1. Personal Information

Tower Defense

Jooel Mäntymaa 628563

Bachelor's program: Tietotekniikka, tekniikan kandidaatti ja diplomi-insinööri, 1. year student

28.4.2021

2. General description

I created bloons td inspired tower defense game, in which a group of enemies aim for a target through a predetermined route. The player has to stop the enemies by building towers next to the path that shoot or in other way defend against the enemy. Every time an enemy gets to the end of the path, player loses health. If health drops to zero the player loses the game. In my opinion, the project was done in intermediate level of difficulty

3. User interface

One can run the program by opening Game.scala file and running the Game app. After the app has started, the main menu of the game should appear on the screen. Player can choose the map and start the game from the menu. After one has chosen the map, they can start the game. Further instructions appear on the top of the screen.

4. Program structure

The program code is divided into four packages: game, objects, path and utils. 'Game' package contains files that are linked to gameplay and GUI. It contains for example game loop and other essential information in Game file. MainMenu and MapMenu are also in 'game' package. 'Objects' package includes GameObject class and all classes that inherit it. All Towers and Enemies are GameObjects. 'Path' package contains PathTile classes. 'Utils' contains all the useful classes like FileManager, CollisionEngine and Vector2D that the other classes of the project use.

5. Program structure

For the collision of projectiles and enemies I used Circle Collision. When buying new towers I checked that the new tower did not collide with path or other towers and for this I used Axis-Aligned Bounding Box. They both are presented in Mozilla's tutorial on 2D collision detection. For the sale of towers I used a simplified version of Axis-Aligned Bounding Box because I just had to check if the coordinates of the mouse cursor where inside a tower.

6. Data structures

I used mostly mutable Buffers to store and process the data I needed, because I could easily remove and add elements to them. I also used immutable Vectors whenever I had to store something and not change it later.

7. Files and Internet access

The program deals with basic text files for map and level files.

The map files are formatted like this:

“2

-48, 100

300, 100

300, 500

100, 500

100, 768”

Where the first line contains just the map number. The first and last coordinate pairs’ x or y coordinate has to be that big or small that the 48x48 PathTile is just over the border of the map. Adjacent coordinate pairs must have same x or y coordinate so that the path does not go diagonally.

The level files are formatted like this:

“4

E1E1E1W1E2W1E2W3E1E1

E2E1E2E1E2E1W4E2E2E2E2”

The first line is the level number. E = enemy and W = wait. The single number after each E or W represents the type of the enemy or wait. There are only 2 types of enemies. The type of the wait tells how much extra the Spawner waits until it spawns next enemy. The bigger the number the bigger the waiting time. Each line represents a wave. There is a standard waiting time between waves.

8. Testing

I did all the testing manually. Whenever I programmed new code, I tested it usually with GUI. Everything should work fine, but because I tested everything manually there might be little bugs that I did not notice. I planned to do unit tests a lot but did not have much time to do those.

9. Known bugs and missing features

I had no time to make the game configurable through setting files.

10. 3 best sides and 3 weaknesses

In my opinion, the adding of a new tower is one of the best sides. It shows the view range of the tower and clearly points out where one can place the tower and where they cannot.

I think that the gameplay is one of the weaknesses. I did not have much time to make sure that the game was fun and challenging to play. It could have also made the program more readable if I had divided the Game app into two different files. I added the Vector2D a little late so the GameObjects have a location(Vector2D, coordinates point to the center of the object) and variables for x(xPos) and y(yPos) position which refer to the top left coordinates of the object. This can be a little confusing at some parts of the code.

11. Deviations from the plan, realized process and schedule

In first two weeks set the coding environment and git repository up. I also created simple game loop in Game, basic GameObject, Tower and Enemy classes. The next two weeks I added FileManager, game maps' file handling and first version of enemy path and enemies' movement. After that I changed the file format of the original to the new because I could not implement enemies' movement properly with the original map file format. On these weeks I added game pausing feature and towers' building feature. On the last two weeks I implemented Tower's shoot function and Projectile's collision with Enemy. I added game levels and their file handling. I also added menu, different types of Enemies and Towers and created player's health and money systems. My time estimate of the plan matched quite well with the reality. In other words, there were not big differences. The order of the progress was pretty much as planned but I worked far more in the last four weeks than in the first four weeks.

