Project Plan

Tower Defense: Group 5
Lim Po
Ashraf Syed
Yeonwoo Yook

<Introduction>

The project is a tower defense type of game. The features the game will have are the capability to build towers on a pre-set location.. The map will be static, for now, and the enemies will run on a pre-made path. The game ends if any enemy reaches the end of the path.

The difficulty scaling will have either more enemies spawn and they can have different attributes, for example, their speed could be faster on higher difficulties. They can also be harder to destroy, drop less money or the player generates less money.

We are currently planning to use SFML, and maybe Qt to make a GUI for the game. The first week will be spent on researching and testing how to make basically anything with these libraries. Some other obvious libraries that are going to be used are the standard library and some of its components. Boost library will most likely not be used.

The game program avoids using dynamic memory allocation and attempts to use stack as much as possible. If dynamic memory allocation is done, it should be done via shared pointers or other pointers from the standard library. Raw pointers should not be used unless absolutely necessary.

Preferred coding style is either Google C++ or https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines.

Preferred compiler is g++, and target C++ standard is C++17.

When the program is executed, it should load a small window with a grid in the middle. The sides of this window should have buttons that allow the users to build towers. The idea is to click on an empty and valid square on the grid, and then click on the “build tower” button which builds a selected tower. Figure 1 shows the idea.

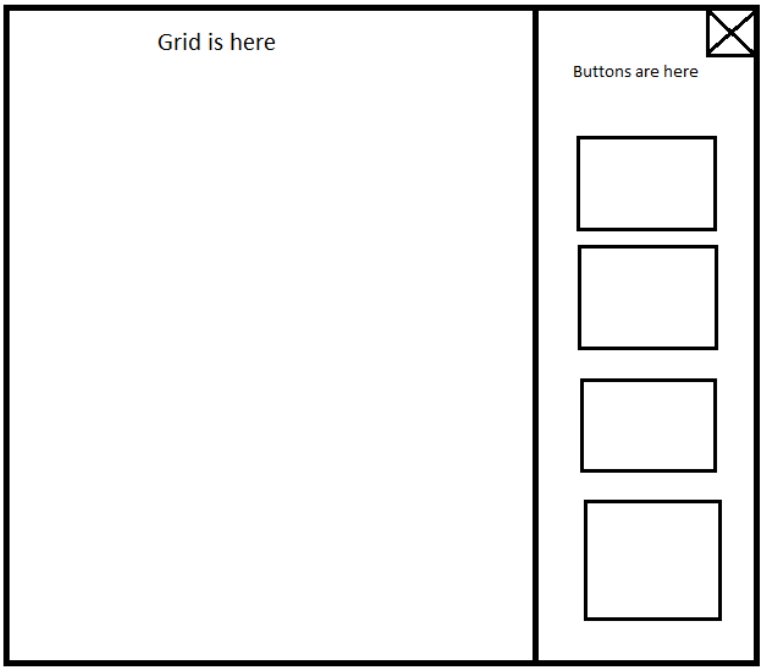

Figure 1 (note that grid is not visualized)

The whole window in Figure 1 will be built with either SFML or Qt and the buttons are simple events. First we expect an “OnClick” mouse button click event on top of a grid, and then a click on one of the buttons to build or sell a tower. All towers most likely start as “basic” towers, which can then be upgraded to be either more effective or do something else, like slow enemies.

As this is only the initial planning stage, we will be focusing on making the window first and adding towers there. The idea is to get something tangible before we decide on what to add or remove. Feature freeze of the game will happen towards the end of this month and the game should be on its alpha stages at the beginning of December. Final deadline is most likely on the 13th or 14th of December but it should be demo-able a few days prior to that.

<High-level Structure>

To implement the functionality of the game we will make use of abstract classes. For towers, we will have an abstract class that will hold general information about towers. Then we will have specific tower classes like basic_tower that will hold information about those specific towers like range, attack speed. The same follows with enemies which will have a bass abstract class that holds general information and we will have specific classes that like one_hit enemy that will store how many hits it takes, its speed.

There will be different types of towers. The types will be:

1. A basic tower, shoots enemies within its range
2. A slowing tower, slows down enemies inside its range
3. A bomb tower, shoot a bomb when enemies in range, can kill multiple enemies

The enemy types and other basic features are shown in figure 2. The idea is to have one base class that is then derived into other classes, for example, one tower class that is derived into base tower, slow tower, and multi-hit tower. This idea will, and should be extended towards enemies as well.

