# Project Summary & Guide

## 1. Overview

Tower_defence.cpp is a programming-themed tower defense game, with different programmers and programming tools as towers and bugs as enemies. The objective of the game is to defeat the bugs and survive through all the waves.

Each tower type has its own ability and the player can place towers strategically to fight against the waves of enemies. Every enemy that reaches the end of the map will deal a certain amount of damage to the player, if the player runs out of health, the game ends and the player's score will be saved to the leaderboard. (More on the specifics of the leaderboard will be discussed later along with the gamemodes.)

## 2. 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: damage (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.

Each tower can target certain enemy types depending on its level and ability.

The planned attack towers are as follows:

- TA (support/direct attack): Boosts towers around it and also deals 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 compiler errors, 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 obstruct 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 three classes: Memory errors, Compiler errors and Runtime errors. Towers can only shoot at the enemy type that matches its target types.

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-stat enemy
    - Invalid write of size x: medium-stat enemy
    - X bytes are definitely lost: tank-type, slow-moving but high hitpoints.
    - Mismatched delete / free: high-stat enemy
- Bosses (Runtime Errors)
    - Stack overflow: Spawns 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.

*(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.)*

- **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 "duration", which is the amount of time it will hold back enemies before being destroyed.

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 “mental health points” (or just “health points” for short) 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 towers**

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

- **Upgrading Towers**

Some towers can be upgraded. Upgrading a tower will improve it in a unique way. Some upgrades may improve attack speed, damage, range, or even its ability to buff other towers better. Upgrading can also unlock the tower's ability to target enemy types that it couldn’t before.

- **Selling Towers**

If the player decides that they need to place the tower in another location, they can sell the tower and place it in another place. Towers can be sold to return 70% 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 can be entirely controlled by mouse inputs. However, keyboard hotkeys are available if the player wishes to speed up their playstyle.

- **Loading maps**

Maps are loaded from a custom file. The custom file can be edited to create new maps. Instructions for writing to this file are in the same directory as the map file itself.

- **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. The sound effects used consist of keyboard sounds and mouse click sounds.

## 3. Software structure

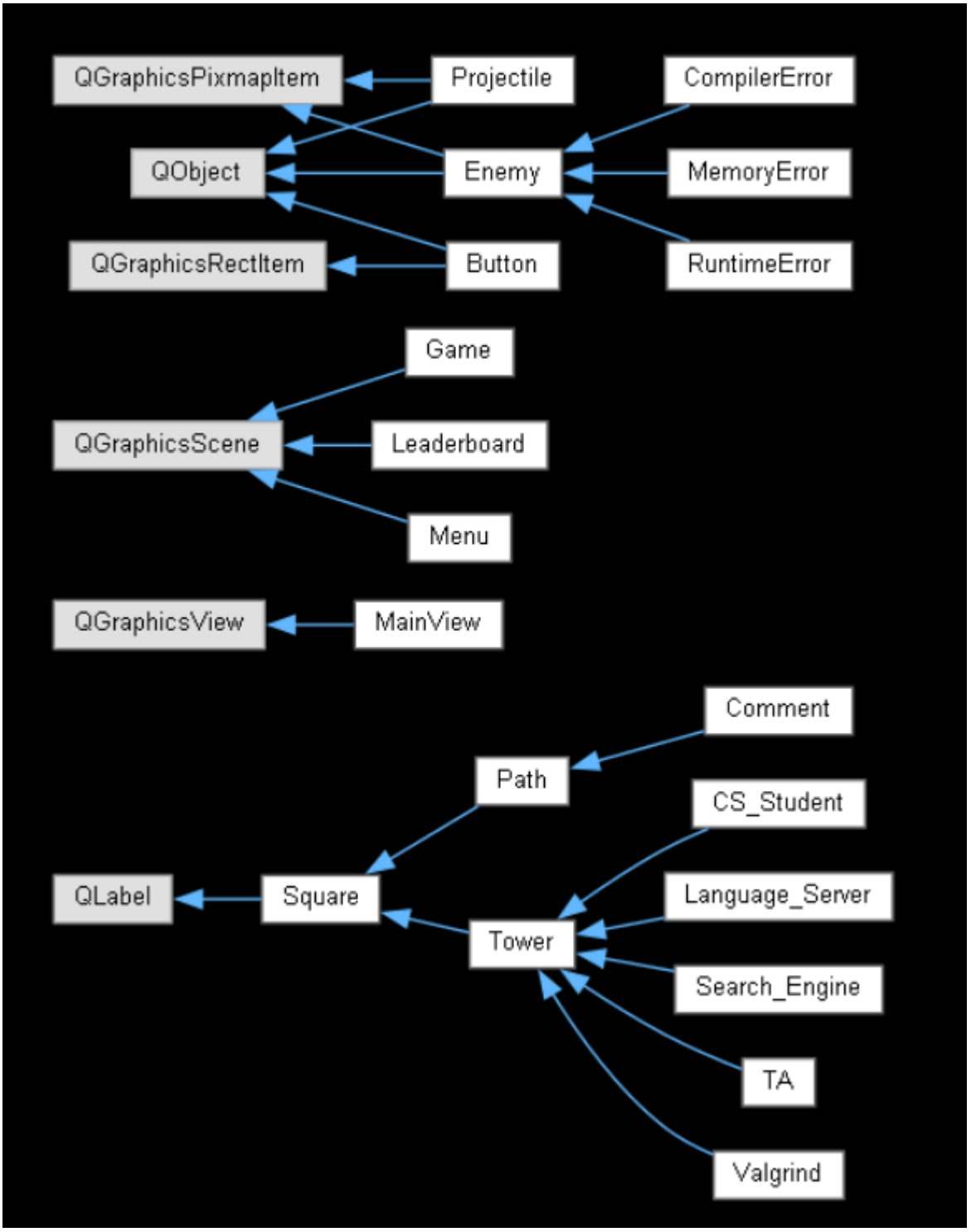



*Figure 1: Project's graph diagram*

A graph relationship diagram can be seen from Figure 1. In essence, the interaction between the custom objects and the external library (Qt) is handled solely through the base class. The inherited classes can thus make use of the available functions from Qt to perform needed actions such as displaying graphics, handling logic, and

memory management. The further detailed relationship between the classes can be seen from Figure 1.

The game has a general management class, called Game, which is inherited from the QGraphicsScene class. There is also a MainView class, which handles displaying the game graphics. The game has a QGraphicsGridLayout which is essentially a map that contains a matrix of Squares. The Tower and Path class both inherit from the Square class, so they can be placed onto the game's map. The Tower can fire and deal damage to the enemy by creating a Projectile object.

As for the enemies, the base class Enemy is inherited from the QObject class. Through the QObject's functions, the program can display the enemy and handle interactions such as movement and positioning.

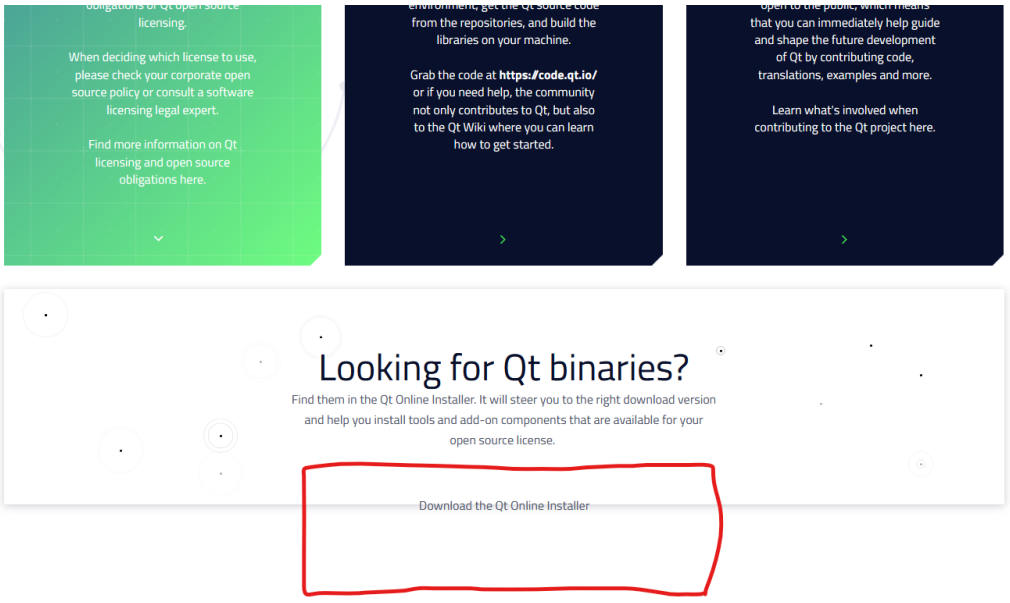
The game's GUI has 3 scenes: Game (which is also the game class) is where all the action takes place; Menu is the main menu of the game; Leaderboard is where the highscores are displayed. Each scene contains a variety of buttons for controlling the game, which is either a custom-built Button object, or Qt's built-in button selection.

# 4. Build instruction

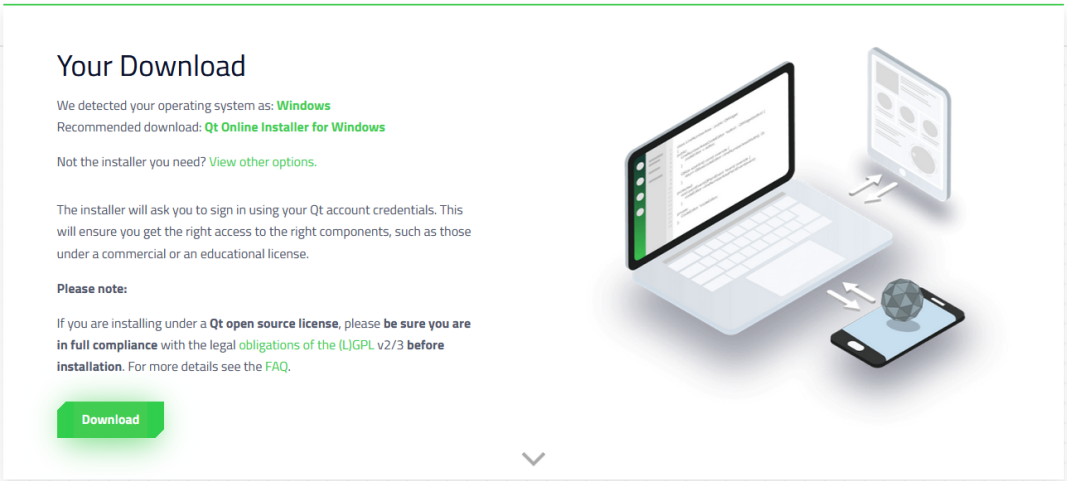
## 4.1. Install Qt

Since the program was developed using the recommended Qt framework, Qt library is required for building and executing the program.

