ELEC-7151 - Object oriented programming with C++

Project Plan: Tower Defense

Antti Haavikko, 714066, antti.haavikko@aalto.fi
Perttu Jalovaara, 714396, perttu.jalovaara@aalto.fi
Anna Huttunen, 711881, anna.a.huttunen@aalto.fi
Atte Tommmiska, 666716, atte.tommiska@aalto.fi

Turned in: November 5, 2021

1 General project description

Our group's project is a tower defense game. The theme of our game is humoristic and close to our lives. In the game there are geese defending their environment and younglings from teekkaris, who are wandering through different popular locations in Otaniemi. There are different kinds of geese, that have different defense abilities against teekkaris. Geese may be able to slow teekkaris down, hurt or even kill them. Geese also either have a certain radius of effect or they may choose their victim. Teekkaris may be able to walk through the field with different velocities and using different routes. Teekkaris are not able to hurt the geese, but if too many teekkaris make it through the geese's defenses, they will cause damage to baby geese. This leads to the geese losing the game.

There will be five levels that represent different environments like Alvari Square, Otabeach and Ossi's Puddle. Our tentative plans for the three aforementioned levels are visualized in Figure 1. The difficulty of the game increases with each level. In addition to the five levels, it will be possible for the player to upload their own level to play. A level editor, where the player may design their own level, will be implemented if we have the time.

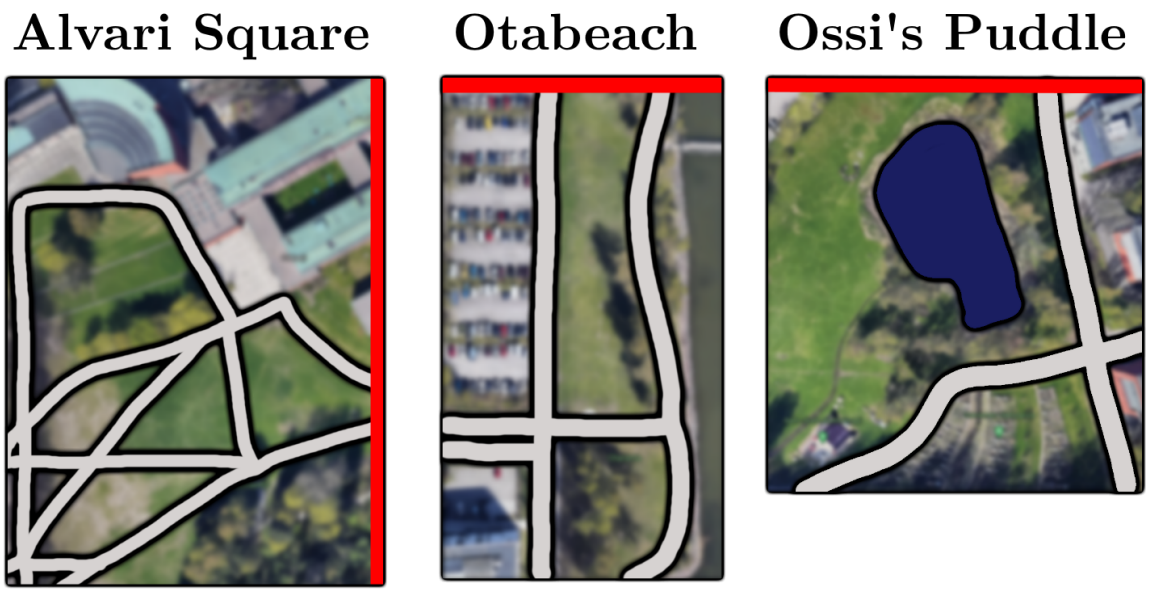

Figure 1: Illustrations of three tentative levels. The enemy paths are drawn in gray and the area to protect is highlighted with red.

2 Scope of the work

What features and functionalities will be implemented?

When the game is started, the player may choose a level to play. Each level has a unique map. The player has some money in the beginning to add towers (geese). Waves of enemies (teekkaris) run through the map following single non-branched paths on their journey by default, but some enemies choose their path with some intelligence as they calculate the most optimal path and some enemies may have paths that branch depending on the level. Towers can be moved in between of enemy waves and also when enemies are on the map. The towers can attack enemies inside their range in different ways. The game is lost when enough enemies reach the end of the path. The player gains money by destroying enemies. Different enemies are worth different amounts of money. Money is used to build more towers.

