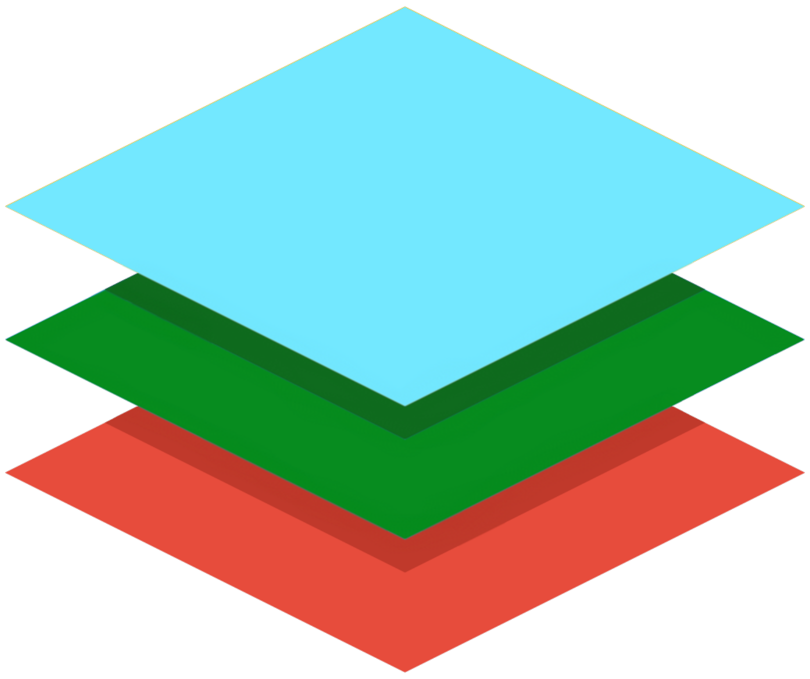
Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering



TOWER HOPSCOTCH

Project Analysis

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

Table of Contents

1. Project Management

Current Iteration	**Iteration #2 [Elaboration Phase]** 16. October 2017 – 27. October 2017
Next Iteration	**Iteration #3 [Elaboration Phase]** 30. October 2017 – 10. November 2017

1.1. Previous Activities

Phase	It	Activity	Assignee	Expected Time [h]	Effective Time [h]
Inception	1	Create Project Outline	Team	10hrs	11.5hrs
		Identify Use Cases	Team	8hrs	6hrs
		First architecture determined	Team	4hrs	4hrs
		Setup development environment	Team	4hrs	6hrs
Elaboration	2	Detailed analysis and formulation of all Use Cases	Team	16hrs	16hrs
		Domain Model defined	emberrap	2hrs	1.5hrs
		Create Project Analysis document	Team	16hrs	14hrs

1.2. Time Comparison

Our effective time expenditures were in expected ranges so we did not need to take any actions to compensate for lost time.

1.3. Risk List

Risk	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	High
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	High

2. Use Cases

The following use cases are prioritized from top to bottom. The biggest use cases "Play Game" was split up into smaller use cases to keep it organized and clear, because it's the biggest and most time consuming one.

Name	Details	Priority
1. Play game		High
1.1. Place tower	UC 1.1 – 1.6 are all use cases extending UC 1. Play Game.	High
1.2. Upgrade tower		Medium
1.3. Tear down tower		Medium
1.4. Call next wave		High
1.5. Pause game		Low
1.6. Unpause game		Low
2. Create map		High
3. Edit map		Medium
4. Import map		Medium
5. Export map		Medium

2.1. UC 1 – Play game

Primary Actor: Player

Stakeholders and Interests:

- Player: Wants a stable framerate with short load times and no crashes to interrupt his experience.

Preconditions: The player has selected a map and started a new game on it.

Post conditions: The player has either defeated all enemy waves and won or his central structure has taken a critical amount of damage and has been destroyed resulting in the player losing the game.

Main success scenario:

1. The player has started a new game and the map is loaded.
2. The player spends his starting currency on building up his defenses.
3. The player clicks on begin, indicating they are done preparing and ready for the first enemy wave.
4. Incoming enemies are destroyed by the defensive structures and the player spends the money gained on new defenses.

Step four repeats itself so long as there are enemies remaining in the current wave and the main structure has not been destroyed.

