries

The use of external libraries will be fully determined as the project progresses. Only a library for the GUI module is intended to be included. For the GUI, the SFML (https://www.sfml-dev.org/) library will be explored as recommended in the project instructions. Other libraries will be included only if deemed necessary during the development phase.

External libraries:

- GUI module: SFML (https://www.sfml-dev.org/)

4. Schedule and division of work

The Gantt chart of the schedule has the planned times and slacks of each main activity, milestones for each week and the responsible person for each task.