26.04.2020

CS-C2120 Ohjelmointistudio 2: Projekti

Tower defense

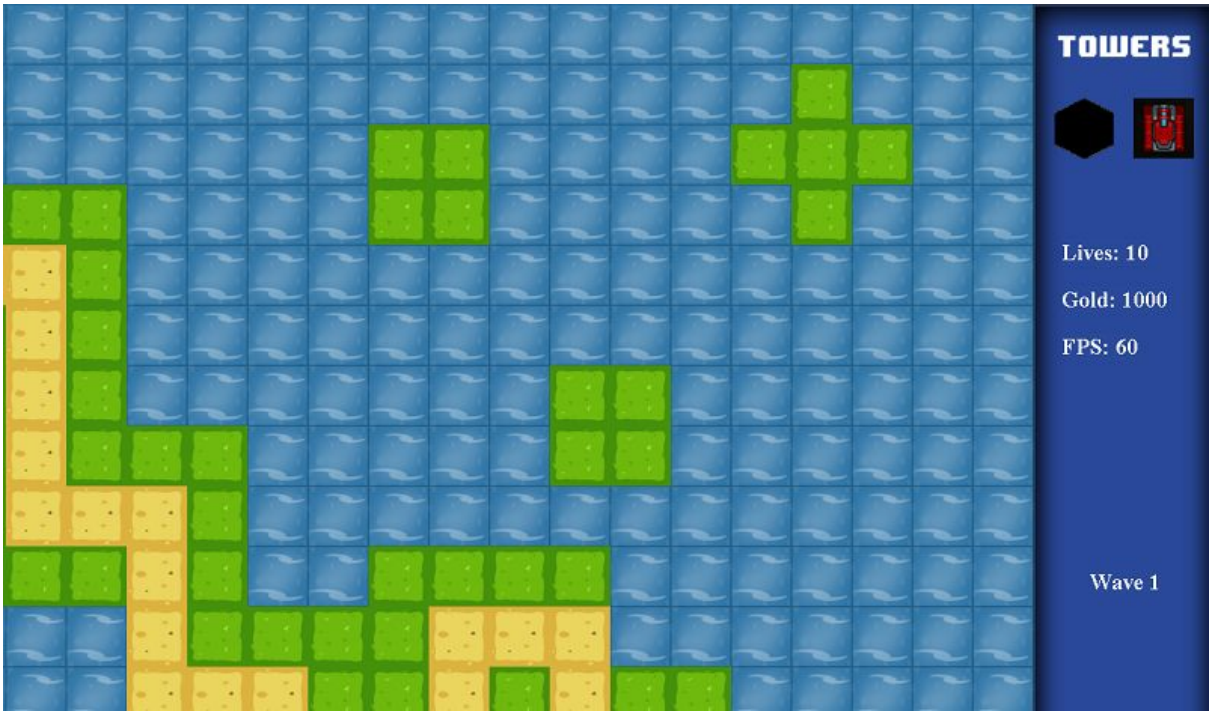
Nikita Kallio
714477
Computer science 2019

General description

Project is a basic tower defense game where players has to build towers to prevent enemies from passing through the level. Enemies spawn in waves and each wave is harder than previous. By killing enemies player gets gold to buy more towers. Player lose if one's health drops to zero.

