Requirements and Analysis Document for and I OOPP

Jacob Bengtsson, Anton Ekström, Elin Nilsson
Amanda Papacosta, Arvid Svedberg

October 24, 2021

1 Introduction

This project intends to create a tower defense video game similar to other games in the genre like *Bloons Tower Defense* and Plants vs. Zombies. The purpose of this game is to entertain the user who is playing it. The game is enjoyable because some strategy is needed. You need to consider where and when to place your towers to defeat all the incoming enemies. You must also balance your spending to make sure all aspects of your defense are up to the task of protecting the end of the playing field. The intended audience benefits from having an entertaining and an enjoyable experience while using this application.

1.1 Definitions, acronyms, and abbreviations

Tower defense are a sub genre of video games within the more encompassing strategy genre. In general, the player has to place towers in order to defend something and it's up to the player to make sure that the correct towers are placed to ensure a strong enough defense to protect the objective at hand.

Towers are the characters which are placed in the game world which by different means defeat incoming enemies to protect the objective.

This tower defense game is built upon the concept of **lanes** and **cells**. Lanes are horizontal paths in which the enemies can move. These lanes are then divided into cells to create a grid and within each of the cells a tower can be placed. The tower can then attack all enemies which are approaching in the same lane as the tower.

The attacking enemies are sent out in **waves**. A wave is a collection of enemies sent out to different lanes at different intervals. Once all enemies in a wave have been

defeated a new wave can be started. Waves further into a session will have more enemies as well as tougher enemies. This means the player has to improve their defense over time to accommodate the increasingly more dangerous competition.

2 Requirements

2.1 User Stories

1. **Epic: Tower defense game**
   As a gamer I want a tower defense game to play to have fun.
   Confirmation: Can approaching enemies be defeated by placing down towers on the game world?

2. Render cell grid
   As a gamer I want a grid of cells to place towers within to be able to create a defense.
   Confirmation: A grid of cells renders in the game window.
   Towers can be placed within these cells.

3. Tower - Mario
   As a gamer I want the tower Mario to help defend.
   Confirmation: A Mario tower can be added to a cell. Mario then correctly shows up in the correct cell.

4. Attack - Fireball
   As a gamer I want Mario to be able to attack with a fireball to kill enemies.
   Confirmation: A fireball appears in front of Mario and the moves forward along the game lanes.

5. Attack targeting
   As a gamer I want towers to only attack when they need to, this to reduce clutter in the game world.
   Confirmation: Towers only attack when an attackable enemy is within range of the attack the tower currently uses.

6. Enemy - Goomba
   As a gamer I want there to spawn Goombas at the end of the game world which move across it to create something to defend against.
   Confirmation: A Goomba enemy is created which moves across the game world.

7. Render entities
   As a gamer I want all towers, enemies and projectiles in the game world to render so that I can see what is happening inside the game.
   Confirmation: Towers, enemies and projectiles from attacks are rendered in their correct positions and are updated over time.

8. Defeating enemies
   As a gamer I want projectiles from attacks to able to damage enemies to defeat them.
   Confirmation: Enemies take damage when their hitbox intersects an attack projectile's hitbox. Eventually the die when hit enough times and the enemy is removed from the game world.

9. Enemies can attack towers
   As a gamer I want enemies to be able to attack and kill towers to make the game more challenging.
   Confirmation: Towers take damage when an enemy is close infront of them. When they've taken enough damage the tower is dead and is removed from the game world.

10. Money
    As a gamer I want there to be money that can be used to buy towers to make the game more challenging.
    Confirmation: Towers have a cost. Towers can only be added when the player has enough money in their wallet.

11. Tower - Toad
    As a gamer I want another tower, Toad, to generate money so that towers can be afforded.
    Confirmation: A Toad is able to be placed in the game world. Toad occasionally increases the amount of money in the player's wallet.

12. Enemy - Koopa Troopa
    As a gamer I want another enemy to defeat to increase variation.
    Confirmation: A Koopa Troopa enemy can be created which moves across the game world, with unique specifications to previous towers (e.g. movement speed, hitpoints

13. Wave of enemies
    As a gamer I want there to be multiple enemies attacking and their spawning locations to be different each playthrough to increase replayability.
    Confirmation: A 'Wave' (or group) of enemies are sent out where the type of enemy is random. The interval between enemies and into which lane they appear is also random.

