Name: Tran Quang Anh Tuan

Student number: 894041

Degree: Aalto Bachelor of Economics

Year of studies: 2

Date: 27-04-2022

2. General description

The topic that I chose was "Tower Defense Game". The result is similar to plan, but the implementation of some main classes has been changed for better extendability.

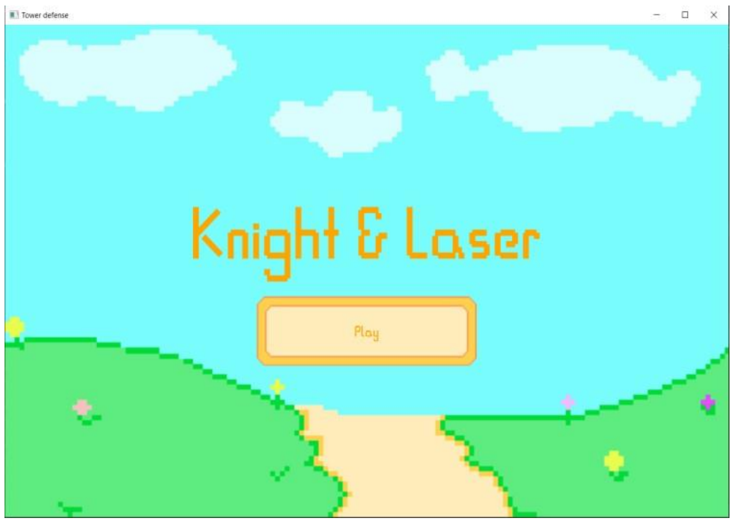
3. User interface

The game uses inputs completely from the mouse

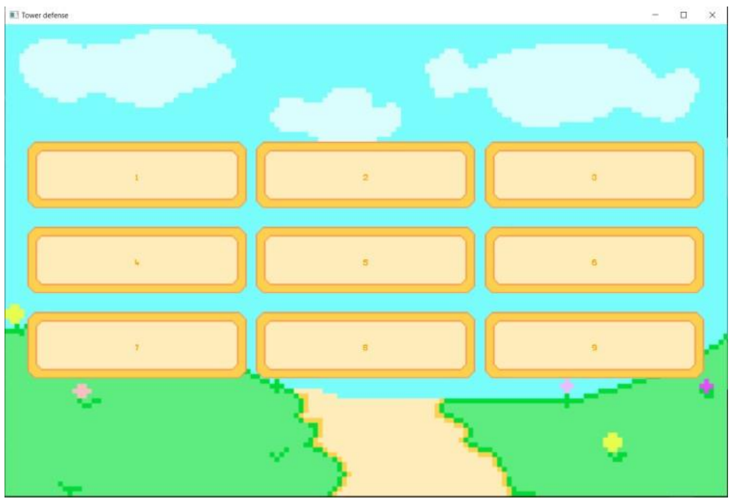
The app starts with a main menu with a play button. When pressed, the player is brought to the level menu where 9 levels are to be chosen. Choosing a level will start the game.

During the game, the player can change game speed, add, remove tower, pause, restart the game or go back to the level menu.

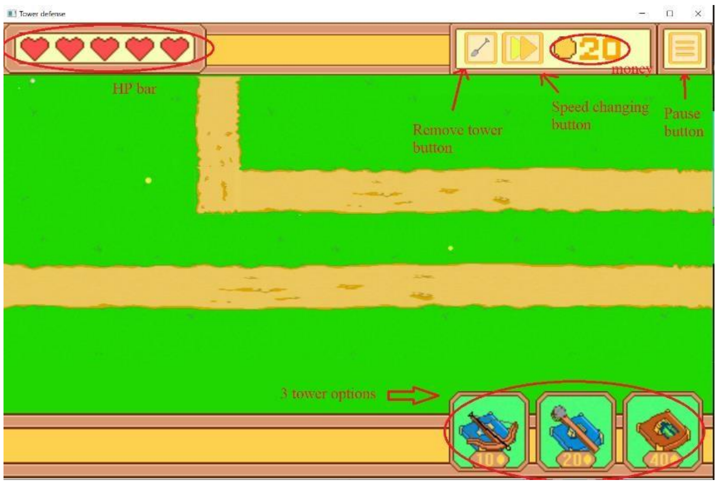
Main menu:



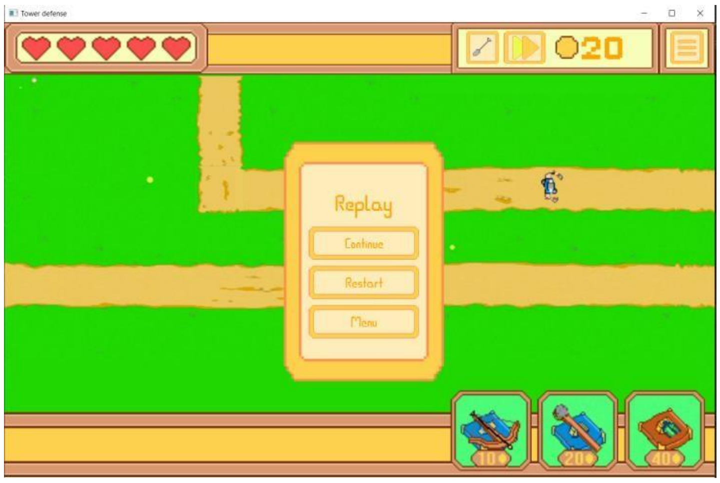
Level menu:



In game:

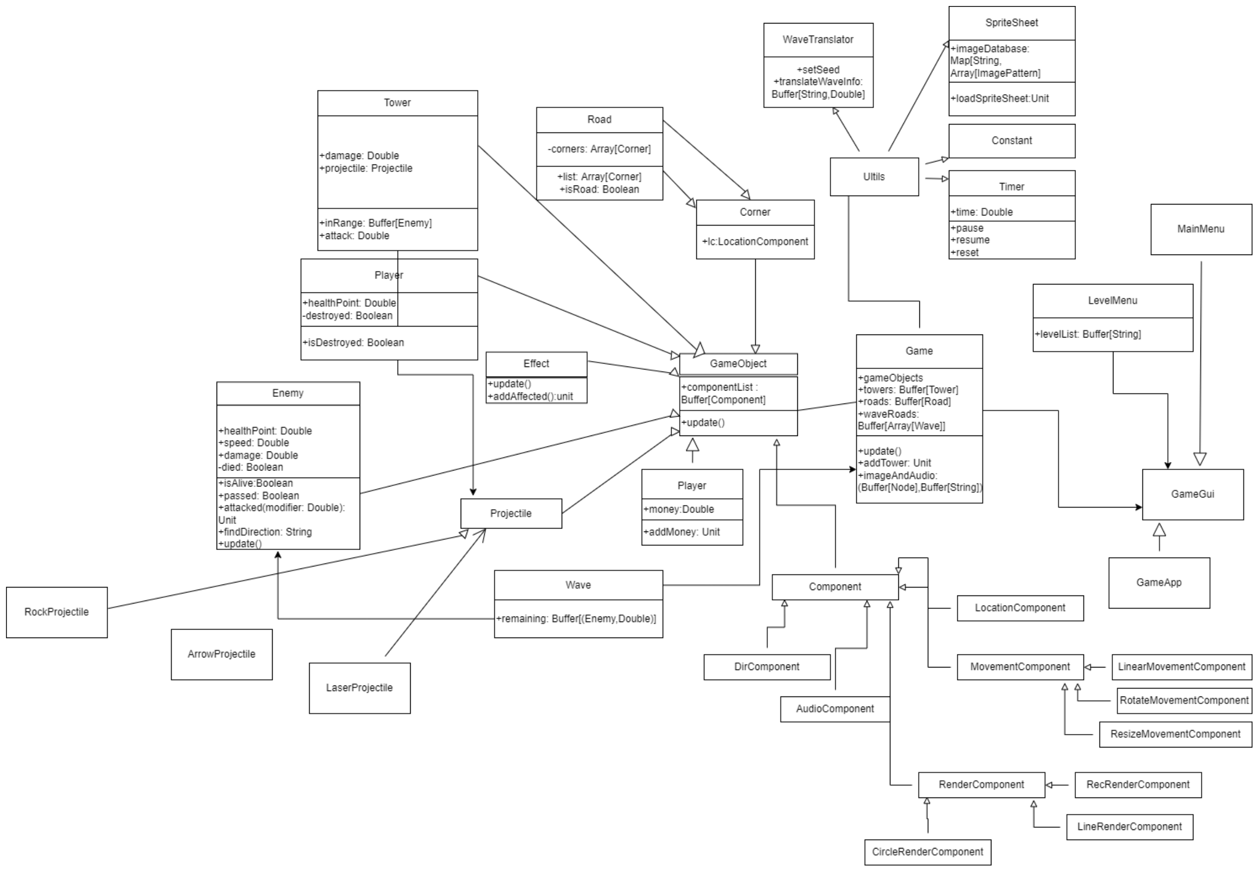


When pause button is pressed, a pause menu appears



4. Program structure

The game is split into the main game logic and the graphic classes. The game logic consists of game objects and util objects.



Some key methods:

In class: + Game: method imageAndAudio: This method grabs the images and audio name

for GameGui to use

Method addTower: this method check the requirements. If met, the tower is created and added to the game.

Method update: this method call all update method on other objects and check the game state (won,lost).

+ Tower:

method inRange: this method returns the enemies that are inside the range of the tower +

Enemy:

Method findDirection: the game checks the location of the next Corner in the road and return the direction of moving as String.

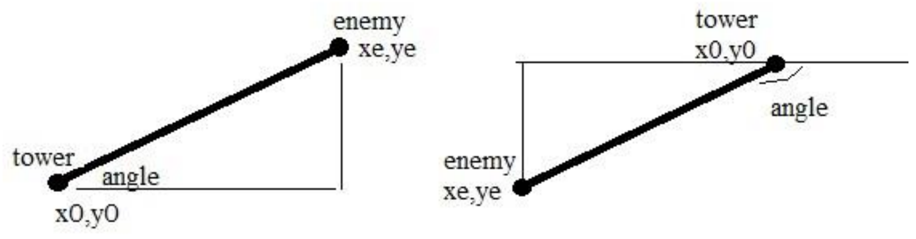
Method getdxdy: This method get the current direction String and returns the change in location.

5. Algorithms

Some of the most important algorithms are those that perfrom the visualization of towers and projectiles.

The tower barrel is rotated to point at the closest enemy using the location of the enemy and the tower.

The differences in X and Y is first calculated and then arccos is calculated to get the angle.



```
if (ye>y0) angle = arccos((ye-y0)/(xe-x0))*180/pi
else angle = 360 - arccos((ye-y0)/(xe-x0))*180/pi
```

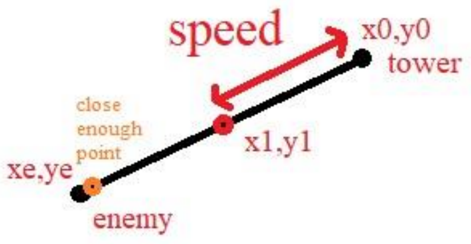
(Initially, the barrel is pointed to right or 0 degree)

The RockProjectile and the ArrowProjectile use almost similar algorithms.

The distance between the enemy and the projectile is first calculated.

Then, the next x,y coordinates of the projectile is calculated using the speed of the projectile: The next point on the line between the enemy and the tower whose distance from the projectile's position is equal to speed.

