Requirements and Analysis Document for Alchemy-defense

Willem Brahmstaedt, Felix Jönsson,
Johan Lindén, Valdemar Stenhammar

October 24, 2021
Version 1.0

1 Introduction

The purpose of this application is to create an entertaining tower defense game that includes dynamic pathfinding. The map is an emtpy grid and the player has to create a path that enemy foes will take and this can be done in many different unique ways. This not only differentiates the game from other tower defense games but also makes it more challenging.

1.1 Definitions, acronyms, and abbreviations

- Foe: Tries to reach a set end goal to inflect damage on the player.
- Spawn: Instantiate a new foe.
- Tower: Placed by the player upon the map to hinder the foes from reaching a set end position, where upon reaching the foe will inflict damage to the players hit points.
- Board: The main stage of the game. Uses a underlying 2D-grid which can contain and update both enemies and towers.
- Tile: Each cell in the 2D-grid represents a tile that can hold a tower or an enemy.
- Pathfinding: Plotting and generating the (shortest) route possible given a speciefied search condition.
- Breadth First Search: A simple pathfinding algorithm.

2 Requirements

2.1 User Stories

2.1.1 Diagram and definitions

1. Epics. A user story that is too large to fit into a sprint.
2. Critical. Required to be implemented in order for the application to function.
3. Essential. Things that are essential for the game to be enjoyable for a player, but not required for the application to run.
4. Features. Features that would be nice to have given there's enough time to implement them.

Critical		Essential		Features	
Create game board	HDM01	Menu screen	HDM06	Sell tower from game board	HDM12
Damage enemies	HDM02	Play-button	HDM07	Pause game session	HDM13
Spawn enemies	HDM03	Gold	HDM08	Change settings in menu	HDM14
Access to towers	HDM04	Health	HDM09	Quit game from start menu	HDM15
Grid	HDM05	End goal	HDM10		
		Enemy healthbar	HDM11		

Figure 1: Green cells are completed

2.1.2 Epics

- As a player I want to be able to begin a game to start playing.
- As a new player I'd like to be introduced to the concept of the game in a seamless way.
- As a player I'd like to have some time at the start of the game to configure the board.
- As a player I want the game to be visually pleasing.

2.1.3 Critical

Story Identifier: HDM01

- Story Name: Create game board
- Implemented: Yes

Description
As a player I'd like to have a game board to play on.

Confirmation:

Functional

- Can the player get a visual representation of the board?
- Can the player interact with the graphical user interface through mouse clicks?

Non-functional

- Can the code of the board handle different scaling options?

Story Identifier: HDM02

- Story Name: Damage enemies
- Implemented: Yes

Description
As a player I would like to have enemies that are able to receive damage from my towers.

Confirmation:

Functional

- Can the tower deal damage to the enemy?

Non-functional

- Is it possible to scale the damage dealt to an enemy?

Story Identifier: HDM03

- Story Name: Spawn enemies
- Implemented: Yes

Description
As a player I would like enemies to spawn in game.

Confirmation:

Functional

- Does the player get a visual representation when the game spawns enemies?
- Can the player decide when to spawn enemies?
- Can the enemies spawn in the form of a wave?

Non-functional

- Can there be different enemies?
- Do the enemies spawn in a fixed pattern?

Story Identifier: HDM04

- Story Name: Access to towers
- Implemented: Yes

Description

As a player I'd like to have access to the towers to be able to place them on the board.

Confirmation:

Functional

- How does the player place the tower on the game board?
- Can the player see where it can place towers on the game board?
- Can the player place a tower at any point in the game?

Non-functional

- Can the player place a tower at any point in the game?
- Is it possible to introduce new towers?

Story Identifier: HDM05

- Story Name: Grid
- Implemented: Yes

Description
As a player I'd like to see a grid of where one can place the towers to make that process more forgiving.

Confirmation:

Functional

- Do the game present a visual representation of the grids with clear boundaries?
- Does the game register the mouse clicks in a correct way?