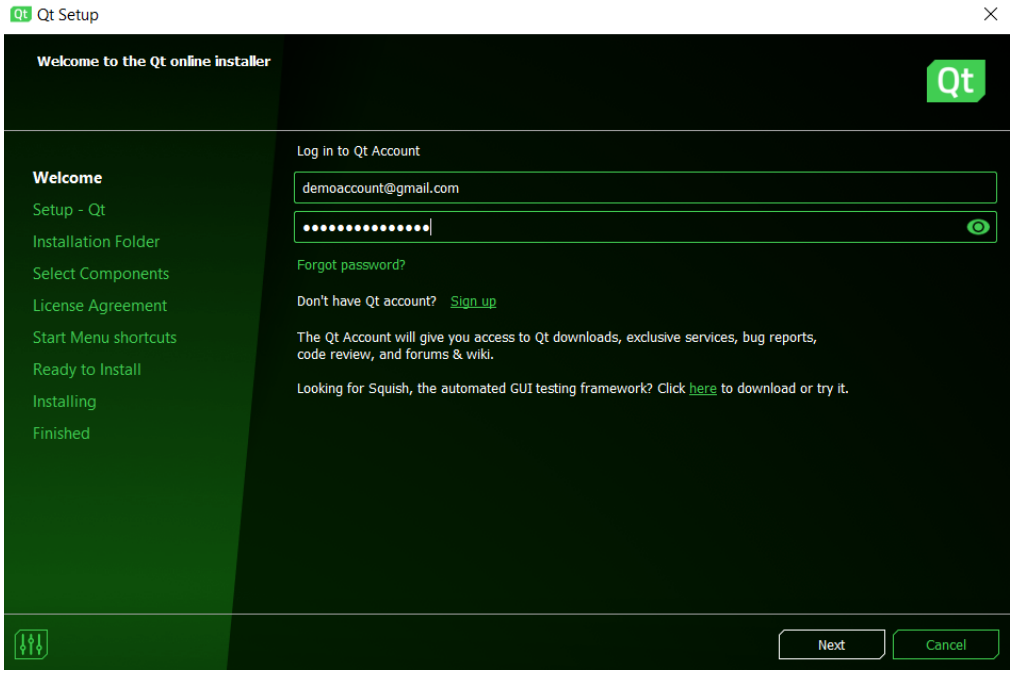
- Go to https://www.qt.io/download, select "Download Qt for open source use". Click on "Download the Qt Online Installer".



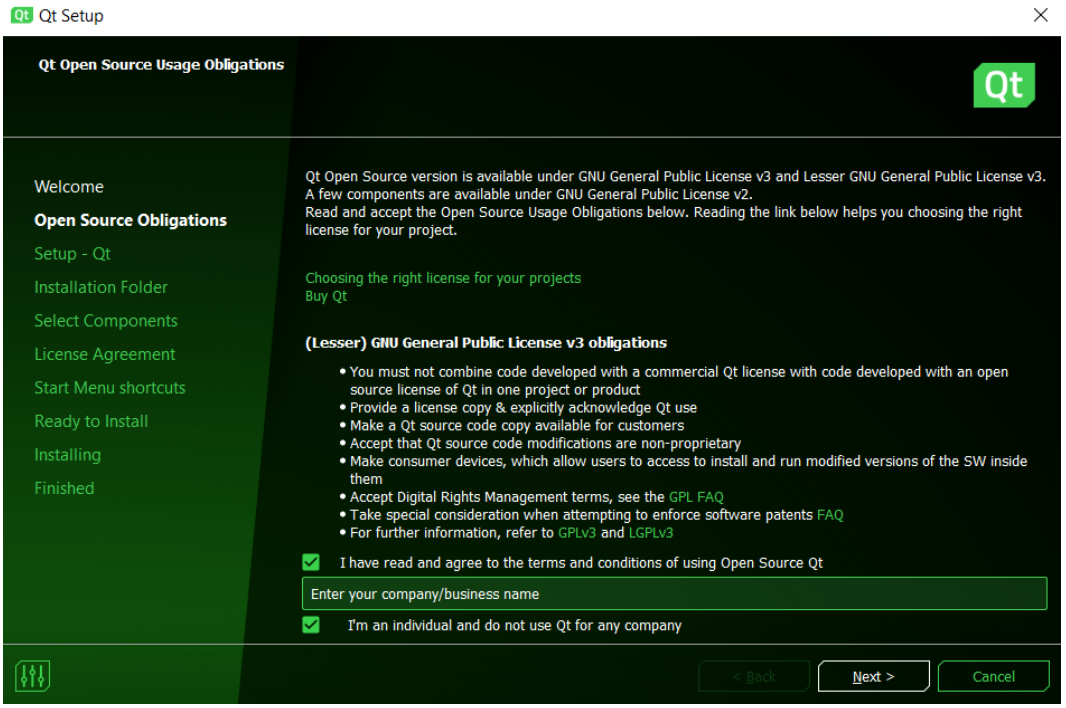

- Click on Download.



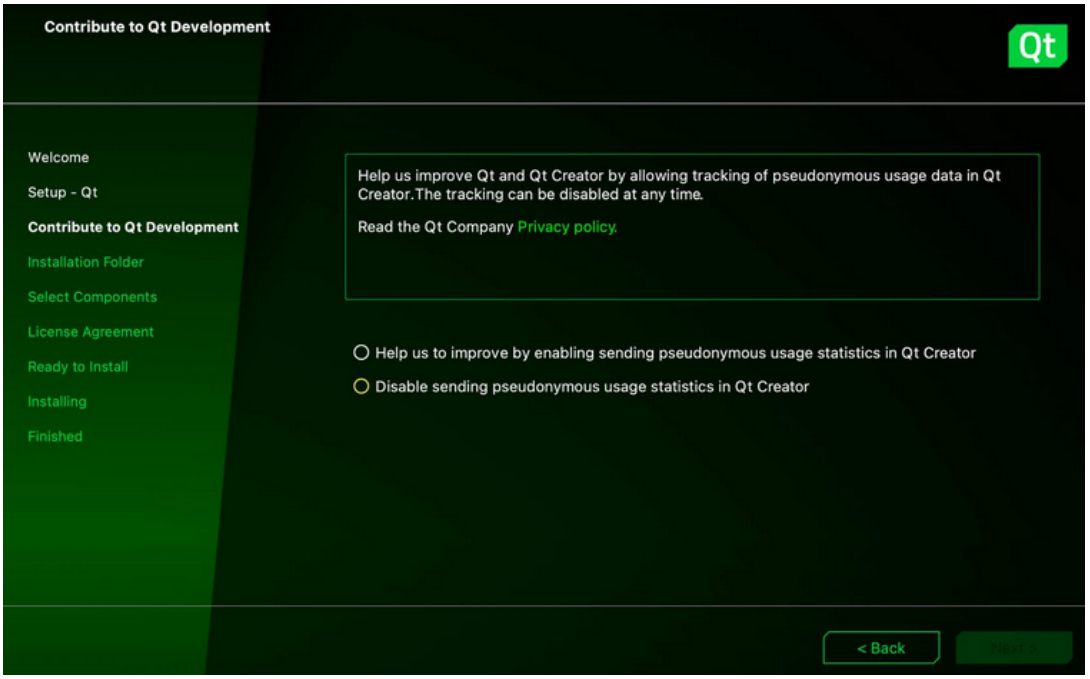

- Run the installer, on the first time opening the installer the user needs to create an account. Login after creating an account.



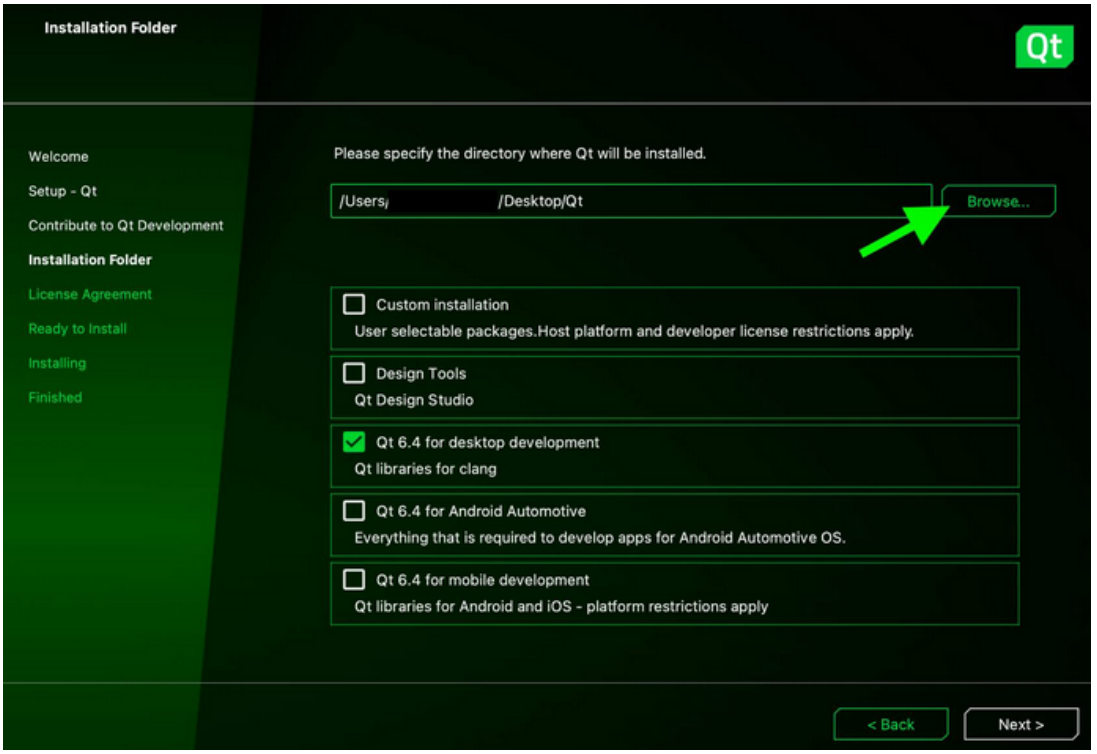

- Click the checkboxes after reading the terms and conditions and click next.


Qt Setup
Qt Open Source Usage Obligations
Welcome
Open Source Obligations
Setup - Qt
Installation Folder
Select Components
License Agreement
Start Menu shortcuts
Ready to Install
Installing
Finished
Qt Open Source version is available under GNU General Public License v3 and Lesser GNU General Public License v3.
A few components are available under GNU General Public License v2.
Read and accept the Open Source Usage Obligations below. Reading the link below helps you choosing the right license for your project.
Choosing the right license for your projects
Buy Qt
(Lesser) GNU General Public License v3 obligations
• You must not combine code developed with a commercial Qt license with code developed with an open source license of Qt in one project or product
• Provide a license copy & explicitly acknowledge Qt use
• Make a Qt source code copy available for customers
• Accept that Qt source code modifications are non-proprietary
• Make consumer devices, which allow users to access to install and run modified versions of the SW inside them
• Accept Digital Rights Management terms, see the GPL FAQ
• Take special consideration when attempting to enforce software patents FAQ
• For further information, refer to GPLv3 and LGPLv3
I have read and agree to the terms and conditions of using Open Source Qt
Enter your company/business name
I'm an individual and do not use Qt for any company
< Back
Next >
Cancel

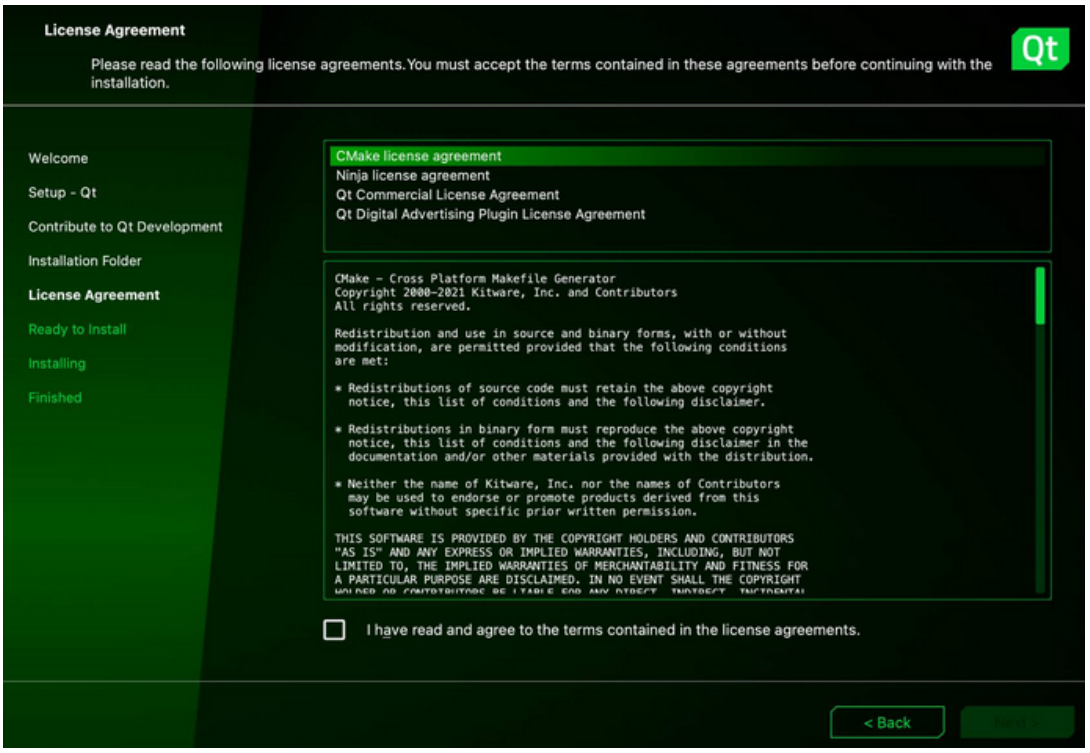
- Wait for Qt to finish downloading metadata. Choose whether you want to track usage data or not.



