Project Documentation

15.12.2016

ELEC-A7150 - C++ Programming

Konstantin Ionin, 348555
Lauri Jääskeläinen, 220819
Pyry Viita-aho, 352648
Joni Väisänen, 82017R
Matti Yli-Ojanperä, 355153

Table of Content

Overview

The goal of the project was to create a functional tower defense game similar to the one shown in Figure 1. Our result is shown Figure 2. The game was designed to be easy to play, meaning that there is basically no instructions needed. The game was designed to be challenging, but still fun to play. The are endless possible ways to play, which makes the game enjoyable for a long time.

The basic concept is to have multiple maps, which have multiple levels. The levels have different enemy waves with increasing difficulty. The difficulty of the game comes from tower placing and money spendage, which isn't as easy as it might sound. The game stays challenging and the whole gaming experience is very pleasant due to simple user interface and game logic.

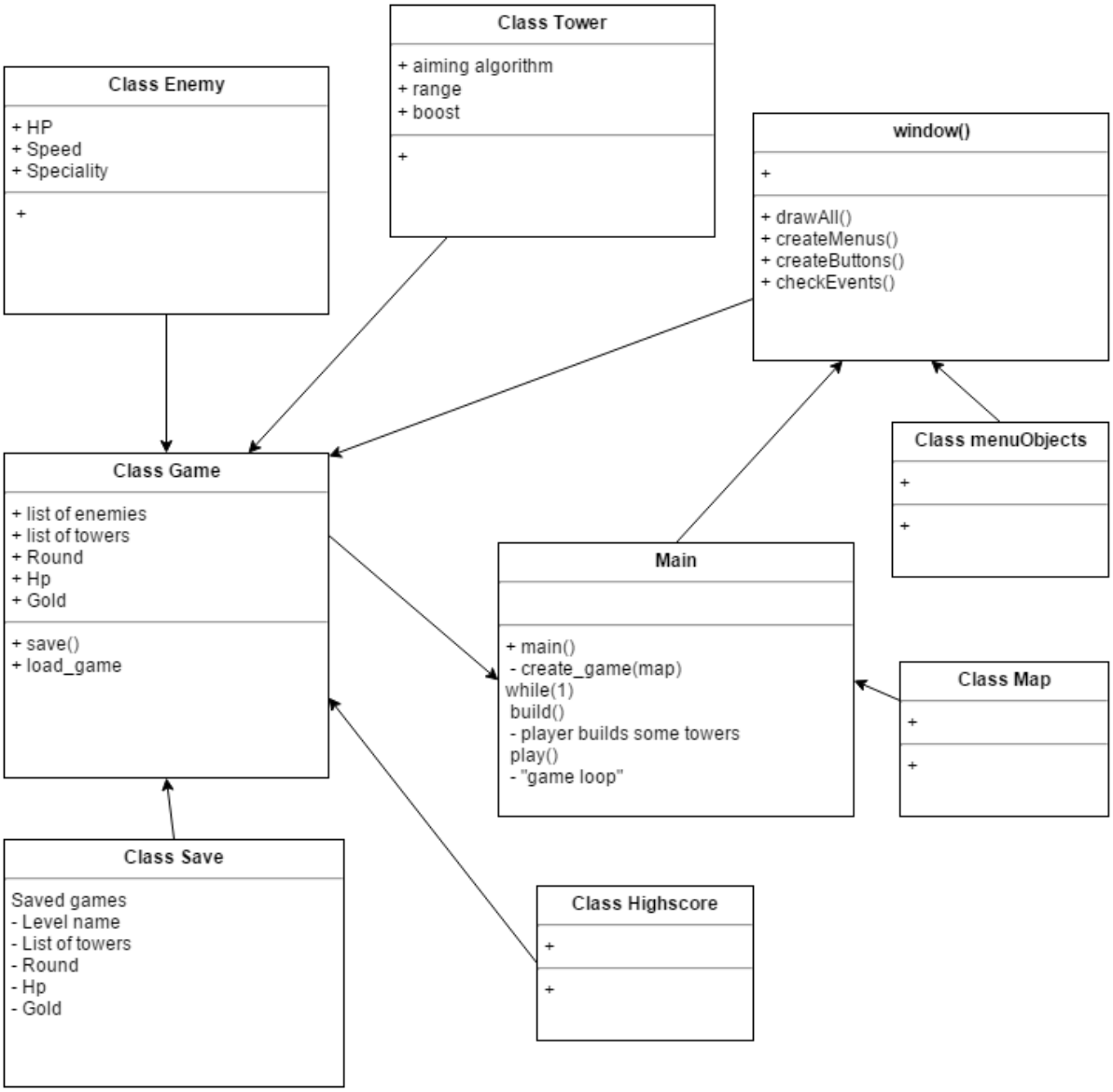
The game graphics were designed to be easy on the eyes. Everything from map design to graphical user interface is very simple, but elegant design.

