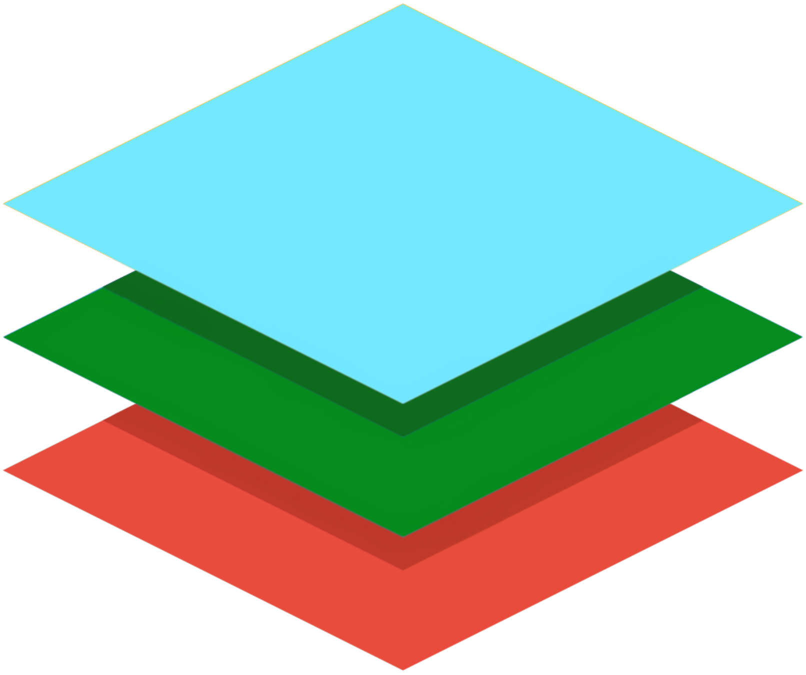
Zürcher Hochschule
für Angewandte Wissenschaften

zh
aw
School of
Engineering



TOWER HOPSCOTCH

Project Design

S. Bösch, N. Eckhart, R. Emberger and P. Meier

Table of Contents

It would be better to insert the whole plan as an overview.

1. Project Management

Current Iteration	**Iteration #5 [Construction Phase]** 20. November 2017 – 01. December 2017
Previous Iteration	**Iteration #4 [Elaboration Phase]** 06. November 2017 – 17. November 2017

1.1. Current stand

With the completion of the fourth iteration, the project now enters the construction phase. All design related parts have been conceptually planned out and summarized in this document. We have already begun with developing some of the basic features such as the game class and loop, the map and layer concepts.

1.2. Review Iteration #4

Iteration #4				
Elaboration Phase		06.11.2017		17.11.2017
No.	**Task**	**Assignee**	**Expected Time [h]**	**Effective Time [h]**
1	Project Management	N. Eckhart	4hrs	3.5hrs
2	Compile all Artifacts from Iteration #3 into design document.	P. Meier	6hrs	8hrs
3	Add additional descriptions required for design document diagrams.	S. Bösch	6hrs	5hrs
4	Define UI Prototype parameters for implementation in the next iteration.	R. Emberger	8hrs	8hrs
5	Create game class with game loop.	N. Eckhart	5hrs	3hrs
6	Create map class that can load a simplified map from a file with only path and non-path tiles. (Only one layer.)	P. Meier, S. Bösch	12hrs	9hrs
7	Update Glossary as needed	Team	2hrs	-

Total?

1.3. Time Comparison

We lost a bit of time in the third iteration due to our design class diagram being more time demanding than expected. We were able to compensate for that in iteration #4 so our effective time expenditures ended up being in acceptable ranges.

1.4. Risk List

Our risk has not undergone any major changes, however since starting coding we have decided that the likelihood of the project complexity exceeding our expectations is fairly low as we now have a general idea of how to implement these features. We have accordingly also lowered the potential impact it might have on our project.

We have also decided on lowering the potential impact that the temporary loss of a developer might have. We did this because we have all features planned out and thus everybody is able to do any task.

A good summary of the risk situation!