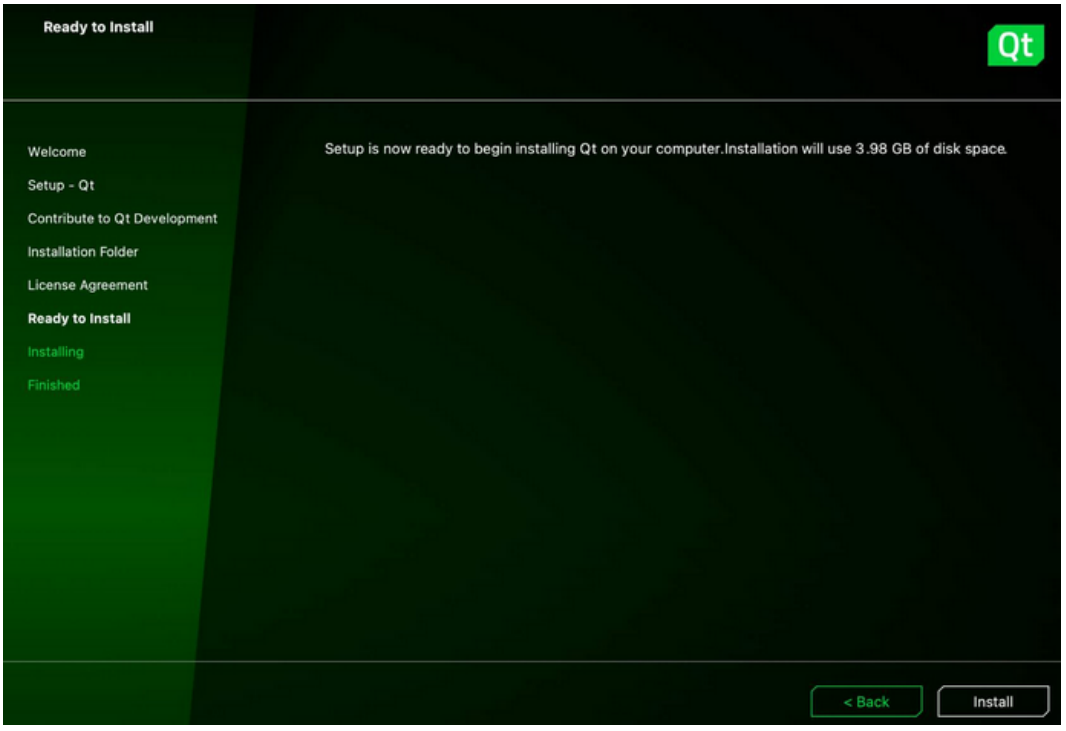

- Choose an install location. Then click on “Qt 6.4 for desktop development”. Note that there is a known bug in Qt 6.4.1 that makes the audio stop working, so make sure to install a different version. We recommend Qt 6.4.0, but Qt 6.5.0 should also work.



- On the next screen, read and accept the license agreements.

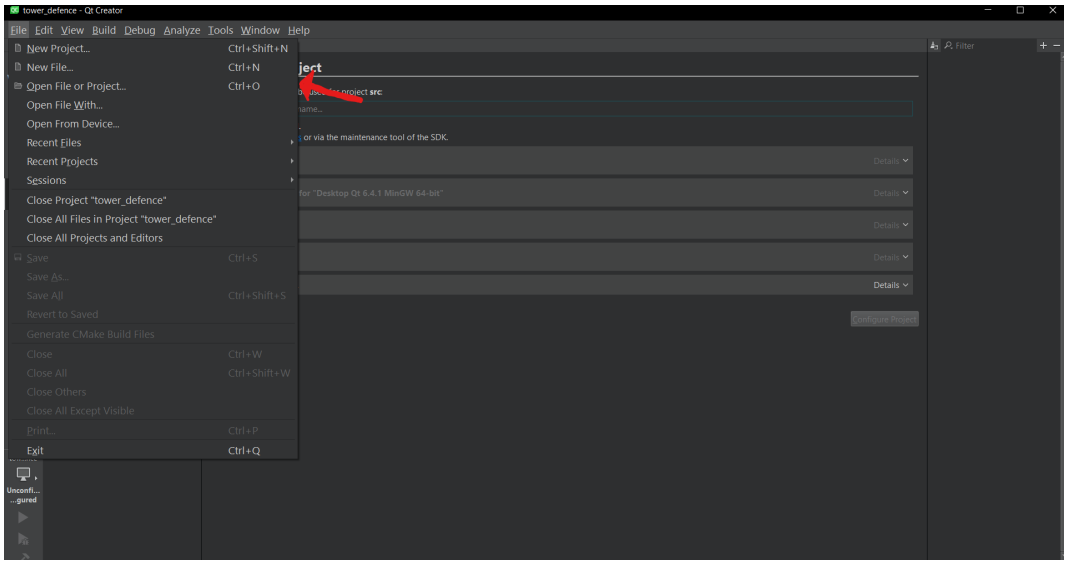



- Click install to start the installation

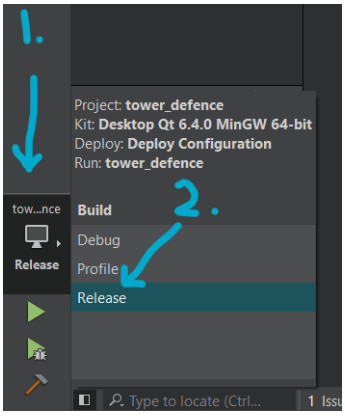


## 4.2. Build the program using Qt creator

- Get the repository from [https://version.aalto.fi/gitlab/cpp-homies/cpp-homies](https://version.aalto.fi/gitlab/cpp-homies/cpp-homies).
- Open the Qt creator application. On the toolbar, click on File>>Open File or Project, then navigate to the directory where the project repository was saved. Open the “src” folder, choose the “tower_defence.pro” file and click open.




- After the project is open, select the monitor icon and set the build mode to “Release”.




- After configuring the build mode to release, select Build>>Build project “tower_defence.pro” on the toolbar.
- After completing the steps above, the application should be built in the build directory.

# 5. User guide

## 5.1 Main menu


In the main menu, you are presented with five buttons and the title of the game, “tower_defence.cpp”.
The buttons do the following:

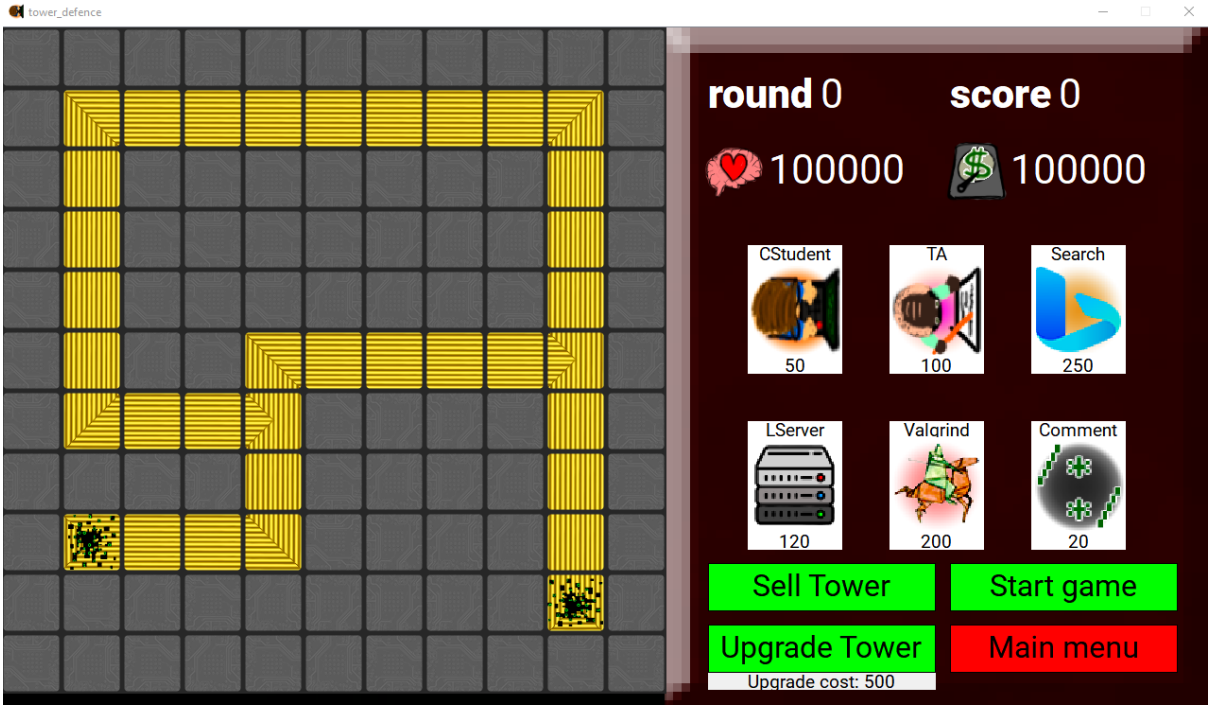
- Play buttons:
    - Sandbox
        - The yellow “Sandbox” play button will start the game in sandbox gamemode. In sandbox gamemode you will have practically an infinite amount of the currency and health points (hp).
    - Easy
        - The green “Easy” play button will start the game in easy gamemode. In easy gamemode you will start with 100 hp and 200 of the currency. Your income will start with a multiplier of 2 and decrease over the rounds.
    - Hard
        - The red “Hard” play button will start the game in hard gamemode. In hard gamemode you will start with only 1 hp and 100 of the currency. Your income will start with a multiplier of 1 and decrease over the rounds to 0.12. This is the only mode that can place you in the leaderboard.
    - Leaderboard
        - The green “Leaderboard” button will lead you to the leaderboard which contains the local leaderboard of top 10 scores from hard gamemode.

- Quit
  - The green “Quit” button will quit the game and close the program.