Each level starts with a certain amount of money. The only target money is spent on are towers. The player's wealth increases by destroying enemies. Money can be spent only during build phase and only on new towers or upgrades to existing ones. There are only three basic tower types and only few enemies to learn. Starting to play the game is therefore very easy and compelling.

Figure 1 shows a similar game, which was used, in some extent, as a target for the game development.

Software Structure

Below is the basic software structure:



Description of Software Logic

Our Tower Defense game has a user interface where we have a route going from one end to the other. First the game starts with a building phase. Player is able to choose from right side some towers to build on the field. Player can click with mouse all 4 different kind of towers, which can be dragged. At the start player has 650 gold which can be seen from the left corner of the UI. Player puts his towers somewhere and clicks the play button to play. After the play button has been clicked enemies start coming on the field in waves. If the player sets up towers wisely enemies will be cleared and player earns a nice bonus after a

round. If player forgot to build towers or towers are in bad positions, enemies will find their way to the end of the path and player's hit points/base health will be decreased.

The Base has 100 hit points at the beginning of the game. The number of hit points left can be seen from the upper corner at the left side of the UI just under the gold amount. Red indicates hit points and yellow the gold amount.

The building phase will start again after enemies have been either killed or they have reached the end of the path. Now player can build more towers. After the player have gathered enough gold they will be able to build more, and possibly better, towers and/or upgrade existing ones.

There are five types of towers: basic tower, freeze tower, multifreeze tower, precision tower and blast tower. Basic tower deals one damage to one enemy at a time. Freeze tower slows enemies. Multifreeze can slow up to four enemies. Precision tower is much stronger than the basic tower and prioritizes stronger and more dangerous enemies. Blast tower damages all enemies in an area and is effective at destroying groups of enemies.

Game ends when the player runs out from life points. Game over text will come to the screen when this happens. Points are shown at the bottom left corner while playing but after the game is over you can see list of high scores by clicking button scores. If your score was good enough it updates to the high score list.

Instructions for Building and Using

We use two external libraries SFML 2.3.2 and sfml-tmxloader. In addition sfml-tmxloader has dependency for zlib. We assume that user has zlib and SFML installed. We have added the sfml-tmxloader to our repository. We use cmake to build our software.

How to Build the Game

The recommended workflow for building is following:

First step:	**cd "*path/to/project/root*"**
Second step:	**mkdir build, cd build**
Third step:	**cmake ..**
Fourth step:	**make**

After that you can run the game with ./Towerdefense

How to play the game

The game opens to a window shown in Figure 2. All the gameplay happens in it.