12. Final evaluation

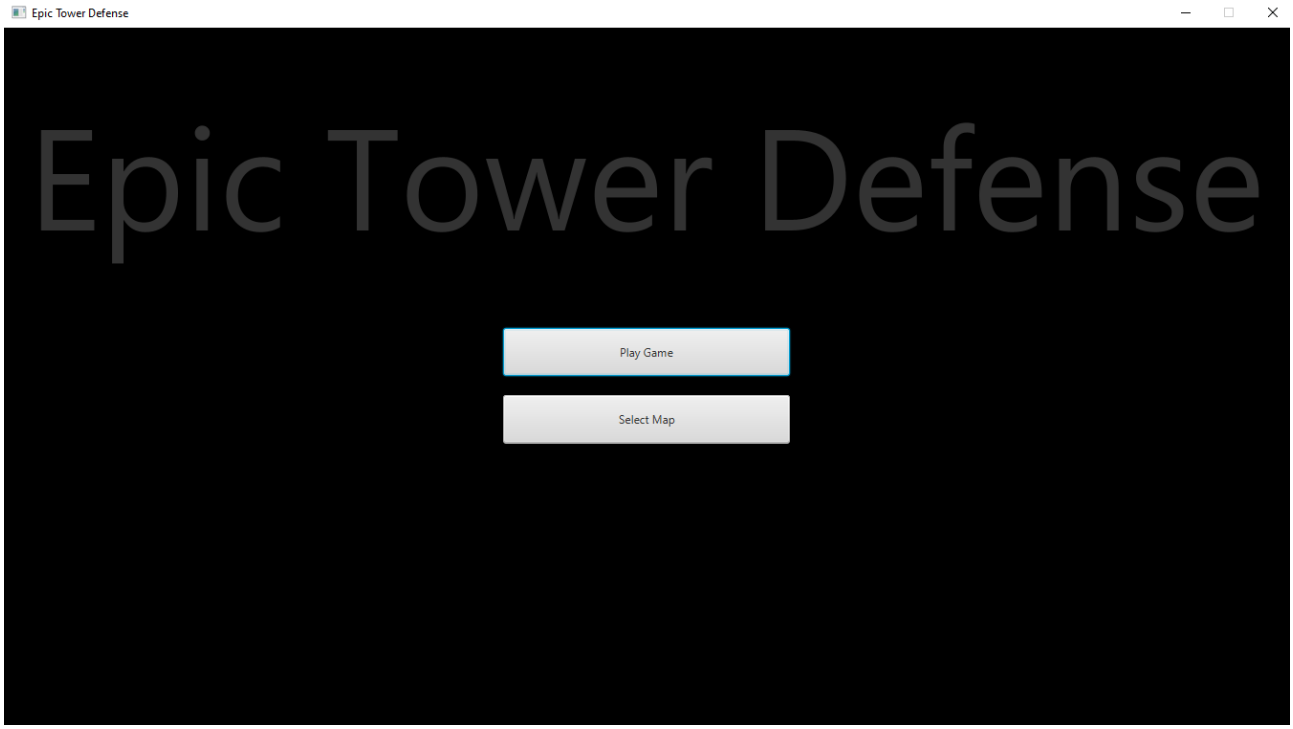
In my opinion the project was pretty good, although it could have been better. If I started the project again now, I would work more on the first weeks. I would also implement unit tests every time I programmed a new feature to the project.