## 5.2 Leaderboard

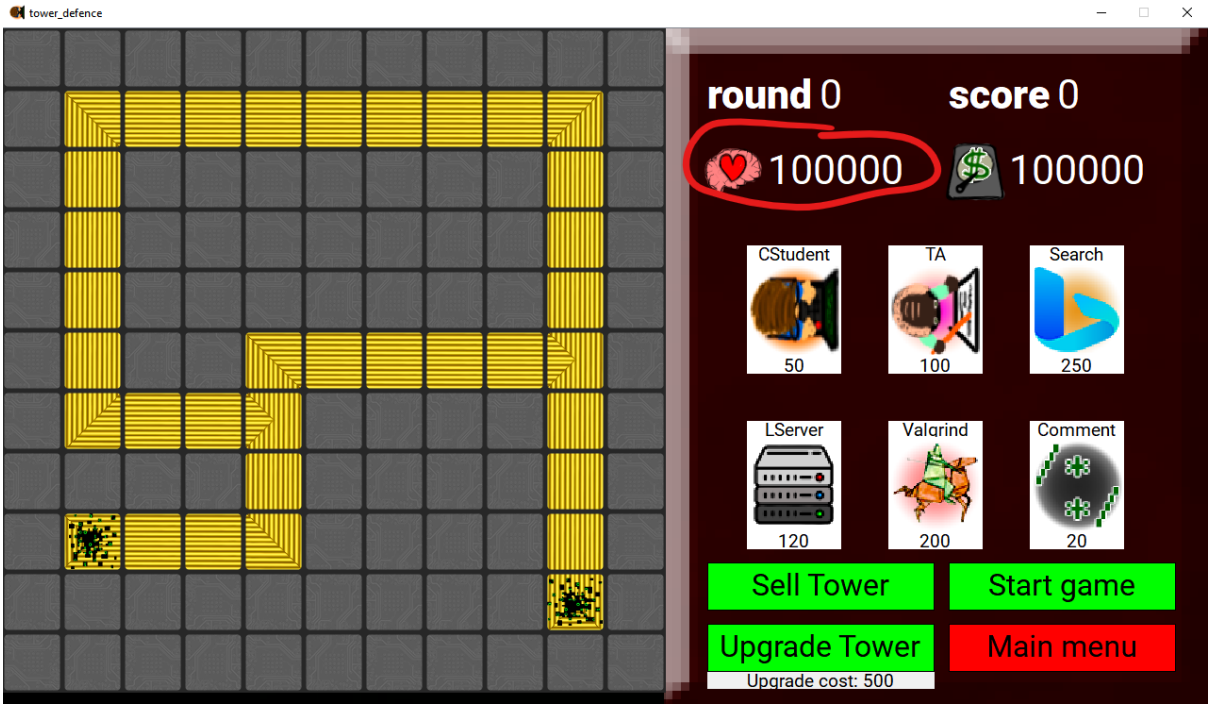

In the Leaderboard, you will find the top 10 scores from “hard” gamemode. You cannot get placed in the leaderboard from playing the “Sandbox” or “Easy” gamemodes. The displayed names only contain the first 30 characters from the name you input after the game has ended. You can also return to the main menu by clicking the green Main menu button.

## 5.3 How to play

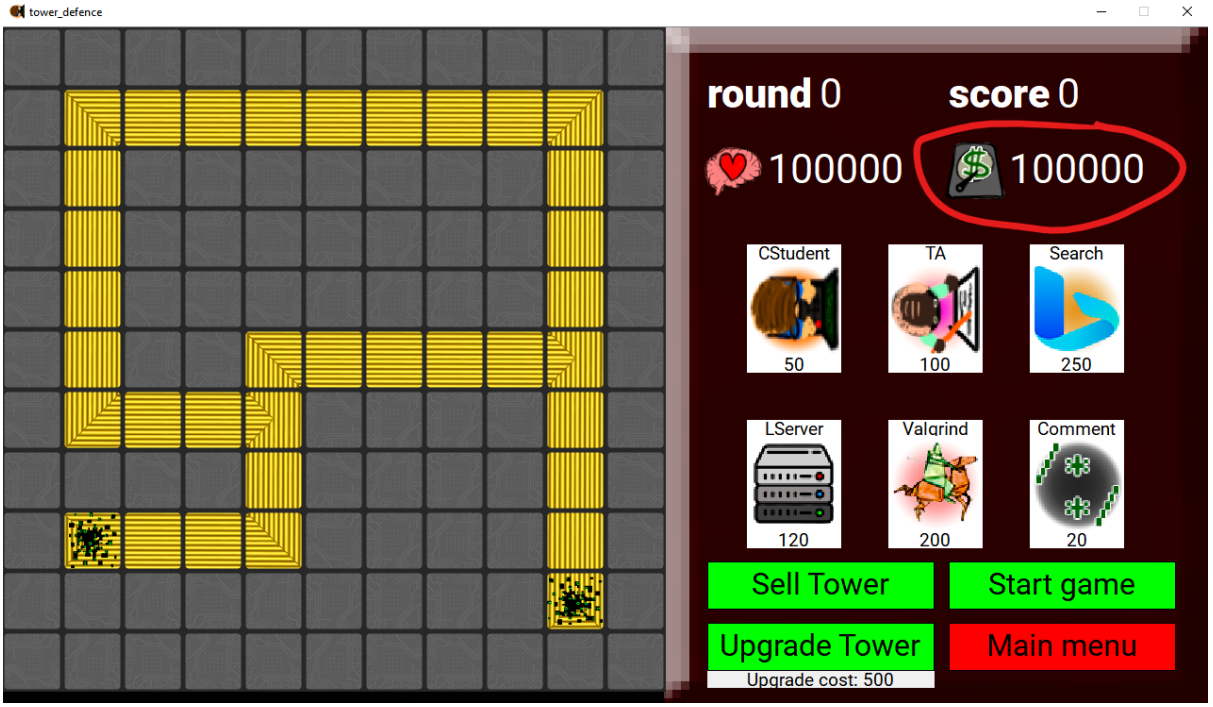



When you select any of the gamemodes, you will get to the game scene. Here, you have the map on the left, where the path that the enemies will take, when they attack, is shown with the golden bus. You can place comment towers on the path. The enemies will spawn from the glitched black hole on the left end of the path. You can place regular towers (all towers except the Comment tower) on the gray squares.

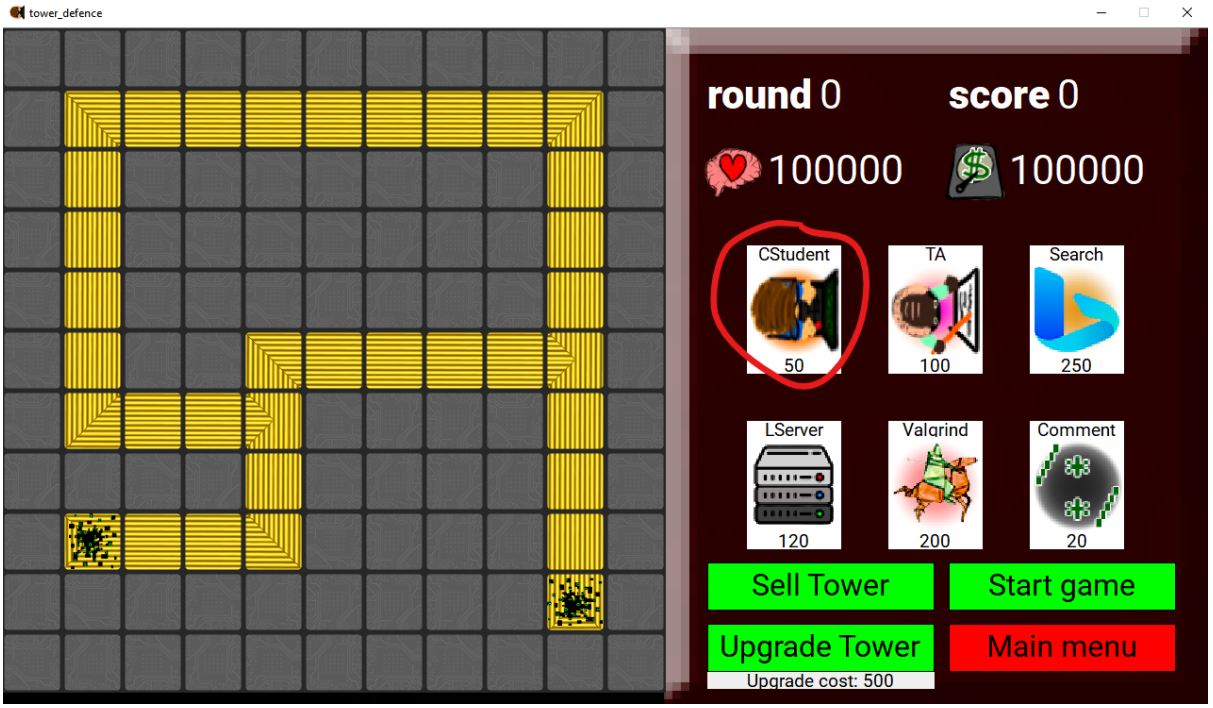
On the right, you find the game controls. Using the controls, you can build towers, sell towers, upgrade towers, start the game, and exit to the main menu.



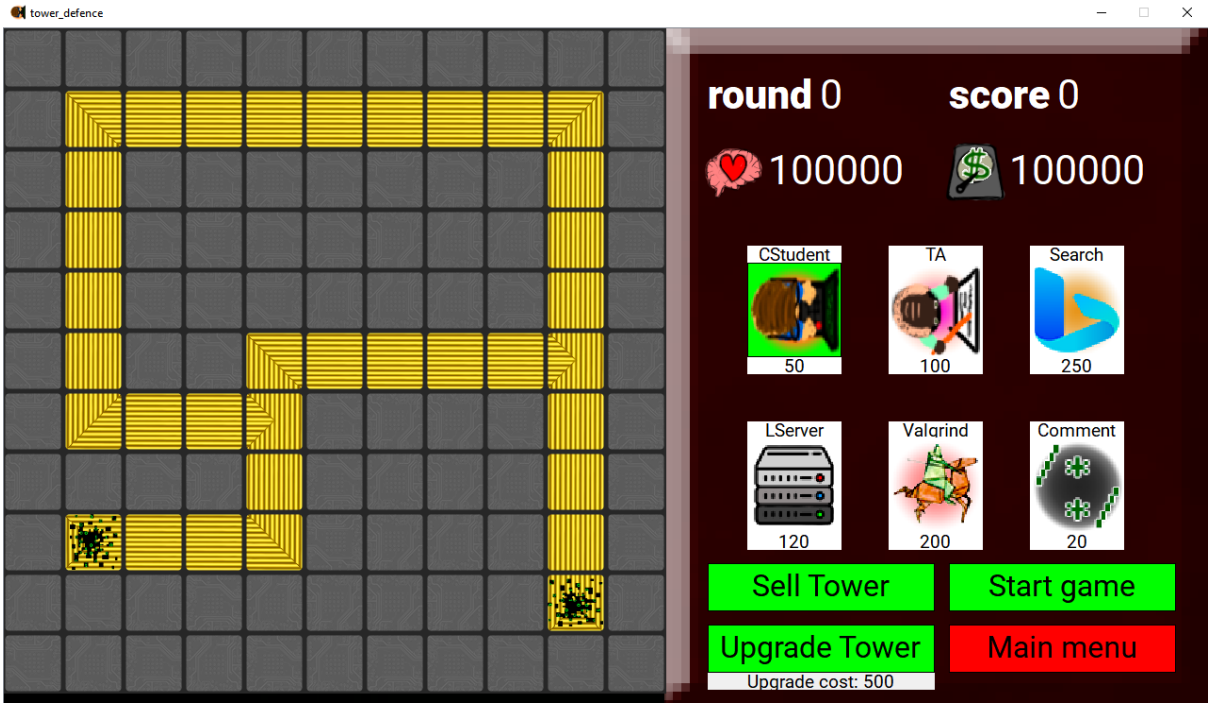

Here you can find the number of health points you have left.



