1. **Personal information**

Title: Tower Defense Game in Python using PyQt5 library

Student's name: Sanzhar Saidnassimov

Student number: 786667

Study program: Digital Systems and Design

Year: 2022

Date: 13.03

2. **General description and difficulty level**

The main objective of this task is to create a tower defense game in Python using the PyQt5 library. The idea of the game is to stop the enemy troops from reaching the exits, by placing defensive structures along the path.

At the initial stages of the project, the aim would be to implement the program at moderate difficulty, which includes a basic graphical user interface, as well as the unit tests for parts of the code. In case the development of the game would reach the set requirements, the difficulty level would be increased. To achieve that, conditions for winning/losing the game would be introduced. Furthermore, the gameplay would be extended with additional modes/levels/objects.

3. **Use case description and draft of the user interface**

Almost all events in the game take place on a game board that is not visible to the player. The player sees the game in a PyQT graphical window, which consists of the game map and the objects bar. The objects bar displays the initial balance of the player and the towers that player can purchase. Each towers costs a specific amount of in-game currency. The game starts as soon as player places the first tower with the "wave" of enemies following the path to reach the exit. If enemies reach the end of the path, the game will halt automatically. In order to stop them, the player has to place defensive structures (towers) along the path, which will eliminate the enemy wave. The placing happens by dragging the tower icons from the objects bar to the game map. The fire radius of the tower will be visible to the user when placing it on the map. Optimally, the health conditions of the enemy troops will be displayed to the player. Each eliminated enemy gains player more currency, which can be spent on more towers.

The current plan is to develop game up to a medium level, however, more game options might be added in the future. In addition to the visual personalization of the gameplay, the stand-out feature of the game will be multiple enemy "waves", so that, the game would not end with one enemy wave being eliminated.

4. **Program's structure plan**

The program structure represented below is the class architecture of the tower defense game. Due to limited understanding of the subject in the initial stages of the development, only a brief description of the class relationships were given in the UML diagram (Figure 1). Implemented at moderate difficulty, the tower defense game must at least contain the following elements:

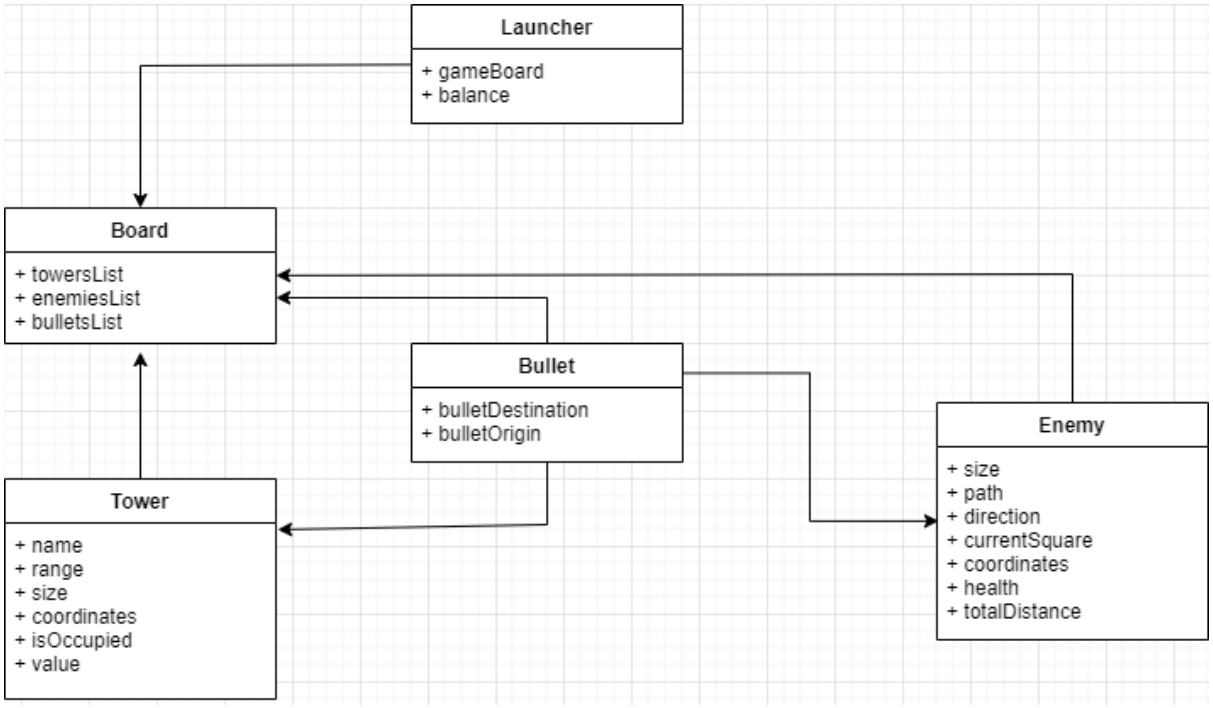

Figure 1: UML draft of the game

The GUI part of the game has not been represented in the class diagram due to limited knowledge of PyQt5 at this stage of the development.

5. **Data structures**

In order to keep track of the squares of the game board lists will be needed. In case multiple objects would be in the range of the defensive tower, a list will be used to monitor them. Since path-following algorithm is under research for the project, the data structure necessary for the enemy path is unknown.

6. **Files and file formats**

The tower defense game is going to be implemented at moderate difficulty, which means some of the graphical interfaces of the program will not be as sophisticated. The menu of the game, if implemented, might use different images, imported icons to represent parts of the GUI.

7. **Algorithms**

The program might have plenty of different classes implemented, each with its own constructor. This means that the majority of the data will be handled via newly created data types. For instance, multiple parameters have to be set when defining the object of the towers/enemies. Their constructors might use a great deal of Python's built-in structures, including both numeric (integers) and sequence (tuple) types of data (for the coordinates).

The idea for the algorithm is still under research, but the general understanding suggests that the game would base itself on the grid of squares, each containing either an enemy or a defensive structure. In order to eliminate the enemy, a defensive object will search for an enemy within the set range. In case multiple objects are in their range, priority will be given towards the object with the most distance traveled. The movement of the enemy would most likely depend on a certain path-finding technique. Proper research is still necessary to find suitable algorithms.

8. **Testing plan**

Since the game is still in the initiation phase, it is quite difficult to plan possible tests for the final program. However, one obvious way of testing would be to just play the game. By testing out different locations for the towers, monitoring particular enemy objects during the attack waves, as well as keeping an eye on the results of the game sessions, the program could be drastically improved.

9. **Libraries and other tools**

To create a graphical interface PyQT Python library will be used. While the development is in the initiation phase, it is still unknow if other external libraries are going to be utilized.

10. **Schedule**

First of all, debugging and testing happen throughout the development as the program expands. The previous week, as well as the first week of the development are dedicated towards planning and researching the topic. Before the first deadline, development of the basic mechanisms of the game, including the game board, square grid, and classes of the main objects (towers, enemies) will be prioritized. Developing those features would require minimum 3 weeks of coding. Next period before second programming checkpoint would be needed to improve the game to the next level, by adding variety and graphical appeal to the project. This includes some form of main menu, polished objects bar and map, release of multiple towers (3-4 weeks). Second session of development heavily relies on achievements of the first, hence it is quite important to sort out the basics of the game. After coding is completed, final testing happens to make final tweaks. Then project documentation is conducted, and project demonstration happens.

Timeline and deliverables:

25.2.2022 - Project plan

4.3.2022 - Plan demo session

25.3.2022 - Programming checkpoint 1

15.4.2022 – Programming checkpoint 2

6.5.2022 - Project documentation and program codes to Git

13.5.2022 – Project demo

11. **Literature references and links**

1. https://en.wikipedia.org/wiki/Tower_defenseAttachments
2. https://plus.cs.aalto.fi/y2/2022/project_topics/topics_pelit_105tornip_en/
3. https://github.com/mlisbit/wander-tower-def