Risk Add an ID to each risk so that the risks can be referenced	Counter Measure	Likelihood of occurrence	Impact on Project
Code becomes hard to understand or maintain.	Follow clean code guidelines and have code reviews.	Low	Medium
Project complexity exceeding expectations resulting in increased time consumption.	Thorough analysis and detailed iteration planning.	Low	Medium
Lacking technical knowledge of one or more team members in a certain area resulting in increased time consumption.	Regular sharing of gained knowledge and assign tasks based on the developer's skills.	Medium	Low
Loss of one developer for an extended period due to unexpected circumstances such as illness.	None	Low	Medium

Risk list as of the 18th of November 2017

2. Architecture

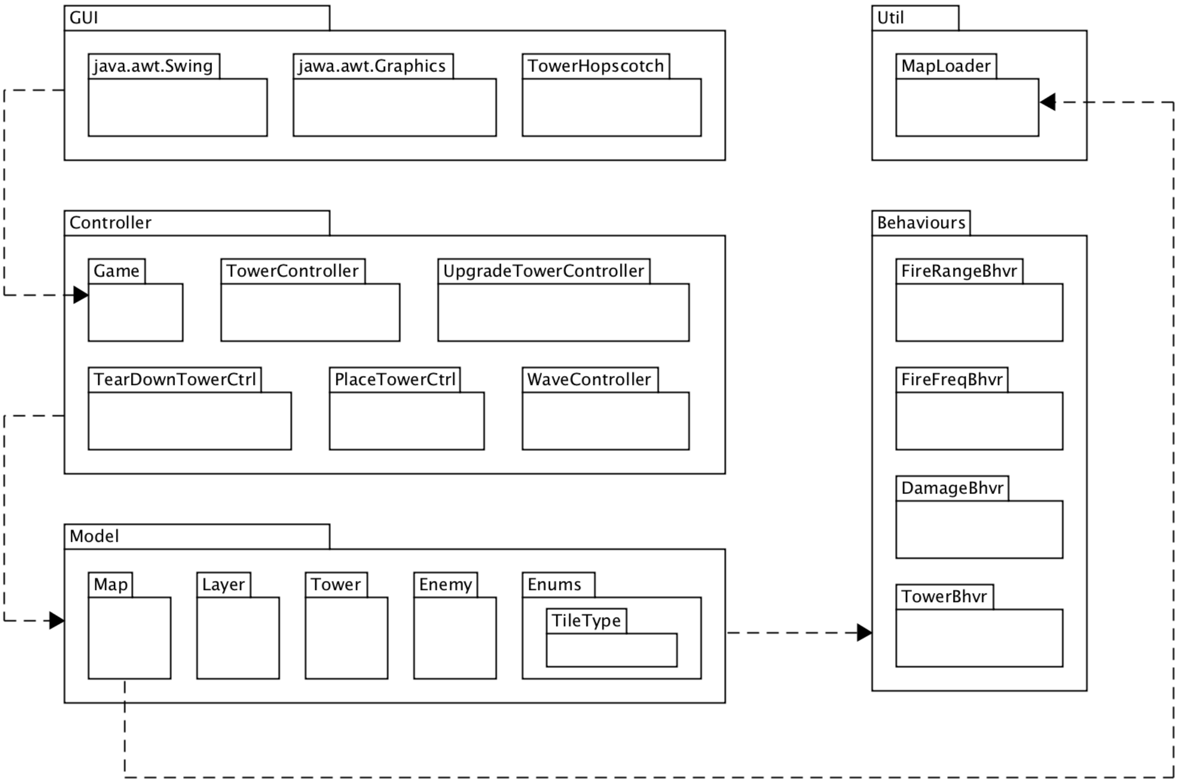
Tower Hopscotch is going to be a standalone desktop application. Data persistence required for features such as custom maps will be accomplished with text files that the program reads from and writes to on the file system. The user interface will be two-dimensional in the game's first iteration.

Design details that should explained after the overall architecture.

2.1. Package Diagram

During development of our design class diagram we decided to follow the model view controller pattern or MVC. By following this pattern we're able to have a user interface than can be easily changed or even replaced. The separation of the business logic and the model classes we can also guarantee higher cohesion and lower coupling.

There is a difference between de GRASP controller and the MVC controller?