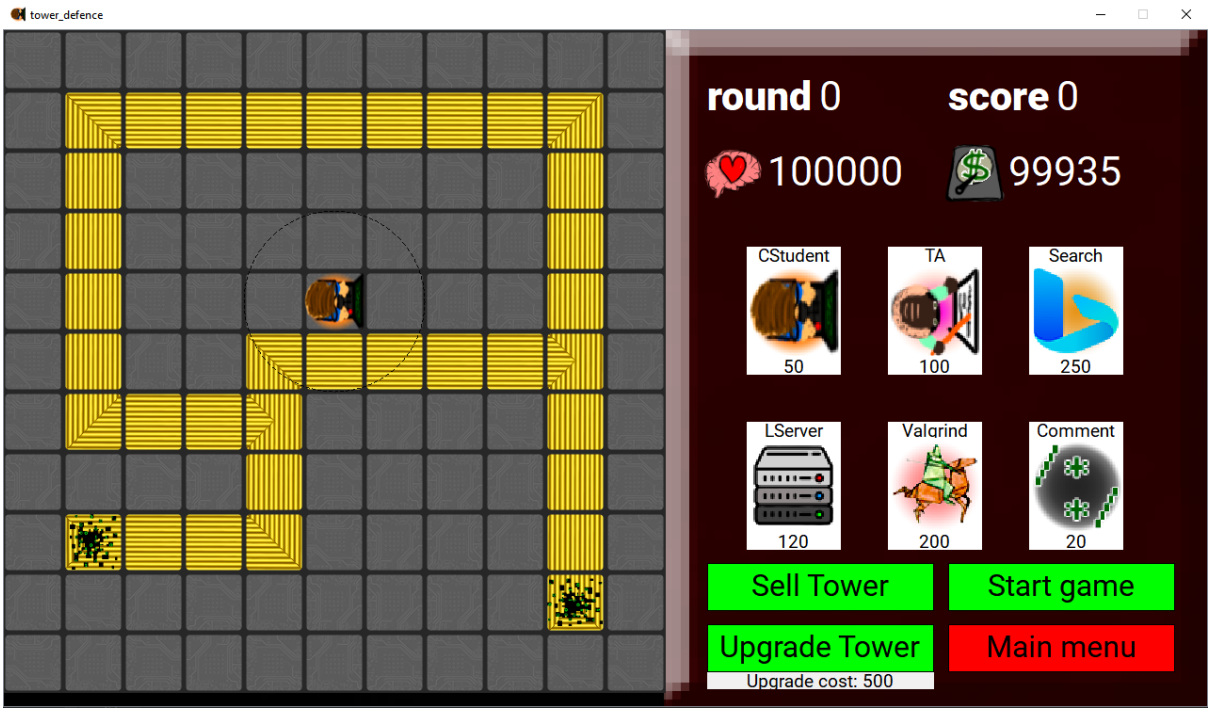

Here you can see the amount of currency you have.



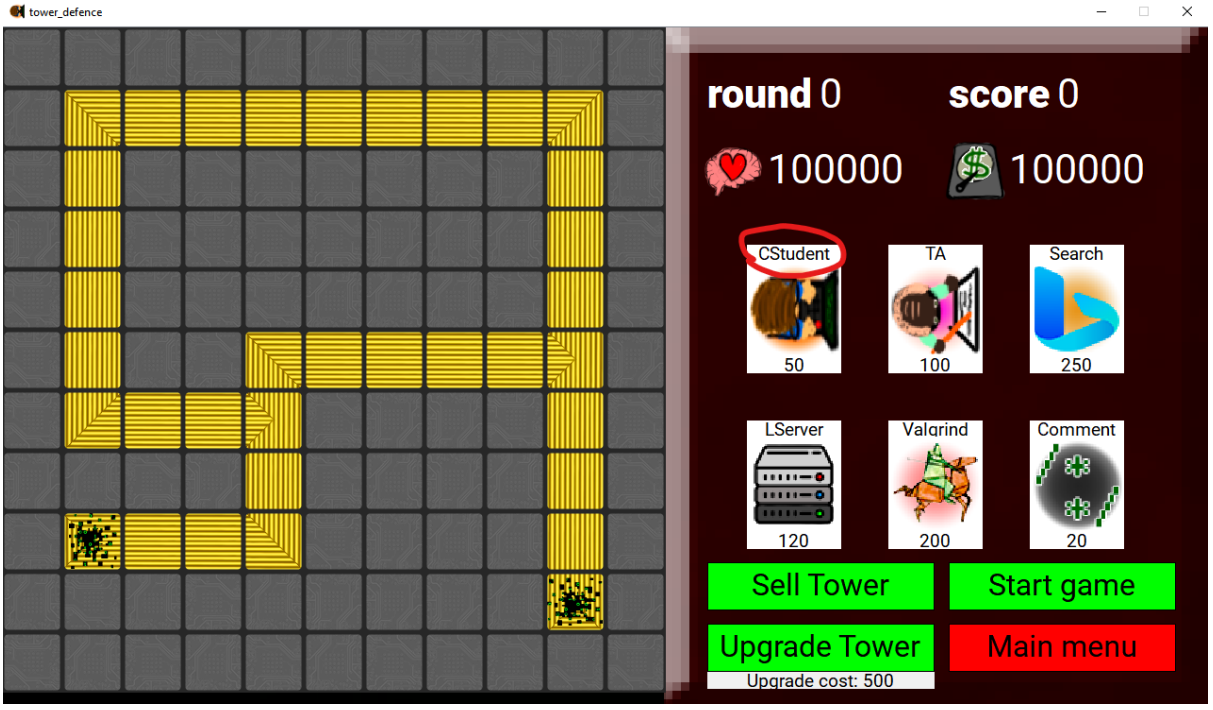

This is one of the tower building buttons. By clicking this, you will enter “building mode” with the CStudent tower type.



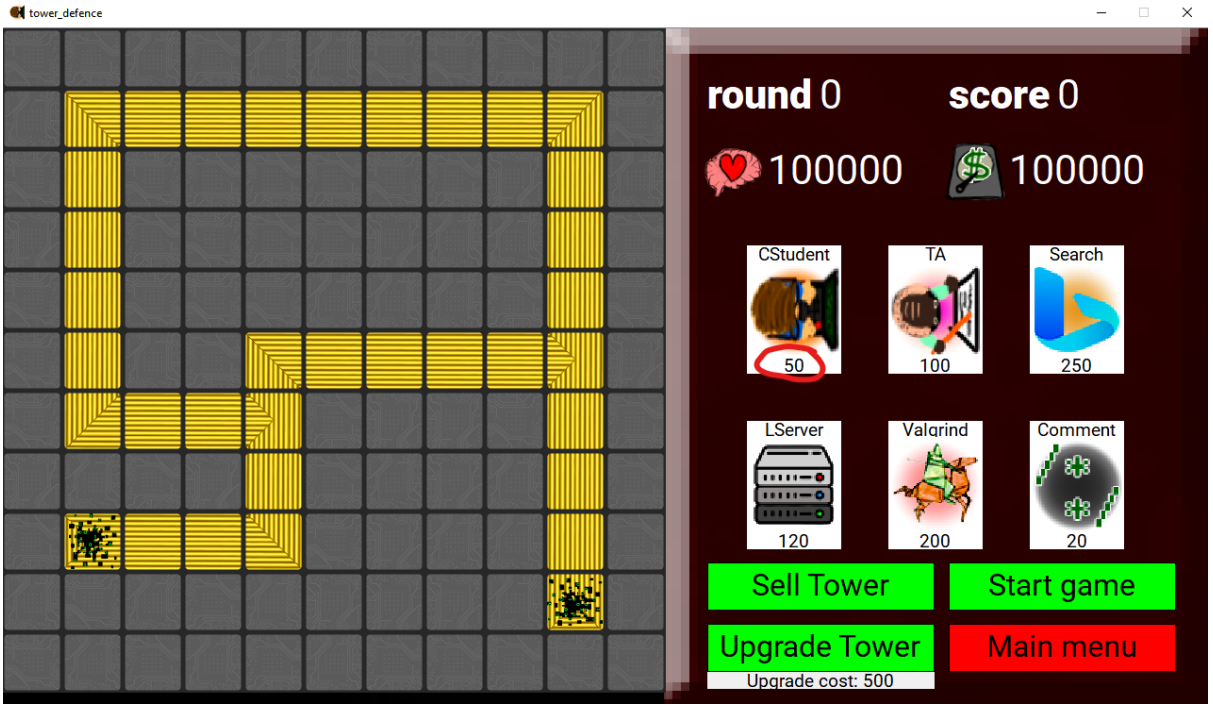

After you click a tower building button, the button’s color will change to green. Now, you can click on a suitable square on the map to place your tower.



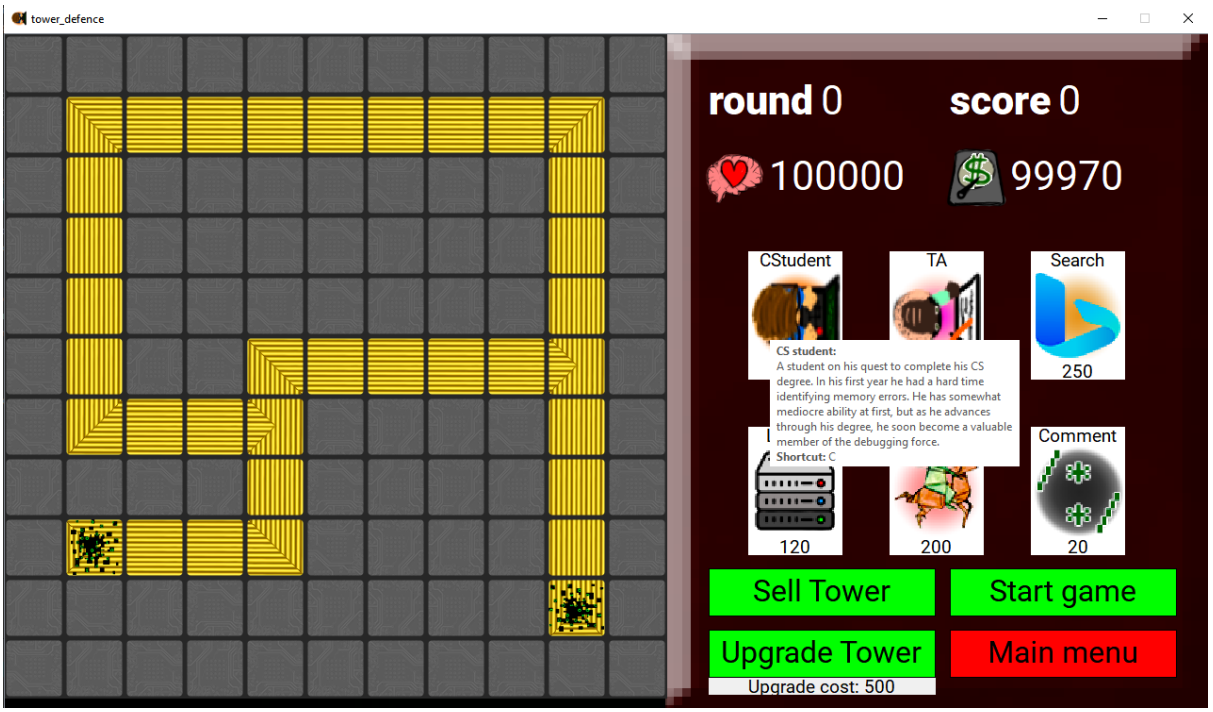

Here, you can see a level 1 CStudent built on the map. The circle around the tower indicates the range of the tower. This is the area in which the tower can target enemies.