At first, I wanted to let the projectile reaches the location of the enemy, but this is impossible since the enemy is constantly moving. I change the algorithm to let the projectile be close enough to the enemy to considered it "reached".



```
distance = (xe-x0)^2+(ye-y0)^2
x1 = (xe-x0)*speed/distance
y1 = (ye-y0)*speed/distance
```

The roads in the game consist of many corner square which are points to connect lines of roads.

The enemy will look for it next direcion when reaching a corner square.

To find direction for the enemies, the game calculates horizontal and vertical distance between the current corner square and the next one. From that, it concludes to what direction is the next corner position from the current corner.

```
dX = nextCorner.loc._1 – currentCorner.loc.x dY
= nextCorner.loc._2 - currentCorner.loc.y
  if(abs(dX) > abs(dY)) {
if(dX >0) "East"
     else  "West"
  }
else {
     if(dY >0) "South"
else  "North"
  }
```

The game use pseudo-random Random object to randomize the spawning location within the first corner and also the turning point inside the next corners. This was done to not let the enemy turning to the next direction as soon as it reaches the corner. This help the game looks more smooth and satisfying

At first, finding direction was done by repeatedly comparing the x and y distance between the enemy and the corner. When the enemy reaches the corner(the enemy coordinates in inside the square), the enemy will ask for the next corner and continue running. This was not efficient enough as it keeps comparing the distance.

6. Data structures

With collections that requires removing and adding object, I use Buffer. For constant collections, I used Array.

For collections that requires getting values using keys, Map is used.

Eg: The stats of different towers, I let the tower name be the key. When tower is created, a tower name is passed as constructor and used as key to get tower statistics.

For audio playing, I used the AudioClip class.

At first, I used the Media class to play audios, this implementation is ineffective as many Media features are not used and the game runs significantly slower than if AudioClip was used.

7. Files and Internet access

The program deals with images, audios and file to store game map.

For images, I use ".png" file format.

For audios, I use ".mp3" file format.

For game map, I use Json files along with JSON library upickle and ujson.

The game does not require internet access.

8. Testing

I did not use any unit testing object. All classes and methods are tested thoroughly when implemented by running the program and also printing different internal variables to make sure everything works as planned.

Partial implementation of Gui are implemented early on to help with the testing of different classes and objects.

Each classes and objects are tested when implemented.

The game is tested as a whole by running the game with different scenarios.

The game has also been tested to see if the game can run smoothly as more objects are added.

The game has also been tested on Aalto computer to see if the game run smoothly. The game can run but the audio can not be played. The problem has not been resolved.

Although the game runs without bugs, I'm not quite happy with the game runs badly when many towers are added. I believe the algorithms and variables used to calculate distances are not efficient enough. Large number of enemies does not cause any problem.

9. Known bugs and missing features

There is a bug with the warcart enemy type. This enemy type feature is that when it die, an infantry will spawn at its location.

However, sometimes the infantry is killed instantly at the wrong position. I believe this is a problem with my implementation of the infantry spawning (the infantry is first created at the first corner and then teleported to the warcart location). I can not fix this as I have been too close to the deadline.

There is not any other known bugs to the game, but the game runs badly when the number of object increases. (Specifically the towers, when more than 10 towers are added, the game runs badly).

Most planned features are implemented.

However, some features thought of during the project have not been implemented.

Most importantly, the feature to zoom in and out of the game and the feature to let the game go fullscreen.

I also plan for more towers types but this was not possible. A more balanced gameplay between the towers is an upgrade not implemented.

10. 3 best sides and 3 weaknesses.

The best sides are:

- The random enemy wave. In the Json file, the wave is represented by a pair of number: the total hp and the latest spawn time. This information is then translated into random enemies and random spawn time. This has helped the wave creation becomes much easier as I don't have to specify each and every enemy. The number of enemy in a wave can also be much larger simply by increasing the total HP.

- The visual as a whole. The road was drawn in a way to create the effect of no edges, the enemies running randomly along the road, the tower and enemies animations.

- The different interesting enemy types, with berserker going berserk (higher speed) and warcart spawning infantry when died.