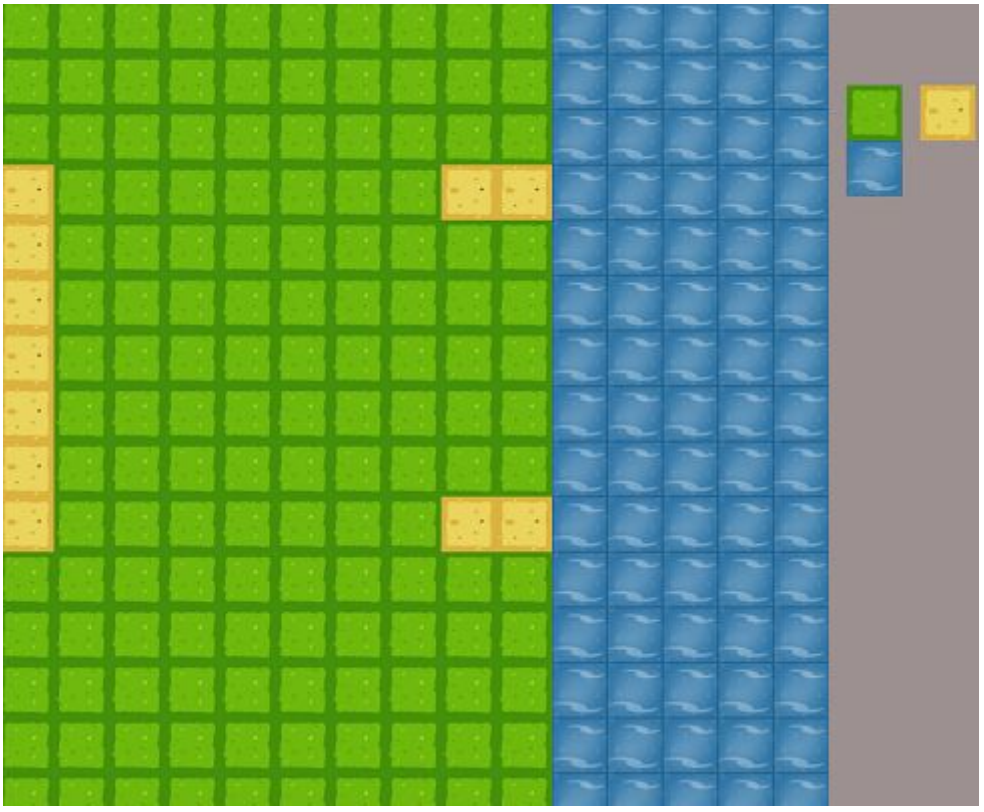
User interface

Game starts by running bootCl class from src/game root. At beginning game launches the main menu where one can start new game, open editor or quit.



In game mode player can choose tower from the menu on the right by clicking “red” or black buttons. Didn’t have time to implement tower info so red button is for cannon tower and black button is for frost tower that slows enemies. On click tower will stick to the cursor until it will be placed on the map. There’re also live, gold, wave and fps counts. With arrow keys player can slow or speed up game.

In editor mode there's a menu for selecting tile on the right, clicking on one of tiles picks that tile. After picking a tile player can change map tiles by clicking on one. Player also can hold left mouse button to wipe/change tiles faster. New map can be saved by pressing s on keyboard.



Program structure

There is UML diagram in src/res.

Game starts by booting BootCl. BootCl starts StateManager that extends enumeration and has a value. By default the value is MainMenu so starting the game will open a MainMenu. By clicking on buttons in the MainMenu, StateManager changes state of the game. For example if one clicks play button then StateManager sets its value to Game and creates new Game.

BootCl and many other classes use Helper object. Helper has openGL methods such as drawing textures, blending, creating screen and more. Helper is located in src/settings. In the same package one can find Clock class. Clock is like an action listener in java. Purpose of clock class is to create time so that enemies and tower

could know when to shoot/move. To be honest, I should have made Clock object instead but class is still working. It's just one needs to pass clock as a constructor.

Game has also trait named Entity. Entity has useful methods such as replace x , y, width etc. Game would be fine without it.

Game creates a new player and player picks towers from ingame menu. Tower is an abstract class that implements basic methods for towers. Tower use projectiles. Projectile is also an abstract class. ProjectileType and TowerType are for adding new towers and projectiles. With their help it becomes much easier. In case of new tower, user should only override shoot method and make new TowerType.

Enemies come in waves and waveManager manages them. In the game there's only one type of enemy due to me running out of time.

Editor mode allows user to create own maps and save them. Saved maps can be found in Reverenced Libraries in project folder. User can load saved maps. Currently there's a bug with saving maps that I could not get rid of. I'll tell more about the bug in "Known bugs and missing features".

For more details please rid the code. There's commentaries in the code.

Data structures

Mostly, data are stored in Buffers although some of data structure would probably fit better.

Algorithms

Game has 2 main algorithms which are: pathfinding, aiming and shooting projectiles.

<u>Pathfinding algorithm:</u>

In Enemy class there are 3 methods that implements the pathfinding algorithm: findNextDir, findNextCorner, fillList. The idea is that before enemy can spawn we first fill a list with corners. Corners are corner tiles. That means that direction changes

upon stepping on the corner tile. In findNextDir method we first check what are the possible directions enemy can take after arriving at a corner tile. If there is a possible direction we go that way. That means that there is a tile next to the corner tile that is same TileType. FindNextCorner method finds first tile that is a corner, marks it as a corner and returns it. And finally fillList method. In this method we loop until there's no directions to go or counter is large enough and while looping we are adding all found corners to the list. After all calculations are done we use this list of corners in update method to move forward. In case there's no corners our enemy just go straight line forward.

Me personally like this algorithm since you don't need to code a specific path for enemies. As long as enemies are being spawned, they will find their way around by themself. Although enemies move by themself they still need to be spawned on a road/path. For example if path ends in the middle of a map, enemies won't jump over the last tile of the path.

For more information check the code.

Aiming algorithm:

In Tower class we have methods: enemyTarget, isInRange, findDistance, calculateAngle and updateEnemyList. EnemyTarget return the closest enemy to the tower. Method first calls isInRange and findDistance methods. In other words, we first check if the enemy is in range and that the distance between enemy and our tower is less that orbitary distance. CalculateAngle method calculates how our tower head/weapon moves/aims. In other words, we use simple geometry to make tower wepon/head look in the direction of a closest enemy. In update method we check whether targeted enemy is dead and if so, we find new target.

