Project plan - Eero Saarro, Juho Karhu, Niko Somila, Patrick Sothmann

Overview

The game we are programming is a Tower Defence-type game similar to Bloons TD where the player attempts to prevent a horde of enemies/balloons from reaching the end of the map. This is done by placing towers around the enemies' path. The towers automatically interact with enemies that enter their effective range in various ways depending on the tower type. With each enemy kill, the player gains gold. Gold can be used to buy new towers and upgrade existing ones.

Each level consists of various waves of increasingly growing difficulty. Each time an enemy gets through the entire path, a certain amount of lives is deducted from the player (depending on the enemy type that got through). If the player loses all lives, the game ends. A level is cleared if the player manages to survive through all waves without losing all of his/her lives.

Classes:

Game

Launches the game, assigns the required resources and frees them at the end of the program

Map

Generates the map based on a text file, and stores them into a two dimensional vector (map).
Map consist of tiles/placeholders which can be either balloon route or area where you can place defenses.

Level editor is used to create custom maps

GUI

User interface for the game

GUI_GraphicsItem

Generates graphics for tower, tile and enemy objects.

Defense - Towers

Towers can be placed wherever on the map except on the enemies' path. Towers cost a specific amount of gold to be bought and can be upgraded with gold.

Defense is the base class of all towers. Has parameter size which is equal for all tower types.

Basic_shooter - Shoots balloons in 3s intervals, makes normal damage

Fast_shooter - Shoots balloons in 1s intervals, makes normal damage

Sniper - Always shoots the first balloon, 5x damage, needs 5s to reload

Mortar - Explosion damage, makes 20x damage and can destroy black balloons

Enemy

Red - Basic balloon, has parameters ‘Hp’, ‘speed’, ‘color’

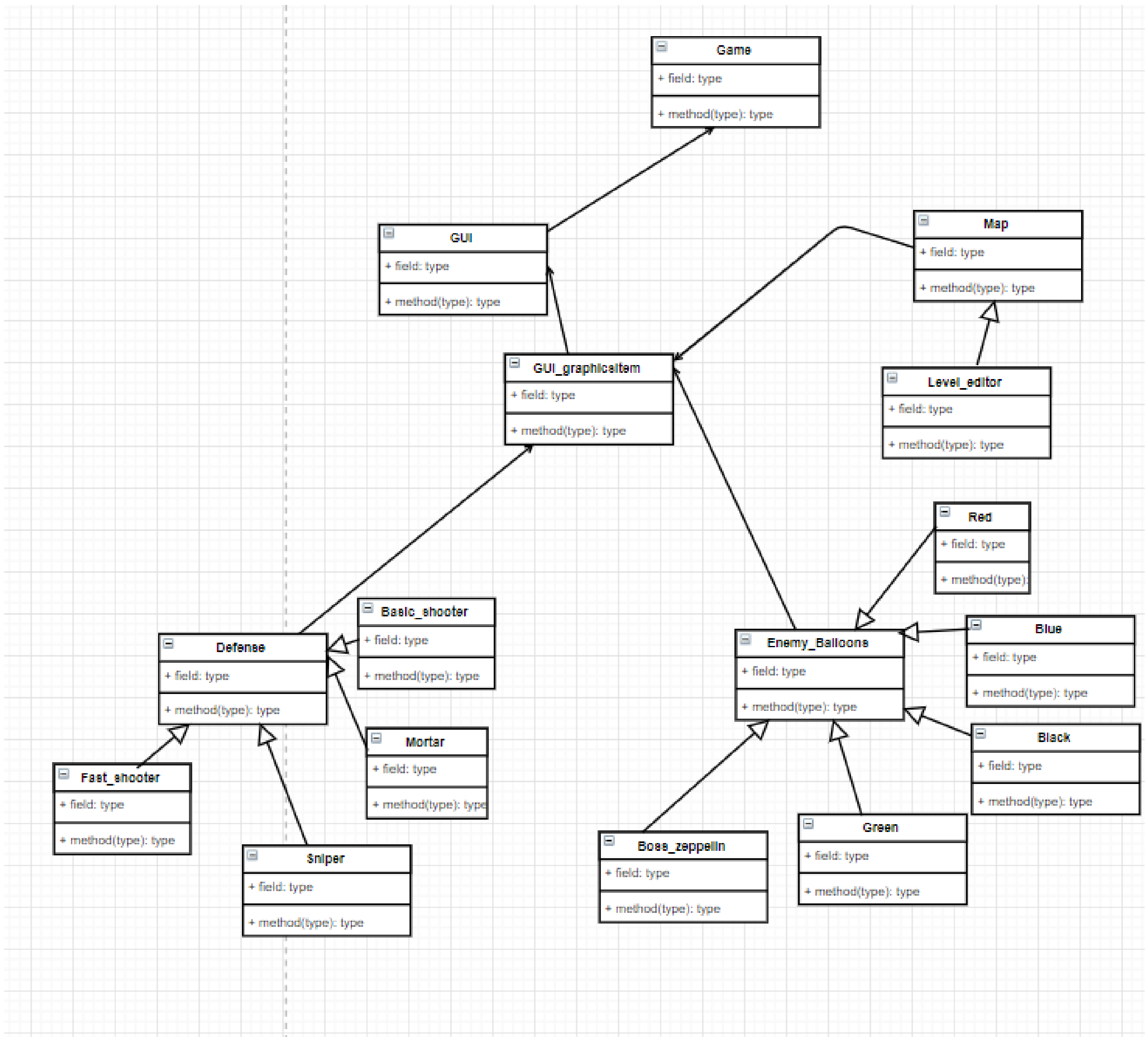
Blue - Basic balloon, has parameters ‘Hp’, ‘Speed’, double hp

Black - Needs explosion, has parameters ‘Hp’, ‘Speed’

Green - Basic balloon, Double speed, Hp

Boss_zeppelin - final round boss, bigger in size, hp is x 100

UML



Game
+ field: type
+ method(type): type
GUI
+ field: type
+ method(type): type
Map
+ field: type
+ method(type): type
GUI_graphicsItem
+ field: type
+ method(type): type
Level_editor
+ field: type
+ method(type): type
Red
+ field: type
+ method(type):
Basic_shooter
+ field: type
+ method(type): type
Defense
+ field: type
+ method(type): type
Enemy_Balloons
+ field: type
+ method(type): type
Blue
+ field: type
+ method(type): type
Mortar
+ field: type
+ method(type): type
Black
+ field: type
+ method(type): type
Fast_shooter
+ field: type
+ method(type): type
Sniper
+ field: type
+ method(type): type
Boss_zeppelin
+ field: type
+ method(type): type
Green
+ field: type
+ method(type): type

Libraries:

- SFML: Simple and Fast Multimedia Library will be used for the main graphic representations and rendering of the game. Due to the object oriented nature of this project we chose to implement it using SFML which is based on classes unlike SDL. Additionally, SFML has a larger documentation than SDL, which will come in handy as we have no experience with C++ game design.
  -Qt: Game graphics can be created with Qt.

Division of work:

Patrick - Algorithms and game engineering

Juho - Testing, Level editor

Eero - Basic classes and graphical design

Niko - Algorithms and map design

Roles will become clearer as the project progresses

Schedule:

Week 1 - Basic classes (game, tile), Map-class - Running main generates the map in a graphical interface

Week 2 - Tower, enemy classes and graphics items - User can deploy towers onto the map

Week 3 - Enemies can move, towers can attack

Week 4 - Algorithms, memory management, difficulty levels

Week 5 - Perfecting graphical interface, additional features, sound effects