There are five different types of geese. The basic type is Röi Goose, which causes a low amount of damage to enemies and shoots at an average frequency at enemies inside its range. The second type is Pooper Goose which slows enemies within its range. The third type is Sniper Goose, which chooses it's victim based on which enemy is most powerful. Sniper Goose causes significant damage. The fourth type is Machine Gun Goose, which has a high rate of fire, shoots enemies within its range and causes average damage. The fifth type is Mama Goose, which can eat multiple enemies at the same time. Eating causes the enemies to be destroyed at once. Each type of goose can have different levels. The higher the level, the bigger its range, rate of fire or damage. Different types of towers cost different amounts of money. Figure 2 showcases concept art for Sniper Goose and Mama Goose.

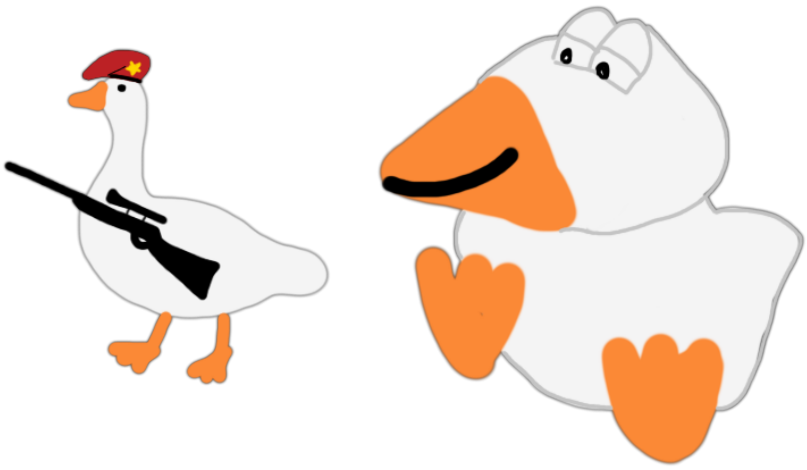

Figure 2: Illustrations of Sniper Goose (left) and Mama Goose (right).

There are also different types of enemies. The first type is Koneteekkari which can be killed by one hit. Other types are harder to kill. The second type is Physicist, which chooses the most optimal path to cross the game field, but is easy to kill. The placing of towers changes the optimal path, so this enemy has a dynamic path. The third type is Kylteri, which is a bit harder to kill, but the reward for killing is significantly higher than for the earlier types. The fourth type is TelaVeera, which takes many hits to kill and once it has been destroyed it splits into many Koneteekkaris. The fifth type is Dokaani, which is really difficult to kill. The effect of a tower on an enemy depends on their types. The Pooper Goose for example is really effective at slowing most enemies, but it can not slow down TelaVeera.

The game is controlled by mouse. The user can drag and drop towers on the map and upgrade the towers by clicking. The number of enemy waves, enemies that have crossed the field and the amount of money are displayed on the user interface at all times. Once the game ends, either on a victory or a failure, the player is shown their points and asked for a nickname to be displayed on the high scores board that is saved locally per each map. The game contains sound effects that represent towers being built, enemies being destroyed, winning and losing.

How is the program used? How does it work?

The graphic user interface (GUI) responds to the commands given with a cursor. The program contains some "loading screens" just to add a little humour and information for the player. The player chooses a level from the default levels 1 to 5 or, alternatively, specifies the file path of a custom level. Custom levels are saved as ordinary text files through the level editor, if implemented. Once the game begins, towers can be dragged and dropped from a side bar on the right if the player has enough money for them. The towers automatically attack the enemies in their range. The money bar grows as enemies are destroyed and shrinks as towers are bought or upgraded. The health bar starts from maximum and decreases the more enemies have crossed the entire map. If all health points are lost, i.e. enough teekkaris have managed to sneak through, the game is over. If the player wins the level, i.e. survives all enemy waves, they may enter their name to be added on the score board that displays high scores. After that another level can be chosen. The player may exit the game via the exit button.

3 The high-level structure of the software

The diagram in Figure 3 introduces the main modules and main classes according to the group's current understanding. We acknowledge that the diagram is not the base of a functioning and complete program, but it is the beginning of starting to piece together the idea of the entire program.