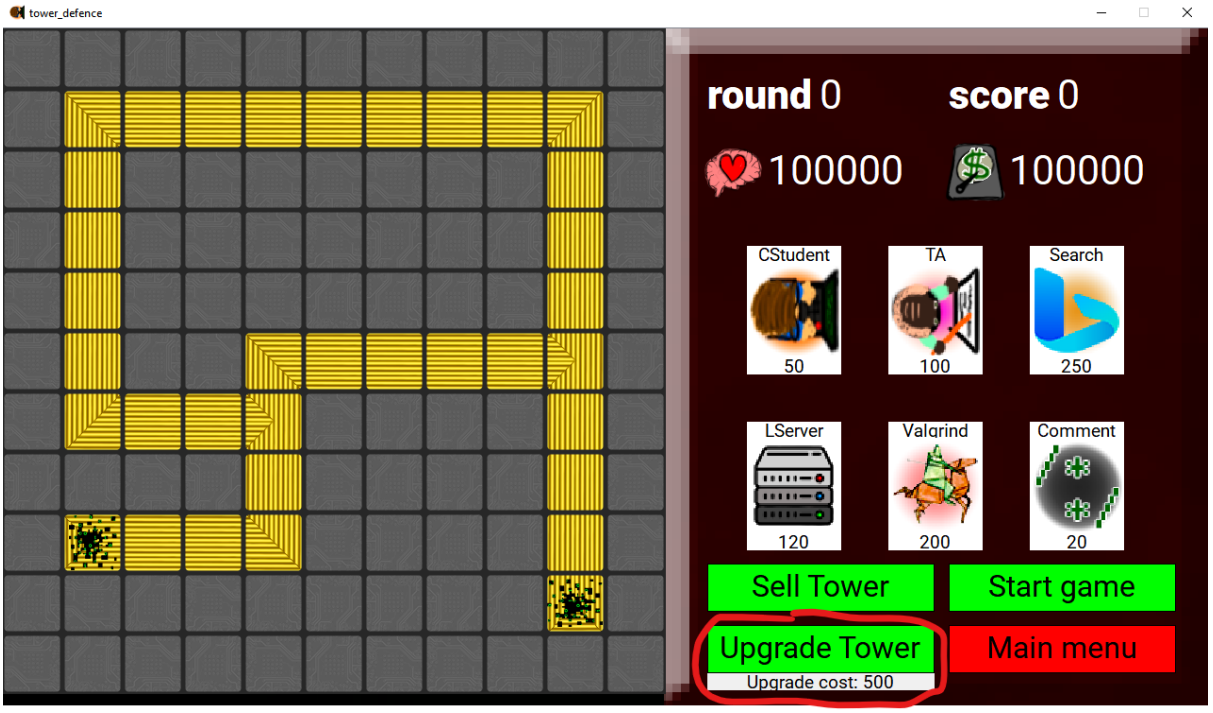

Above the picture of a tower is the (shortened) name of the tower.



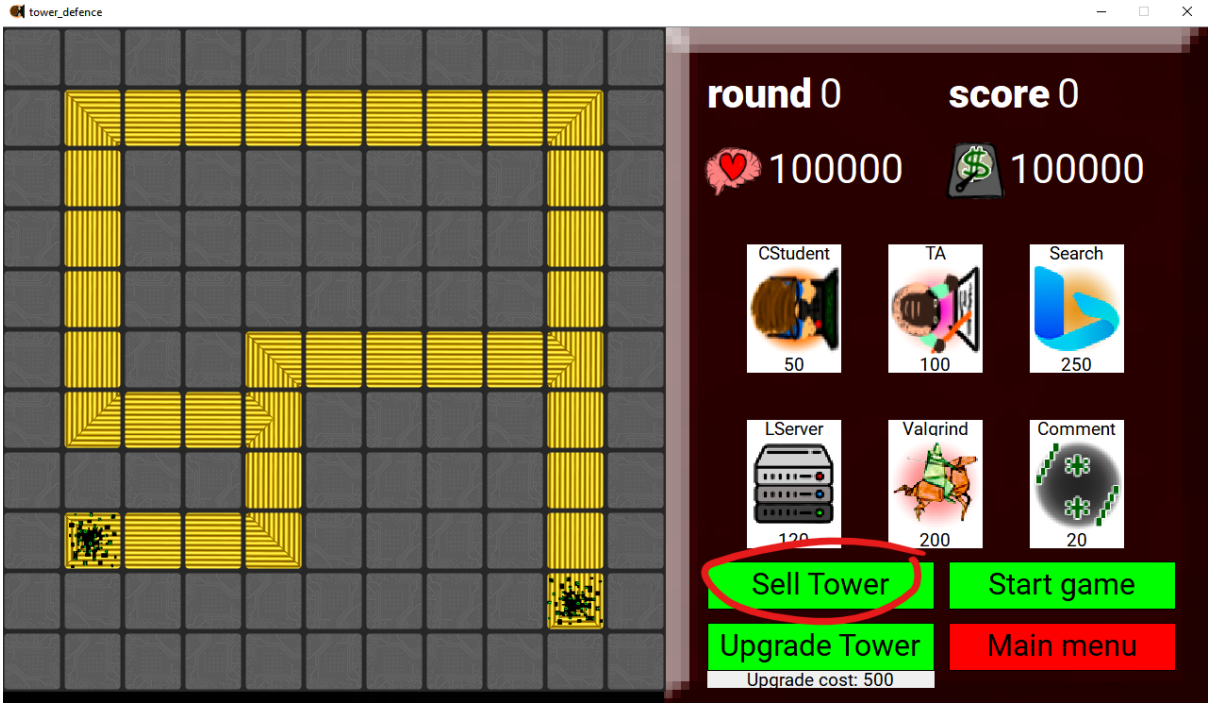

Below each tower building button is the price of the corresponding tower.



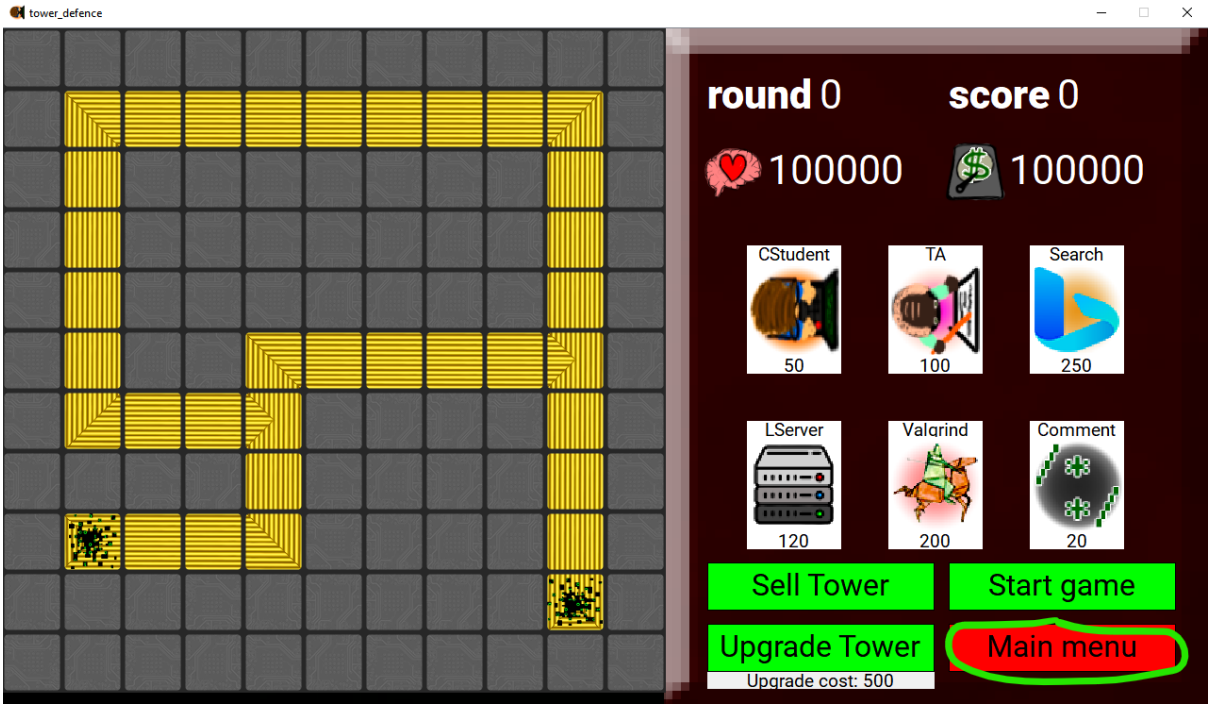

When you hover your mouse over any of the tower buttons, a tooltip will pop up, displaying the tower description and the shortcut key for building the tower.



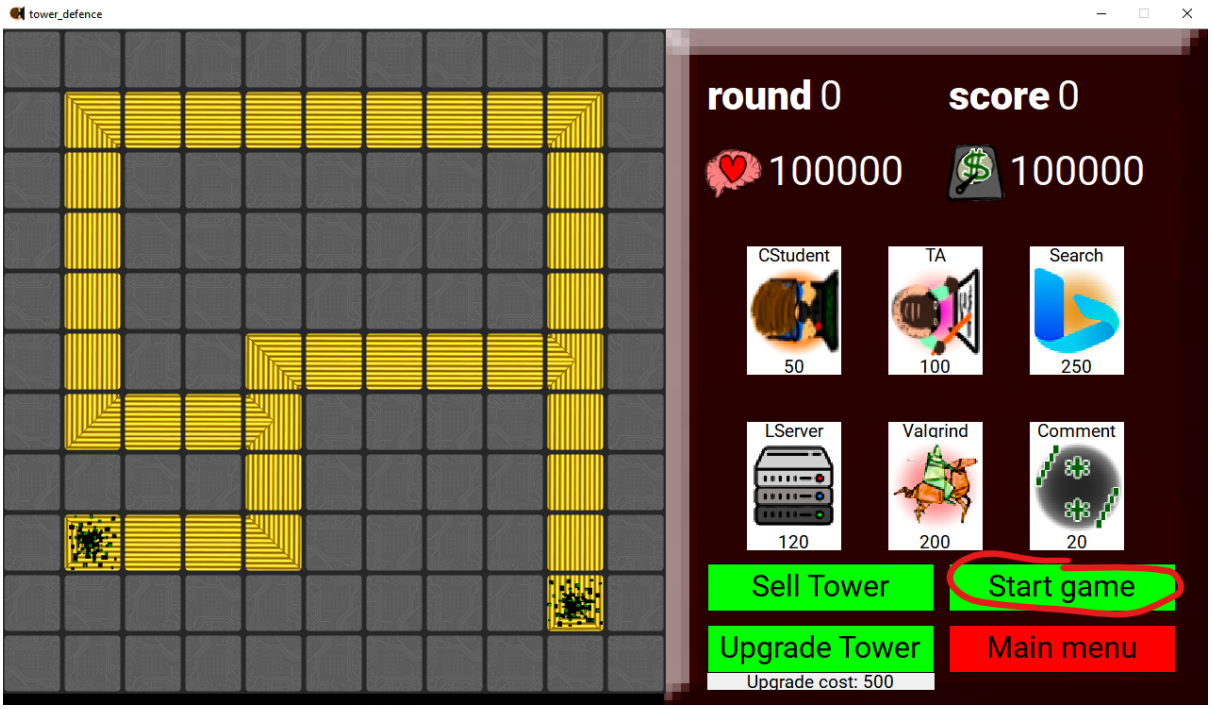

By clicking this button, you will enter the upgrade mode. In upgrade mode, you can click a tower you want to upgrade. The tower will get an upgrade if it is upgradable and you have sufficient currency. Upgrading any tower costs 500. You can also enter the upgrade mode by pressing the “U” key on your keyboard.



