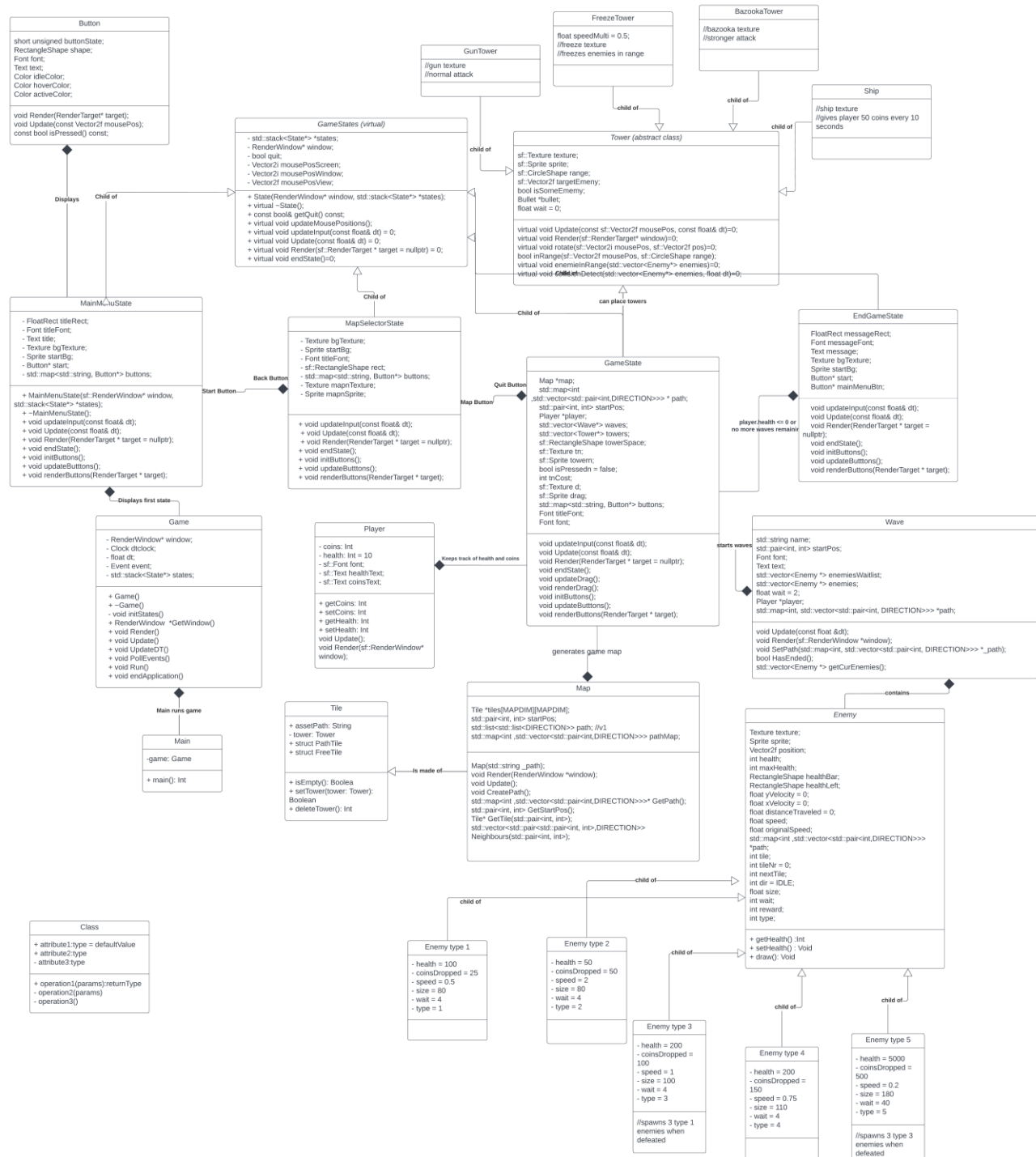# Project Documentation

Group 3 (TD)

Overview

This project is an UI based Tower Defence game built using c++ SFML library. All the basic features and some optional features have been implemented. The window is fixed size and trying to resize it will break the game. This is because all the buttons and layout has been done with relation to a fixed window size. There are three level with increasing difficulty. Within each level there are 5-10 waves of enemies that spawn. With the progression of waves, spawned enemies are stronger (harder to kill) and move faster. The player has 10 hitpoints, where I hitpoint correspond to one enemy reaching the end before being defeated. If 10 enemies reach the end, the game is over. The player wins if all the waves are complete with more than 0 hitpoints. There are total of 4 towers. Where the basic tower shoots a bullet every 2 seconds, with a price of 100. Then there is a freeze tower in the form of sinper which halves the enemies' speed within the sniper's range. Then there is a bazooka tower with a price of 300, which shoots bigger bullets, hence hitting multiple enemies and doing more damage than the basic tower. Lastly there is a boat which can only be placed on water tiles. This boat gives the player 50 coins every 10 seconds. The towers are placed by holding the right mouse button and releasing in the desired tile. If the player does not have sufficient money or places on the wrong tile, the placement doesn't go through. Once the tower is placed it cannot be removed, hence one must be strategic where they place towers.

