Project plan - Kim Amir, Sanni Tähtinen, Meeri Särkkä, Matias Honkasalo

Overview

The game we are programming is a Tower Defence-type game (similar to Bloons TD or Kingdom Rush) where the player attempts to prevent a horde of enemies 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

Tile

Represents a single tile within the map. Can be occupied by tower and enemy objects.

Map

Generates the map based on a text file, generates Tile-objects and stores them into a two dimensional vector. Example (not drawn to scale):

```
######################~~~~~~~E
########~~~~~~~~######~#######
#~~~~~~#~######~######~#######
#~####~~~######~##~~~~~#######
S~############~~~~##########
################################
```

\# - tiles where a tower can be placed
~ - The enemy's path
S - Starting tile of the enemy
E - Ending tile of the enemy (if an enemy gets past this point, lives are deducted from the player)

GUI
User interface for the game

GraphicsItem
Generates graphics for tower, tile and enemy objects.

Tower

Towers can be placed wherever on the map except on the enemies' path. Towers require a specific amount of gold to be bought.
The base class of all towers. Has parameter size which is equal for all tower types.

Class 'Tower' is Inherited by:

Mortar - Area of effect damage within a certain radius. Has parameters 'price', 'range', 'fireRate', 'explosionRadius'

Caveman - Slows all enemies that enter its effective range. Has parameters 'price', 'range'

Quicky - Shoots single enemies quickly. Has parameters 'price', 'range', 'fireRate'

Basic - Shoots single enemies. Has parameters 'price', 'range', 'fireRate'

Enemy

Inherited by:

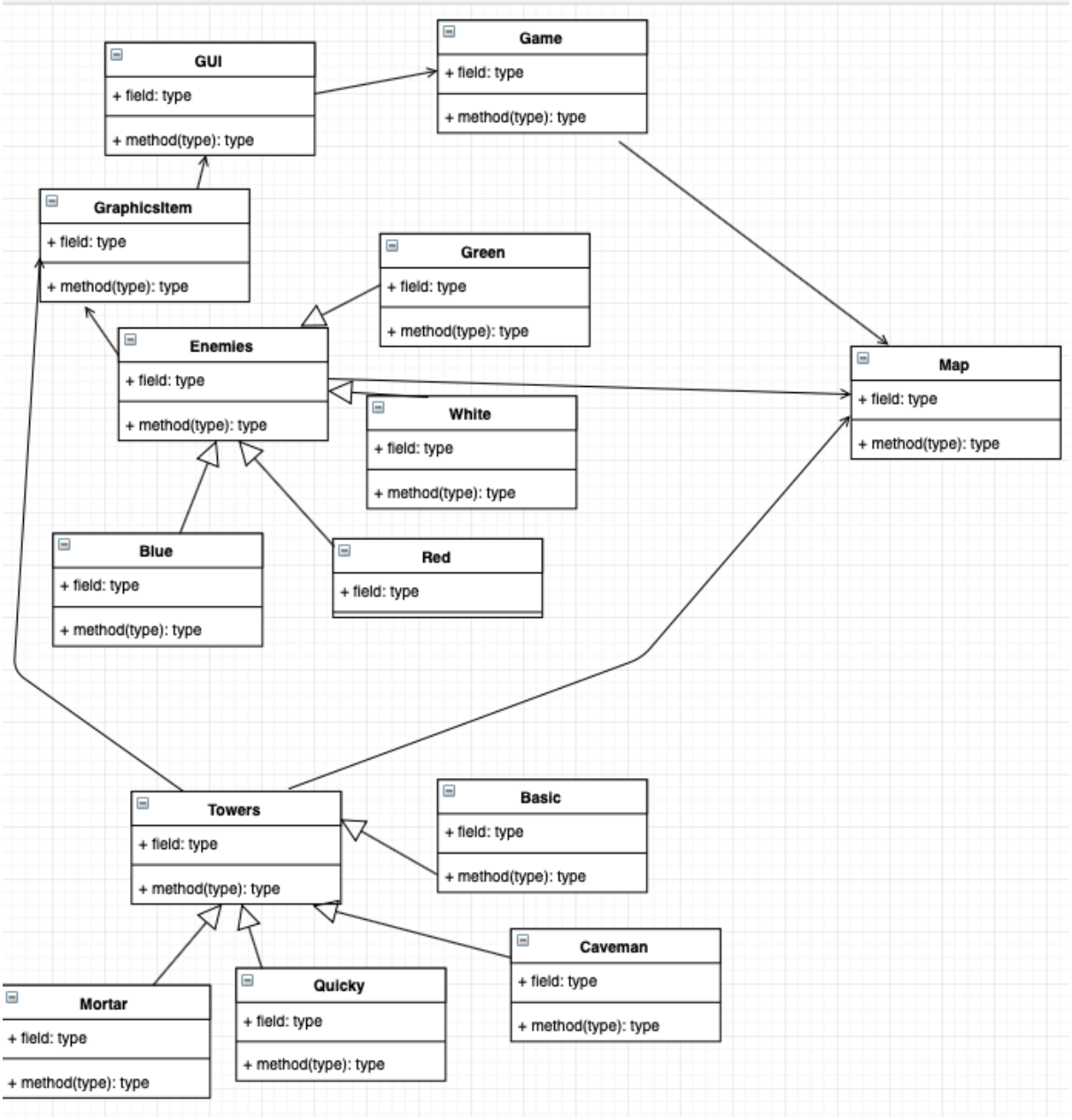
Red - Basic balloon, has parameters 'Hp', 'speed', 'color'

Blue - Basic balloon, has parameters 'Hp', 'Speed'

White- Does not freeze, has parameters 'Hp', 'Speed'

Green- basic balloon but faster, has parameters 'Hp', 'Speed'

UML (missing Tile-class)



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:

Meeri - Class implementation
Kim - Graphical design
Matias - Graphical design
Sanni - Algorithms

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