Siim Kasela - 1006443
Saku Kovanen - 1024713
Harvey Lim - 1026986
Hung Vu - 1006058

ELEC-A7151 Object oriented programming with C++

Project Plan

Scope and features

Description

The aim of this project is to build a classic tower defense game [1] with its own theme and learn to use the Qt framework [2] in C++ while implementing the game. Furthermore, the final game should be easily extendable for future development, should work robustly and implement enough features to provide an enjoyable experience for the player.

The theme of this game is going to be “programming students versus C++ errors”. The towers will represent different people from the university (i.e. students, teaching assistants (TA)) or tools that can detect and help eliminate errors (i.e. valgrind and language server). Enemies will be represented by various errors that a programmer might encounter during development.

The game will be divided into successive waves of enemies that follow a path and the time flow will be continuous throughout a wave. When an enemy reaches the end of the path, it will deal damage to the player's reputation (health). During and between the waves, the player can buy and place towers at predetermined locations on the map and they have a certain range where they can shoot at the enemies. The game will have a saved local leaderboard, which means it will keep track of the points received. More in-depth descriptions of the features can be found in the next section.

The features

(Features titled with *italics* are “additional features”)

- **Attack towers**

These towers shoot at enemies within their range and deal damage. Attack towers have 2 main stats that affect its damage output: power (the damage per hit) and attack speed (the frequency at which the tower fires).

There are two types of attack towers, direct attack (e.g. TA) and AOE (e.g. Language Server). The AOE towers deal damage to all enemies within a certain radius, while the direct attack towers can only target a limited number of enemies at once. In addition, the direct attack towers further split into two types: high damage, long-range slow-firing towers and medium-damage fast-firing towers.

Aside from the classification above, the attack towers also have two target classifications: memory attack and compiler attack. The towers can only target enemies that match their target classification (i.e. memory attack towers only shoot at enemies of memory error type).

The planned attack towers are as follows:

- TA (support/direct attack): boost towers around it and also deal damage to the enemies. Can be upgraded to the Teacher tower.
- Valgrind (direct attack): sniper type tower that has a long range, targets memory enemies.
- CS student (direct attack): deals damage to all enemies, medium range, attack speed, and damage.
- Language server (AOE attack): deals damage to all enemies within range. Can be upgraded to level 2 (Treesitter) and level 3 (Github Copilot)

- **Support towers**

The main role of support towers is to buff towers and/or debuff enemies within their range. This means that support towers can increase the stats of ally towers or decrease the stats of enemies accordingly. However, some support towers also deal a small amount of damage to the enemies within its range.

The planned support towers are as follows:

- Search engine: boosts towers within its radius.
- TA: boosts towers within its radius and also deals some damage to the enemies.
- Comment: can block the enemy's path, forcing them to either break the Comment tower or choose another path.

- **Enemies**

Enemies are the attacking units in a tower defense game. They are divided into two classes: Memory errors and Compiler errors. Towers can only shoot at the enemy type that matches its target type. Some towers get damage multipliers and deal extra damage to certain types of enemies.

The planned enemies are as follows:

- Memory errors

The spawn rate of these errors are low and they are harder to solve. They will also drop more money when solved.

    - Invalid read of size x: medium stats enemy
    - Invalid write of size x: medium stats enemy
    - X bytes are definitely lost: tank type, slow-moving but high hit points.
    - Mismatched delete / free: high stats enemy

- Bosses (Runtime Errors)
    - Stack overflow: spawn from the depths of the abyss, at the very bottom of the recursion tree. Stack overflow is a boss with a large number of hitpoints, with 5 different stages. After each stage, the boss spawns a memory stack minion from its stack. The minion is a fast-moving, medium-hitpoint unit that will run in front of the boss and take some damage for the boss.
- Compiler errors

The spawn rate of these errors are high and they are easier to solve.

    - Exception: medium-hitpoint enemy, splits into multiple smaller and lower-hitpoint exceptions when solved.
    - Syntax errors: one-hit KO enemies, spawns in groups.

- ***Multiple Paths***

(From this point onwards, the place the player is trying to stop the enemies from getting to will be referred to as the 'end' of the map. There may be more than one 'end'.)

Depending on the map, there can be multiple possible paths enemies may take to get through the map. If an enemy comes across a junction where there is more than one path they can take to get to an end of the map, it will randomly choose one of the said paths.

- **Special Tower: Comment**

The Comment tower is a special type of tower. It does not do anything to towers nor damage enemies, however, it can be placed on the enemy path to block or redirect enemies.

Comment towers also have a special stat called "mental health points", which they are given a set number of when they are initially placed.

If all possible paths enemies can take to get to the end of the map are obstructed by Comment towers, the enemies may damage a Comment tower to destroy it and restore a path for them to use to get to the end of the map.