Shooting projectiles:

In projectile class we have method called calculateDirection. In this method we calculate x and y distances from target to tower and speed for projectile.
Check code for more information.

Files and Internet access

Most of the files needed for the game are images. Images are in .png format. Textures for towers and enemies are 64x64 pixels and for projectiles are 32x32.

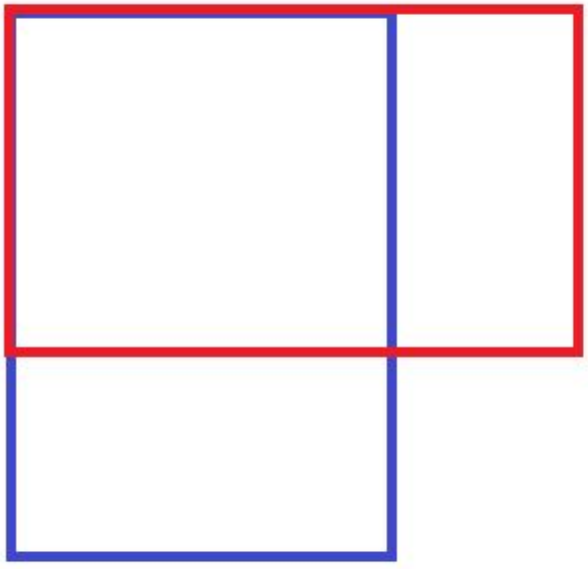
Background is 1280x960 but resized in game to 1472x960. Maps are saved in .txt format.

Testing

Since games usually has visuals, the testing of the project happened mainly by running the program. Most errors were noticeable while playing. In additional console was used a lot. Simple println line helps to find out size of buffers, current gold, target by ID, tiles, coordinates and whether if else statements are evaluated (this applies for loops too). Also I didn't have time to implement unit test.

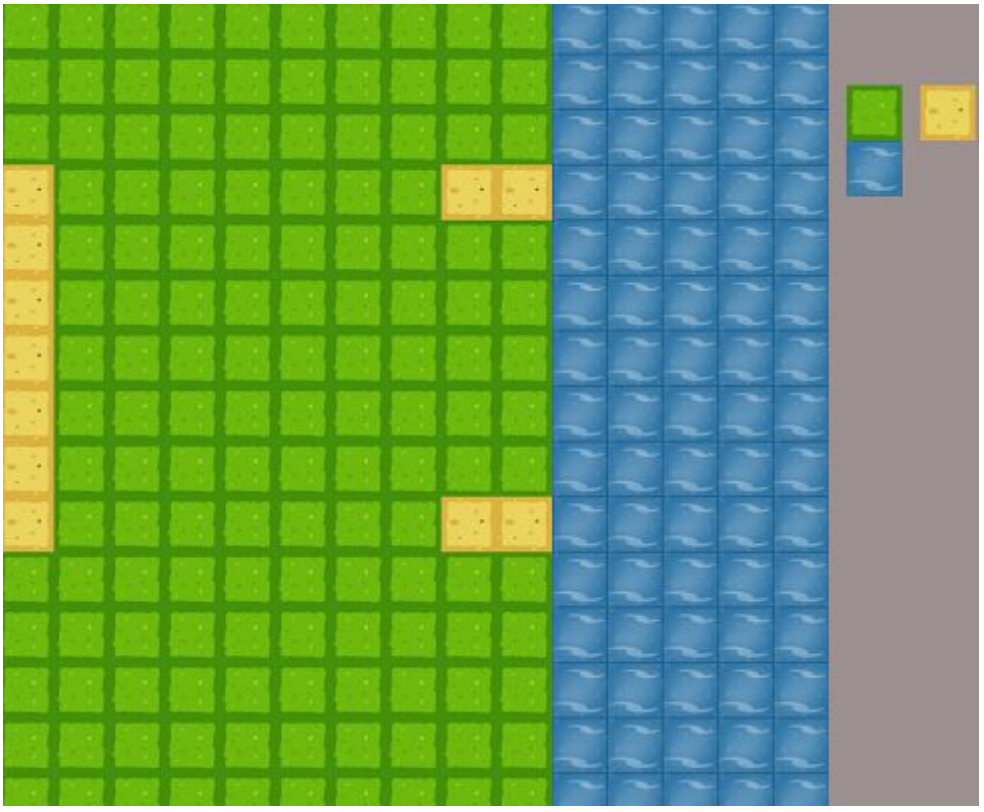
Known bugs and missing features

Firstly, I did grid in a strange way. I messed up with axis so while reading code you can find lines such as x = y etc.