The weaknesses are:

- Many hard to extend, hard to read implementation. Eg: The road implementation, the enemy type features (warcart and berserker) being implemented in different places, the GameGui messy implementation (many buttons and their placings), algorithms as a whole is hard to understand. I think this could be done with better planning of features in the game and better understanding of gamemaking principles.

- Lack zooming and chaging the game visual based on window size(the game is fixed to a predetermined size). I am not sure about the implementation of this, I believe I will have to implement a camera object.

- The ineficient distance calculations and algorithms. This could be better by learning the different data system. Besides, the coordinates system can also be improved and implemented to improve distance calculation implementation. (An implementation of a pos class with distance calculation methods to improve extendability)

- The bloated Constant object. This was due to my lack of ordering when implementing the project.

11. Deviations from the plan, realized process and schedule.

In the first 2 weeks, all the main classes are first partly implemented, then a partial visualization was implemented. This is when testing started.

Different game features is added and previous ineficient implementaion fixed in the next 2 weeks In the

next 2 weeks, The visual representations for the game objects are created and added to the game.

Lastly, time is spent testing features, adding menus, json files, audios to the game.

More specifc progress can be found in the file project_log within the project.

The process start to deviate from the initial plan as soon as the second week. This was because the initial planning lack understanding. I first believe that each classes can be created and tested independently of each other. However, the project has been upgraded simultaneously in different parts as time goes, as this is much more effective for testing and fixing bugs. Therefore, the initial plan has not been of much help to the progress of the project.

By finishing my first project, I have learned more about progressing and therefore I can make better plans.

12. Final evaluation

I believe I have learned a lot from this project. Before the project, I have doubts about my planning and implementation prediction, as I have never done this before. My biggest problem was the lack of understanding how app object and gui implementation works. But now I have much better understanding of what I can get from different libraries and I can make better choices during planning in the future.

I believe the program has significant shortcoming regarding its readabilty and extendability. This could be improved by better planning. The program when firstly implemented is much different from how I picture it in my head, this has caused many problems. This can be improved in the future as I can make better choices of algorithm implementation.

I think the class structure can be vastly improved, as some features are implemented in a way that lack a common system.

If I started the project again from the beginning, I will plan the class structures in a more effective way, different features implementations will be put in the right classes and methods.

The time plan will also be better as I understand the progress of the project better now.

More efficient data structures will also be used to improve efficiency.

13. References

I mostly used the model project and the scalafx tutorial from the course for my implementations.

I also search online for an easy JSON library and come up with upickle. https://github.com/com-lihaoyi/upickle

Some of my audio files are from the game Knight and Merchant and Age of Empires

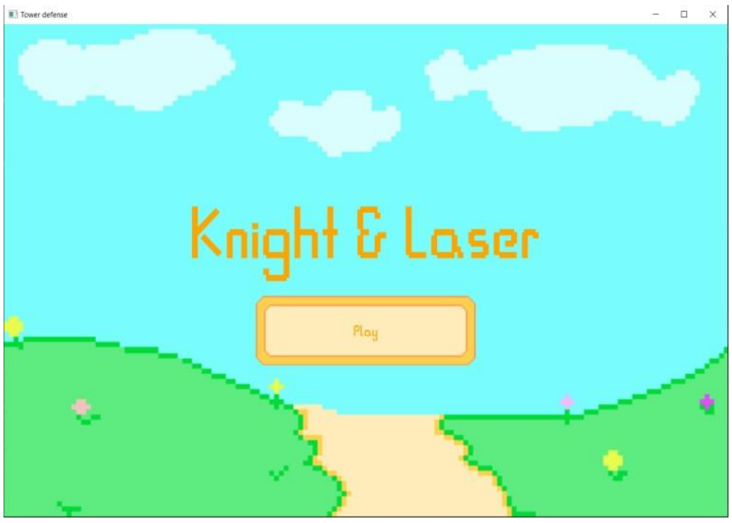
https://en.wikipedia.org/wiki/Knights_and_Merchants:_The_Shattered_Kingdom