2.1.4 Essential

Story Identifier: HDM06

- Story Name: Menu screen
- Implemented: No

Description
As a player I would like to be greeted by a Menu Screen to be able to start playing the game.

Confirmation:

Functional

- Does the menu screen open when the player opens the program?
- Where can the player go from the menu screen?

Non-functional

- What does the menu screen look like?

Story Identifier: HDM07

- Story Name: Play-button
- Implemented: Yes

Description
As a player I would like to be able to press a "Play"-button to start the game.

Confirmation:

Functional

- What exactly happens when the player presses the play-button?

Non-functional

- Where should the play-button be located on the screen?

Story Identifier: HDM08

- Story Name: Gold
- Implemented: Yes

Description
As a player I'd like be able to earn gold.

Confirmation:

Functional

- Does the player earn gold from enemies?
- Does the player receive gold from selling a tower?

Non-functional

- Does the player visually see when it earns gold?

Story Identifier: HDM09

- Story Name: Health
- Implemented: Yes

Description
As a player I'd like to be able to lose health.

Confirmation:

Functional

- How much health does the player lose if example one enemy crosses the finish line?
- Does the player lose health during or after the round?
- How much total health should the player have?

Non-functional

- Does the player visually see when it loses health?
- What different ways can the player lose health?

Story Identifier: HDM10

- Story Name: End Goal
- Implemented: Yes

Description
As a player, I would like to receive clear feedback when the enemy has passed the end goal so I can take stock of the current situation.

Confirmation:

Functional

- Where on the game board will the end goal be positioned?
- How will the player know when an enemy has passed the end goal?
- What happens when an enemy passes the end goal?

Non-functional

- How will the end goal look like?

Story Identifier: HDM11

- Story Name: Enemy health bar
- Implemented: No

Description
As a player, I want to receive graphical feedback of the enemies health to be able to estimate the current situation.

Confirmation:

Functional

- Will the enemies have a total health bar on the screen or one separate for each enemy?
- Will the health bar follow the enemy when it crosses the game board?

Non-functional

- How will the health bar be visually represented?

2.1.5 Features

Story Identifier: HDM12

- Story Name: Sell tower from game board
- Implemented: Yes

Description
As a player I would like to be able to sell towers from the game board.

Confirmation:

Functional

- How does the player sell a tower?
- Can the player sell a tower at any point in the game?
- Can the player "press" a tower to sell it?
- How will the view show the player that a tower can be sold?

Non-functional

Story Identifier: HDM13

- Story Name: Pause game session
- Implemented: No

Description
As a player, I would like to be able to pause the game session.

Confirmation:

Functional

- Can the player pause the game in the middle of a round?
- What can the player do when the game is in a pause?

Non-functional

- Where will the pause-button be located on the GUI?

Story Identifier: HDM14

- Story Name: Change settings in menu
- Implemented: Yes

Description
As a player I would like to adjust the game settings in the menu

Confirmation:

Functional

- Does the tweaked settings correctly apply to the game?

Non-functional

- Is the settings presented in clear manner to the player?
- Is it easy to extend with future setting tweaks?

Story Identifier: HDM15

- Story Name: Quit game from start menu
- Implemented: No

Description
As a player I would like to be able to quit the game from the start menu.

Confirmation:

Functional

- Present a button for the player to exit the application through.

2.2 Definition of Done

A user story is done if it fulfills all of the following conditions:

- Passed all tests
- Is merged in git without unresolved conflicts
- Meets all acceptance criteria
- Is accepted by another group member.

2.3 User interface

The prototype only consists of one single view of the main game. It consists of a view of the grid where the game takes place and the player can place towers upon and a user interface bar covering the bottom part of the screen. The user interface bar offers tower that are able to be bought and placed upon the board. The bar also presents both some general game information and information about the current toggled tower. The player is also able to sell the tower from the toggled tower window.