5. When all enemies of the current wave have been destroyed, there is an indication that the next wave will be incoming soon.
6. The player has a set amount of time to improve his fortifications before the next wave begins automatically.

Steps four through six are repeated while the last wave has not been defeated and the main structure has not been destroyed.

7. The player has defeated the last wave and a message is displayed indicating that they have won the game.
8. The game automatically returns to the main menu after the message has disappeared.

Extensions:
*a. At any time, the game crashes:

The game shuts down and the player must restart the game if he wishes to continue playing. Any game progress will not be saved.

*b. The player closes the game window:

The current game ends and no game progress is saved.

*c. The Player pauses the game:

The ongoing game is paused, and a menu is brought up allowing the player to leave to leave the game or to resume it.

Special Requirements:

- Windows or Mac computer with Java 8.
- Computer with mouse or a touch display.

Frequency of Occurrence: However often it is initiated by player.

2.2. UC 1.1 – Place tower

Main success scenario:

1. The player selects a tower type to build. The selection is valid if the chosen tower type is unlocked and the players budget is not lower than the costs of the chosen tower type. If the selection is not valid, then the scenario ends here and can be restarted at any time.

2. The player selects a tile on the map to build the tower. The selection is valid if the chosen tile is neither a path tile nor is already occupied by another tower and can fit the tower in if the tower is multilayered (Towers spanning two layers can only be placed on the lower 2 layers. Towers spanning all three layers can only be placed on the lowest layer). If there is no tile available which could be selected, the scenario ends here and can be restarted at any time.
3. The tower gets built on the selected tile and its cost gets subtracted from the player’s budget.

2.3. UC 1.2 – Upgrade tower

Main success scenario:

1. The player can upgrade a tower so that the tower makes more damage to the enemies. To upgrade a tower the player needs enough amount of money.
2. By clicking on an existing tower, a menu shows up with different upgrade possibilities like double damage, shoot faster etc. If the player doesn’t have enough money for a specific upgrade, the upgrade is still shown in the menu, but grayed out.
3. By clicking on the desired upgrade, the player loses money based on the cost of the upgrade and the tower gets upgraded.
4. The player closes the upgrade menu and is back in action.

2.4. UC 1.3 – Tear down tower

Main Success Scenario:

1. The player clicks on the tower he wants to tear down.
2. The tower menu opens in which he can upgrade or tear down the tower.
3. The player clicks on the tear down button.
4. The tower gets destroyed and the player gets a fractional amount of the money he invested in building the tower.
5. The tower menu gets closed.

2.5. UC 1.4 – Call next wave

Main success scenario:

1. When all enemies are defeated from the previous wave and the player has modified his defense, he can click on the button "Call Next Wave". This results with the beginning of the next wave.

Alternate Scenario

When the player has defeated the last wave of enemies, the player can't call a next wave because he has won the game.

2.6. UC 1.5 – Pause game

The player can pause an ongoing game to stop all ongoing actions until resumed.

2.7. UC 1.6 – Unpause game

The player can resume an ongoing game from a paused state to continue playing.

2.8. UC 2 – Create map

The player can create custom maps for him to play on.

2.9. UC 3 – Edit map

The player can edit his custom maps.

