MAC0218 - Técnicas de Programação II
EP Galaxy Raiders - Parte 2

Alfredo Goldman, Cainã Setti, Dayanne Gomes e Renato Cordeiro

Instituto de Matemática e Estatística – USP

21 de maio de 2023

1 Introdução

Galaxy Raiders é um jogo inspirado por clássicos da Era de Ouro dos Arcades: Space Invaders (1978), Star Raiders (1979), Asteroids (1979) e Galaga (1981). Esses jogos se popularizaram graças aos fliperamas (máquinas especializadas para jogos eletrônicos) e aos primeiros videogames (dispositivos caseiros para jogos eletrônicos), introduzidos pela empresa americana Atari.

No ambiente espacial onde o Galaxy Raiders está inserido, o jogador precisa controlar a nave para desviar de objetos que possam colidir na nave e causar dano na sua estrutura e também controlar os canhões de laser para lançar projéteis para destruir os asteroides.

O jogo é composto por duas partes. Um componente para visualização do jogo, chamado de front-end ou interface Web, onde o usuário consegue interagir com o jogo; e um componente de lógica do jogo, chamado de back-end ou API, onde o motor do jogo processa os comandos do usuário, atualiza o estado do jogo, e o envia para visualizar. O repositório do projeto pode ser encontrado em `https://github.com/galaxy-raiders`. Nesta parte do EP, a API será explorada.

A API foi estruturada utilizando o estilo arquitetural hexagonal. Este estilo arquitetural propõe desacoplar a lógica de negócio com dependências externas utilizando estruturas conhecidas como portas e adaptadores. A estrutura de pastas do código está dividida em três pacotes:

- Pacote core, com a lógica do jogo. Ele é subdividido em dois pacotes para as funcionalidades de física (core.physics) e do jogo (core.game).
- Pacote ports, com as interfaces para dependências externas, fornecendo um ponto de testabilidade para a lógica do jogo. Ele é subdividido num pacote com portas para a interface do usuário (ports.ui).
- Pacote adapters, com a implementação das dependências externas. Ele é subdividido em dois pacotes que implementam uma interface de texto (adapters.tui) e uma interface web (adapters.web) para interagir com o jogo.

2 Tarefa

Esta parte do EP aborda herança de classe e persistência de dados. Novamente, para o desenvolvimento do EP, deve-se atentar à utilização de Docker. A documentação inclui um README com instruções para execução. O front-end encontra-se no repositório `https://github.com/galaxy-raiders/galaxy-raiders-web`. Não é necessário alterá-lo.

2.1 Herança de classe

Faremos a extensão da classe SpaceObject para criar a classe Explosion. Essa classe tem como função adicionar efeitos visuais de explosões ao acertar um asteroide. A classe deve se chamar Explosion. Você pode incluir um parâmetro booleano (is_triggered) para indicar que a explosão já aconteceu ou não. A escolha dos parâmetros e das funções é livre. No entanto, a explosão deve acontecer quando o projétil acertar um asteroide e depois desaparecer. A informação da explosão é passada pelo front-end por um JSON. Portanto, é necessário modificar partes da API no pacote core.game.

Para verificar a funcionalidade da classe Explosion, a criação de testes automatizados similares aos implementados para outras classes é recomendado. Ao criar uma classe é importante estender e compatibilizar os testes já existentes. Além disso, é recomendável adicionar novos testes novos onde for necessário. Confira o arquivo app/build/reports/jacoco/test/html/index.html para verificar a cobertura de testes.

2.2 Persistência de dados

O asteroide (/core/game/Asteroid.kt) possui os atributos `radius` e `mass`, que devem ser utilizados para atribuir uma pontuação ao acertar um asteroide. Essa pontuação deve variar de acordo com o raio e a massa, cuja lógica de cálculo deve ser definida pelo aluno. Você deve criar uma pasta /core/score que deve conter os arquivos Scoreboard.json e Leaderboard.json, estando ambos no formato JSON. O arquivo Scoreboard.json deve conter o histórico de todas as execuções do jogo, com a data e hora de início, pontuação final e quantidade de asteroides destruídos. Já o arquivo Leaderboard.json deve conter as informações das três melhores partidas, ranqueadas pela pontuação final.

3 Entrega

O EP pode ser feito em até duas pessoas. A entrega deve feita por uma url do repositório de git. Cada pessoa deve fazer um fork do repositório `https://github.com/galaxy-raiders/galaxy-raiders-api` e colocar a url do repositório no eDisciplinas. Caso seja feito em dupla, cada um dos integrantes deve fazer sua entrega no sistema, indicando qual é sua dupla.

No repositório é necessário criar uma branch p2, que deve conter as seguintes modificações:

- a criação da classe Explosion dentro da pasta /core/game/. O arquivo precisará ser nomeado Explosion.kt. Essa inclusão exigirá alterar o código do motor do jogo e conseguir mandar o dado das explosões dentro do JSON para o front-end;
- a criação da lógica de atribuição de pontuação, que deve criar os arquivos Scoreboard.json e Leaderboard.json dentro pasta /core/score.

4 Avaliação

O EP será avaliado a partir da implementação do código de cada parte da tarefa. A segunda parte (p2), de herança de classe e persistência de dados, considera os pontos:

- Código compila sem erros e warnings: 2.0
- Boas práticas de programação e clareza do código: 2.0
- Código executa sem erros e produz os resultados desejados: 6.0