Requirements and Analysis Document for Alchemy-defense

Willem Brahmstaedt, Felix Jönsson,
Johan Lindén, Valdemar Stenhammar

date
version

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. (Stakeholders??)

1.1 Definitions, acronyms, and abbreviations

Create a word list to avoid confusion and give a definition of every abbreviation you use in the document.

2 Requirements

2.1 User Stories

2.1.1 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.

2.1.2 User Stories

1. Criticial
    a) As a player I'd like to create a game board (1)
    b) As a player I would like to have enemies to fight against. (1)
    c) As a player I would like enemies to spawn in game. (1)
    d) As a player I'd like to have access to the towers to be able to place them on the board (1)
    e) As a player I'd like to see a grid of where one can place the towers to make that process more forgiving (1)
2. Essential
    a) As a player I would like to be greeted by a Menu Screen to be able to start playing the game. (2)
    b) As a player I would like to be able to press a "Play"-button to start the game. (2)
    c) As a player I'd like to see how much gold/HP I have left to get a understanding of my game status (2)
    d) As a player, I would like to receive clear feedback when the enemy has passed the end of the line so I can take stock of the current situation. (2)
    e) As a player, I want to receive graphical feedback of the enemies health to be able to estimate the current situation. (2)

3. Features
    a) As a player I'd like to be able to sell a tower from the game board. (3)
    b) As a player, I would like to be able to pause the game session. (3)
    c) As a player I would like to be able to tweak the settings in the menus. (3)
    d) As a player I would like to be able to quit the game from the start menu. (3)

1. Critical. Required to be implemented in order for the application to function.
2. Essential. Things that are essential for the game to be enjoyable for a player, but not required for the application to run.
3. Features. Features that would be nice to have given there's enough time to implement them.

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

Include sketches, drawings and explanations of the application's user interface. Describe the navigation between the different views.

2.3.1 Primary view

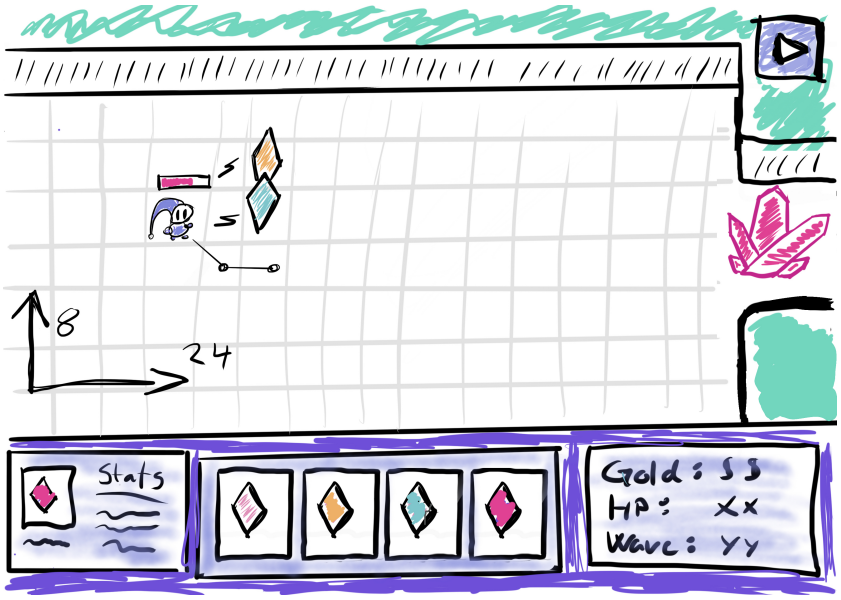

Figure 1: First draft of GUI

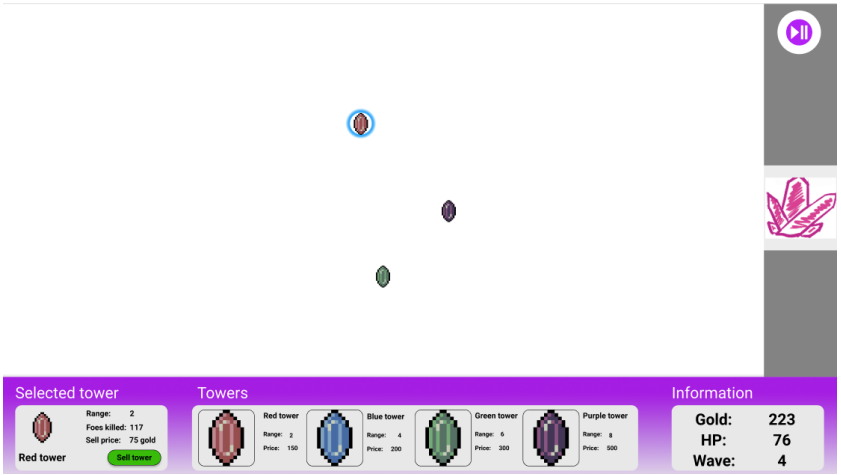

Figure 2: Model of GUI in Figma (Cite figma?)

3 Domain model

Give a high level view overview of the application using a UML diagram.