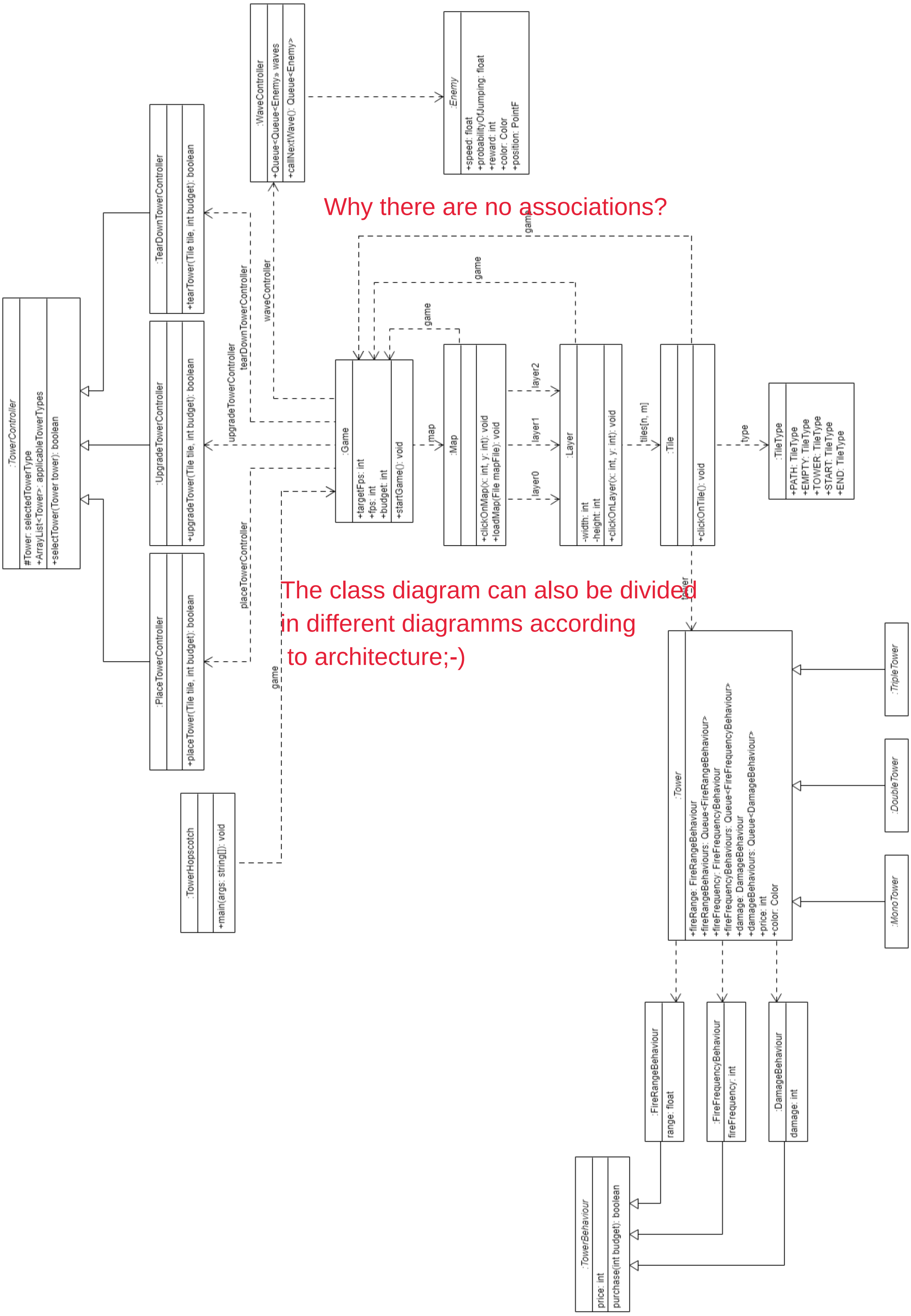
Short description of the main packages is missing!

All important design decisions (such as communication protocols, persistence, etc.) and their reasons should be listed here. This makes it easier to ensure that the architecture is not "overgrown"during further development.

Formal aspects are evaluated and commented on in the exercise design.

3. Design Class Diagram

What part of the logical architecture does it show?



Why there are no associations?

The class diagram can also be divided in different diagramms according to architecture;-)

4. Class Responsibilities

The following table lists all classes and gives a brief summary of the classes responsibilities and the other classes they interact with.

Class Name	Responsibilities	Knows	Does
TowerHopscotch Which of the two is the GRASP controller?	The starting class, which contains the main method and starts the game and the GUI.	The game class	Instantiate the game class and start it
Game	The main controller for the game.	The map ~~class~~	Control the game loop and flow
Map	The class Map is the representor for the whole map, which contains the three layers.	The controllers for the use cases, the layers, deployed enemies and placed towers	Channel the events and hold together the essential parts
Layer	The layers contain the tiles.	The tiles or its area	Channel click and mouse movement events
Tile	Representation of a tile on a layer	The tower that has been placed on it and its own type	Decide the action to take when click or mouse move events get triggered
TileType	An enum which determinates what type of Tile it is. (For example: Path or Tower)	Its own state	Provide a state based type representation
Tower	A base class for Towers with the base logic	Its upgrade possibilities and the enemies it has to attack	Upgrade and attack enemies
Upgrade	Base class for upgrades	Its own price	Provide a purchase method

