1. **Personal information :**

   Tower Defense Game – Nguyen Khac Hieu – 901251 – First Year Student - Bachelor Program of Data Science – 23/4/2021

2. **General description:**

   This is a basic Tower Defense game, where you build tower to prevent the enemies form reaching a finished point. Enemies will run on a fixed track. In my game, there will be 3 types of towers and 2 types of enemies with different ability. Each tower has 3 levels of upgrading.

3. **User interface**

   This game will have a menu mode , where you can select the map and choose to continues from the game state where you have saved later or not. Press “Start” to start play the game. Choose the location to build your tower, when the tower location is marked, you can select and build the type of tower you want by pressing key. Tower can be selected to upgrade or remove for a price. You can pause the game by pressing the Pause button. Pressing the Home button to exit or save the game.

4. **Program structure**

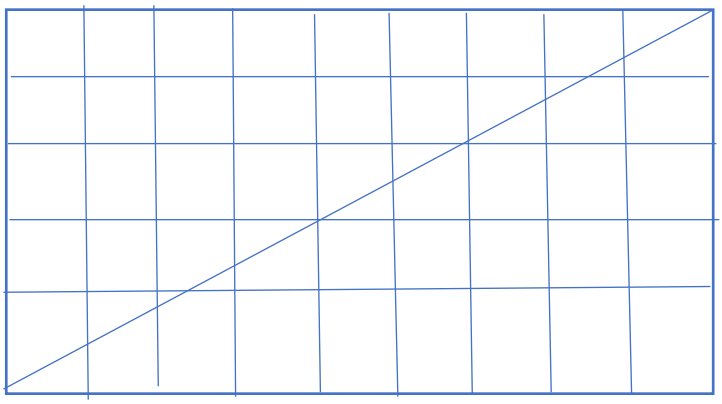
   The program structure had been modified compare to the last UML diagram. The new program structure is described in the UML diagram file

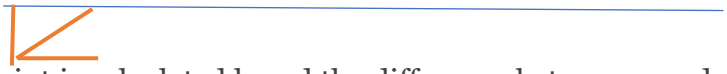
   https://drive.google.com/file/d/1BEazBQEd2hXSNBoWJSCX6rC6l95DGrdY/view?usp=sharing

5. **Algorithms**

   There are 2 main algorithms in my game.

   + Enemies Path: First I select some representation positions on the enemies ‘path of the map. The algorithm will create a full path for the enemies so that they will move from each representation position to another 1 pixel in x or y or both direction a time -as described below -with least deviation from the line matching two points.





Each next point is calculated based the difference between x and y coordinate of 2 representation points, the ratio of width and height and if needed the distance to the line mapping 2 representation points.

+ Shooting mechanism:

I want the projectile to move in a parabol shape. Inspired by the mechanism of throwing, I supposed that the x is a linear function of time and the y is a parabolic function of time – x velocity is constant . Assume that the projection reached the enemies after 10 updates.

xTarget = velocity * time + xStart. (time = 10)

yTarget = gravity * time^2 + time + yStart. (time = 10)

We know the the start and the target so we can compute the gravity.

6. **Data structures**

   I am using the data type provided by scala such as Buffer(mutable), Array(immutable)

7. **Files and Internet access**

   -The game initial file and saved file are saved in Json file. The library are used to write and read the data is “com.typesafe.play:play-json_2.13:2.9.2” downloaded as Maven external library.

   - The graphic contents are downloaded from various Website

8. **Testing**

   Most of testing is by observing how the game work. Sometimes I use the command System.out.print to test whether the output is the same as expected.

9. **Known bugs and missing features**

   The game has a bug due to thread I supposed. When the init method have not load all the enemies or towers and other function comes and the game will receive a None.get exception.

10. **3 best sides and 3 weaknesses**

I can't think of best sides but the weakness of the game is the game take a long time to set up such as reading and rendering the initial state of object.

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

My plan has been changed vastly after watching some episode of Chreno Game Programming in Java. I decide to create the main game loop first, after that I create Player, Maps, Enemies Path algo, Enemies, Waves to handle enemies, Spawner to handle Waves, Towers, Projectiles, PlayMode to handle Spawner and Towers. At last I create Menu and End mode, then I create InitReader and Saver object to read and save the game.

12. **Final evaluation**

The game still quite slow so if I can start again I will try to use more threads to handle the game.

13. **References**

I watch some episode of Chreno Programming in Java on Youtube and learn to use the Play Json to write and read json file conveniently