Gebze Technical University
Tugay Apaydın
1801042081

CSE443 - OOAD
Homework 1 - 2021

Report

Menu Controls and General Rules

The game menu comes with 2 button ’Start’ and ’Exit’. After starting the game, the game can be paused by pressing ’ESC’ button. Pausing the game will make a third button visible which is ’Resume’. At each pause, the main game thread is blocked and at each start, a new thread is created to run the game.

The character has 3 lives. If the character collapses to a monster, it will start over. At third failure, the game will be over and it must be started again from the menu.

Main Character Design

The main character has a virtual point which simulates that the character is running, and a real coordinate which represents that the character is not moving at all. The locations of the components in the enviroment are updated according to virtual point of the main character.

According to keyboard listening, main character is notified about the next move. If ’moveRight’ flag is set, the character’s horizontal movement can be updated. If the character is set to be jumping, the jump class (LowJump or HighJump) is notified.

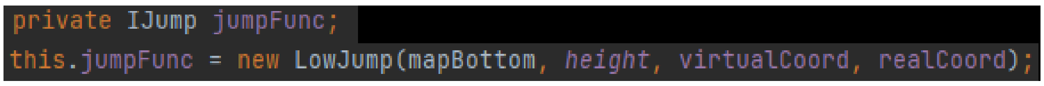
At each frame that is processed in the world thread, main character is updated for the next move (jump or move). This update function calls walk and jump functions from the main character.

Strategy Design Pattern and Jump Classes

There are 2 jump classes (LowJump and HighJump) and an interface IJump in the game. IJump includes jump function and some other helper functions.

According to strategy design, an instance of IJump interface used in the main character. The IJump instance is defined with the required jump class to use high jump or low jump abilities.

The jump function implementations inside the LowJump and HighJump classes are similar. At each update from the main world thread, the next update is handled. If the character is set to jump or it is already jumping or landing, the coordinate of the hero is edited accordingly.



```
private IJump jumpFunc;
this.jumpFunc = new LowJump(mapBottom, height, virtualCoord, realCoord);
```

Figura 1: Jump Definition

Monsters Class and Monster Creation

At the start of game, some number of monsters are created with randomly set coordinates. The total number of existing monsters and the minimum distance between each monster are defined by the programmer.

At each update in the world thread (main thread for game), the monster collision with the hero and monster pass is checked. If the hero passes a monster, the first monster in the list is removed and a new randomly created monster is added to the end of the list. At each update, all monsters are painted to the GUI.

Decorator Design Pattern and PowerUp Classes

Firstly, I designed the PowerUpCreator class that creates a randomly created PowerUp. The class also includes collapse and pass checks.

There are 3 decorator classes (DecoratorA, DecoratorB, DecoratorC). Each class implements IPowerUp interface and in the constructor, takes an instance of an object (IPowerUp basePu) that is declared as IPowerUp interface.

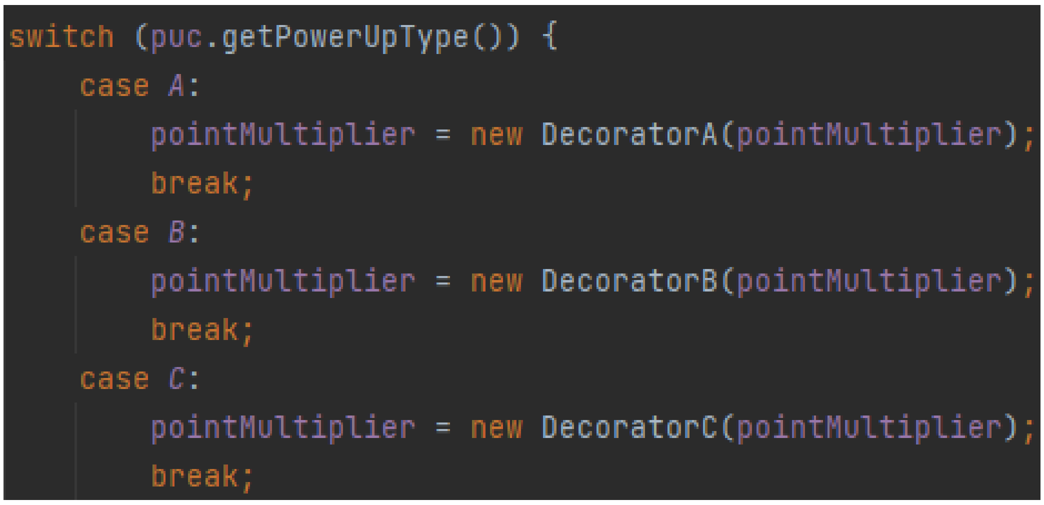
According to PowerUpOp function that is inherited from IPowerUp interface, each class multiplies the result of the basePu.PowerUpOp() with their multiplier value.

The base power up class (Multiplier), returns base_multiplier value which is defined at constructor.

Basically, at each power up collapse, the instance of IPowerUp pointMultiplier is sent to a decorator, and this decorator is hold in the main world class. By doing that, the decorator class operations are saved recursively at pointMultiplier instance.

Design of World Class and Game Environment

In the world class, the main thread is listening the keyboard and button actions and notifying the character for the next move. Besides, there is a separated thread which runs the game.



```
switch (puc.getPowerUpType()) {
    case A:
        pointMultiplier = new DecoratorA(pointMultiplier);
        break;
    case B:
        pointMultiplier = new DecoratorB(pointMultiplier);
        break;
    case C:
        pointMultiplier = new DecoratorC(pointMultiplier);
        break;
```

Figura 2: Jump Definition

The world class is runnable. After starting the game, it runs a loop in a thread which sleeps for 15 milliseconds at every iteration (that makes approximately 60-70 fps). At each iteration, the main character is updated, collapse checking for monster and power ups are done and all images (including background, fps counter, main character, menu etc.) are printed.