- ***Pathfinding***

Enemies also have basic pathfinding capabilities, as the Comment tower has the ability to obstruct paths, thus the enemies are able to find the next shortest unobstructed path to the end of the map when a Comment tower is placed. In the scenario where all paths to the end(s) of the map are obstructed by Comment towers, the enemies will find the shortest route to the end of the map (without considering Comment towers) and attempt to destroy any Comment towers they come across on that path.

- **Reputation Points**

In this game, the player is given a fixed number (positive integer) of "reputation points" when the game begins. When an enemy reaches an end of the map, the number of reputation points the player has will be deducted by a set amount, depending on the enemy that reached the end. If the number of "reputation points" the player has reaches 0, the player will be "fired" and the game will end with the player's loss.

- **Currency System (IOPS)**

This game has a currency system called “IOPS”. IOPS may be used to purchase towers, and is earned when a tower kills an enemy. The number of IOPS gained from killing an enemy depends on the strength of said enemy.

- ***Building and Upgrading Towers***

Towers may be upgraded and built at any time during the game. The action of building a tower and upgrading a tower require a set number of IOPS.

- **Selling Towers**

Towers can be sold to return 80% of its total value back to the player. The tower’s total value is denoted by the collective number of IOPS used to build and upgrade the tower up to that point of time.

- **Game Interface**

The game has a visual interface, which is entirely controlled by mouse inputs.

- **Loading and Saving Maps**

Maps can be saved into files, and can also be restored and loaded.

- **Local Leaderboard**

A locally-saved file will contain a list of high scores achieved by the player, and can be viewed in the in-game leaderboards.

- **Sound Effects**

Sound effects are used in the game for when shots from towers hit enemies, when towers are placed, and when enemies get to the end of the map. Most of the sounds used are keyboard and mouse sounds.

If we have extra time:

- **Level Editor**

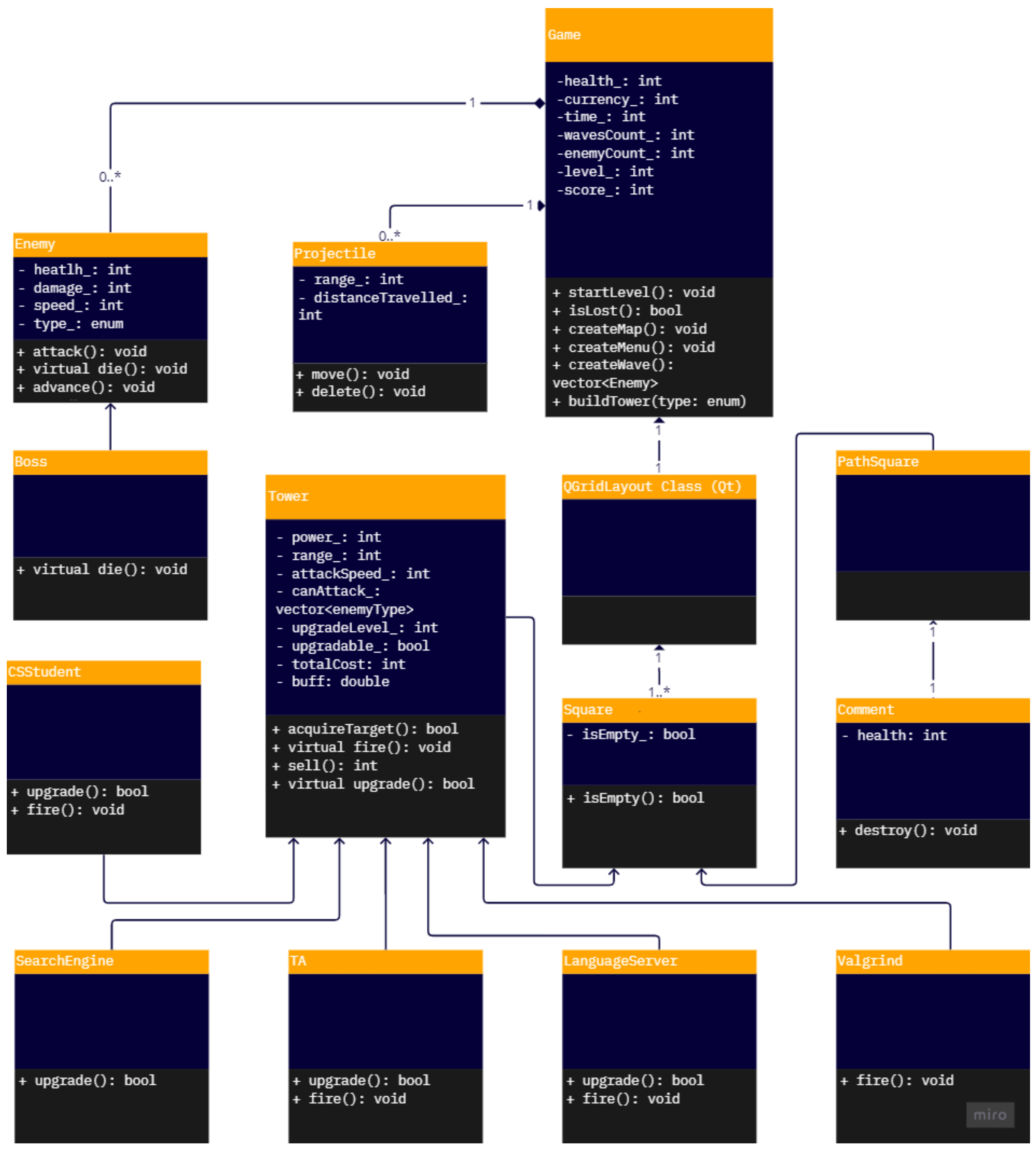
A level editor is built into the game to allow the player to create their own custom maps and set their own custom enemy waves. This can be used in combination with the loading feature to share custom maps around.