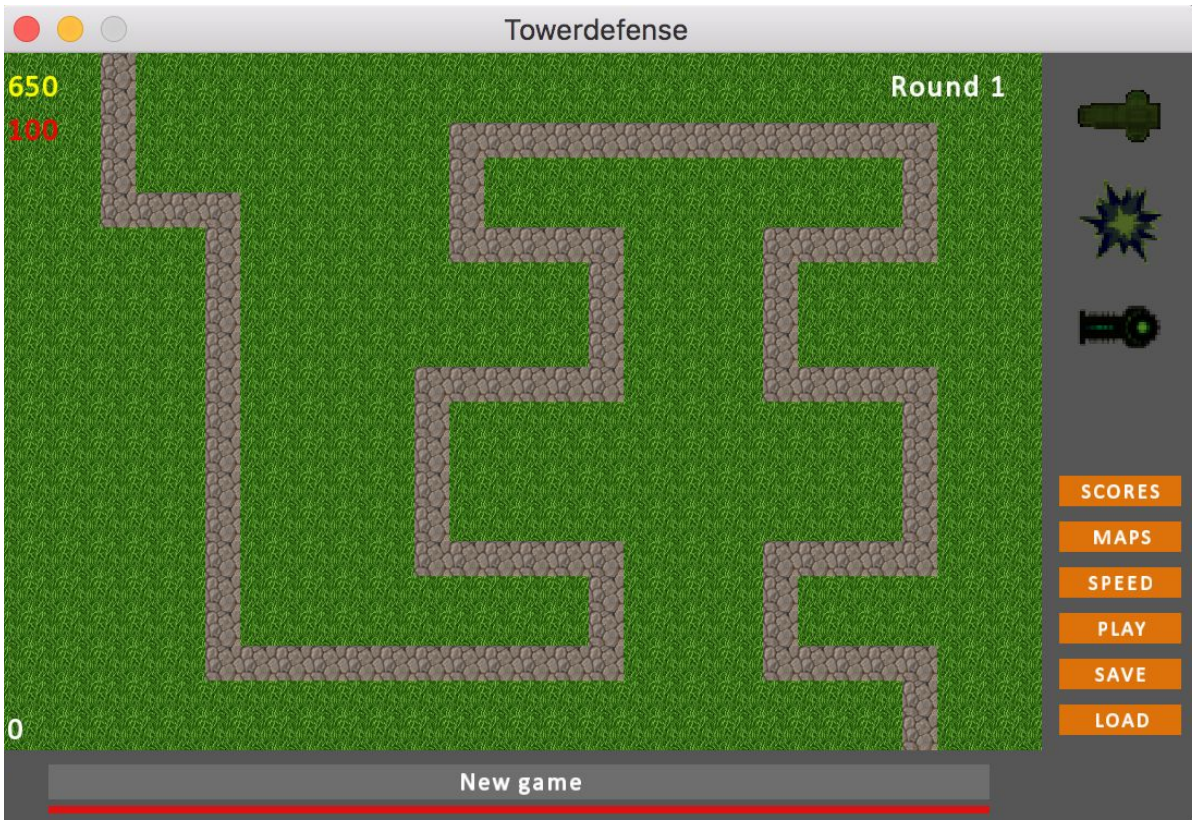


Figure 2: The opening window of the Towerdefense game.

The menu can be seen on the gray background and the game area is all the rest. The menu has a textbar for game information and a lifebar for life visualisation. There are six menu buttons and three tower buttons. The game related stats can be seen on three different corners of the game area. The bottom left corner shows the player score. The top left corner shows the player money (yellow) and the player health (red). The top right corner shows the current round.

There are three different kind of towers to build. They all have their specific purposes with varying amount of range, cost, and damage. The towers can be built during build phase by clicking on one of the towers on the upper right side of the window. The selected tower can be placed on the map, if the player has sufficient funds. By clicking a tower on the game area, a menu opens right next to the tower. The player is able to upgrade and sell the tower, and also view the range of the selected tower. The game area is basically the whole window excluding the menus. There is a clear path for the enemies to travel. No towers can be built on this area. The player has health of 100 in the beginning and it is decreased by enemies that pass the map. This means all the enemies the player isn't able to destroy. The window can be closed by pressing the cross at the upper left corner.

Menu buttons
Score: Shows the leaderboard.
Maps: Map selection. Only maps that have been reached, can be selected.
Speed: Speeds the game by x2 or x4.
Play: Used to play and pause the game.
Save: Save the current game. Cannot be done during active play. Only local saves.
Load: Loads the saved game, if such exists. Only one game can be saved at a time.

Towers
BasicTower: Shoots one enemy at a time, no special effects, weakes of towers.
FreezeTower: Slows a single enemy at a time, but doesn't damage them, prioritizes enemies that aren't yet slowed.
BlastTower: Damages all enemies in an area where it hits, significantly stronger than Basic.
PrecisionTower: Quickly kills most dangerous enemies from long range one at a time. Strongest against single targets.
MultiFreezeTower: Slows up to three targets, otherwise similar to "normal" FreezeTower.

Game play
The game is played in two phase. The window opens as build phase. When the player is ready to play, the play-button is to be pressed. Now the game phase is on. When the round is complete, the game return to build phase. This goes on as long as the player has health. When the player's health runs out the game is over. At this point it can be restarted by clicking "New game" or selecting a map. Figure 3 and Figure 4 shows the game during gameplay in two different maps.