14. Game end
    As a gamer I want the game to end when an enemy has traveled across the entire game world to give the game a losing condition.
    Confirmation: When an enemy reaches the opposite side of the game world of where it was created the game should stop.

15. Money GUI
    As a gamer I want a graphical display to see my current balance in the wallet.
    Confirmation: The current balance in the players wallet is rendered in the game window.

16. Tower card menu
    As a gamer I want a card menu of the towers to see the aviliable selection and cost for each tower.
    Confirmation: A card is rendered in the GUI for each tower featuring the tower's cost and its name.

17. Drag and drop
    As a gamer I want to be able to place towers in the game world by dragging them from the card menu and dropping them in a cell.
    Confirmation: A tower can be placed in a cell by dragging the tower from its card in the menu and then dropping it in the wanted cell.

18. Tower - Bob-omb
    As a gamer I want to be able to palce Bob-omb to increase the variation in the defense.
    Confirmation: A Bob-omb tower is placeable in the game world. It explodes after a couple seconds killing all enemies in close proximity. Bob-omb is removed from the game world after it has exploded.

19. Dropped coins
    As a gamer I want Toad to drop a coin instead of just increasing the amount in the wallet. I also want enemies to drop a coin when they die. The coin needs to be clicked to then increase the amount in the wallet. This to increase engagment in the game.
    Confirmation: A coin is rendered at Toad's position when he generates money, the same type of coin is rendered where an enemy just died. When the coin is clicked with the mouse its value is added to the player's wallet.

20. Enemy - Boo
    As a gamer I want a Boo enemy so that there are enemies that can only be defeated by certain towers to increase the required strategy.
    Confirmation: A Boo enemy can be created that moves across the screen similar to other enemies, however it can only be hurt by light based attacks.

21. Tower - Luigi
    As a gamer I want a tower with a light based attack to defeat certain enemies.
    Confirmation: Luigi can be placed in a cell in the game world. He attacks using a flashlight which can attack enemies in a short range infront of him. The flashlight does not hurt regular enemies, only specific enemies vulnerable to light (e.g. Boo).

22. GUI feedback
    As a gamer I want to know what towers I can afford and where I can place them to incerase ease of use.
    Confirmation: The tower cards change colors when the amount of money in the wallet is enough to afford the corresponding tower. When a towers dragged into the game world the cell below it is highlighted to show clearer where the tower is going to be placed.

Unimplmeneted user stories

23. Enemy - Hammer Bro (unimplemented)
    As a gamer I want an unpredictable Hammer Bro enemy to increase the games difficulty.
    Confirmation: A Hammer Bro enemy can be created which while moving forward hops between lanes. It also throws hammers which can hurt towers from a distance.

24. Enemy - Buzzy Beetle (unimplemenetd)
    As a gamer I want an enemy immune to fire to further increase strategy.
    Confirmation: A Buzzy Beetle enemy can be created which behaves like a standard enemy but is immune to fire based attacks.

25. Enemy - Spiny (unimplemented)
    As a gamer I want an enemy that can't be jumped on to further increase strategy.
    Confirmation: A Spiny enemy can be created which behaves like a standard enemy but is immune to jump attacks.

26. Enemy - Lakitu (unimplemented)
    As a gamer I want an enemy that can create other enemies to make the game more difficult.
    Confirmation: A Lakitu enemy can be created which is a flying enemy that does not attack on its own but throws Spiny enemies on the ground close to itself.

27. Flying enemies (unimplmented)
    As a gamer I want most enemies to have a flying variation to increase difficulty and vareity.
    Confirmation: Enemies can be set to fly meaning they fly over/past towers and can only be hit by specific attacks that reach up in the air.

28. Boss enemy - Bowser (unimplemented)
    As a gamer I want there to be a final boss enemy to mark the end of the game.
    Confirmation: A large boss enemy, Bowser, can be created in the game world. When he is defeated the player has won. Bowser takes up two lanes and can attack towers by shooting fireballs at them.