Software Structure

Button

short unsigned buttonState;
RectangleShape shape;
Font font;
Text text;
Color idleColor;
Color hoverColor;
Color activeColor;

void Render(RenderTarget* target);
void Update(const Vector2f mousePos);
const bool isPressed() const;

FreezeTower

float speedMulti = 0.5;
//freeze texture
//freezes enemies in range

BazookaTower

//bazooka texture
//stronger attack

GunTower

//gun texture
//normal attack

Ship

//ship texture
//gives player 50 coins every 10 seconds

GameStates (virtual)

- std::stack<State*> *states;
- RenderWindow* window;
- bool quit;
- Vector2i mousePosScreen;
- Vector2i mousePosWindow;
- Vector2f mousePosView;

+ State(RenderWindow* window, std::stack<State*> *states);
+ virtual ~State();
+ const bool& getQuit() const;
+ virtual void updateMousePositions();
+ virtual void updateInput(const float& dt) = 0;
+ virtual void Update(const float& dt) = 0;
+ virtual void Render(sf::RenderTarget * target = nullptr) = 0;
+ virtual void endState()=0;

Tower (abstract class)

sf::Texture texture;
sf::Sprite sprite;
sf::CircleShape range;
sf::Vector2f targetEmeny;
bool isSomeEmemy;
Bullet *bullet;
float wait = 0;

virtual void Update(const sf::Vector2f mousePos, const float& dt)=0;
virtual void Render(sf::RenderTarget* window)=0;
virtual void rotate(sf::Vector2i mousePos, sf::Vector2f pos)=0;
bool inRange(sf::Vector2f mousePos, sf::CircleShape range);
virtual void enemieInRange(std::vector<Enemy*> enemies)=0;
virtual void CollisionDetect(std::vector<Enemy*> enemies, float dt)=0;

MainMenuState

- FloatRect titleRect;
- Font titleFont;
- Text title;
- Texture bgTexture;
- Sprite startBg;
- Button* start;
- std::map<std::string, Button*> buttons;

+ MainMenuState(sf::RenderWindow* window, std::stack<State*> *states);
+ ~MainMenuState();
+ void updateInput(const float& dt);
+ void Update(const float& dt);
+ void Render(RenderTarget * target = nullptr);
+ void endState();
+ void initButtons();
+ void updateButtons();
+ void renderButtons(RenderTarget* target);

MapSelectorState

- Texture bgTexture;
- Sprite startBg;
- Font titleFont;
- sf::RectangleShape rect;
- std::map<std::string, Button*> buttons;
- Texture mapnTexture;
- Sprite mapnSprite;

+ void updateInput(const float& dt);
+ void Update(const float& dt);
+ void Render(RenderTarget * target = nullptr);
+ void endState();
+ void initButtons();
+ void updateButtons();
+ void renderButtons(RenderTarget* target);

EndGameState

FloatRect messageRect;
Font messageFont;
Text message;
Texture bgTexture;
Sprite startBg;
Button* start;
Button* mainMenuBtn;

void updateInput(const float& dt);
void Update(const float& dt);
void Render(RenderTarget * target = nullptr);
void endState();
void initButtons();
void updateButtons();
void renderButtons(RenderTarget * target);

GameState

Map *map;
std::::map<int
,std::::vector<std::::pair<int,DIRECTION>>> * path;
std::::pair<int, int> startPos;
Player *player;
std::vector<Wave*> waves;
std::vector<Tower*> towers;
sf::RectangleShape towerSpace;
sf::Texture tn;
sf::Sprite towern;
bool isPressedn = false;
int tnCost;
sf::Texture d;
sf::Sprite drag;
std::map<std::string, Button*> buttons;
Font titleFont;
Font font;

void updateInput(const float& dt);
void Update(const float& dt);
void Render(RenderTarget * target = nullptr);
void endState();
void updateDrag();
void renderDrag();
void initButtons();
void updateButtons();
void renderButtons(RenderTarget * target);

Game

- RenderWindow* window;
- Clock dtclock;
- float dt;
- Event event;
- std::stack<State*> states;

+ Game()
+ ~Game()
- void initStates()
+ RenderWindow *GetWindow()
+ void Render()
+ void Update()
+ void UpdateDT()
+ void PollEvents()
+ void Run()
+ void endApplication()

Player

- coins: Int
- health: Int = 10
- sf::Font font;
- sf::Text healthText;
- sf::Text coinsText;

+ getCoins: Int
+ setCoins: Int
+ getHealth: Int
+ setHealth: Int
void Update();
void Render(sf::RenderWindow* window);

Wave

std::string name;
std::::pair<int, int> startPos;
Font font;
Text text;
std::vector<Enemy *> enemiesWaitlist;
std::vector<Enemy *> enemies;
float wait = 2;
Player *player;
std::::map<int, std::::vector<std::::pair<int, DIRECTION>>> *path;

void Update(const float &dt);
void Render(sf::RenderWindow *window);
void SetPath(std::::map<int, std::::vector<std::::pair<int, DIRECTION>>> *_path);
bool HasEnded();
std::vector<Enemy *> getCurEnemies();

