INFO1113 Report

Ruixiao Dai

1 Design Principles

This project follows key object-oriented design principles to ensure maintainability, modularity, and clarity. These principles include encapsulation and inheritance. Each class is designed to handle a specific aspect of the game, this design principle makes the project easier to maintain and manage.

2 Classes and Interface

App Class

This class is the main entry point for the game and handles the initialization and main game loop. It manages the setup of the terrain, tank objects, and other components read from configuration and level files. The `App` class is responsible for handling key input and drawing elements on the screen by calling the `draw` methods of other classes.

Terrain Class

The `Terrain` class represents the game terrain, handling the generation and rendering of the landscape. It ensures that the terrain interacts correctly with other game objects from the explosion from tanks and projectiles. Tree is a part of the terrain since it's always on top of the terrain.

TankPiece Class

The `TankPiece` class models the individual components of a tank. this allowed for a modular and detailed representation of the tank's structure. This class handles the movement including turret and tank, collision detection, and rendering of tank pieces.

Projectile Class

The `Projectile` class represents projectiles fired by tanks. It manages the physics, trajectory, collision detection, and damage calculation of projectiles. This class ensures that projectiles interact correctly with the terrain and other game objects.

Explosion Class

The `Explosion` class implements ExplosionInterface. It handles the visual and physical effects of explosions in the game. When a projectile hits a target, an `Explosion` instance is created to manage the impact explosion effects.

ExplosionInterface Interface

The `ExplosionInterface` interface defines the methods that any explosion-related class must implement. This interface ensures consistency and standardization for explosion behaviors across different classes.

Wind Class

The `*`Wind class simulates the wind effect in the game, affecting the trajectory of projectiles. It manages wind speed and direction, providing a dynamic challenge for players to consider when aiming their shots.

UI Class

The `UI` class handles the user interface elements of the game, including menus, score displays, and other on-screen information. It ensures that the game provides clear and accessible information to the player.

DamageCalculation Class

The `DamageCalculation` class is responsible for calculating the damage produced by projectiles and tanks. It considers factors like projectile type, and distance to determine the damage they caused.

Powerups and Extention

My extension is to implement powerups for INFO1113 and INFO9003. For INFO1113 powerups: When r is pressed, If the player has enough points, the hp attribute of that tank will be incremented by 20. I have a checkhp method to make sure the maximum hp is 100.

When f is pressed, if the player has enough points, the fuel attribute of that player will be incremented by 200.

For COMP9003 powerups: When designing the game, the damage calculation, damage to the terrain and explosion animations are set based on the explode radius I passed in. So when designing large projectiles, I only need to create a larger projectile animation, everything else is done by passing in a bigger explode radius to the method I have created. When press x, a boolean value will indicate the next projectile is a large projectile, and relevant messages will be prompted to notify the player, it will then create a large projectile when the player fires.

For additional parachutes, when p is pressed,if the player has enough points, the parachute attribute of that player will be incremented by 1.

3 Conclusion

Each class in this project adheres to object-oriented design principles, ensuring a well-structured and maintainable codebase. This makes the project easier to maintain and manage.