29. Tower - Yoshi (unimplmeneted)
    As a gamer I want more towers to increase the variation of attacks.
    Confirmation: A Yoshi tower can be placed which can eat enemies. Koopa Troopas that have been eaten can be spitted back out where the shell deals damage to enemies in Yoshi's lane. Yoshi can also jump on enemies to stomp them.

30. Tower - Rosalina (unimplemented)
    As a gamer I want more towers with unique attacks to increase variety.
    Confirmation: A Rosalina tower can be placed which can attack all surrounding cells with a spin attack. She can also attack all cells on a diagonal with a starbit attack.

31. Tower upgrades (unimplemented)
    As a gamer I want to be able to upgrade towers to increase the variety of attacks.
    Confirmation: A GUI for upgrading towers can be used to exchange money in the players wallet to increase the power of a tower. The tower could unlock a new attack or it could upgrade the towers current attack by making it stronger in some way.

32. Losing the game (unimplemented)
    As a gamer I want to be able to loose the game, because it makes the game more exiting.
    Confirmation: When an enemy reaches the end of a lane, something tells the user that the game is over.

33. Winning the game (unimplemented)
    As a gamer I want to be able to win the game, because it gives me a purpose of playing.
    Confirmation: The game has a predetermined amount of waves/enemies, and when they are all defeated the game is won. When the game is won, something tells the user that the game is won.

34. Displaying level progress (unimplemented)
    As a gamer, I want to be able to see how much progress I have made on a specific level, because it makes me feel more motivated to continue playing.
    Confirmation: A progressbar is implemented at the bottom of the screen, displaying how many percent of the current level has been completed.

35. Start game view (unimplemented)
    As a gamer, I want a startscreen with a "start game" button before entering the game so that I am ready for when the game actually starts.
    Confirmation: A different view prior to the actual game is show on the screen, with buttons such as "start game". When pressed, the view of the game is showed on the screen, and the game begins.

2.2 Definition of Done

The definition of whenever a user story has been completed is whether or not the code related to said user story has been tested and reviewed, as well as been merged to the main branch in the group's common GitHub repository.

2.3 User interface

Figure 1: A simple sketch of the game showing the basic functions for placing and choosing a tower.

The game mainly consists of a grid where the towers can be placed. One in each cell of the grid. The enemies enter the game from the right side of the screen and approach the towers. From the cards at the top of the screen, towers can be selected and then dragged onto the game grid to be placed down. When a tower is placed, its cost listed at the bottom of its card is subtracted from the player's current in-game balance, which is displayed at the top left side of the screen.

The players current balance of coins is displayed in a brown box with a distinct title "coins". Here the player can always keep track of their coins so that they know which towers they can afford. The cards for the towers have a light-grey background when the tower is available for purchase, but turn into a darker grey when the player can´t afford a tower.

When a tower is dragged for being placed into a cell, the cells changes color to show whether or not the cell is available. White for available and red for unavailable. The tower also expands in size when selected from a card, to help the user realise that they are supposed to be dragged and placed on the grid.