Main

-game: Game

+ main(): Int

Tile

+ assetPath: String
- tower: Tower
+ struct PathTile
+ struct FreeTile

+ isEmpty(): Boolea
+ setTower(tower: Tower):
Boolean
+ deleteTower(): Int

Map

Tile *tiles[MAPDIM][MAPDIM];
std::::pair<int, int> startPos;
std::::list<std::::list<DIRECTION>> path; //v1
std::::map<int ,std::::vector<std::::pair<int,DIRECTION>>> pathMap;

Map(std::::string _path);
void Render(RenderWindow *window);
void Update();
void CreatePath();
std::::map<int ,std::::vector<std::::pair<int,DIRECTION>>>* GetPath();
std::::pair<int, int> GetStartPos();
Tile* GetTile(std::::pair<int, int>);
std::::vector<std::::pair<std::::pair<int, int>,DIRECTION>>
Neighbours(std::::pair<int, int>);

Enemy

Texture texture;
Sprite sprite;
Vector2f position;
int health;
int maxHealth;
RectangleShape healthBar;
RectangleShape healthLeft;
float yVelocity = 0;
float xVelocity = 0;
float distanceTraveled = 0;
float speed;
float originalSpeed;
std::::map<int ,std::::vector<std::::pair<int,DIRECTION>>> *path;
int tile;
int tileNr = 0;
int nextTile;
int dir = IDLE;
float size;
int wait;
int reward;
int type;

+ getHealth() :Int
+ setHealth() : Void
+ draw(): Void

Class

+ attribute1:type = defaultValue
+ attribute2:type
- attribute3:type

+ operation1(params):returnType
- operation2(params)
- operation3()

Enemy type 1

- health = 100
- coinsDropped = 25
- speed = 0.5
- size = 80
- wait = 4
- type = 1

Enemy type 2

- health = 50
- coinsDropped = 50
- speed = 2
- size = 80
- wait = 4
- type = 2

Enemy type 3

- health = 200
- coinsDropped = 100
- speed = 1
- size = 100
- wait = 4
- type = 3

//spawns 3 type 1 enemies when defeated

Enemy type 4

- health = 200
- coinsDropped = 150
- speed = 0.75
- size = 110
- wait = 4
- type = 4

Enemy type 5

- health = 5000
- coinsDropped = 500
- speed = 0.2
- size = 180
- wait = 40
- type = 5

//spawns 3 type 3 enemies when defeated

Displays — Child of — child of — child of — child of — child of — can place towers — Child of — Back Button — Start Button — Quit Button — Map Button — Displays first state — Keeps track of health and coins — starts waves — player.health <= 0 or no more waves remaining — generates game map — contains — child of — Main runs game — Is made of — child of — child of — child of — child of

External libraries used: SFML

Instructions for building and using the software

In order to compile the program running 'make' is sufficient and './main' in order to run it.

In order to use the program is enough to follow the description of the buttons, as the UI is self evident.

Testing

Different modules in the software were tested by running the main program. Sometimes when there was a seg fault, we used gdb to pin point the location of the error in our code. The main testing was done visually. By compiling and checking if the indented output was rendering on the screen and if we could interact with it.

Work log

Week 1

During week on we mainly focused on getting the environment set up and rendering a basic screen. In this week we also took time to go through the sfml documentation and learn how to implement things.

Erald - created the main.cpp and game.cpp

Shreyas – cleaned up and made it more modular by adding state.cpp

Week 2

During this week we had planned to implement the tile and map class, which would dynamically generate the map from a file. This week we had a meeting on Monday and divided the tasks.

Erald – Map and tiles class (15 x 15 tiles), Player class, enemy abstract class

Shreyas – MainMenuState, LevelSelector state and tower abstract class

Week 3

By week 3 we had most of the basic things working and we now had to focus on implementing the actual movement of the enemies and the shooting of the towers. We decided that bullets are difficult to implement with the given time and planned to use a laser instead. However we found a way to rotate the tower to the closest enemy and if the bullet is relatively faster than the enemy it is most likely to hit the target. Hence we implemented the bullet class. Along with the other implementations we also did some visual bug fixes

Erald – Waves and enemies concrete classes

 Shreyas – Bullet Class and towers concrete classes

Week 4

We had completed most of our tasks, however we only had 2 towers but planned to have 4 in total. This week we also had a meeting with the advisor and did the demo presentation. After that the advisor

advised that it would make the game feel more polished if we had a game over and game win state. Hence we spent this week to polish our game and fix minor bugs and added 2 more towers boat(grants money every 10 seconds) and bazooka(shoots bigger bullet thus hitting multiple enemies with more attack damage). We also finished out project documentation this week

Erald – Polishing enemies, writing documentation

Shreyas – boatTower and bazookaTower, writing documentation