The game phases:

1. Build phase
    a. Build, sell, upgrade towers
    b. Save or load game
    c. Change map
    d. Look at leaderboard
2. Game phase
    a. Start, pause game
    b. Select game speed
    c. Change map

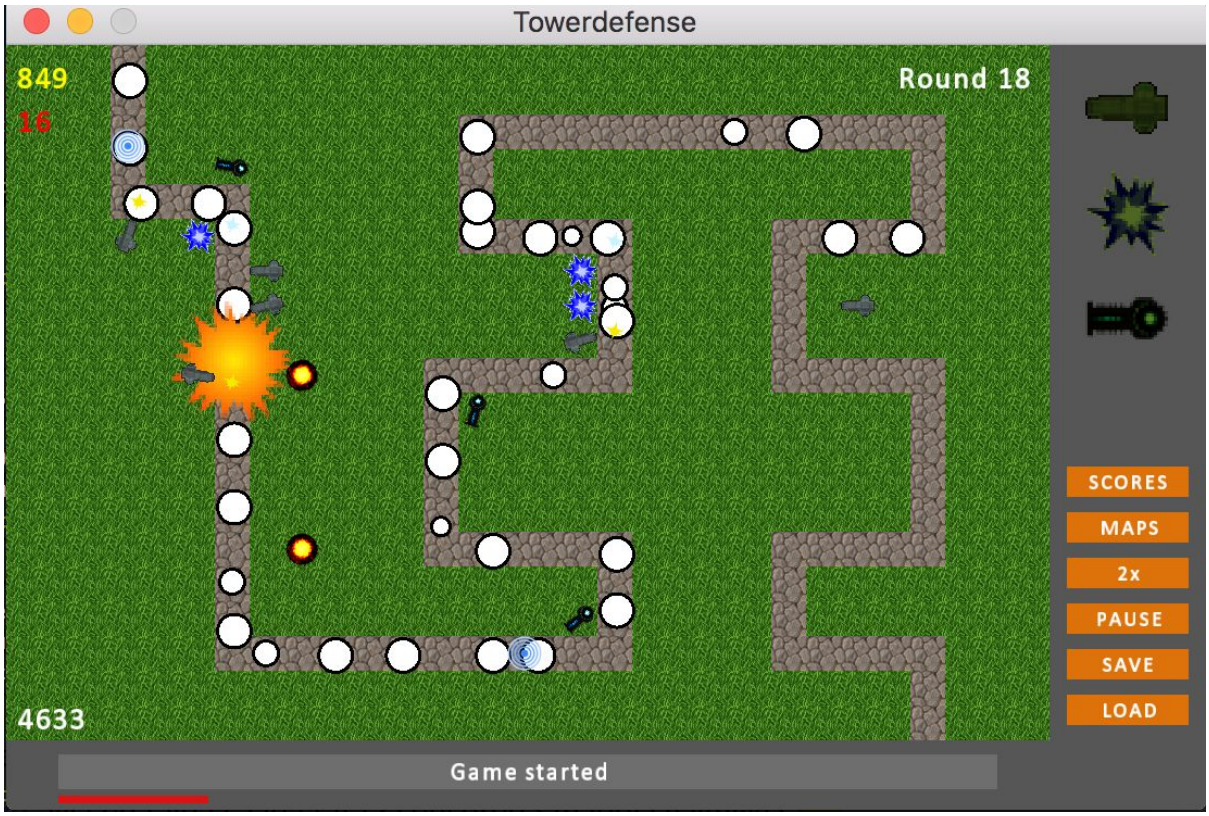


Figure 3: Window during gameplay.