13. References

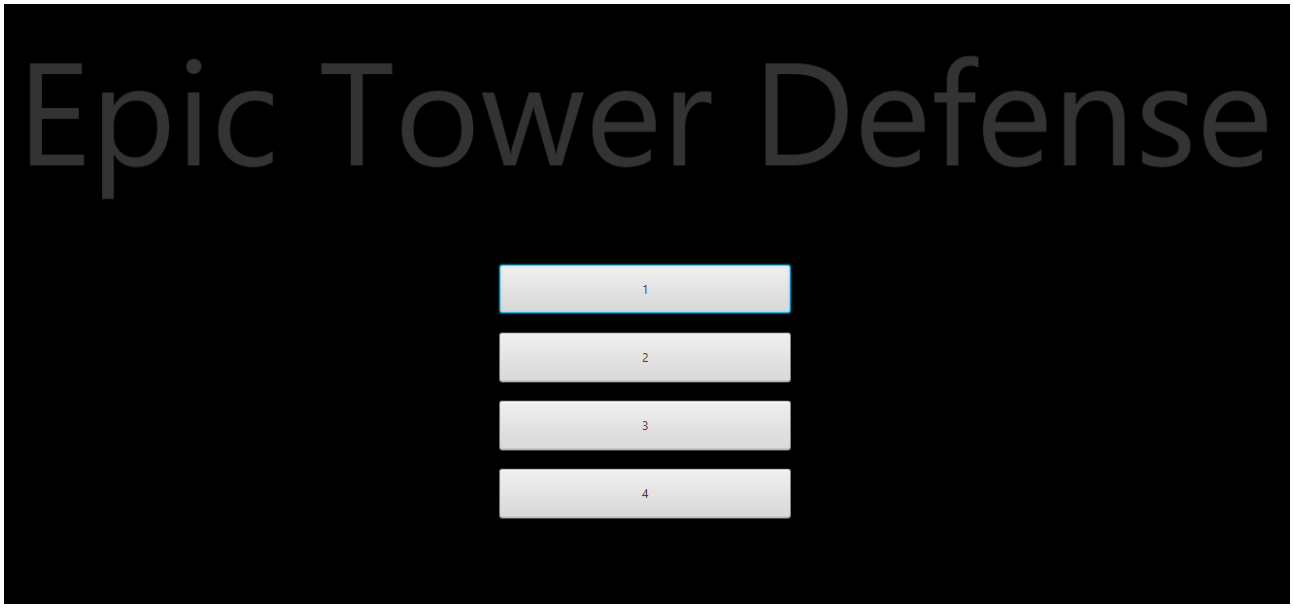
https://developer.mozilla.org/en-US/docs/Games/Techniques/2D_collision_detection