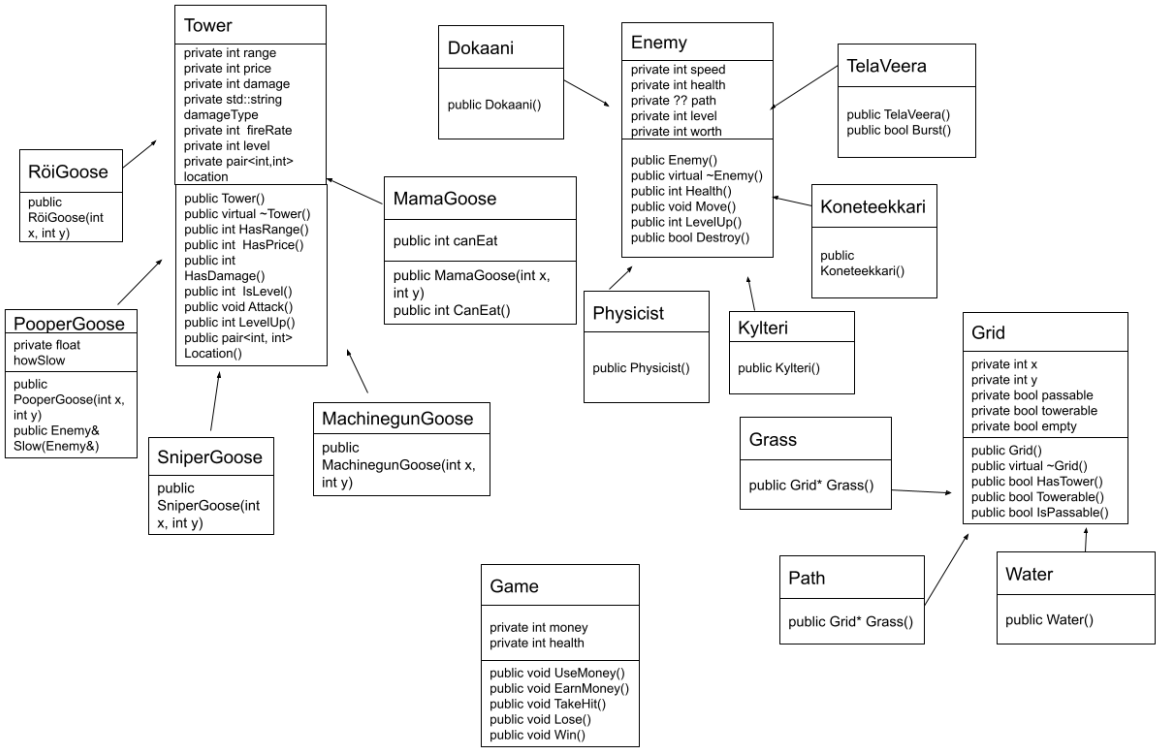

Figure 3: The main classes of our game. The arrows represent an "is a"-inheritance relationship, e.g., SniperGoose is a Tower.

The abstract base classes are Tower, Enemy and Grid. In addition to the base classes, there are multiple subclasses that inherit properties from the base classes. In the diagram the name of the class is presented in the first box and beneath are the attributes and functions of that class. The arrows show the inheritance relation of the classes. The base classes are abstract meaning they do not have any instances. By default the attributes are private, since there is no need for anyone to access them directly.

4 External libraries (Qt)

We chose to use Qt as our external graphics library. It seems to suit our needs well and is relatively beginner friendly. Also, Qt has cross-platform support, including Android & iOS, in case we really want to challenge ourselves and write game good enough to run on other platforms. Based on our quick research, we came to the conclusion that for our purposes

$$\text{Qt} > \max(\text{SDL, SFML}).$$

None of us have used Qt previously but as we do not have experience of any other graphics libraries either, it seems like a good choice. We have started experimenting with it to create a single goose. So far our efforts have been futile but we remain hopeful.

5 Division of work and responsibilities

Every group member will naturally participate in writing code. Additional duties and responsibilities are listed below.

Antti's responsibilites:

- Project Overseer
- Writing the Makefiles (CMake)
- Updating the `README.md` (Markdown)

Perttu's responsibilites:

- Project Overseer
- Writing and updating the `Meeting-notes.md` (Markdown)

Anna's responsibilites:

- Project Plan (NOTE: submission deadline Fri 5.11.2021)
- Object diagram (layout of main classes and modules)
- Graphical design

Atte's responsibilites:

- Final source code documentation (Doxygen)
- Coding style
- The use of Git

6 Planned schedule and milestones

Fri 22.10.2021	Project topic/group questionnaire (deadline)
Tue 26.10.2021	Project group distribution and confirmation
Fri 5.11.2021	Project plan submission (commit) to git (deadline)
Fri 12.11.2021	First running prototype
Fri 12.12.2021	Project final commit to git (deadline)
Fri 17.12.2021	Project evaluation (deadline)