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

Tower Hopscotch

Final Report



Table of Contents

[1. Outline 4](#_Toc500432478)

[1.1. Starting Position 4](#_Toc500432479)

[1.2. Idea 4](#_Toc500432480)

[1.3. Customer Value 4](#_Toc500432481)

[1.4. State of the Art – Competition Analysis 4](#_Toc500432482)

[1.5. Main process 4](#_Toc500432483)

[1.6. Economics 5](#_Toc500432484)

[2. Analysis 6](#_Toc500432485)

[2.1. Domain Model Diagram 6](#_Toc500432486)

[2.2. Use Cases 7](#_Toc500432487)

[UC 1 – Play game 7](#_Toc500432488)

[UC 1.1 – Place tower 8](#_Toc500432489)

[UC 1.2 – Upgrade tower 8](#_Toc500432490)

[UC 1.3 – Tear down tower 9](#_Toc500432491)

[UC 1.4 – Call next wave 9](#_Toc500432492)

[UC 1.5 – Pause game 9](#_Toc500432493)

[UC 1.6 – Unpause game 9](#_Toc500432494)

[UC 2 – Create map 9](#_Toc500432495)

[UC 3 – Edit map 9](#_Toc500432496)

[UC 4 – Import map 9](#_Toc500432497)

[UC 5 – Export map 9](#_Toc500432498)

[2.3. Use Case Diagram 10](#_Toc500432499)

[3. Design 11](#_Toc500432500)

[4. Implementation 12](#_Toc500432501)

[5. Results 13](#_Toc500432502)

[6. Appendix 14](#_Toc500432503)

[6.1. Current stand 15](#_Toc500432504)

[6.2. Review Iteration #4 15](#_Toc500432505)

[6.3. Time Comparison 15](#_Toc500432506)

[6.4. Risk List 16](#_Toc500432507)

[7. Architecture 17](#_Toc500432508)

[7.1. Package Diagram 17](#_Toc500432509)

[8. Design Class Diagram 18](#_Toc500432510)

[9. Class Responsibilities 19](#_Toc500432511)

[10. Interaction Diagram 21](#_Toc500432512)

[10.1. Sequence Diagrams (System) 21](#_Toc500432513)

[Place Tower 21](#_Toc500432514)

[Call next Wave 21](#_Toc500432515)

[Get Tile Type 22](#_Toc500432516)

[Shoot Enemies 22](#_Toc500432517)

[Upgrade Fire Range 23](#_Toc500432518)

[10.2. Communication Diagrams 24](#_Toc500432519)

[Call Next Wave 24](#_Toc500432520)

[Get Tile Type 24](#_Toc500432521)

[Tear Down Tower 25](#_Toc500432522)

[Load Map 25](#_Toc500432523)

[11. Glossary 26](#_Toc500432524)

[12. GUI-Design 27](#_Toc500432525)

[12.1. In-Game Layer Positioning 27](#_Toc500432526)

[12.2. In-Game User Menu 27](#_Toc500432527)

# Outline

## Starting Position

The market for video games is currently in a continuous state of growth with an estimated increase of almost eight percent in 2017. [1] While many start-ups and small businesses are entering the business to find success, the demand for original and entertaining games is still growing. While all this new content is being released, the strategy game subgenre and its sizeable and enthusiastic fan base finds itself somewhat neglected. We believe that this is something that can and should be changed.

## Idea

The plan is to develop a tower defence strategy game for desktop PCs. The primary objective will be to defend a central structure over multiple landscapes against waves of enemy units. This is accomplished by building defensive structures to divert, impede and destroy the incoming foes. Each enemy that is stopped will provide the player with a set amount of currency that allows him to further improve his defences.

In our game the player must manage multiple different maps at once, all of which have enemies that move towards the central structure. These maps can be either automatically generated or the player may design his own levels and play on them.

## Customer Value

* The customer will be provided with an entertaining and unique strategy game with multiple original twists.
* The game will run all major operating systems that support a Java environment.
* It will feature increased challenge by having the player manage multiple maps at once.
* The customer will be able to choose to create his own unique maps for improved replayability and enjoyment.

## State of the Art – Competition Analysis

There are many tower defense games which gained much popularity over the years. The most successful game released 2009 and is still a name people know: Plants vs. Zombies. [2] It uses a small by grid where the enemies approach from one side and the “towers” are placed on the grid to fight against them. The enemies can destroy the towers when they’re near enough. Killed enemies give the player money to upgrade or place new towers.

## Main process

The main process is the player who plays the game

* The player starts the game, which he already has installed.
* He selects the standard mode.
* The player receives a specific amount of game currency.
* The player spends the money to place towers.
* The enemies are coming in waves.
* The towers try to shoot down the enemies.
* A tower kills an enemy and the player receives money, which he can spend again on towers.
* An enemy reaches the players base and the base loses some health points.
* The player survives all waves and wins the game.

## Economics

We'll each work for 120 hours, which accumulates to a 480 hours project. Our goal is to pay ourselves about 40 francs per working hour. Together with the marketing campaign and a reasonable time buffer for testing and unexpected bug issues or changes, the project will cost about 35’000 francs.

The game will be sold on Steam (biggest online video game store) for 10 francs. Which means that at least 3500 copies need to be sold to break even. Steam has a multi-million audience and is a great platform to sell games on. We estimate to break even after about 4 months.

The first year will probably be the most successful, because the typical gamer craves innovation and new experiences. A 50% decrease in sales per year is a realistic vantage point. After 5 years, the estimated income will be around 220’000 francs minus the first investment.

# Analysis

## Domain Model Diagram



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.

## 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. Place tower | UC 1.1 – 1.6 are all use cases extending UC 1. Play Game. | | High |
| * 1. Upgrade tower | Medium |
| * 1. Tear down tower | Medium |
| * 1. Call next wave | High |
| * 1. Pause game | Low |
| * 1. Unpause game | Low |
| 1. Create map |  | | High |
| 1. Edit map |  | | Medium |
| 1. Import map |  | | Medium |
| 1. Export map |  | | Medium |

### 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.*

1. When all enemies of the current wave have been destroyed, there is an indication that the next wave will be incoming soon.
2. 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.*

1. The player has defeated the last wave and a message is displayed indicating that they have won the game.
2. 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.

### 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.

### 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.

### 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.

### 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.

### UC 1.5 – Pause game

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

### UC 1.6 – Unpause game

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

### UC 2 – Create map

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

### UC 3 – Edit map

The player can edit his custom maps.

### UC 4 – Import map

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

### UC 5 – Export map

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

## 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.



# Design

## Architecture

## Design Class Diagram

## Interaction Diagram

## Design Decisions

# Implementation

## Tests

# Results

## Goal Summary

## Open Points

## Possible Future Developments

# Appendix

## Project Management

## Bibliography

|  |  |
| --- | --- |
| [1] | E. McDonald, “NEWZOO,” 20 April 2017. [Online]. Available: https://newzoo.com/insights/articles/the-global-games-market-will-reach-108-9-billion-in-2017-with-mobile-taking-42/. |
| [2] | EA. [Online]. Available: https://www.ea.com/en-gb/games/plants-vs-zombies/plants-vs-zombies-2. [Zugriff am 02 10 2017]. |

## 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. |

## Manual