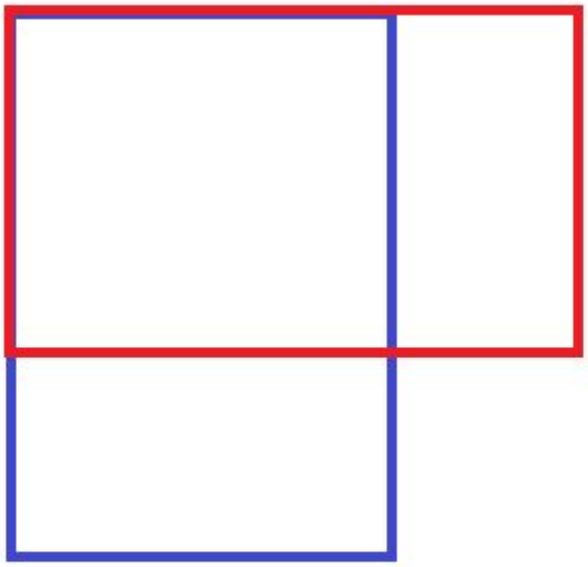
Blue rectangle is what the grid looks like and red is what it should be. I tried to find a solution but have not managed to and don't understand why (I'm getting index out of

bound or null point and just changing x and y won't help). So what I do is just swap x and y of blue rectangle to make it looks like red rectangle. Due to that I had tons of problems with pathfinding algorithm since my axis are messed up. I spent like 30 hours to just get enemies move properly. But the biggest problem with this bug is that saving game is not working as intended. It works but while saving the game we lose data



If you take a look at a picture above you can see 5 by 15 water tiles. These are so-called none tiles and are created when we get null tile error. In other words there's no tiles at these coordinates. And yes, none tile has same texture as water tile.

Funny thing that in editor mode you can edit these water tiles but they won't be saved. So they stay water tiles after saving map.



So from what I understood. I try to access tiles of red rectangle with width 20 while my actual width is 15 but somehow when I try go down 20 tiles instead of 15 and 15 to the right instead of 20, I still have index out of bound. So I thought what if my tiles that do exist are at negative index but because one cannot use negative indexes in get function I still get an error.

Loading game works as it should so actually player can save new maps and load them but will lose 5 by 15 tiles on the right.

Due to that bug I created a class that have only 2D array (mapTest). Game uses this array as a map and in fact you can edit it to create different maps.

If you want to try loading maps into actual playable game, you need create a GameLevels variable in the class and change line 19 in Game.scala from “val grid = new TileMap(map)”
to
val a = new GameLevels
a.loadMap(mappi2)
But the problem is that in all other classes we need to change variable grid value from TileMap to TileGrid since GameLevels works with TileGrid class and 2d array works with TileMap class.

On the other note, I do miss some features. Abstract enemy class. That means that I have only one kind of enemy. And the reason is that I simply run out of time.
In additional, my in game ui and editor ui are kinda lacking. Also I have not done start wave button. That means that enemies spawn immediately but you can make them stop moving by using time feature (arrow keys in game). And while they are not moving you can build towers. I also don't have tower info in game and you cannot sell towers.

3 Best sides and 3 weaknesses

Best sides:

1) Good enemy AI
2) Good shooting and aiming implementations (you can see towers aiming woah)
3) Easy to add new towers and projectiles

Weaknesses:

1) Saving is not working as intended
2) No enemy abstract class
3) In game UI lacking

UI is kinda hard to make since I'm not using java.swing (the only taught UI libraries in this course). I'm just using openGl's mouse tracking system and creating menus by myself.

Deviations from the plan, realized process and schedule

At first I wanted to create a game where enemies find the shortest path to move and players build path in real time with towers (something like warcraft 3 modes) but I realised that I'm not able to create that at the moment. I also planned to have 3 enemies instead of 1. The reason to all that is I simply ran out of time. I had and still have 7 courses (I took more courses than needed). And having databases, programming 2, c-language basics and this course at the same time is madness

(also had maths and physics). But although I say that I ran of time, I still spent twice the amount of hours I thought I needed. Roughly 200h were spent on this project. Also I think I should have ask for help in slack, since something like 50 of 200 hours were spent on figuring out errors. Also due to c-course and databases course having projects I started to push almost at the last second. In the last week I had 3 all-nighter just to make this project.

Final evaluation

I am quite satisfied with the outcome, even if the end result turned out different from the plan. On the other hand, it is good that I did not strictly follow
plan, because while doing the project I found better solutions. I felt that I learned a lot while doing the game.
The extensibility of the program is relatively good, but I would add more enemies which would improve extensibility.

If I started the project again from the beginning I would definitely add abstract enemy class.

References

Youtube, stackoverflow.