2.10. UC 4 – Import map

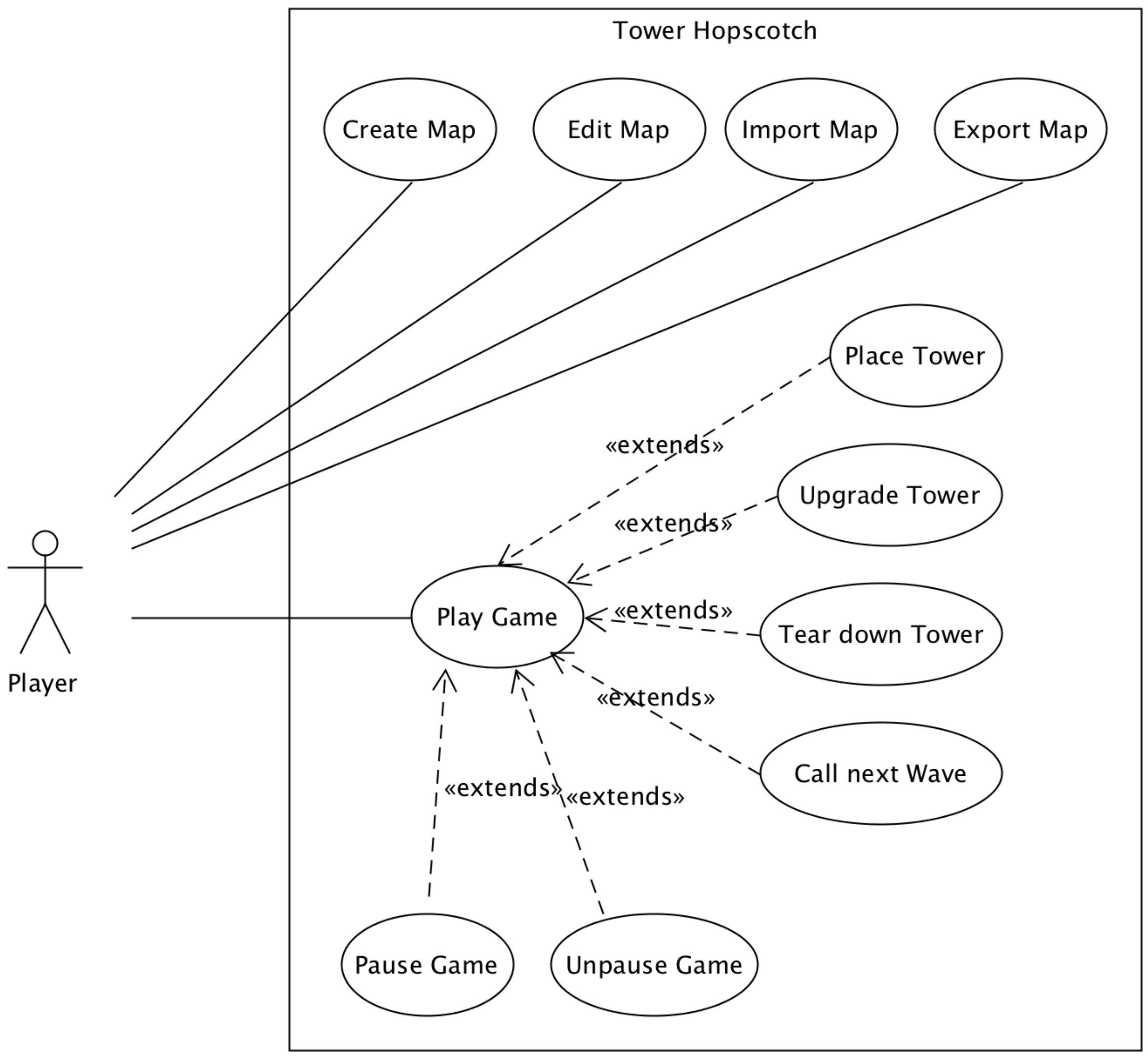
A map can be imported from a chosen directory to be used or edited in the game.

2.11. UC 5 – Export map

The player clicks on the "Export Map" button and the program exports a selected map to a chosen file location.

