UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ARTES, CIÊNCIAS E HUMANIDADES
BACHARELADO EM SISTEMAS DE INFORMAÇÃO
COMPUTAÇÃO ORIENTADA A OBJETOS

CAIQUE ALVES DE SOUZA
GUSTAVO FERREIRA BOTELHO DE SENA
JÚLIA DU BOIS ARAÚJO SILVA
WILLIAM JUN OKINAKA SUZUKI

RELATÓRIO DE COMPUTAÇÃO ORIENTADA A OBJETOS

São Paulo - SP
2024

Sumário

1 Críticas ao código original

O código original possui o problema principal de ser procedural. Isso dificulta tanto a debugação quanto a expansão do código.

A debugação é dificultada porque, por ser procedural, as responsabilidades são misturadas na classe Main. Portanto, para modificar qualquer função, considerando a falta de encapsulamento, pode ser que seja necessário modificar todo o programa. Essa falta de encapsulamento também atrapalha a expansão do programa, pois adicionar novos inimigos ou elementos.

Portanto, a forma inicial do programa o torna desnecessariamente complexo, mas pode ser melhorada com a implementação de boas práticas de programação orientada a objetos, como os princípios SOLID (*Single Responsibility Principle*, *Open Closed Principle*, *Liskov Substitution Principle*, *Interface Segregation Principle* e *Dependency Inversion Principle*).

2 Descrição e justificativa para a nova estrutura de classes/interfaces adotada

A nova estrutura implementada utiliza as seguintes interfaces e classes:

- GameElement (abstrata)
- Player
- Enemy (abstrata)
- Enemy1
- Enemy2
- Enemy3
- Projectile
- HP
- Powerup
- Background
- Game (main)

Cada uma dessas é explicada a seguir:

2.1 GameElement

GameElement é a classe principal para herança de todos os elementos do jogo, e reúne os atributos de tamanho e estado. Além disso, essa interface reúne os getters e setters relacionados aos seus atributos.

Essa classe é utilizada como mãe por Player, Enemy, Projectile, HP e Powerup (ou seja, por tudo que define um elemento móvel com o qual exista interação).

Ela foi utilizada como herança principalmente por conta da facilidade da reutilização de código e pelo polimorfismo exigido pelas classes que a utilizam.

2.2 Player

Player é a classe que define tudo relacionado ao jogador. Essa classe herda os atributos de GameElement e define outros atributos e métodos úteis a ela. Esses são:

- Como atributos:

        - velocidade x e y
        - momento de inicio e fim da explosão
        - momento do próximo tiro
        - ativação do powerup
        - momento da última ativação do powerup
- Como métodos:
        - instanciação
        - setters:
            - setPowerupEnabled
            - resetLastPowerupStartTime
            - setNextShot
        - updateState
        - renderização

2.3 Enemy

Enemy é uma classe abstrata utilizada pelos diferentes tipos de inimigos. Essa interface herda atributos e métodos de GameElement e seu propósito é facilitar a criação de novos tipos de inimigos por meio da herança. Essa classe define os seguintes atributos e métodos:

- Como atributos:
        - velocidade
        - angulo
        - rotação
        - explosionStart e explosionEnd
        - nextShoot
- Como métodos:
        - instanciação
        - getters e setters:
            - explosionStart
            - explosionEnd

- velocidade
- ângulo
- velocidade de rotação
- nextShoot

2.3.1 Enemy1

A classe Enemy1 herda os métodos e atributos de Enemy. Essa classe caracteriza o inimigo representado por um círculo ciano, que possui como comportamento descender e atirar projéteis um a um em uma linha reta vertical para baixo, na direção do jogador.

2.3.2 Enemy2

A classe Enemy2 herda os métodos e atributos de Enemy. Essa classe caracteriza o inimigo representado por uma sequência de losangos magenta, que possui como comportamento descender e fazer uma curva, saindo pela lateral, enquanto atira projéteis múltiplos em ângulo.

2.3.3 Enemy3

A classe Enemy3 herda os métodos e atributos de Enemy. Essa classe caracteriza o inimigo representado por um círculo amarelo, que possui como comportamento descender em zigue-zague e atirar dois projéteis em ângulo. A implementação desse inimigo é descrita em mais detalhes na seção 4.3 deste relatório.