14. Appendixes

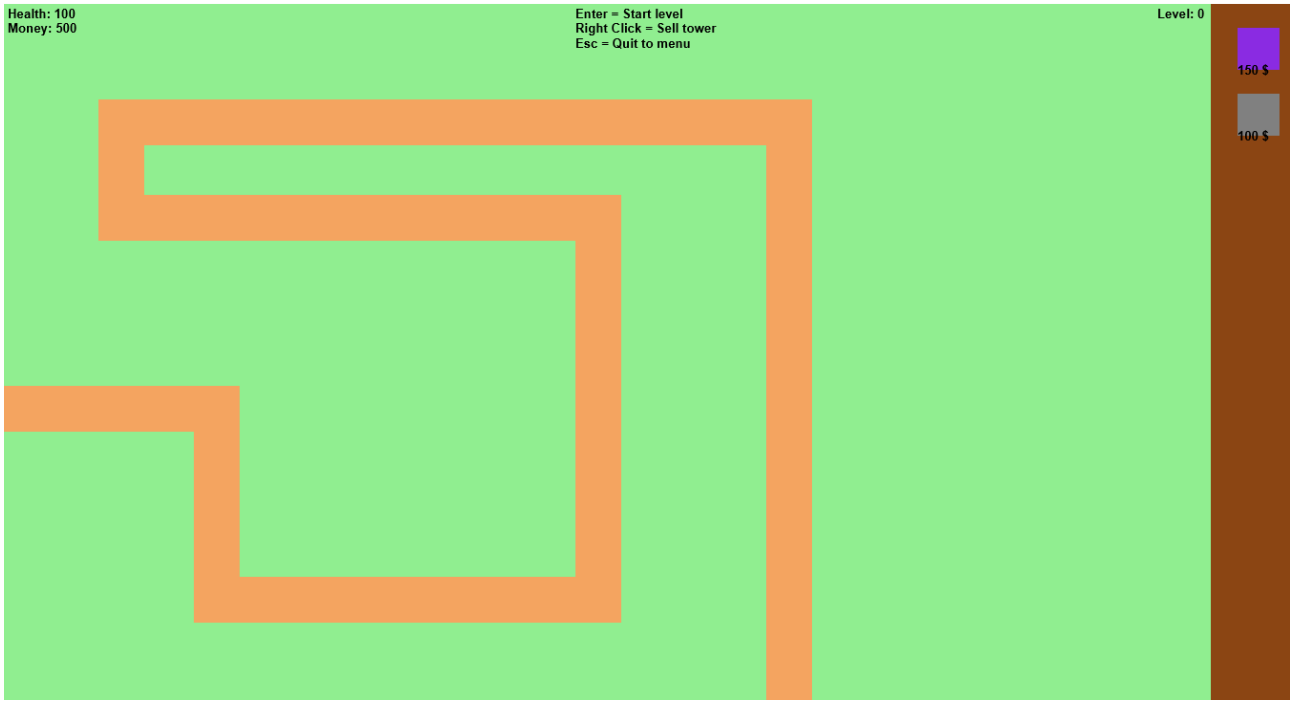
Main menu



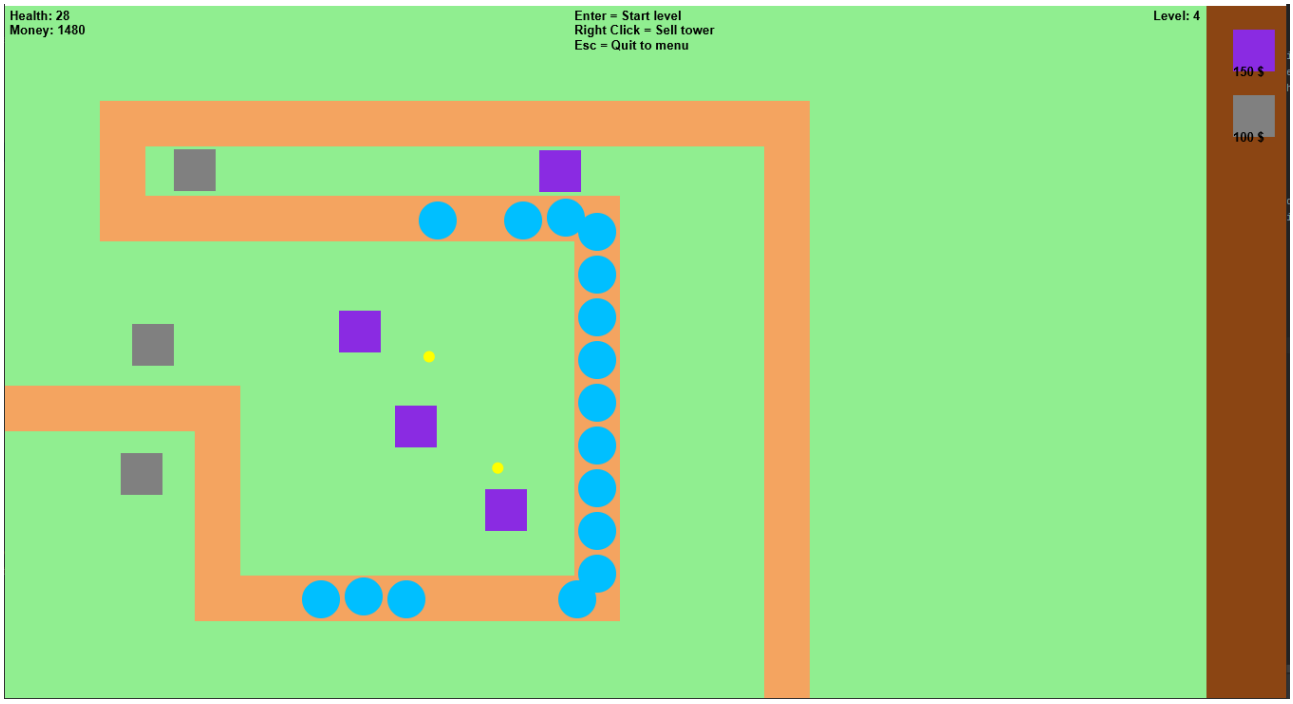
Map selection



First Map


Health: 100
Money: 500
Enter = Start level
Right Click = Sell tower
Esc = Quit to menu
Level: 0
150 $
100 $

Gameplay


Health: 28
Money: 1480
Enter = Start level
Right Click = Sell tower
Esc = Quit to menu
Level: 4
150 $
100 $


Health: -4
Money: 2120
Enter = Start level
Right Click = Sell tower
Esc = Quit to menu
Level: 4
150 $
100 $
You Lost.
Press Enter