FloatUpgrade	An upgrade which controls a float value	Its value	Provide a purchase method
IntegerUpgrade	An upgrade which controls an integer value	Its value	Provide a purchase method
WaveController	The WaveController contains a queue of queues of enemies which can get called.	The enemies for every upcoming wave	Deploy enemies for the new wave
Enemy	Enemies are to be destroyed and give a reward or take some of the players lives if they reach the end of the map.	Its own position	Move on a map and try to reach its end
TowerController	This class is the superclass for the other tower control classes.	Which tower has been selected	Allow for selecting a tower
PlaceTowerController	This class is the controller for placing a tower at a specific tile.	The tower that has to be built, the tile it has to be built on	Allow for placing a tower
UpgradeTowerController	This class is the controller for upgrading a tower.	Which tower to upgrade	Allow for upgrading a tower
TearDownTowerController	This class is the controller to tear down a tower.	Which tower to tear down	Allow for tearing down towers

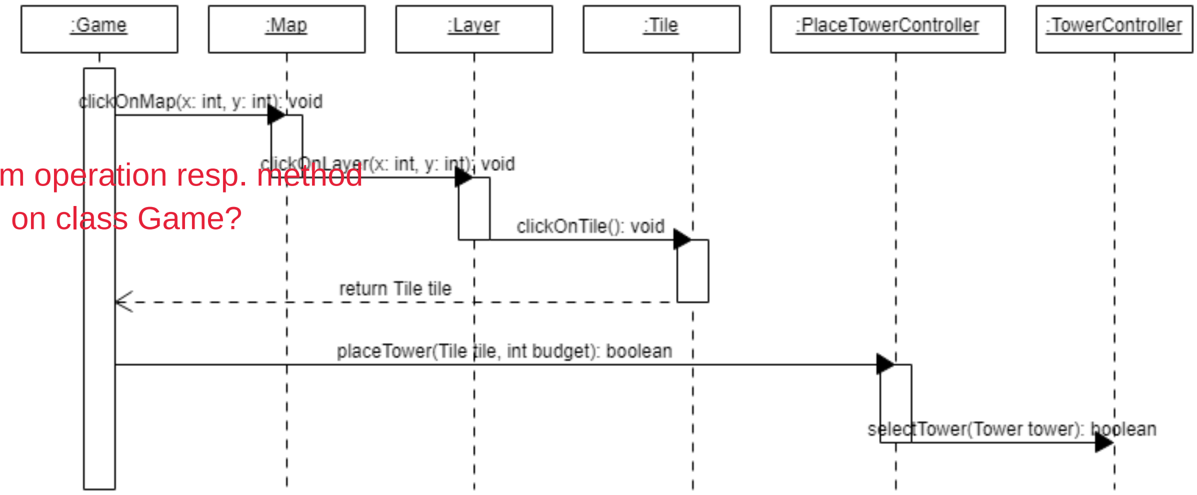
Formal aspects are evaluated and commented on in the exercise design.

5. Interaction Diagram

It would have been good enough to explain the most important system operations to understand the dynamics of the application.

5.1. Sequence Diagrams (System)

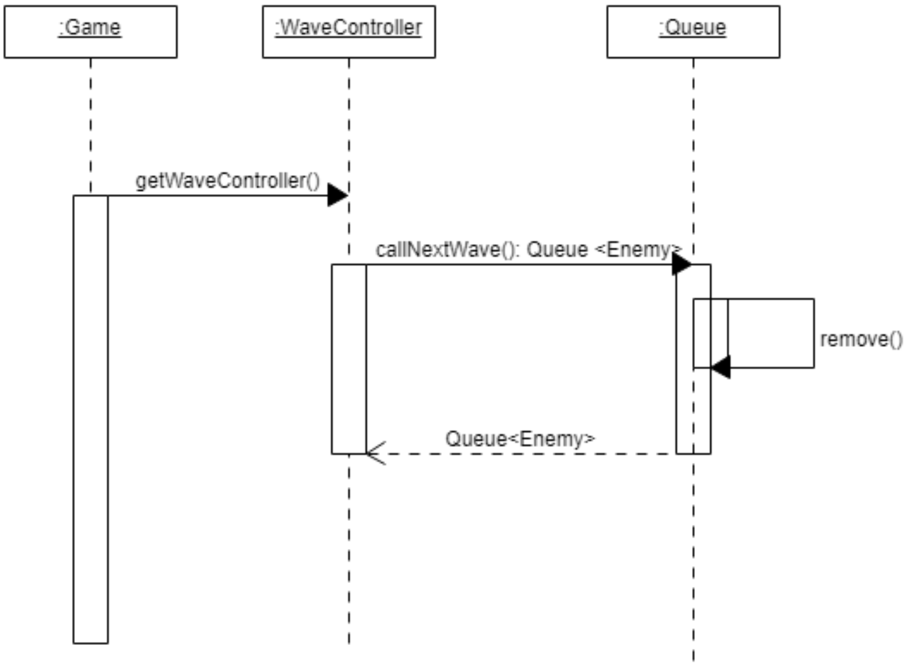
Place Tower


:Game
:Map
:Layer
:Tile
:PlaceTowerController
:TowerController
clickOnMap(x: int, y: int): void
clickOnLayer(x: int, y: int): void
clickOnTile(): void
return Tile tile
placeTower(Tile tile, int budget): boolean
selectTower(Tower tower): boolean