2.4 Projectile

A classe Projectile herda os atributos e métodos de GameElement. Essa classe caracteriza os projéteis tanto do jogador quanto dos inimigos. Seus atributos e métodos são:

- Como atributos:
  - velocidade no eixo x

        - velocidade no eixo y

- Como métodos:

    - instanciação
    - getters e setters:
        - velocidade no eixo x
        - velocidade no eixo y
    - updateStateP e updateStateE
    - renderP e renderE

2.5 HP

A classe HP herda os atributos e métodos de GameElement. Essa classe caracteriza o comportamento da barra de HP, representada por três corações no canto superior esquerdo, que desaparecem um a um à medida que o jogador é atingido por projéteis e inimigos. Quando essa barra é zerada, o jogo acaba. Seus atributos e métodos próprios são:

- Como atributos:

    - hp

- Como métodos:

    - instanciação
    - getter e setter de hp
    - redução
    - renderização

Mais sobre o processo de implementação da classe HP é discutido na seção 4.1.

2.6 Powerup

A classe Powerup herda os atributos e métodos de GameElement. Essa classe implementa apenas novos métodos:

- instanciação
- renderização

- localização e momento de renderização

Mais sobre o processo de implementação da classe Powerup é discutido na seção 4.2.

2.7 Background

A classe Background apenas caracteriza a geração do plano de fundo, em duas renderizações diferentes. Ela possui como atributos tudo o que modifica as estrelas utilizadas, como velocidade, contagem, localizações, tamanho e cor; e possui como métodos apenas a instanciação e a renderização.

2.8 Game

A classe Game caracteriza o funcionamento do jogo em si. Ela organiza as coleções de inimigos e projéteis e o momento de renderização dos múltiplos elementos. Além disso, essa classe processa o input do jogador.

3 Descrição de como as coleções Java foram utilizadas para substituir o uso de arrays

3.1 Java Collections

O Java Collections Framework é um conjunto de classes que implementam diversas estruturas de dados clássicas, como listas, filas, árvores e hashing, organizadas em uma estrutura unificada. Essas estruturas são implementadas da forma dinâmica mais eficiente possível e de forma que troca de uma por outra causa o mínimo possível de impacto no código.

As principais interfaces desse framework são Collection, List, Set, SortedSet e Map, e as principais classes são ArrayList, LinkedList, HashSet, TreeSet, HashMap e TreeMap.

3.2 Como foram utilizadas

No novo código, para substituir os arrays, foi utilizada a classe ArrayList do Java Collections Framework. Essa classe foi selecionada por conta de sua facilidade de uso, que ajuda a inserção e remoção fácil dos elementos necessários e permite a alteração das classes que nela são utilizadas.

4 Descrição de como as novas funcionalidades foram implementadas e como o código orientado a objetos ajudou neste sentido

4.1 HP

A classe HP foi implementada como filha da classe GameElement para evitar a repetição de código. Ela é uma classe simples, com apenas um atributo (hp), que possui um getter e um setter, e dois métodos simples para renderização e manipulação desse atributo. Por conta da utilização dos princípios SOLID, sua implementação foi simples.

Esse elemento é renderizado como três corações, das cores verde, amarela e vermelha, localizados no canto superior esquerdo da tela. À medida que o jogador é atingido e o hp diminui, os corações desaparecem. Quando o hp chega em 0, se o jogador for atingido, o jogo é encerrado.

4.2 Powerup

A classe Powerup é filha de GameElement. O Powerup é caracterizado por um losango preto com bordas brancas. Quando ele é coletado, o jogador fica com a capacidade de lançar projéteis laranjas, que são mais velozes que os projéteis normais. Sua implementação foi facilitada pelo encapsulamento do código, que permitiu que não fosse necessário alterar múltiplos arquivos para que essa funcionalidade operasse.

4.3 Enemy3

A classe Enemy3 é filha da classe Enemy. Esse novo tipo de inimigo é caracterizado por um círculo amarelo que percorre a tela de cima para baixo em zigue-zague, atirando projéteis em pares e em ângulo.

Sua implementação foi facilitada por meio da herança em relação tanto à classe Enemy quanto à classe GameElement, que permitiu a reutilização de código comum e a implementação do novo inimigo em métodos já existentes, tais como o método checkCollisions, além da interação com classes já existentes, como a classe Projectile.