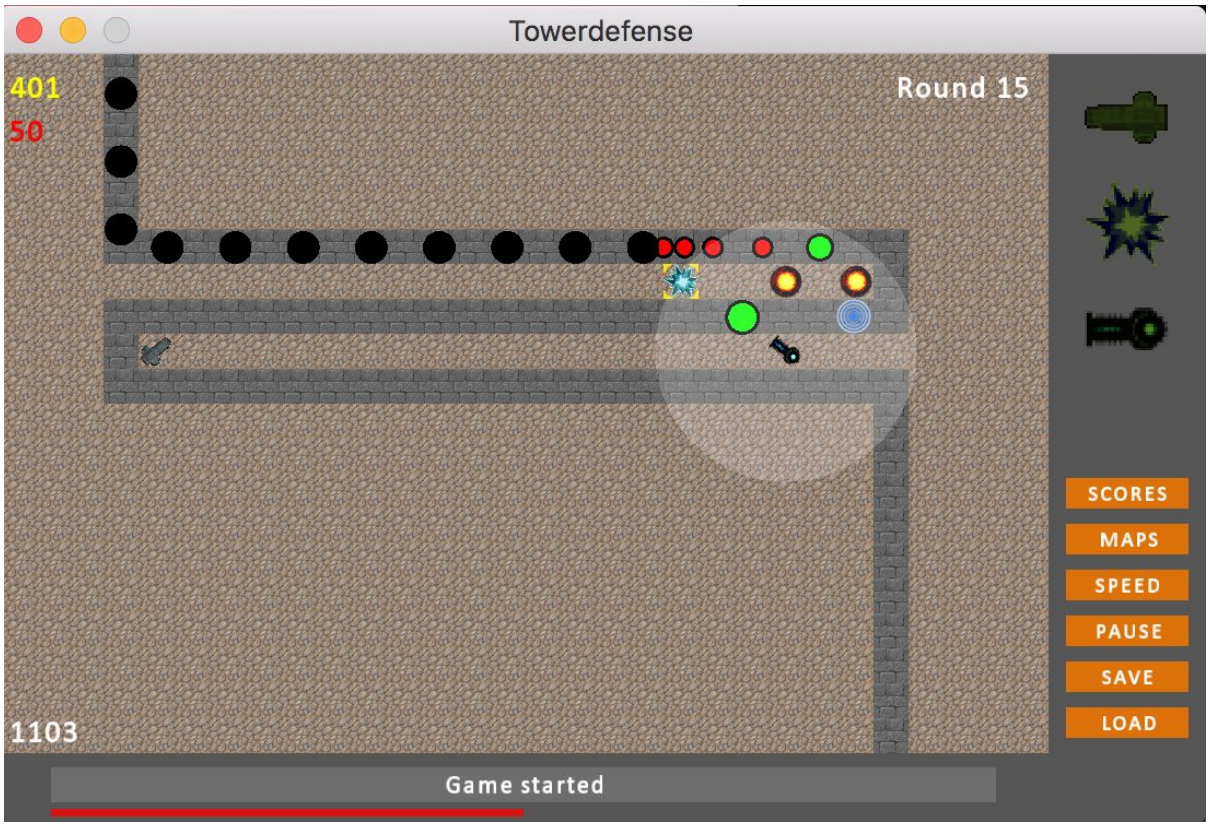


Figure 4: Another gameplay scenario on another map.

Creating and adding maps
As said before we have support for Tiled Map Editor. Here is a very extensive tutorial for creating maps.
There's some requirements concerning the map.

- Size of tiles must be 32px
- Height of map must be 20 tiles and width 30 tiles
- The map must have Object Layer named Collision
    - objects in this layer should cover every spot on the map that tower can't be placed
    - we recommend using rectangle objects.
- The map must have Object Layer named Enemies
    - the layer must have Polygon Object named Route which contains the coordinate points that enemies will follow.

If in doubt take example from maps we have provided.

After you have created the map make sure that you place the tmx file and the tileset image to maps folder. Also note that the name of the map must be map_x.tmx where the x is the next number. For example if there's already three maps you must name the map map_4.tmx. Also add the map name to maps.txt and make sure there is no extra line changes after it.

Testing

Most software classes and functions were tested with actual gameplay. For non-visual elements, most of the testing was based on stream prints and valgrind. The main rule was that everything that goes into master must work, and the whole team obeyed that rule very well.

Graphics were tested visually and their functionality by playing. Map functionality was seen during gameplay. Enemy and tower functionality was seen during gameplay and also tested the whole time during their development. Game logic was tested mainly by gameplay.

Game testing included actual gameplay testing and game-logic testing. Most of this was done during development by simply playing the game. In this section, most of the illogical behaviour and general game sensibleness were found and fixed, at least to some extent.

Future Improvements

Discussed improvements to the software and logic testing:

- Earlier game beta, which could be divided to as many people as possible.
- Code reviews, so that errors would be kept in minimum.
- Unified programming guidelines to have a unified look, which would help the testing and fixing of team members' code.
- Automated testing and TDD (test driven development)

Worklog and Responsibilities

The game development had many areas of responsibility. We had an initial division, but the closer we got to the end, the more everyone fixed and refactored other team members' code. The following shows the team members' main responsibilities:

Main responsibilities

Lauri Jääskeläinen: GUI (menus and functionalities), save and load
Joni Väisänen: Gameplay, Game class
Pyry Viita-aho: Enemy class, gameplay
Konstantin Ionin: Towers, some visual effects, some interactions between towers and enemies
Matti Yli-Ojanperä: support for Tiled Map Editor, high scores

Weekly Worklog and Time Spendage

Week 45:

- We used the week to get familiar with the scope of the project and setting up our personal development environments for the upcoming coding.
- Also we got together couple of times and wrote the project plan.
- Approximately 10 hours per member.