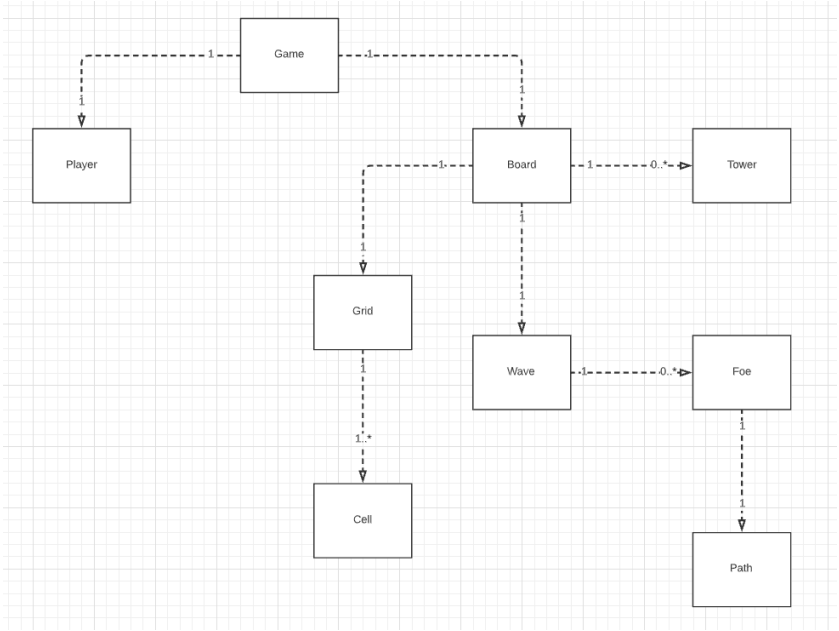

Figure 3: Domain model diagram

3.1 Class responsibilities

Explanation of responsibilities of classes in diagram.

1. Game
    a) Has a board which resembles the game.
    b) Has one player.
    c) Has the ability to spawn wave of foes.
2. Player
    a) Stores values of different types, for now it stores gold and health.
3. Board
    a) Holds a grid with cells.
    b) Has the ability to place concrete towers in grid.
    c) Has the ability to place concrete foes in grid.

4. Wave
    a) Represents a list of foes.
    b) Has the ability to spawn foes in different sequences.
5. Grid
    a) represents a two-dimensional grid of cells.
6. Grid
    a) represents a cell (x,y coordinates) in a grid.
7. Tower
    a) Abstract class
    b) Has all common functionality and attributes for towers.
    c) Concrete towers implement this class.
8. Foe
    a) Abstract class (Not at the moment)
    b) Has all common functionality and attributes for foes.
    c) Concrete foes implement this class.
9. Path
    a) Defines the pathfinding algorithm.
    b) Returns a path that foes can use
    c) ConcreteBoard has-a pathfinder.

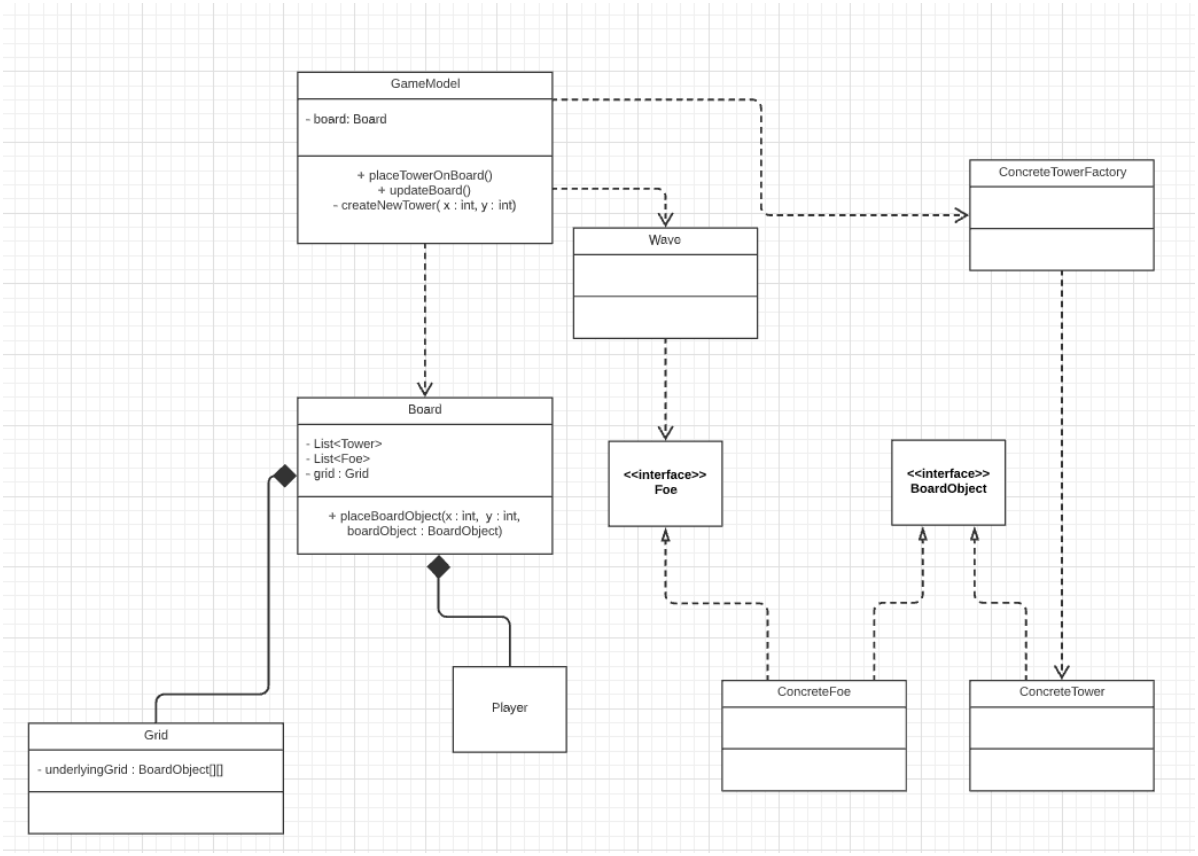

Figure 4: Class uml diagram (Temporary)

4 References

List all references to external tools, platforms, libraries, papers, etc.