KATOUZIAN Pouria s192865
HOUYON Manuel s203802

Project 2

INFO0027 : Tower Defense game

Université de Liège
Faculté des Sciences Appliquées
Année académique 2023-2024

1 Introduction

This project consists of developing a tower defense game coded in Java programming and employing different types of design patterns. To launch the project, simply run the commands make all then make run.

2 Choices of patterns

Based on several research and referencing the course INFO0027 - Programming Techniques, the patterns chosen for each class in this project are as follows :

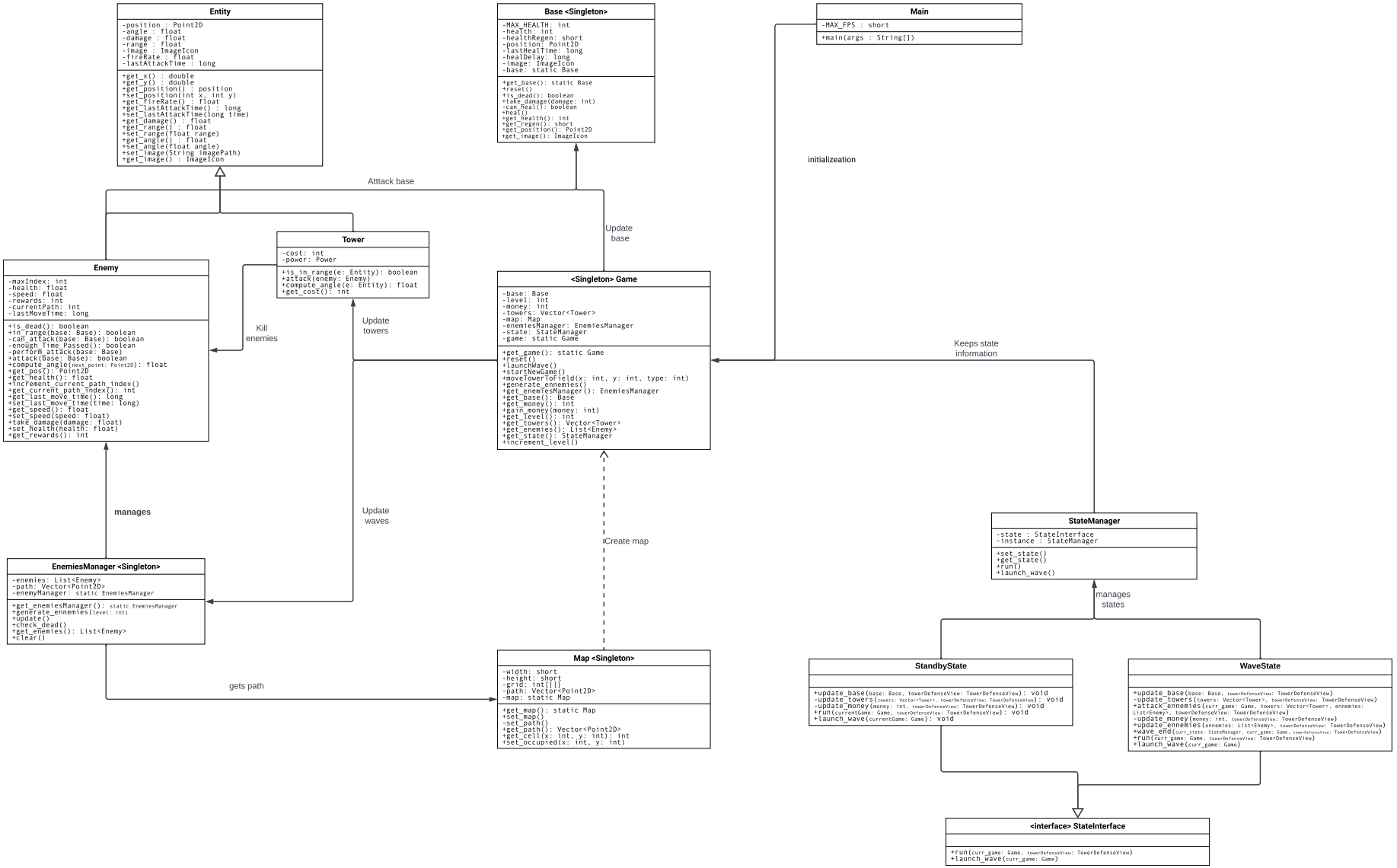
In the Base, EnemiesManager, Game, and Map classes, the singleton pattern is employed. The state pattern is used to implement the WaveState and StandbyState classes. also, some classes in this project, such as Tower and Enemy, are developed using the factory design pattern.

3 Feedback

This project proved to be quite interesting.Selecting the appropriate design pattern posed a challenge initially, prompting the need for some research and reviewing the course material.However, the implementation phase was relatively complicated. This project took 3 weeks to analyze the problem and finding the right path to develop and reach the final stage of the development.