There will be different types of enemies. The types will be:

1. An enemy that's killed immediately when it's hit
2. An enemy that takes multiple hits to kill
3. An enemy that splits into multiple kind-1 after killed

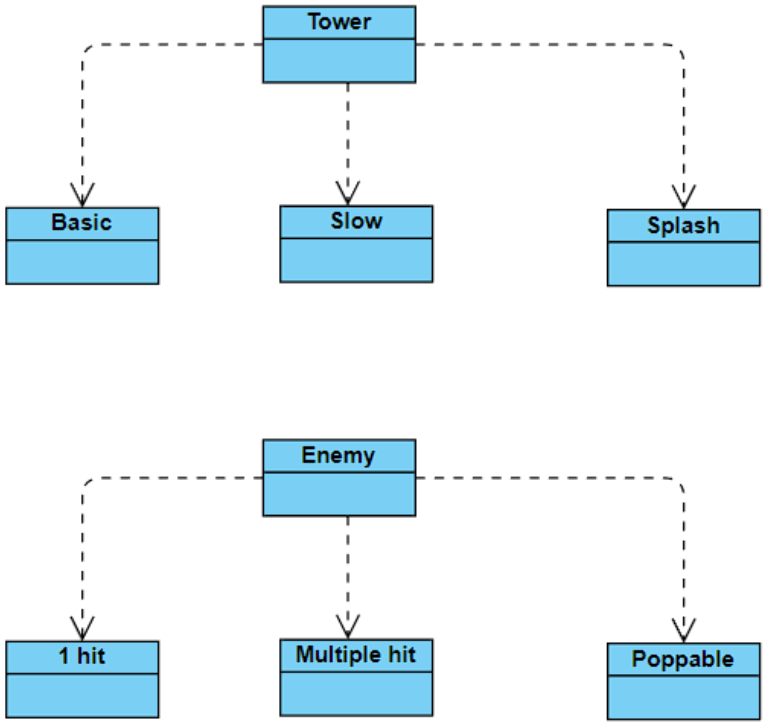

Figure 2, the basic structure

To make the map, we will make a sort of grid that will be composed of different types of objects. These objects will include ground objects, node objects, and path objects. The ground nodes will be squares on the map where no towers can be placed and no enemies can pass through. The node objects will be squares where the player can place down towers, remove towers and upgrade towers. The path objects will be squares where the enemies can pass through to reach the end of the map.

There will also be a money system in place for the player to purchase towers to place down and upgrade towers. The money can be earned through killing enemies, when the round ends, and when a player sells a tower. There will be a class that will hold the amount of money that the player currently holds and the total amount of money spent, total towers bought, and total towers sold.

There will be a total of 5 levels that can last multiple rounds where there will be different types of enemies that appear depending on the difficulty of the level. For example the first round of the easiest level would only have one_hit enemies in it. The player will also have the ability to start the wave, stop the wave, and fast forward the wave. All of that will be stored in a time class.

<Division of Work and Responsibilities>

1. All team members divide the workload as evenly as possible.
2. The team should have weekly meetings, and the date of the meeting for the following week is always decided at the end of a weekly meeting.
3. All team members should try to be present in the weekly meeting. However, if someone is absent, the other members should let the member know what was discussed and planned in the meeting.
4. Meeting notes of weekly meetings are uploaded to the project repository each week.
5. All team members respect the given deadlines and work according to the planned schedules and milestones, as well as contributing to the learning objectives of the project.
6. All team members respect other team member's perspectives. If opinions clash, it can be solved by discussion.
7. Team members share feedback about other members' works to develop the project.

<Planned Schedule and Milestones>

- First week will be spent on studying SFML and possibly Qt. The studying involves testing how these libraries work and maybe we will build a crude GUI with that.
- The second week will be spent on finalizing the GUI and starting some rudimentary tests on how to influence the GUI.
- Third week will be spent on checking how to influence and build our own code on top of the GUI through events.
- Fourth week will be spent on finalizing classes required for the game and testing.
- Fifth week will be spent on adding the classes and making the game work.
- Final week will be spent on testing and adding additional features.