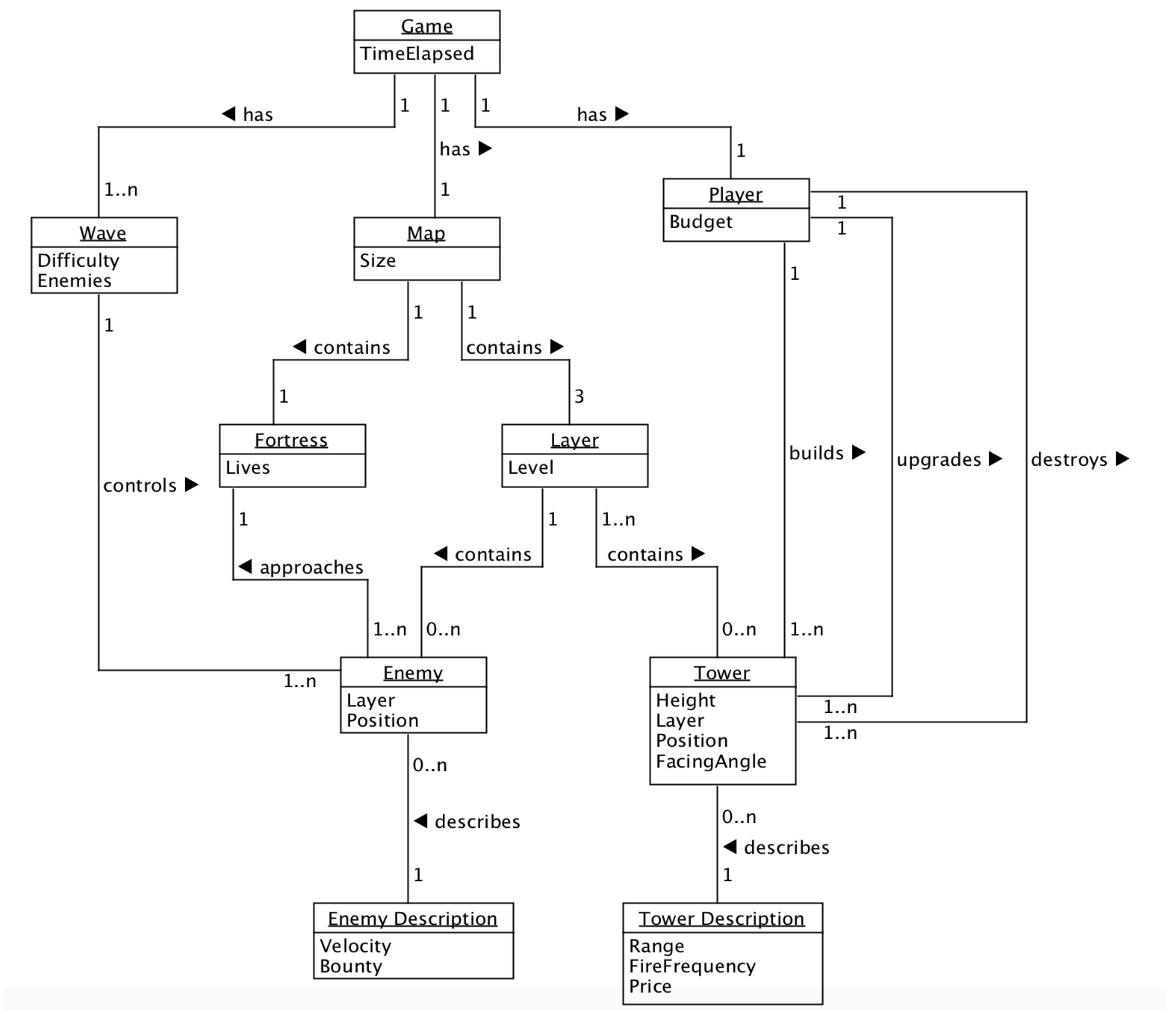
3. Use Case Diagram

As briefly mentioned above, most of the use cases are extensions of the primary “Play Game” use case. Everything related to the map on the other hand stands alone.