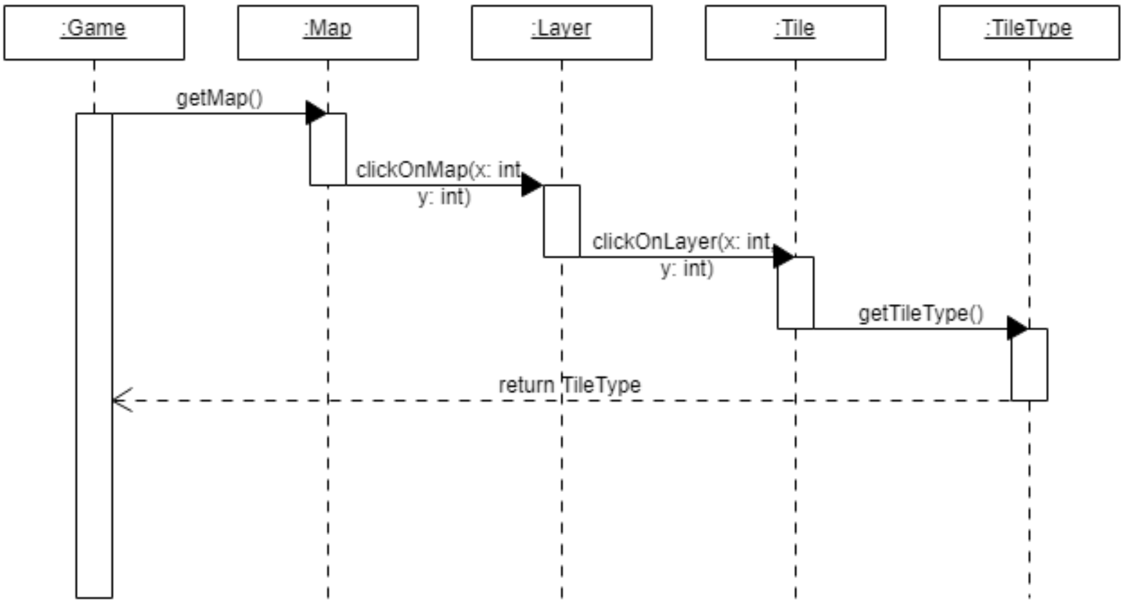
System operation resp. method called on class Game?

A short explanation of the context (use case, scenario) of the system operation in each diagram would be useful.