Week 46:

Matti approximately 20 hours:

- Researched different Tiled Map framework libraries for SFML. Tested two of them STP (SFML TMX Parser) by edoren and C++/SFML Tiled map loader by fallahn. Came to a result that the second one is better for our requirements.
- Studied what is cmake and how it works. Implemented CMakelists.txt file and got cmake work with SFML and SFML-tmxloader.
- Created first maps with Tiled Map editor.
- Implemented simple main to show the map.

Pyry approximately 20 hours:

- Get to know to sfml and trying some ready made games to figure out how to make games with SFML.
- Learning Tower defence game logic by testing different kind of tower defence games.

Joni approximately 25 hours:

- Played tower defense games
- Tried desperately to install SFML to Windows environment and finally gave up
- Tried to install SFML and TMX libraries to the current Ubuntu virtual machine without success. Reinstalled Ubuntu virtual machine and managed to make our programming environment to work.

- Studied SFML libraries and found them to be old fashioned and awkward.
- Created basic structure for game class and sketched gameplay (what happens and when).

Lauri approximately 20 hours:
- Added menuObjects (menus, buttons, bars) = GUI
- Refactoring and new functionalities to menuObjects' classes
- Added proper main function

Konstantin approximately 10 hours:
- Looked into how to make game to function at specific pace
- Sketching tower classes, their functionality etc.

Week 47:
Matti approximately 20 hours:
- Implemented Map class.
- Implemented first working versions of functions move and draw in Enemy class.
- Implemented first version of enemy_create and move_enemies in Game class.

Pyry approximately 20 hours:
- Planning and working on enemy class.

Joni approximately 10 hours:
- Updated game class and added more functionalities
- Was in midpoint meeting and left to pre-Christmas party while other group members went to code.

Lauri approximately 20 hours:
- Added game font
- Added window (window related functionalities like button press)
- Added tower placer and gui colors
- Game vs. build phase separation
- Much refactoring of window's and menuObjects' classes

Konstantin approximately 10-20 hours:
- Continued tower sketches
- Various ideas passed between group members
- Some suggestions for enemy class, discussing tower-enemy interactions with Pyry

Week 48:
Matti approximately 20 hours:
- Made adaptations for pause support.
- Made adaptations for speed change support.
- Tried to figure out problems with integrating Tower class to the rest of the code base.

Pyry approximately 20 hours:

- Working on enemy slow functionality
- Changed Enemy class to template form

Joni approximately 15 hours:
- Tested current version of our game and tried different ways to add more enemy waves to game loop.
- Debugged operation of the tower class and fixed memory leak bug.

Lauri approximately 20 hours:
- Added texts
- Button and event functionalities
- Game speed related fuctionalities
- Added map selection
- Refactoring

Konstantin: approximately 10-20 hours:
- “Finished” tower classes that didn’t work with the rest of the program
- Attempts to fix problems with dynamic memory with Matti (turned out later that those functions simply couldn’t work with the rest of program)

Week 49:
Matti approximately 20 hours:
- added score calculation
- changed the way towers are handled in the gui
- added possibility to show tower range
- created a pop up menu for selling and upgrading towers

Pyry approximately 20 hours:
- added money calculation
- implemented tower rotation function

Joni approximately 15 hours:
- Was amazed at how gorgeous our game is.
- Made functionalities for game ending and implemented some gameplay indicators.

Lauri approximately 20 hours:
- Much refactoring and unifying
- Added save game
- Button textures

Konstantin approximately xx hours:
- Worked on game rounds and file(s) that would contain that data
- Added support for delays between waves of enemies
- Added alternative way for towers to “remember” which enemy was their target
- Created MultiFreezeTower

Week 50:
Matti approximately 20 hours:

- Fixed minor bugs
- implemented Highscore class
- support for user added maps without rebuilding
- Wrote parts of this report

Pyry approximately 20 hours:

- Tower sprites instead of rectangles
- Made some reasonable rounds on txt.file
- Adjusting towers
- Added additional map
- Wrote parts of this report

Joni approximately 15 hours:

- Tested game with different rounds.txt files and different tower locations.
- Added procedure if game ends to player's victory.
- Wrote parts of this report.

Lauri approximately 20 hours:

- Added load functionality
- Generating many small functions
- Game and button logic fixes
- Testing and refactoring all
- Initial implementation of newGame

Konstantin approximately 20 hours:

- Added initial upgrade function, later fixed/improved by other members
- Fixed MutliFreezeTower
- Made a bunch of pictures for tower sprites and tower hit effects
- Implemented tower hit effects in game
- Various additions to code, such as tower placement in function that loads saved game
- Started working on an additional map (possibly too late)
- Wrote parts of this report