INFO1113: gremlins-mik Report

Mikael Sebastian Indrawan
SID: 520484597

Gremlins Game and features

Alongside the expected features outlined by the assignment task description and demonstration video, gremlins-mik implements a unique powerup and gameplay extension to the game core functionality.

> Powerup

Description
In gremlins-mik, a speed powerup is implemented where double speed is enabled for a brief period of 10 seconds. Its location is determined by a powerup tile with character **'P'** determined by the level .txt file layout. The tile is represented in-game by an empty podium in its inactive state. When active, a speedboots icon is displayed. The time interval for a powerup reset is dictated randomly, between 15 to 30 seconds.

Implementation
Implementation of the speed powerup is determined by level design. At the designer's disposal, one may wish to implement this feature by editing a level file and placing the **'P'** character in a desired tile location.

> Extension

Description
In gremlins-mik, the player possesses a new projectile at its disposal - the iceball. It is unable to break walls, but has the capacity to freeze gremlins for a period of 10 seconds. It can be utilised as a secondary projectile, or tactically as a method of collateral execution (multiple gremlins with one fireball). Frozen enemies eliminated by a subsequent fireball will no longer be frozen. The cooldown period of Iceball projectiles are equivalent to that of fireballs.

Implementation
No further implementation is required as it is embedded within the core functionality of the game. To utilise the iceball, press the **'W'** key to fire the projectile.

Game Configuration and Level Setup

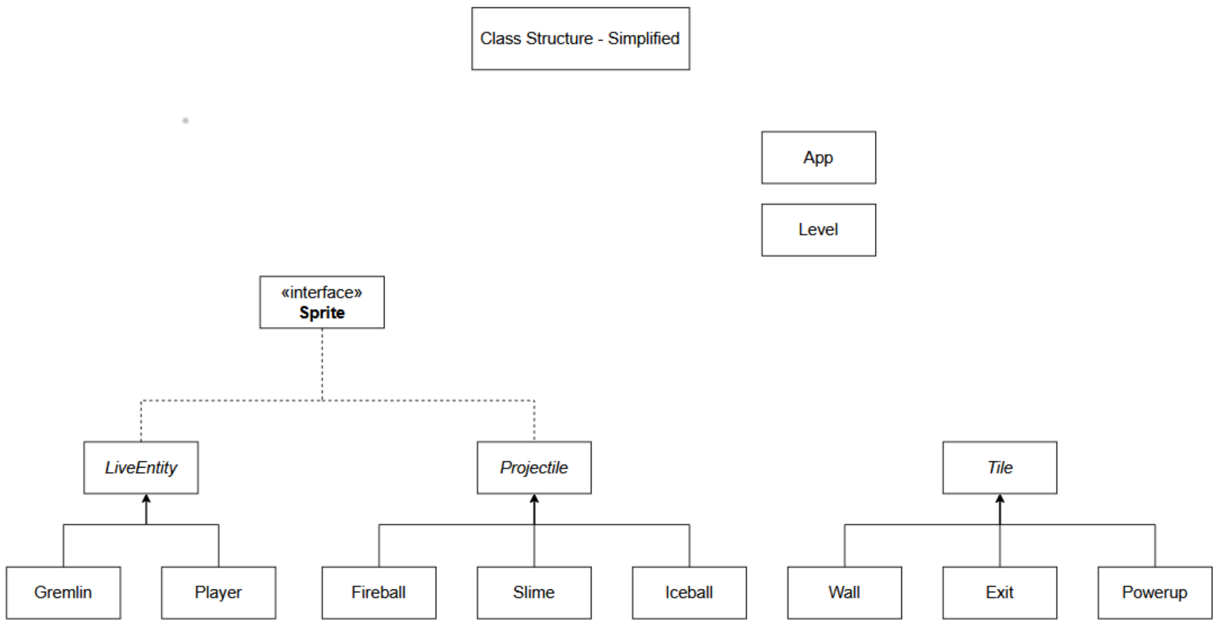
> Configuration

The game directory contains multiple files and folders that contain .json files for configuration and .txt files for levels. Please ignore the **TestConfig** and **TestLevels** directories as they are segregated for testing purposes. To ensure the implementation of desired levels, place your folders in the game root directory - the same directory as the default level1.txt and level2.txt level layouts are located. Either replace or add JSON objects within the **"levels"** JSON array, ensuring that the string associated with the **"layout"** key is the level layout .txt file name.

> Level Setup

Within each JSON object in the **"levels"** JSON array, the double values associated with the **"wizard_cooldown"** and **"enemy_cooldown"** keys represent the firing cooldown of the respective objects.

UML and Design



[fig 1.a: Simplification of UML Diagram Structure]
Please Consider the in-depth UML diagrams for more information.

> App Class

The App class, an extension Processing PApplet class, handles the graphical aspects of gremlins-mik. It is responsible for input handling, image loading, frame display and level-handling logic. It is not responsible for any logic at the *Level* scope, rather the overall game aspects - lives, levels and game-end handling.

> Level Class

The level class handles all in-game logic within the scope of the current level. It is responsible for sprite collision and detection, level layout .txt file parsing and loading, and the importing and updating of any entity (LiveEntity, Projectile and Tile). Hypothetically, all three entities can inherit from a parent, but the mere commonality of pixel-position location and PImage attributes does not warrant the implementation of an arbitrarily primitive class (Entity) for the sole sake of increased object-oriented abstraction.

> Sprite Interface

The Sprite interface embodies any game entity with the possibility of motion. As such, Tiles are separated from Sprite interface implementation due to their immobility. Likewise, it is through Sprite's motive capacity that it requires unique collision detection in the Level class. Whilst Sprite-Tile collision is handled through HashMap index checking, Sprite collision involves overlap and positional comparison. The differentiation of LiveEntity and Projectile abstract classes, however, lies in their contrasting behaviours.

- LiveEntity: creatures and multidirectionality within lifespan.
- Projectiles: '*lifeless'* and unidirectional in motion.

> Tiles

The Tile abstract class represents a static, immobile game entity that is contained in a tileMap. Tiles (any instances of Tile abstract class) compose the map of the level and, providing both a background and interactive plane for the Sprites. Whilst the foreground consists of Sprites, Tiles constitute the background.

Further Information

> Extended UML and Information

Please see the attached UML diagram pdf.