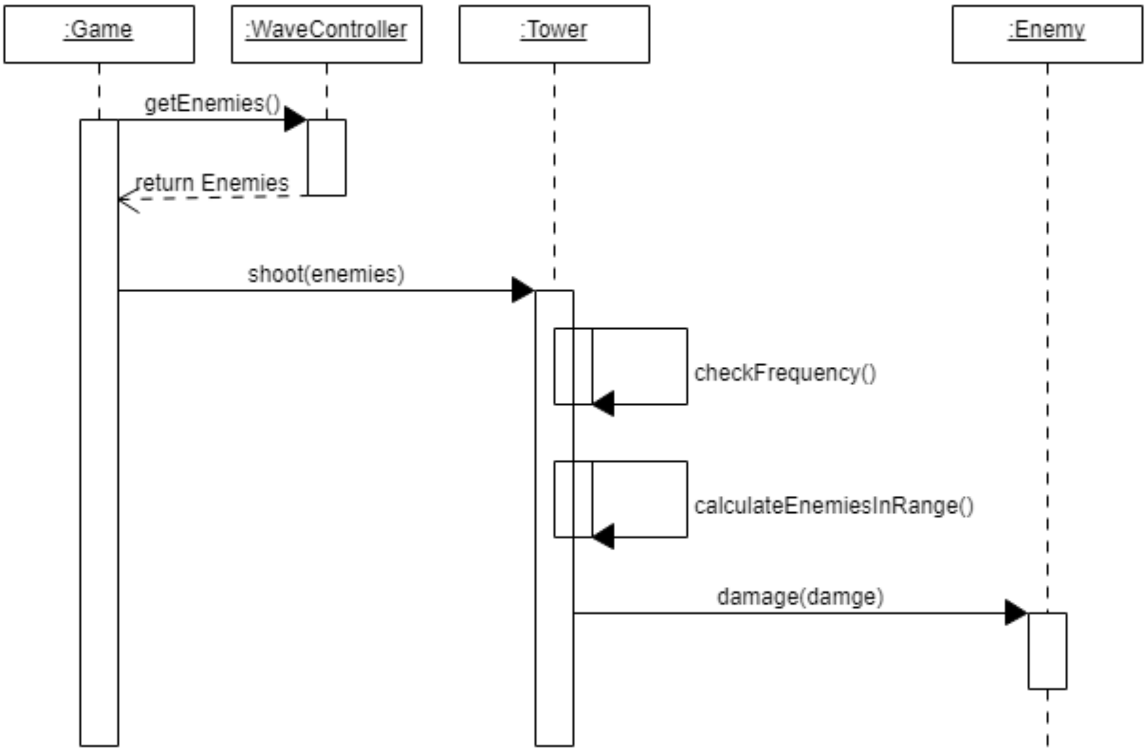
Call next Wave



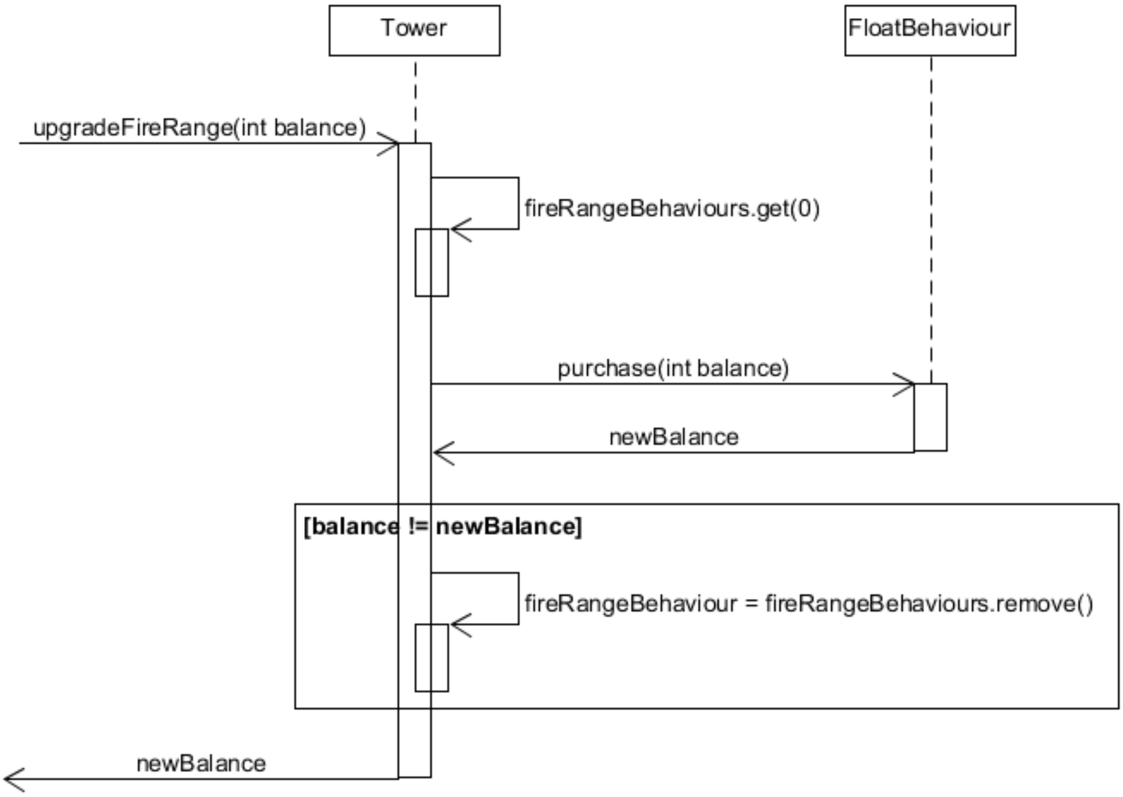
Get Tile Type



Shoot Enemies

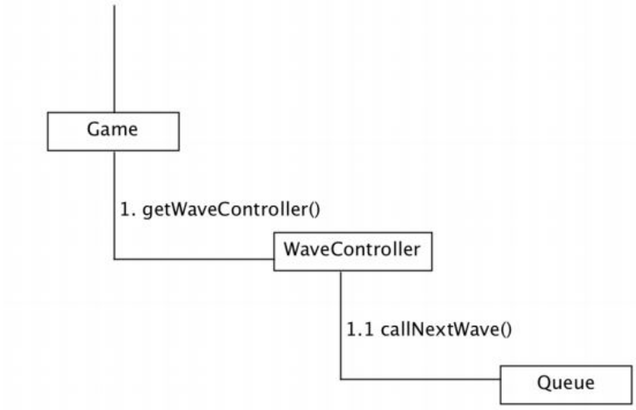


Upgrade Fire Range

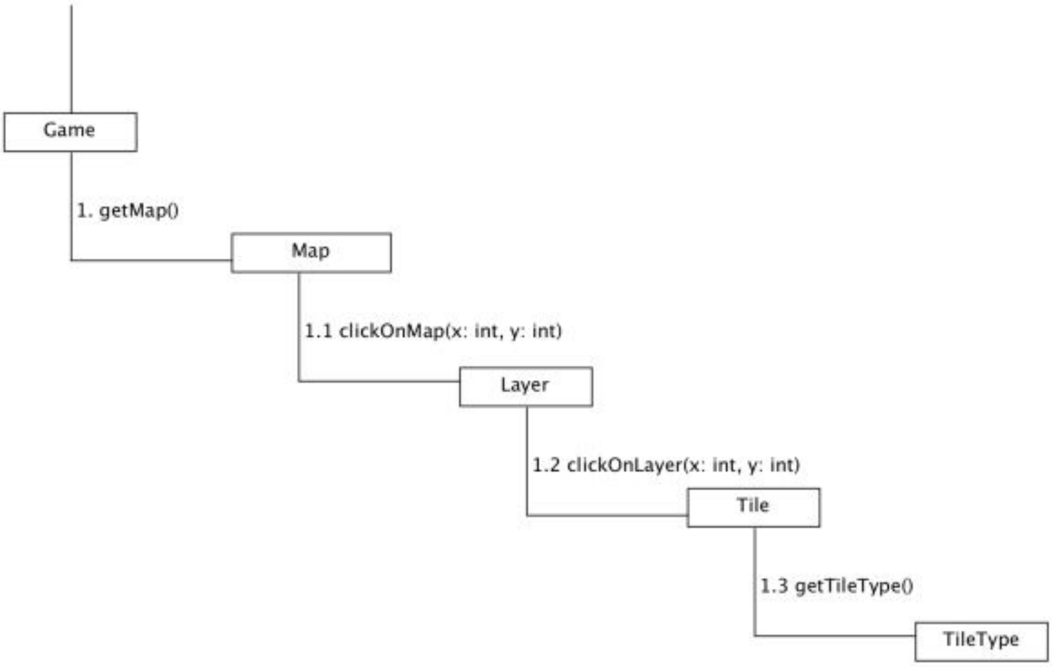

Tower
FloatBehaviour
upgradeFireRange(int balance)
fireRangeBehaviours.get(0)
purchase(int balance)
newBalance