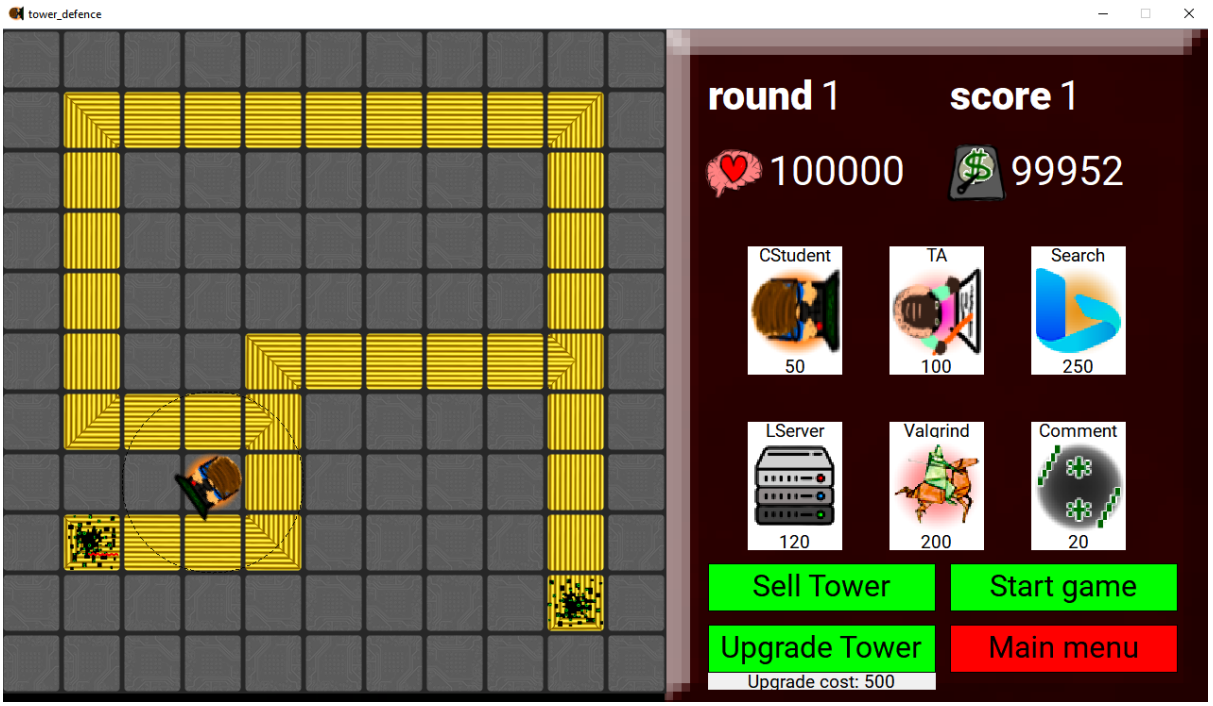

This button will allow you to enter selling mode. In selling mode, you can click on a tower on the board to sell it. You will get back 70% of the IOPS you have paid for it. You can also enter selling mode by pressing the “S” key on your keyboard.




This is the quit to main menu button. Clicking it will quit the current game and send you to the main menu. Your progress will be lost.



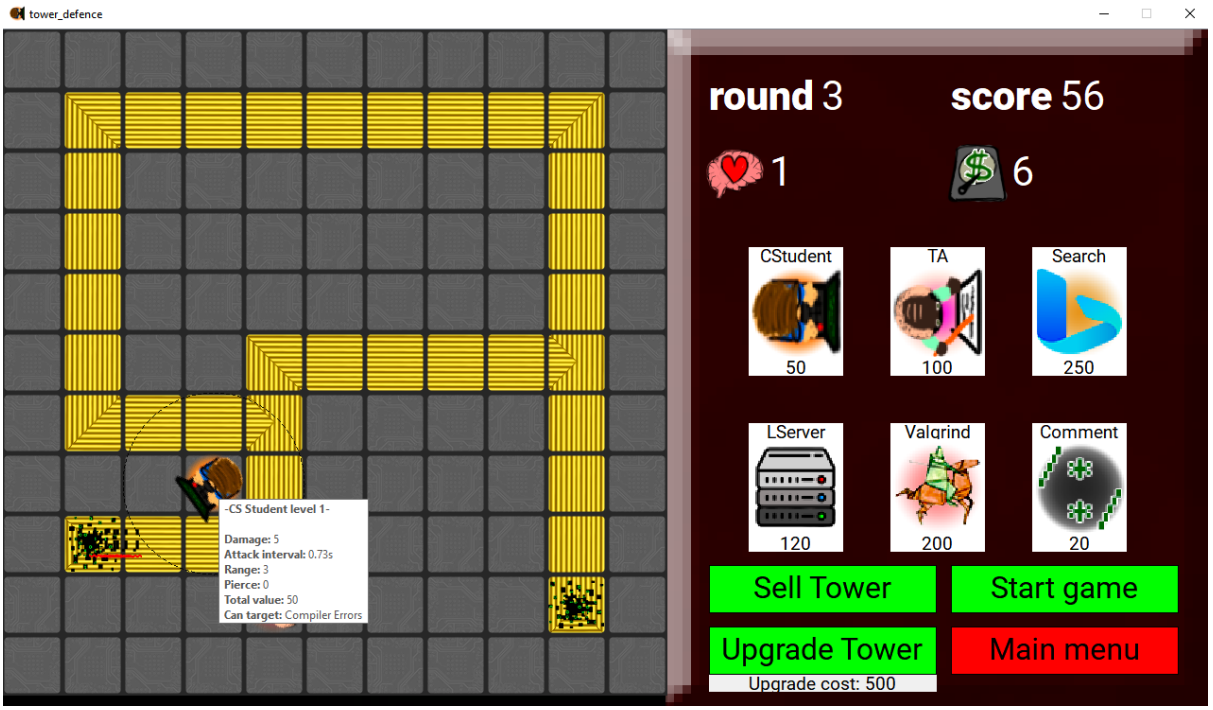

This is the “start game” button. When you are on round 0, you can start the game by clicking it. You can also start the game by pressing space on your keyboard.



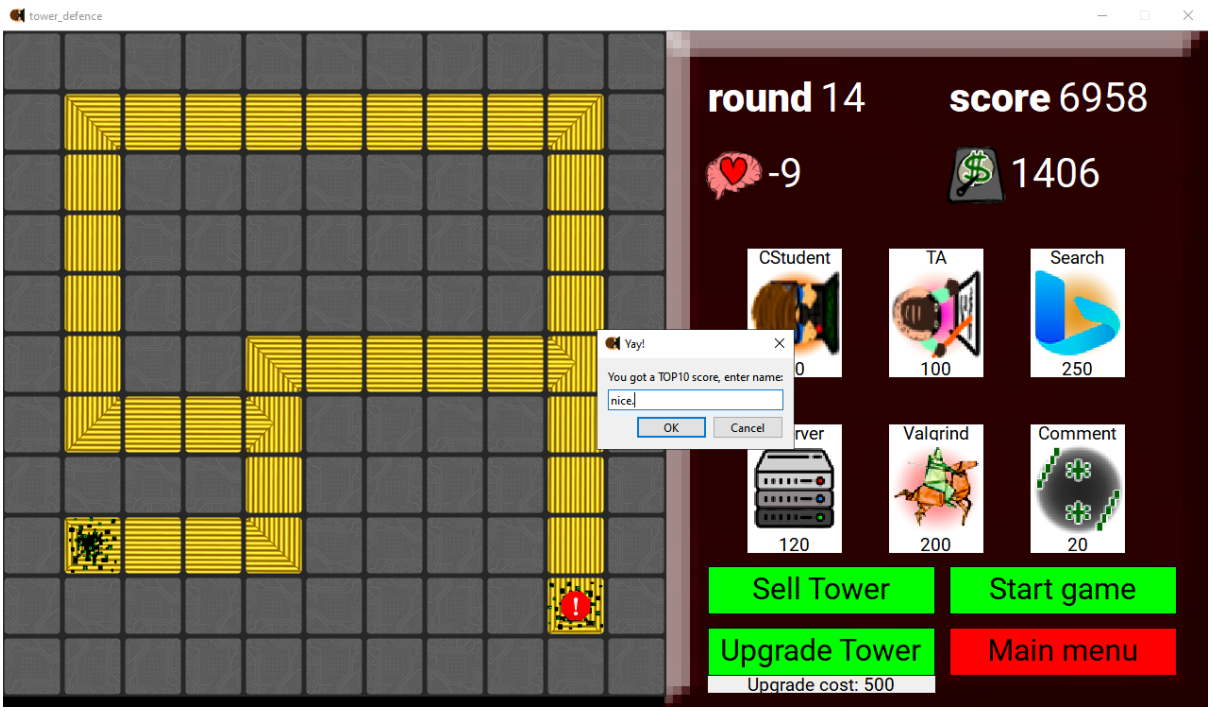

After you start the game enemies will start spawning from the beginning of the path. The enemies will vary in their speed, health, outlook and some towers cannot target some enemy types.

There are 3 types of enemies:

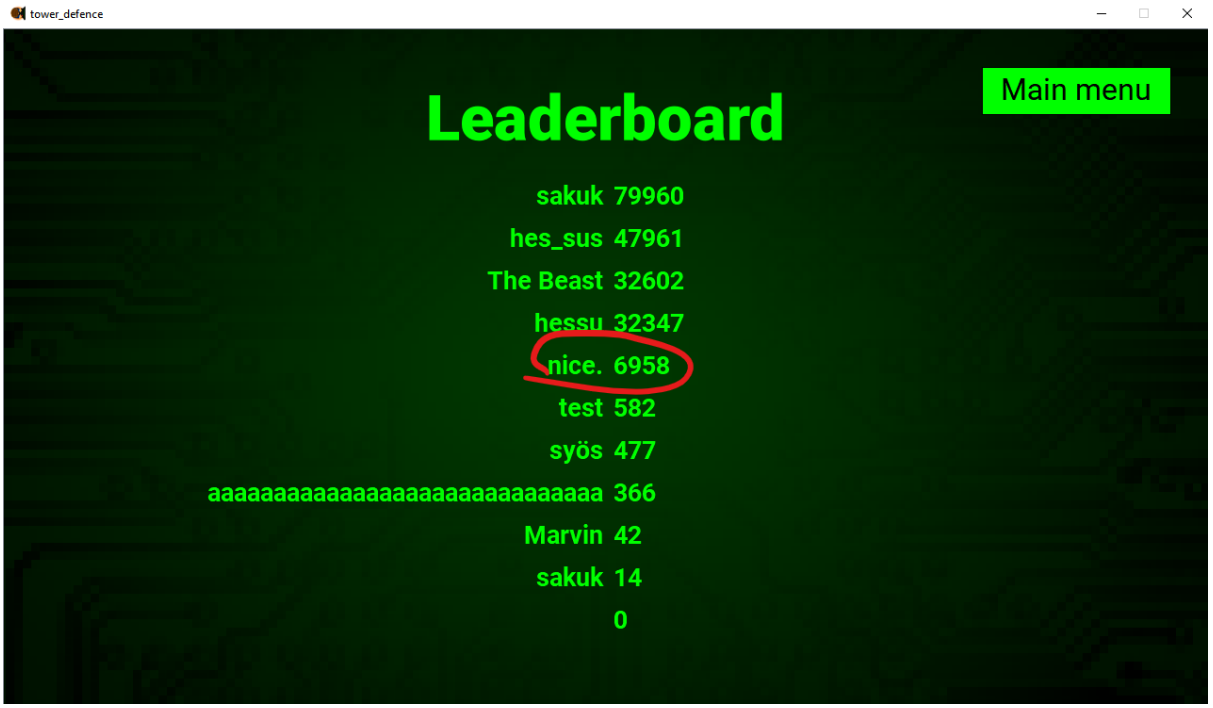
- Compiler errors
  - Compiler errors can be solved (destroyed) by any tower
  - Syntax error 
  - Exception 
- Memory errors
  - These errors are more advanced. They cannot be solved by all towers
  - Invalid read of size X 
  - Invalid read of size X 
  - X bytes are lost 
  - Mismatched delete / free 
- Runtime errors
  - This is the boss type. The boss has 5 stages. After each stage it spawns a minion.
  - Stack overflow 
  - Stack overflow minion 




Hovering over a tower on the map will show a tooltip that displays the stats of the tower.