https://en.wikipedia.org/wiki/Age_of_Empires

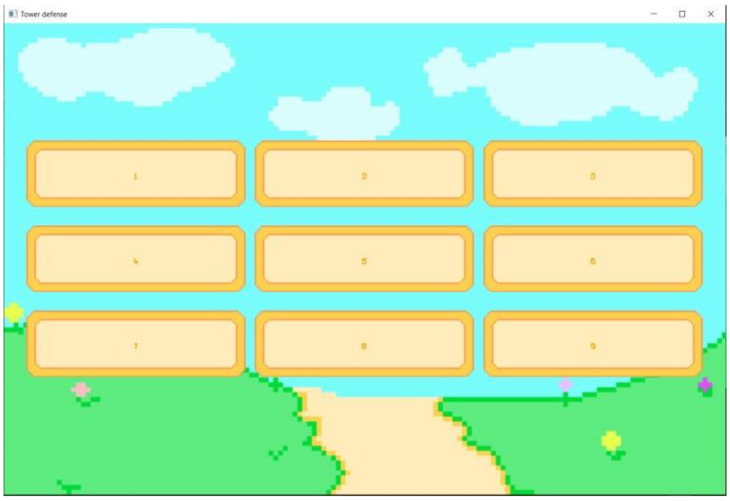
14. Appendixes

https://version.aalto.fi/gitlab/trant17/towerdefense894041/-/tree/master Main

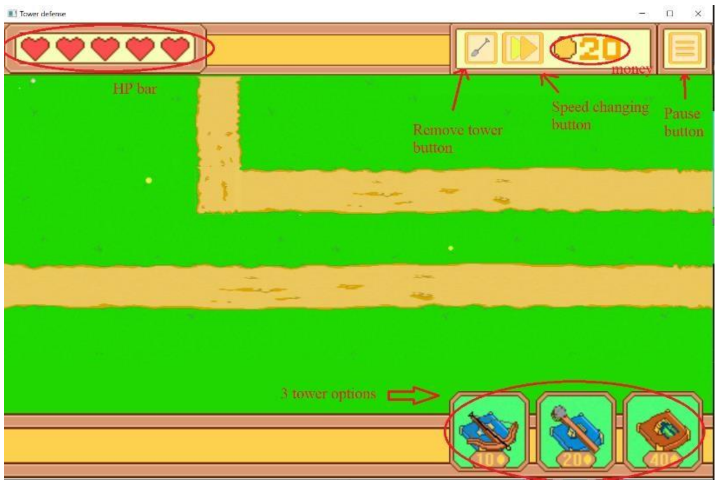
menu:



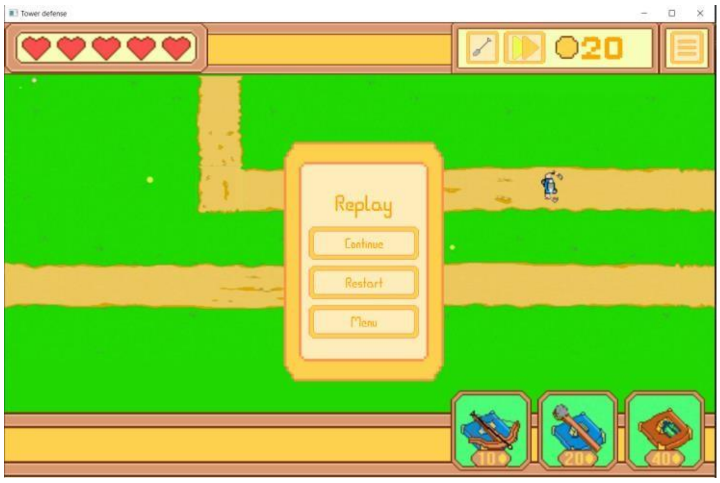
Level menu:



In game:



When pause button is pressed, a pause menu appears


Tower defense
20
Replay
Continue
Restart
Menu
10
20
40