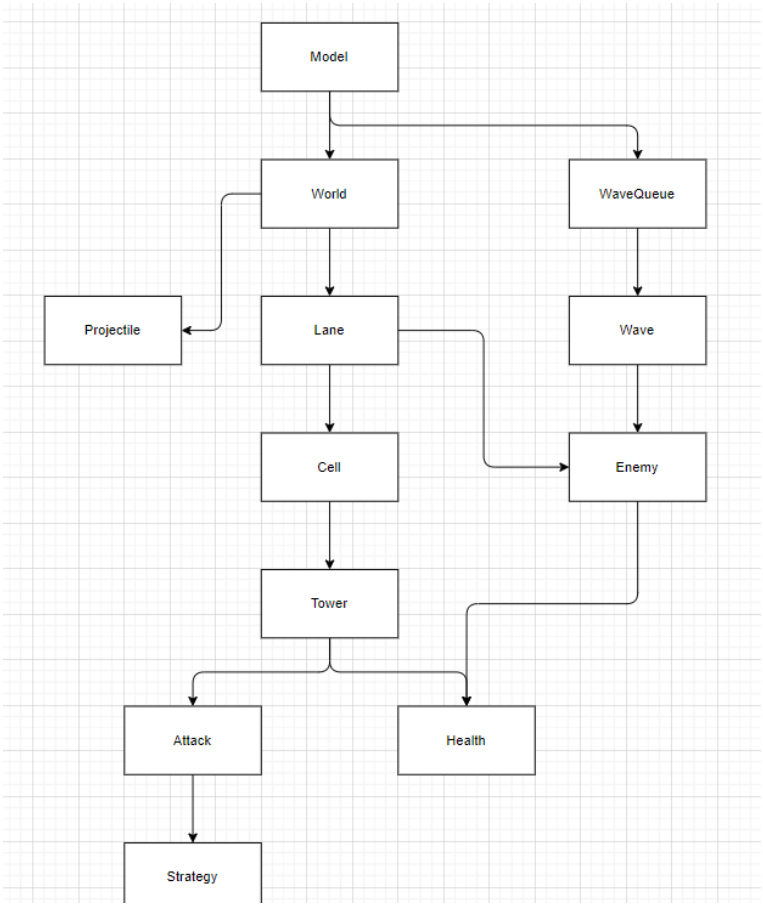

Figure 2: Domain model

3 Domain model

3.1 Class responsibilities

The *Model* has a *World* and a *Player* and is responsible for connecting the two. This is the only thing the Model does. It holds a World and a Player while these two has a lot more logic and information.

The *World* is responsible for tieing together all the different components the game itself consists of. The *World* has a list of all the *Lanes*, *Enemies*, *Projectiles* and *DroppedCoins*. The world provides the information about these that a World would know. For example how many *Lanes* the grid consists of, how many and which *Enemies* are currently on the grid. And the same for *Projectiles* and *DroppedCoins*.

The *Lanes* are a part of the *World*, and consist of a number of *Cells*. The number of *Cells* are the same for every *Lane*. The *Cells* works as a coordinate-system and their task is being able to hold one *Tower* at a time. The *Cell* can provide information about

which *Tower* it currently holds or if the *Cell* is empty. A *Lane* is responsible for holding the *Enemies*, since they move across Cells, but in the same Lane.

Projectiles are entities moving on the grid that are not *Enemies* or *Towers*, and also has a damageSource. One example is *FireBallProjectile* that is a fireball thrown out by a specific *Tower* and deals damages to *Enemies*.

DroppedCoins are entities that appears on the grid on different occasions. Either if the *Tower Toad* makes them appear, or if an *Enemy* is defeated. The *DroppedCoins* can be collected by clicking on them, and this will increase the amount of money a *Player* can use for buying new *Towers*.

Tower is the super-class of all *Towers* and has all the basic properties of *Towers* like Mario, Toad, Luigi or Bobomb. These can be placed on the grid in a *Cell* and attack and defeat *Enemies* in different ways. Each of them are sub-classes of *Tower* and implements their own specific behaviour and attributes, for example different *Attacks*.

Attack is an abstract class where all common logic for performing attacks are located. The purpose of this class is to check if an attack is available and perform it. The class has a few sub-classes which are concrete *Attacks*, for example Explosion and FireBallAttack. An *Attack* also has a *Strategy* in the form of an AttackTargetArea, that has a few sub-classes that represents different targeting areas for an attack.

Enemy is the super-class of all *Enemies* and has like *Tower*, all the basic properties an *Enemy*. The different sub-classes Goomba, Koopa-Troopa and Boo, all have different attributes and behaviour. For example the amount of damage they make on a *Tower* or the pace they are moving forward at.

The *WaveQueue* class is responsible for being a queue of different *Waves*. It adds one *Wave* at a time to the *World*. The class *Wave* is a collection of *Enemies*, and these will be added to the *World*, one at a time as well.

Both *Enemies* and *Towers* uses the class *Health*. The *Health* class is responsible for keeping track of a Tower or Enemies health, decreasing it, and check if one is defeated.