4. Domain Model

4.1. Domain Model Diagram



4.2. Domain Model Description / Problem Domain breakdown

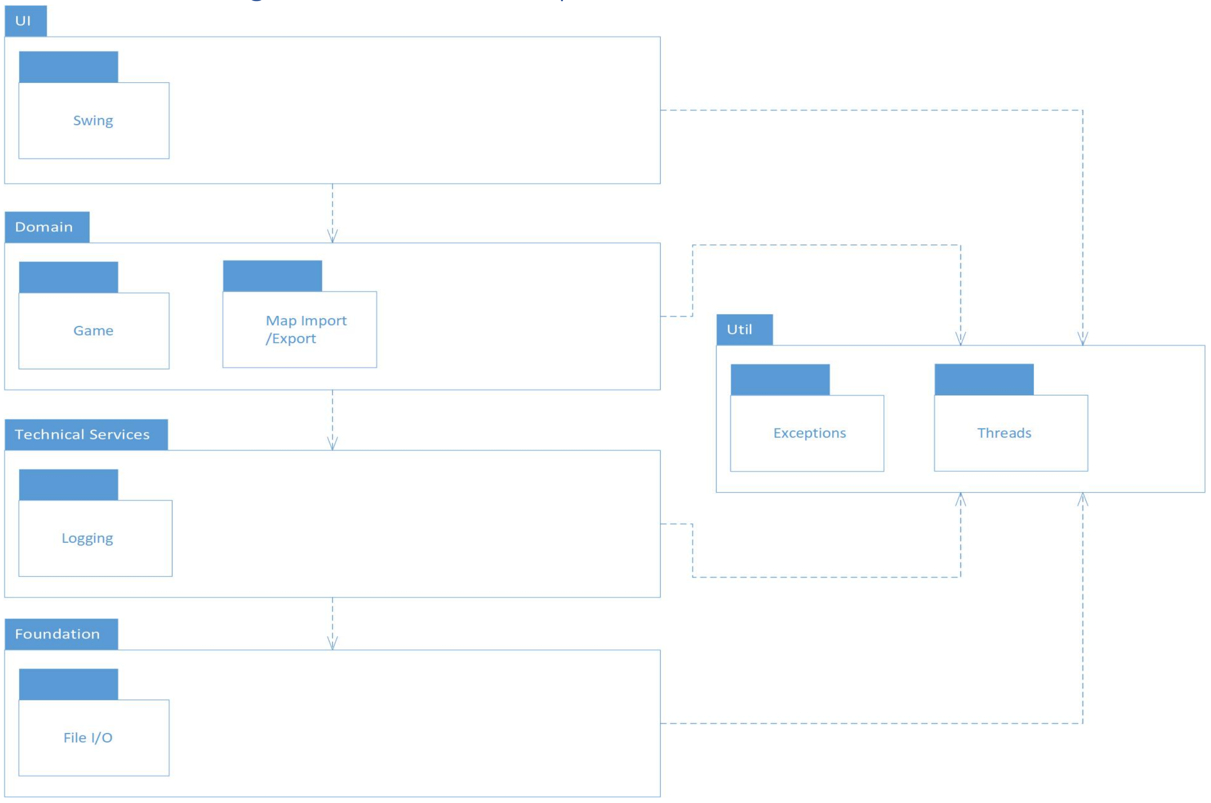
The central problem domain in our application is the game concept itself. It ties everything together as shown in the diagram. The individual parts are explained here:

- **Player:** The Player has a Budget, with which he can build or upgrade towers. He can also destroy towers to get a fractional amount of money back.
- **Map:** The map consists of three layers and a fortress.
- **Fortress:** The Fortress is the players object to defend. The player loses the game, if the fortress has not any health points anymore.
- **Layer:** On a layer are different paths which lead the enemies to the fortress.
- **Tower:** The Player can place towers. The towers attack the enemies and try to stop them before the reach the fortress.
- **Wave:** A Wave contains a specific number of enemies. The wave ends when all enemies are killed or if they are not killed in time and reached the Fortress.
- **Enemy:** The Enemies try to reach the Fortress. If this happens, the fortress will lose health points.

5. First 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.

5.1. First Packege Architecture Concept



5.2. Programming Language

As Programming Language, we have chosen Java, because the Team has the most common experience in Java. The only framework that will be used is JavaFX to help with the user interface.

6. Additional Specifications

6.1. Non-functional Requirements

Number	Description	Category
AS1	The game is supported on Java version 8 and above.	Functionality
AS2	The game may be played with a mouse or a touch display.	Functionality
AS3	The program runs on both Mac and Windows computers.	Functionality
AS4	The application does not crash under normal circumstances.	Reliability
AS5	Starting the game and loading a map should never take more than a few seconds.	Performance
AS6	The game should remain at a constant 60 frames per second while playing.	Performance
AS7	The code meets general quality standards so that it can be maintained or worked on by new developers without effort.	Supportability

