Tower Defense

Group: tower\_defence\_santeri\_salmela\_5

Mikhail Romanov, Aki Oura, Leo Teodosin and Kalle Lindgren

December 2023

1. **Overview**

We have built developed our own defense game. Game has fully own graphics and a lot of features. Our game has 8 different kinds of enemies and 6 different kinds of towers. Enemies and towers have all their own animations when attacking, moving or dying. Game has money system where player gets cash by surviving rounds, player can use that cash to buy new towers or to upgrade existing towers.

Game starts with main menu, where user can choose level and difficulty to play. User can choose between few hard coded maps or randomly generated map. If user chooses randomly generated, game will randomly generate fully new level that player have never played before.

After main menu game will start and players mission is to survive as long as he/she can. User wins game by reaching certain amount of round or loses by letting to many enemies get to the finish.

List of features:

* Main menu
* Animation of objects
* Own graphics
* Pre-existing levels
* Randomly generated levels
* 8 types of towers
* 6 types of enemies
* Mouse control
* Different damage levels depending on type of object
* (High Score)
* Multiple difficulties

Photo of main menu

Kuva, joka sisältää kohteen animaatio, pikseli

Kuvaus luotu automaattisesti

Photo of in game action

Kuva, joka sisältää kohteen teksti, kuvakaappaus, neliö, animaatio

Kuvaus luotu automaattisesti

1. **Software structure**

In our project we have Game class that runs whole game. Game class relies on Level class that runs logic of in game level and on the Render class that renders everything that appears on the screen.

As was mentioned Level class has all logic that runs in game level. It stores current grid, enemies, towers, etc. It relies on two classes, Square and Object. Square class is running logic for all individuals squares in levels grid. For example, it stores occupancy of the specific square. Object class is basic constructor for all objects in the game. Under it we have Tower and Enemy classes that inherited Object class and serve as more specific basic constructors for enemies and towers. Then we have in total 14 subclasses for every type of tower and enemy in the game. They inherited Enemy or Tower class depending on which type they represent.

Render class helps Game class to render all graphics. It uses ResourceHandler class to get right textures for each object in the game. ResourceHandler class is managing all textures by loading them all at the start of the game and storing pointers for each one. It helps to make game smoother, because functions in Render class doesn’t need to load textures every time, but they can just use pointers that point to textures.

On the side we have two classes that are separately, because they don’t relay on any other class, but all other classes relay on them. One of them is class Vector2D that manages in game coordinates. And other class is AttackTypes that manages types of objects and has function in it that decide how much damage one object does to another.

Picture of structure here when ready

1. **Instructions for building and using the software**

**Building instructions**

BUILDING INSTRUCTIONS HERE

**Usage instructions**

Game starts with main menu, where user should choose which type of level they want to play. User can choose to play already existing level, after what in front of users eyes will appear new window where user then can choose which specific level of 6 existing user wants to play. If user chooses to play random level, game will generate fully new and fully random level and game will start.

After game has started, user will see level on the screen with side menu on the right side of it. On the bottom of side menu, user will see stats of the game, as current round, money situation and lives remaining. With money that user has, user can buy towers from side menu by clicking on then and dragging to play field. All towers have different stats as health, damage, etc. That can be seen to the right side from tower. In the side menu is also “Start round” button that will start new round when pressed. Button will start new round that will generated some specific amount of enemies (depending on current round and difficulty of the game). Enemies will spawn on left side on in game level on the first piece of road and enemies mission is to get to the house that is located somewhere on the right side of the in game level. If they get to the house user will lose lives (one enemy takes on life).

Depending on enemies type enemies can only move or they have some ability as well. Enemy’s ability can be ability to attack towers, heal other enemies or make fog around them that make will decrees towers attack range. Also Boss enemy will spawn new smaller enemies when it dies.

Towers can move in side of game, but they can attack enemies. Depending on tower type, they can attack enemies by dealing damage to them, slowing down enemies or by pushing back the enemies. Also all towers can be upgraded by clicking first on the tower and then clicking on “upgrade” button. Upgrading will add damage, health and range to tower.

All objects that can attack other objects by dealing damage to them. Deals different amount of damage depending what type they are and what type object that they are attacking are. User wins game by reaching specific amount of round that depends on difficulty of the game. And loses game by letting to many enemies reach house at the end of the road (amount depends on difficulty).

After winning or losing user can go back to menu and start new game.

To be finished and reviewed when game is fully ready

1. **Testing**

In our project we have folder called “tests” that have all tests for our game. There are tests for classes Level, Square, Object and Drag button. Other classes get tested on the side of testing those four classes so they don’t have their own tests. All tests can be runed from main.cpp file that will run all tests and print error amount that appeared in tests. For actual running, our cmake file makes own executable for running tests that is named “TestRunner”.

Our tests test all function in the class that they are testing. They give functions situation when they should work and situations when function should fail. Then they check what was outcome of function and print out did function pass or fail the test, also if function failed error is added to error counter.

1. **Work log**

**Overall work log**

**Mikhail: Developed fully level class, helped with object class development and connected object, level and game class together. Made animation for objects. And also developed resource handler with Kalle.**

**Leo: Developed object class and all subclasses with different abilities to each object.**

**Kalle: Developed GUI of the game, methods that allow user to interact with the game (for example method to drag towers to games grid) and made main menu of the game with help of Aki.**

**Aki: Made all textures for the game, and they animations. Also made UI of the game with Kalle and helped in design of the GUI.**

**Week by week work log**

**TODOOOOOO**