4 UML

This report contains a UML diagram on the last page, which details the program's functionality.


Entity
-position : Point2D
-angle : float
-damage : float
-range : float
-image : ImageIcon
-fireRate : float
-lastAttackTime : long
+get_x() : double
+get_y() : double
+get_position() : position
+set_position(int x, int y)
+get_fireRate() : float
+get_lastAttackTime() : long
+set_lastAttackTime(long time)
+get_damage() : float
+get_range() : float
+set_range(float range)
+get_angle() : float
+set_angle(float angle)
+set_image(String imagePath)
+get_image() : ImageIcon
Base <Singleton>
-MAX_HEALTH: int
-health: int
-healthRegen: short
-position: Point2D
-lastHealTime: long
-healDelay: long
-image: ImageIcon
-base: static Base
+get_base(): static Base
+reset()
+is_dead(): boolean
+take_damage(damage: int)
-can_heal(): boolean
+heal()
+get_health(): int
+get_regen(): short
+get_position(): Point2D
+get_image(): ImageIcon
Main
-MAX_FPS : short
+main(args : String[])
initializeation
Atttack base
Update base
Tower
-cost: int
-power: Power
+is_in_range(e: Entity): boolean
+atEacK(enemy: Enemy)
+compute_angle(e: Entity): float
+get_cost(): int
Enemy
-maxIndex: int
-health: float
-speed: float
-rewards: int
-currentPath: int
-lastMoveTime: long
+is_dead(): boolean
+in_range(base: Base): boolean
-can_attack(base: Base): boolean
-enough_Time_Passed(): boolean
-perform_attack(base: Base)
+attack(Base: Base): boolean
+compute_angle(next_point: Point2D): float
+get_pos(): Point2D
+get_health(): float
+increment_current_path_index()
+get_current_path_index(): int
+get_last_move_time(): long
+set_last_move_time(time: long)
+get_speed(): float
+set_speed(speed: float)
+take_damage(damage: float)
+set_health(health: float)
+get_rewards(): int
Kill enemies
Update towers
<Singleton> Game
-base: Base
-level: int
-money: int
-towers: Vector<Tower>
-map: Map
-enemiesManager: EnemiesManager
-state: StateManager
-game: static Game
+get_game(): static Game
+reset()
+launchWave()
+startNewGame()
+moveTowerToField(x: int, y: int, type: int)
+generate_ennemies()
+get_enemiesManager(): EnemiesManager
+get_base(): Base
+get_money(): int
+gain_money(money: int)
+get_level(): int
+get_towers(): Vector<Tower>
+get_enemies(): List<Enemy>
+get_state(): StateManager
+increment_level()
Keeps state information
manages
Update waves
Create map
EnemiesManager <Singleton>
-enemies: List<Enemy>
-path: Vector<Point2D>
-enemyManager: static EnemiesManager
+get_enemiesManager(): static EnemiesManager
+generate_enemies(level: int)
+update()
+check_dead()
+get_enemies(): List<Enemy>
+clear()
gets path
Map <Singleton>
-width: short
-height: short
-grid: int[][]
-path: Vector<Point2D>
-map: static Map
+get_map(): static Map
+set_map()
-set_path()
+get_path(): Vector<Point2D>
+get_Cell(x: int, y: int): int
+set_occupied(x: int, y: int)
StateManager
-state : StateInterface
-instance : StateManager
+set_state()
+get_state()
+run()
+launch_wave()
manages states
StandbyState
+update_base(base: Base, towerDefenseView: TowerDefenseView): void
-update_towers(towers: Vector<Tower>, towerDefenseView: TowerDefenseView): void
-update_money(money: int, towerDefenseView: TowerDefenseView): void
+run(currentGame: Game, towerDefenseView: TowerDefenseView): void
+launch_wave(currentGame: Game): void
WaveState
+update_base(base: Base, towerDefenseView: TowerDefenseView)
-update_towers(towers: Vector<Tower>, towerDefenseView: TowerDefenseView)
+attack_ennemies(curr_game: Game, towers: Vector<Tower>, ennemies: List<Enemy>, towerDefenseView: TowerDefenseView)
-update_money(money: int, towerDefenseView: TowerDefenseView)
+update_enemies(ennemies: List<Enemy>, towerDefenseView: TowerDefenseView)
+wave_end(curr_state: StateManager, curr_game: Game, towerDefenseView: TowerDefenseView)
+run(curr_game: Game, towerDefenseView: TowerDefenseView)
+launch_wave(curr_game: Game)
<interface> StateInterface
+run(curr_game: Game, towerDefenseView: TowerDefenseView)
+launch_wave(curr_game: Game)