6.2. Tower Hopscotch Game Rules

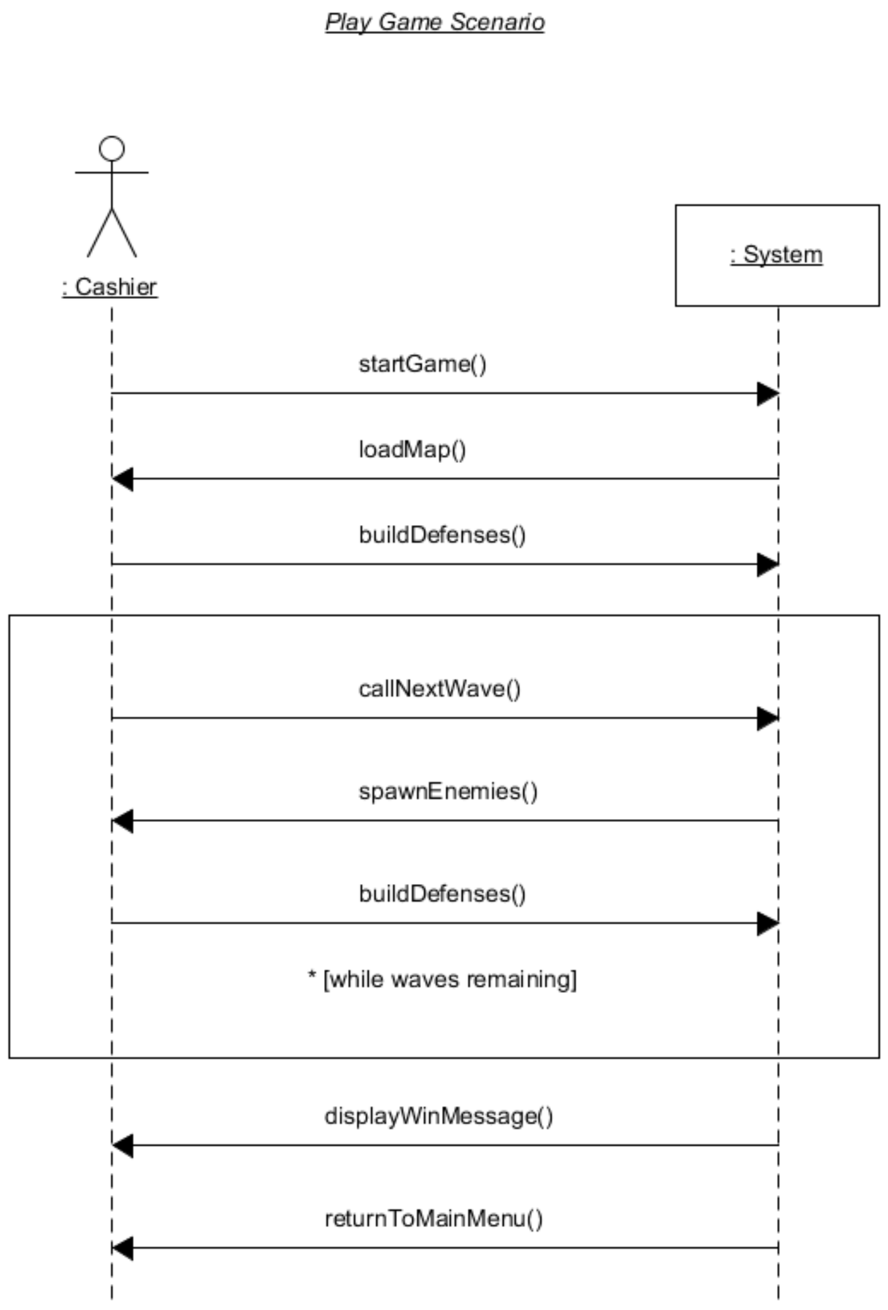
- Whenever an enemy reaches the opposite end of a layer it came from, the proverbial finish line, then a set amount of health points is deducted from the player’s fortress. The amount deducted depends on the enemy in question.
- There are set paths across each layer from one end to the other that can split up but cannot have dead ends.
- Towers and other defensive or offensive structures may not be placed on the path, but may be placed anywhere else.
- The only source of income is the bounty of destroyed enemies. For this reason, each game will start out with a base amount of currency to build the first few towers.

6.3. Tower Hopscotch Win / Lose conditions

- The player loses when enough enemies gotten through that his or her fortresses health points have dropped to zero.
- The Player wins if all enemies of the last wave have been destroyed.

7. Sequence Diagram

The following is a sequence diagram describing the primary use case “UC1: Play Game” as well as most of its extending use cases from UC1.1 to UC1.4.



8. Glossary

Term	Definition
Fortress	The players central structure, the defense of which is the games main objective.
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.