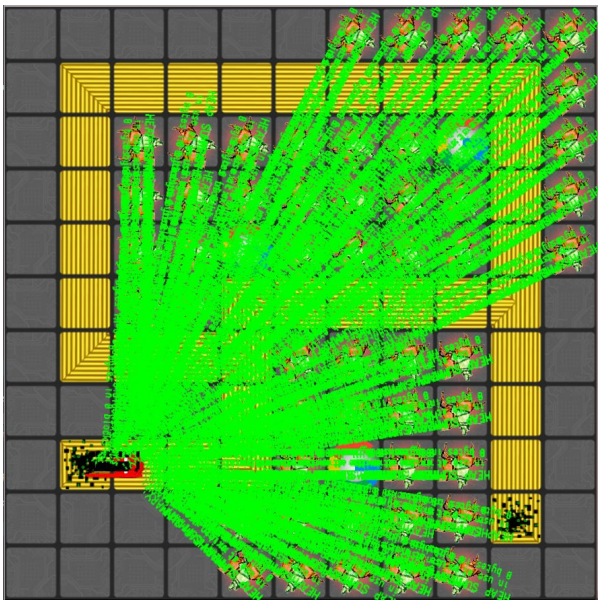

After the game ends, you will be prompted to give your name to be displayed in the leaderboard.

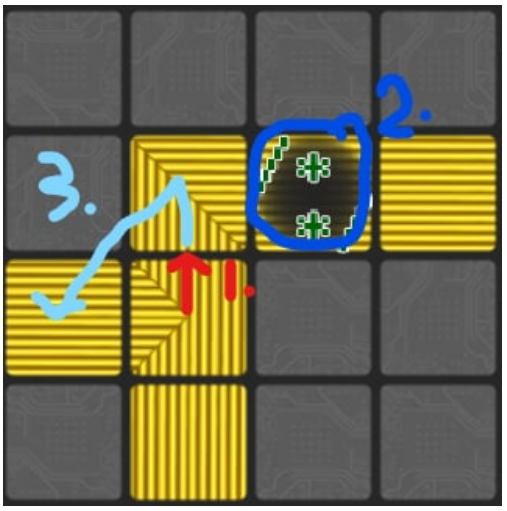



Now, your chosen name will be displayed in the leaderboard next to your score. Of course, if your score is below the 10th best score, your name and score will not be displayed here.

# 6. Testing

Testing was a crucial part of the development, especially because we had no prior experience of using QT. Testing was done manually. When the project advanced and got bigger and more complex, reproducing the bugs found got harder and harder. Thus, we started to use screen capture to investigate the steps done leading up to that bug. Furthermore, QT creator's built-in debugger was used extensively. Git, miro board, and telegram chat were used to track the issues and the progress on solve them.
A few problems required cooperation since some parts of the project were more familiar to other team members. For these situations, we used Telegram or had extra meetings on our Discord channel.





# 7. Work log

## ● 01.11.2022

**Summary of works**

1. Teamwork (Harvey, Hung, Saku, Siim) 3h

First stage project plan brainstorming, initial concept and class hierarchy.
Invite link to Miro board:
https://miro.com/app/board/uXjVPIee4qA=/?share_link_id=19406790451

**Challenges**

1. Level editor to create levels and to save them in a file?
2. Multiple paths of the enemies, so branched paths and enemies choose one with some intelligence?
3. Tower placement can be altered during an enemy wave ?

**Project status**
Project plan ideation is almost done. Basic theme of the game is done. Class hierarchy is done. Scope of work is done. High level structure is done. Use of external libraries is done.

## ● 01.11.2022 - 08.11.2022

**Summary of works**

- Teamwork (Siim, Saku, Harvey, Hung) - 3h
    - UML diagram
        - Can be found on Miro board (Linked in meeting notes of meeting 01.11.2022 16:00)
    - Scheduling of project
    - Decide on libraries to use
    - Qt vs SFML

- Individual work (Siim, Saku, Harvey, Hung) - 15h

    - Get familiar with Git
    - Get familiar with Qt

**Challenges**

1. Needing to learn Qt is quite a challenge

**Project status**

Project plan is almost done. UML diagram is done. Next big step is to start working on the program in Qt creator. Before that each team member has to learn a little of Qt since none of us has prior knowledge of it.

## ● 08.11.2022 - 16.11.2022

**Summary of works**

1. Teamwork (Siim, Saku, Harvey, Hung) - 3h
   - Project plan
2. Individual work (Siim, Saku, Harvey, Hung) - 15h
   - Mess around with Qt to get comfortable with Qt
   - QGridLayout
3. Saku - 4h
   - Graphics for
     - CStudent
     - Projectile
     - SyntaxError

**Challenges**

1. Need to learn Qt is a quite a challenge

**Project status**

Everyone has a basic understanding of the Qt framework and has practiced on Qt Creator IDE. Have decided on the fundamental Qt classes to use for the map.

- **16.11.2022 - 23.11.2022**

**Summary of works**

1. Individual work (Siim, Saku, Harvey, Hung) - 10h
   - Mess around with Qt to get comfortable with Qt
2. Saku - 8h
   - Graphics for
     - Path
   - Graphics implementation for
     - Tower
   - Facing direction
   - Regular tile
   - Orientation randomization
3. Siim - 35h
   - Created Qt project
   - Base class for Game
   - Reading Qt documentation

   QGraphicsGridLayout
   - Base class for Square
   - Base class for Projectile
4. Harvey - 8h
   - Added enums for Square (Path, Tile, Blank)
   - Added map value member to game
5. Hung - 15h
   - Initial tower implementation
   - Tower attack area

**Challenges**

1. Merging branches is challenging

**Project status**

Development is ongoing. A basic implementation with one type of enemy, tower and path is scheduled to work by the end of the week. This leaves a full week to implement a working demo of the final game. This leaves us a little bit behind the original schedule but still on track for a fully working game.

- ## 23.11.2022 - 28.11.2022

**Summary of works**

1. Saku - 16h
   - Graphics for
     - Rest of the towers
     - Rest of enemies
     - Game overlay
     - Rest of CStudent upgrades' projectiles
   - Graphics implementation for
     - Main menu
     - Basic game overlay
   - Navigation between scenes (Game, Main menu, Leaderboard)

2. Siim - 47h
   - Completed all enemy classes
   - Implemented game mechanics to know when game/wave is won or lost
   - Implemented scene switching functionality
3. Harvey - 15h
   - Pathfinding implementation
   - Very basic path implementation, not fully compatible yet
4. Hung - 32h
   - Full base tower implementation
   - Tower upgrading feature
   - Partial enemy type targeting logic
   - Some other types of tower
   - Buttons for building towers in the game UI

**Challenges**

1. Typecasting between Qt classes

**Project status**

Most basic functionalities of the game are working. The most difficult challenges have been solved and the mechanics seem to work as intended at the moment. For the next week, there will be some bug fixing to do and finalize the rest of the planned features before releasing the demo.

- ## 28.11.2022 - 07.12.2022

**Summary of works**

1. Saku - 33h
   - Game overlay arrangement
   - Graphics for
     - Rest of the projectiles
     - Start/endpoint path
   - Graphics implementation for
     - Tower graphics over the map tiles
     - Leaderboard
     - Projectiles
   - Level/game design
   - Hotfixes
   - Ghost towers and enemies (Issue#6)
2. Siim - 55h
   - Hotfixes
   - Enemies wondering off the path,
   - Projectile fired in the wrong direction,
   - Waves creation bug (twice),
   - Enemies get stuck on the path,
   - Game crashed when boss enemy is killed in the final level
   - Local leaderboard functionality
   - Wave creating from local file
3. Harvey - 10h
   - Added comment tower
   - Changed pathfinding implementation for comments
4. Hung - 17h
   - Other tower types
   - Enemy targeting logic
   - Tower selling
   - Connect tower buying, selling, and upgrading to game money system
   - Fixed some coordinate bugs in Tower functions

**Challenges**

1. Many bugs in big project.

**Project status**

The project is very close to its final form. Needs some minor bug fixes and non-essential additional features, but overall the game is playable. The deadline seems to be reachable at the moment to release a fully working implementation that fulfills the objectives of our plan.

- ## 07.12.2022 - 11.12.2022

**Summary of works**

1. Siim - 7h
   - Hotfix for some enemies getting past a wall
   - Writing additional checks for user errors (wave creation file)
   - Multiple bug fixes
   - Learning how to generate documentation with Doxygen
2. Saku - 39h
   - Level/game design
   - Modify language server shooting mechanics
   - Creating executable file
3. Harvey - 42h
   - Map loading/saving
   - Documentation
   - Comment breaking
   - SFX
4. Hung - 9h
   - Add tooltip for the Buttons
   - Add tower descriptions
   - Remove stacking mechanic of TA tower
   - Fix segmentation fault for support tower
5. Everyone - 20h
   - Adding comments to code
   - Testing

**Challenges**

1. Too much documentation to do. Roughly 200 pages or more.

### Cumulative Hours:

- Harvey - 141 hours
- Siim - 210 hours
- Hung - 142 hours
- Saku - 148 hours

## 8. Division of work

1st stage game setup:

- Game class(the main setup) (Siim)
- MainView class(Siim)
- QGridLayout class (Siim, Saku)
- Basic tower  (Hung)
- Projectile (Siim)
- Basic enemy (Siim)
- Basic path (Harvey)
- Initial graphics (Saku)
- Meeting notes (Saku)

2nd stage :

- Tower implementation (Hung)
    - Tower upgrades
    - Selling towers
    - Different tower types
    - Tower controls: build, upgrade, sell buttons
- Enemies implementation (Siim)
- Local leaderboard (Siim, Saku)
- Game mechanics(Siim, Saku)
- Path implementation (Harvey)
    - Comment tower
    - Pathfinding [BFS]
- Map loading (Harvey)
- Sound effects (Harvey)
- GUI (Saku)
    - Main menu
    - In-game interface
    - Upgrade interface
- Documentation (Doxygen generated, everyone)
- Meeting notes (Saku)