- **Multiplayer Leaderboard**

Besides the local leaderboard, there is a multiplayer leaderboard which shows the high scores achieved by anyone playing the game, across all devices that the game can be run on.

Structure of the software

UML class diagram



The UML diagram was constructed to the best of our current ability. Since none of us has experience with the Qt framework, some details might change during the development.

The get and set functions for private variables are not referenced in the diagram for readability.

The graphics, time flow, and positions of elements of the game will be set with Qt framework classes.

Class descriptions

- **Game**

The main class for the game with all the necessary game relevant variables like the player's health, game currency, number of current enemies, current level etc. The map will be created in this class' constructor.

- **Tower**

An abstract class for the towers. Has the variables common to all the towers and methods for acquiring a target, selling the tower and firing a projectile at enemies.

- **CSStudent, SearchEngine, TA, LanguageServer, Valgrind**

Classes for specific tower types. These classes inherit from the abstract class Tower and if it is upgradable the virtual function for upgrading the tower is specified. The fire method will be specified according to the type of the tower.

- **Enemy**

A class for enemies. The type of enemy will be specified with an enum type. Each enemy has a certain amount of health and will die once it is depleted.

- **Boss**

Special type of enemy that inherits from Enemy specifies the virtual function die() according to its special ability of splitting to multiple enemies.

- **Projectile**

Class for representing the projectiles fired by the towers. Projectiles will be deleted on collision with the enemies or when they have reached the maximum range of the tower that shot the projectile.

- **QGridLayout**

Qt framework class for representing a grid for the game.

- **Square**

A class for representing one square in the game grid. A square can either be a tower position, general map area, or a path square. A path square can also be blocked for the enemies by a wall(support tower).

- **Comment**

Type of path square that represents a wall that will block the advance of the enemies. Has a certain amount of health and can be destroyed by enemies.

External Libraries

- Qt library [2]
  - Graphics
  - Input handling
  - Audio output

Division of Work

2 stages:

- 1st stage game setup
  - Game class (Siim)
  - QGridLayout class (Siim, Saku)
  - Basic tower (Hung)
  - Basic projectile (Siim)
  - Basic enemy (Hung)
  - Basic path (Harvey)
  - Initial graphics (Saku)
  - CMake/QMake configuration (Harvey)
  - Meeting notes(randomized)
- 2nd stage
  - Tower implementation (Hung)
    - Tower upgrades
    - Selling towers
  - Enemy implementation (Hung)
    - Bosses
  - Path implementation (Harvey)
    - Comment
    - Pathfinding
  - Projectile implementation (Siim)

    - Map saving/loading (Harvey)
    - Tutorial (Saku)
    - GUI (Saku)
        - Main menu
        - In-game interface
        - Upgrade interface
    - Sound effects (Siim)
    - Local leaderboard (Siim)
    - Documentation (Doxygen generated, everyone)
    - Meeting notes (randomized)

Planned Timeline

- 1.11.22 - Brainstorming
- 8.11.22 to 11.11.22 - Creating the project plan
- 8.11.22 to 15.11.22 - Learning Qt framework
- 15.11.22 to 5.12.22 - Implementation of project
    - 15.11.22 to 22.11.22 - Stage 1
    - 22.11.22 to 1.12.22 - Stage 2
    - 1.12.22 to 5.12.22 - Testing
- 05.12.22 to 9.12.2022 - Testing and bug fixes

References

[1] ‘Tower defense’, *Wikipedia*. Nov. 06, 2022. Accessed: Nov. 11, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Tower_defense&oldid=1120316900

[2] ‘Qt | Cross-platform Software Design and Development Tools’. https://www.qt.io (accessed Nov. 10, 2022).