The pictures below presents the different visual iterations of the projects GUI.

2.3.1 Primary view

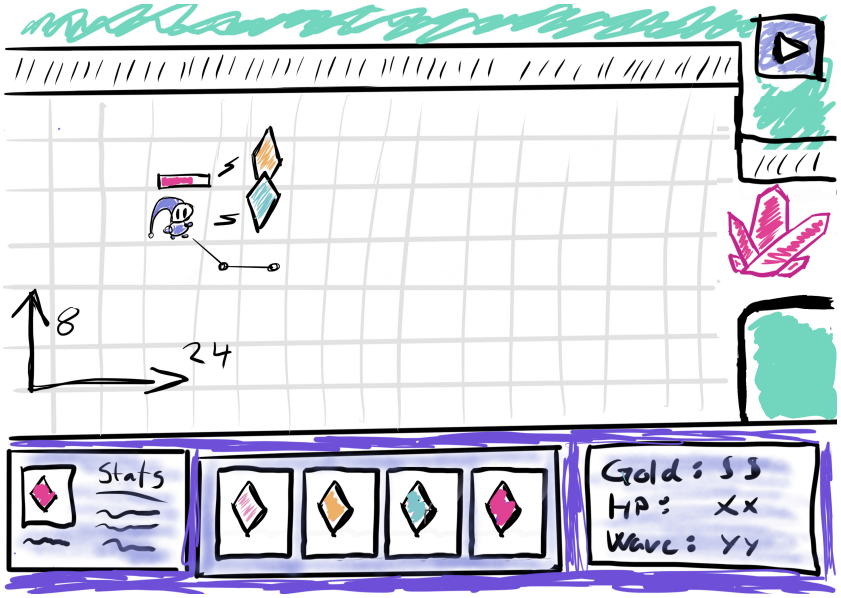

Figure 2: First draft of GUI

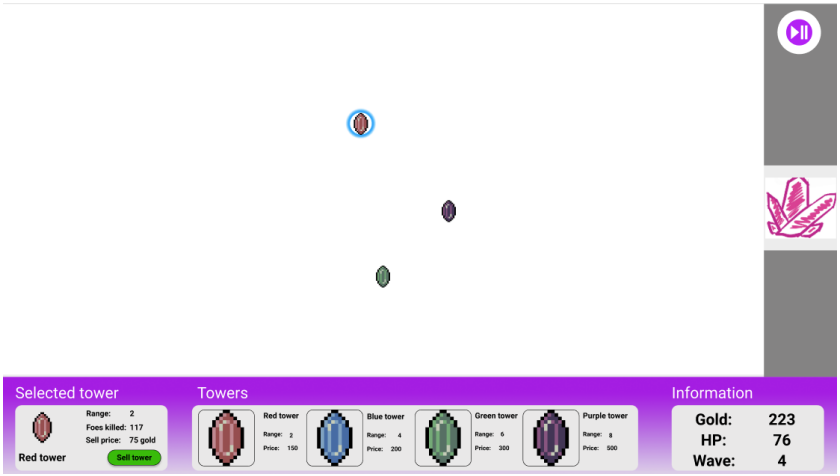

Figure 3: Model of GUI in Figma

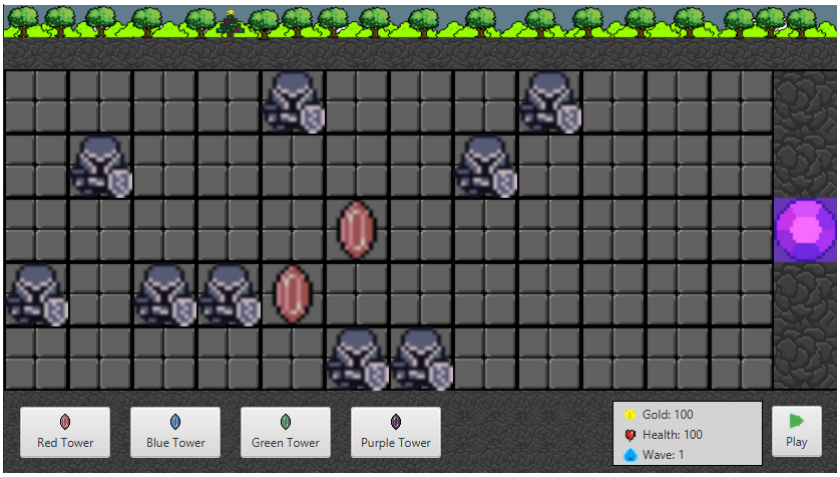

Figure 4: Final visual iteration

3 Domain model

Below is a high level overiew of the application presented through an UML diagram. Each class will be briefly described except for classes whose only responsibility is to keep track of low level data.

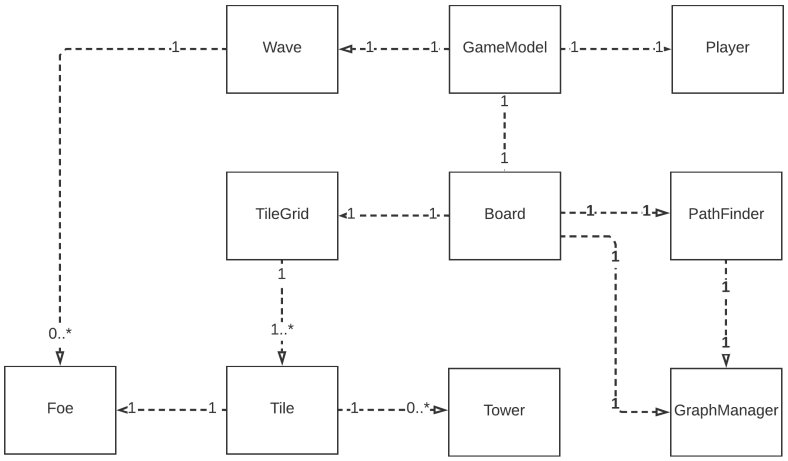

Figure 5: Domain model diagram

3.1 Game

GameModel is responsible for supplying the necessary functionality for the client to be able to run the application through some form of graphical representation. It acts as a communication point to the whole domain and the main update loop resides here.

3.2 Board

Board manages a two-dimensional grid where towers and enemies can be placed upon. It offers functionality for both manipulation and interaction with the underlying TileGrid and sets up a GraphManager that keep tracks of a seperate grid for pathfinding logic.

3.3 GraphManager

Sets up a graph (grid) with different nodes that can be used with pathfinding to both statically and dynamically generate a traversable path, with consideration to some nodes being ”blocked” and so on.

3.4 PathFinder

Responsible for actually generating the path. It needs a graph of nodes to operate on. Can perform checks if a specified path would be valid or not during the current circumstances.

3.5 TileGrid

A simple data container representing a two-dimensional grid of Tiles. No logic is performed here.

3.6 Tile

Responsible for manipulating any object that is placed upon it. Has method to both inject and eject one Foe or one Tower. If the injected object is a tower it can generate a list of adjacent Tiles in the grid that represents that towers attack range.

3.7 Wave

This class is responsible for creating a set of foes that will be spawned into the board by the GameModel in a even interval.

3.8 Player

Tracks the players gold and health and offers some functionality to manipulate these.

4 References

4.1 Production Tools

- Github: Used for managing version control through Git.
- Lucidchart: Used for sketching up and working collaboratively on different UML-diagrams.
- Travis: Used for continious integration.
- IntelliJ: IDE used for writing code.
- Trello: A taskboard application that was used to manage the projects scrum-board.
- Figma: Prototyping tool for graphical interfaces.

4.2 Communication Tools

- Slack: A communication platform.
- Zoom: Teleconferencing software program.

4.3 External Libraries

- JUnit: Used for testing the codebase.
- JavaFX: Used for graphical presentation of the application.