Project Plan

Topic: Tower defense

Worked by:

Erald Shahinas

Shreyas Giridhar

Advisor:

Mark Heidmets

General description

Core functionalities

We aim to implement all the core functions before the optional features.

We plan to achieve “Enemy follow a single, non-branched path” by loading a map instance into a game instance. Which map instance depends on which level the player selects. The game instance also knows the current player’s status which includes health and money. If the player health reaches >0, then the game is over and the high scores will be displayed.

We also plan to have an abstract “Tower” and “Enemy” classes. Different types of towers or enemies inherit these abstract classes. Some basic function members such as health/damage and bullet speed/movement speed will be in the abstract class. However, they will be implemented in classes which inherit the abstract class.

Initially we plan to have at least 5 towers and 5 enemies with 2-3 upgrades for towers. The upgrades improve the stats of the tower and slightly change the visual appearance.

Optional features

We plan to have many optional features included however the time constraints of the project will decide how many of them we will be able to implement in the end. The optional features are:

Ability to generate a random functional map, also we will add a seed input box so we can get the same map always

Upgradable towers, will have different sprites depending on the level of the tower

More towers and enemies

Level editor which can create new maps using the gui interface. Also we want the game to be able to save and load the maps created

At the end of the game the user will be able to input their name, and they will be placed in a high score table with the best local scores

User will be able to delete or change previously inserted towers.

In case of multiple paths enemies will chose the one with the least towers possible

Implementation of 3 distinct difficulty levels: easy, medium, hard

Game will have a soundtrack and different operation will produce sound: click of a button, tower shooting, enemy dying etc. Both sounds can be toggled in the main interface