[balance != newBalance]
fireRangeBehaviour = fireRangeBehaviours.remove()
newBalance

5.2. Communication Diagrams

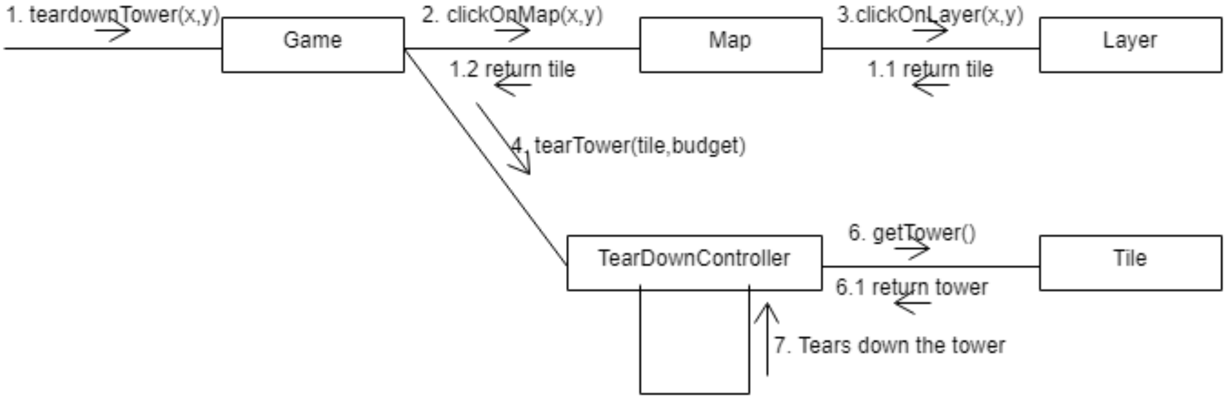
Call Next Wave



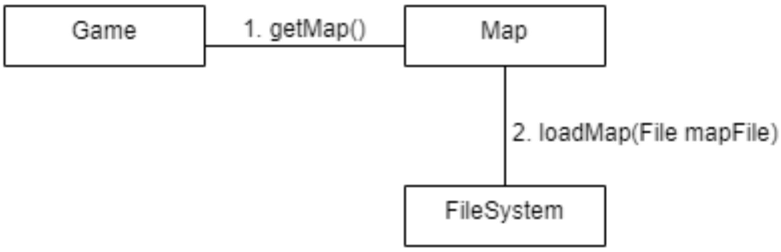
Get Tile Type


Game
1. getMap()
Map
1.1 clickOnMap(x: int, y: int)
Layer
1.2 clickOnLayer(x: int, y: int)
Tile
1.3 getTileType()
TileType

Tear Down Tower



Load Map



In the interaction diagrams it is also possible to comment on the applied GRASP principles/patterns; -)

6. Glossary

Term	Definition
Fortress	The players central structure, the defense of which is the games main objective. Also referred to as Castle.
Tower	Any of a variety of defensive or offensive building created by the player to hinder or destroy incoming enemies.
Wave	A wave refers to a group of enemies. A game encompasses multiple waves that need to be defeated.
Layer	Each map has three layers that simultaneously spawn incoming enemies that may jump between these layers.
Treasury (Budget)	The amount of gold the player has at any given time. Also referred to as budget.
Gold	The currency used in Tower Hopscotch. Gold is obtained by destroying enemies and can be spent on towers and upgrades.
Hit points	Hit points refer to the amount of health an enemy or the players fortress has.

All relevant technical terms should now be defined here!

7. GUI-Design

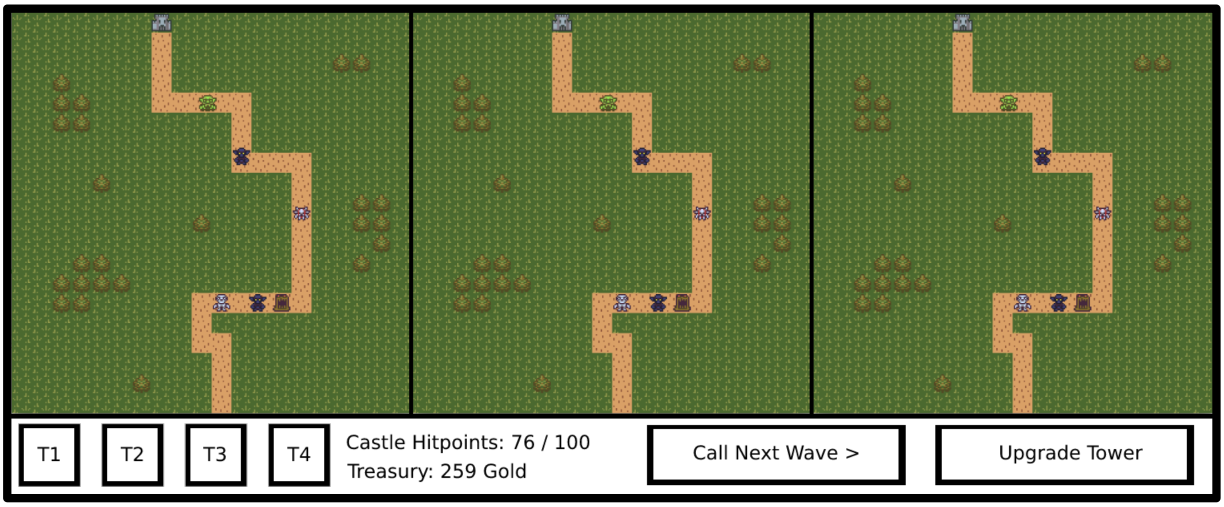
7.1. In-Game Layer Positioning

Much of the screen will be taken up by the three layers. The layers themselves will use different textures to make each of them distinct from the others. The paths as well will vary from layer to layer. The game window will be laid out in a landscape format to fit in all three layers horizontally.

7.2. In-Game User Menu

The bottom part of the game window is taken up by the in-game user menu. It contains information relevant to the player as well as buttons for building and upgrading towers.
On the bottom left there are buttons for each of the towers the player can build. Once clicked, the player can hover his mouse over any part of the map, and there will be a square overlaid at the mouse pointer. This square is green if the tower may be built at that position. If the player does not have the gold, there is already a tower there or he is hovering above a path tile, then the square is red.
If the square is green, then the player can use another left click to build the tower at his mouse's current position. He or she may also at any time cancel his build action by right clicking. Next to the tower buttons, there are two none-interactive labels that show the players remaining hit points as well as his accumulated gold.

On the bottom right, there are two large buttons that are both not clickable by default. The call next wave button gets enabled once all enemies of the current wave have been destroyed. Clicking on it prompts the next wave to begin. The tower upgrade button gets enabled when the player has selected a tower by left clicking it on the map. He also has to have enough gold to purchase the upgrade. The cost of an upgrade is displayed on the button itself when a tower is selected. When purchased, the tower receives the upgrades effects and the cost is deducted from the players treasury.



Castle Hitpoints: 76 / 100
Treasury: 259 Gold