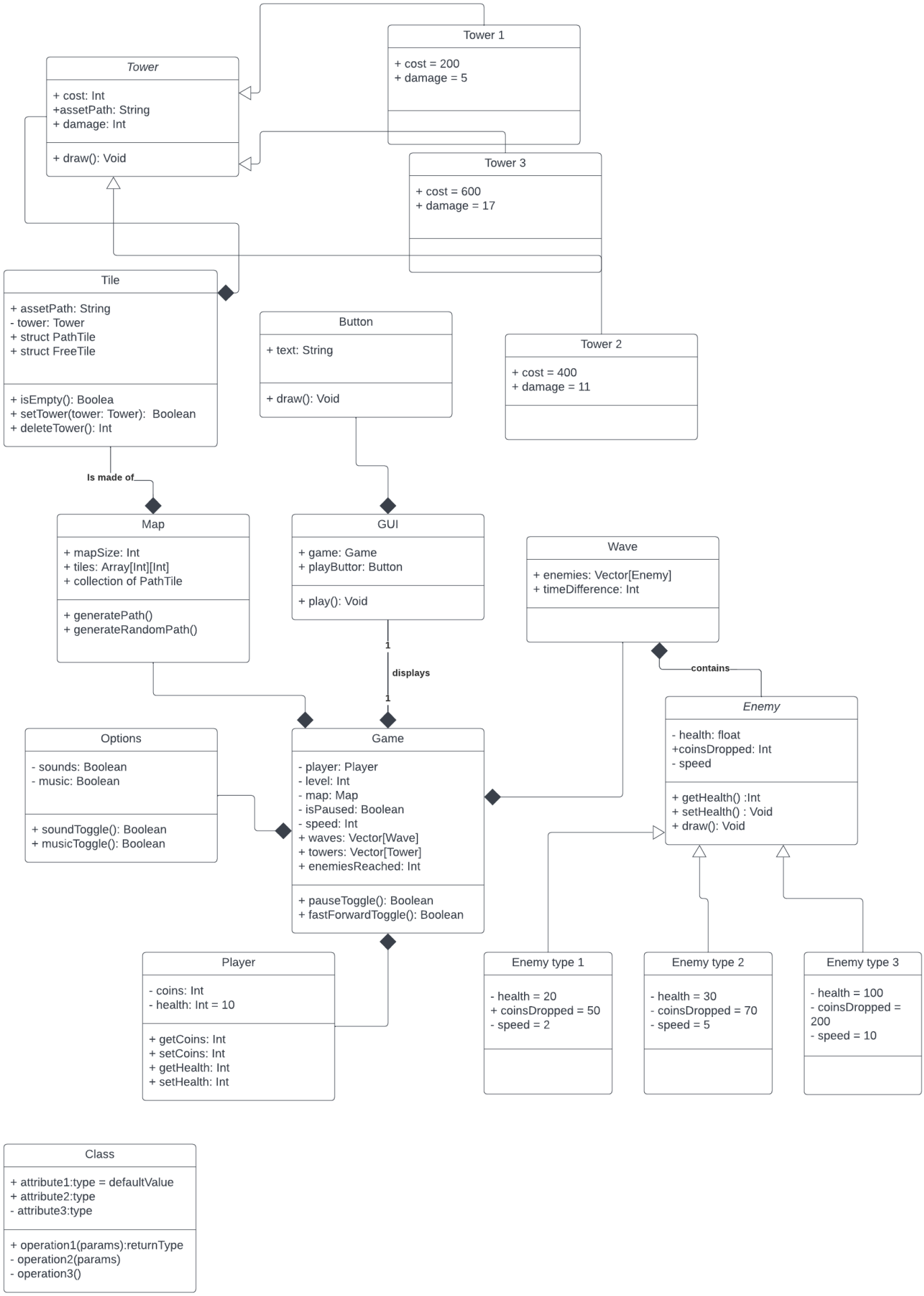
External Libraries

SFML

Font

UML Diagram

The classes will eventually be changed, this is a depiction of how the structure of the project will be based on our current understanding


Tower 1
+ cost = 200
+ damage = 5
Tower
+ cost: Int
+assetPath: String
+ damage: Int
+ draw(): Void
Tower 3
+ cost = 600
+ damage = 17
Tile
+ assetPath: String
- tower: Tower
+ struct PathTile
+ struct FreeTile
+ isEmpty(): Boolea
+ setTower(tower: Tower): Boolean
+ deleteTower(): Int
Button
+ text: String
+ draw(): Void
Tower 2
+ cost = 400
+ damage = 11
Is made of
Map
+ mapSize: Int
+ tiles: Array[Int][Int]
+ collection of PathTile
+ generatePath()
+ generateRandomPath()
GUI
+ game: Game
+ playButtor: Button
+ play(): Void
Wave
+ enemies: Vector[Enemy]
+ timeDifference: Int
1
displays
1
contains
Enemy
- health: float
+coinsDropped: Int
- speed
+ getHealth() :Int
+ setHealth() : Void
+ draw(): Void
Options
- sounds: Boolean
- music: Boolean
+ soundToggle(): Boolean
+ musicToggle(): Boolean
Game
- player: Player
- level: Int
- map: Map
- isPaused: Boolean
- speed: Int
+ waves: Vector[Wave]
+ towers: Vector[Tower]
+ enemiesReached: Int
+ pauseToggle(): Boolean
+ fastForwardToggle(): Boolean
Player
- coins: Int
- health: Int = 10
+ getCoins: Int
+ setCoins: Int
+ getHealth: Int
+ setHealth: Int
Enemy type 1
- health = 20
+ coinsDropped = 50
- speed = 2
Enemy type 2
- health = 30
- coinsDropped = 70
- speed = 5
Enemy type 3
- health = 100
- coinsDropped = 200
- speed = 10
Class
+ attribute1:type = defaultValue
+ attribute2:type
- attribute3:type
+ operation1(params):returnType
- operation2(params)
- operation3()

We decided to uses SFML as the main library to render our game. While doing initial research we found that, we need to clear and redraw the screen in a loop in the main file. Hence we decided to have a separate game class that keeps a collection of towers, enemies, money and health. Then this inherits the GUI responsible for the buttons and Map which is responsible tower/ enemy placements. We have an abstract class for enemies and towers which are inherited by a other instances of towers / enemies (for different type of enemies). Map inherits tile class which has a struct path block (enemy path) and free block. The tile class is used to determine is a tile is free or blocked. It is also responsible for placing and editing/moving the towers.

Schedule and division of work

Week 1 (14th Nov. – 20th Nov.)

- main class: (Shreyas)
- game class: (Erald)
- map/tile: (Erald)
- gui: (Shreyas)
- button: (Shreyas)
- player: (Erald)

Week 2 (21th Nov. – 27th Nov.)

- towers (placement, deleting etc...): (Shreyas)
- enemies: (Erald)
- waves: (Erald)

Week 3 (28th Nov. – 4th Dec.)

- sprites: (Erald(for enemies) & Shreyas(for towers))
- sound effects (2): (Erald)
- upgradable towers (2): (Shreyas)
- more types of towers and enemies (2): (Erald & Shreyas)
- random path generator (2): (Erald)
- high score (1): (Shreyas)

Week 3.5 (5th Dec. – 9th Dec.)

- documentation (Erald & Shreyas)