Manual do Sistema

Projeto Space Invaders – parte2
SCC 604 – Programação Orientada a Objetos

Murilo Mussatto

11234245

1. Introdução

O sistema está divido em cinco "*packages*": *spaceinvaders, Assets*, *ElementosDoSistema*, *Engine* e *InterfaceGrafica*; que serão descritos a seguir. Para facilitar a compreensão, durante esse manual a cor azul foi utilizada para designar métodos e a cor verde para classes.

2. Packages

2.1. spaceinvaders

Esse *package* contém apenas o método *main* que é chamado na hora que o programa é executado. Esse método apenas inicia a classe *GUI* que está presente no *package InterfaceGrafica.*

2.2. Assets

Aqui é o local onde estão armazenadas todas as imagens em formato .png que são utilizadas na execução do programa.

2.3. ElementosDoSistema

Nesse *package* estão armazenadas todas as classes que definem entidades no jogo, inclusive a classe abstrata "Entity.java". São entidades: o canhão, os inimigos, as barreiras e os tiros disparados. Dentro dessas classes estão os métodos que definem as características que determinam o comportamento dessas entidades no jogo, como por exemplo a sua velocidade, o número de vidas do canhão, o método para atirar e o método para renderizar as imagens no jogo.

2.4. *Engine*

Esse *package* inclui a classe "Game.java" que controla a maior parte das funções do jogo. É nela onde são criados os inimigos, as barreiras e o canhão, além de determinar a movimentação deles, a contagem de pontos e as condições de fim-de-jogo.

2.5. *InterfaceGrafica*

Esse *package* controla toda a interação com o usuário por meio da interface gráfica. Aqui é onde está armazenada a classe "GUI.java" que implementa a interface utilizando o JavaFX.

3. Funcionamento e Interface

Quando o programa é executado, o método *main* inicializa a classe GUI que controla a interface gráfica, criando um *canvas* onde serão desenhados os objetos e abrindo a janela do jogo. Dentro desta classe, é instanciado um objeto da classe Game e o método *IniciaGame* é chamado. Esse método é responsável por criar todas as entidades que serão utilizadas no jogo.

Essa criação é dada da seguinte forma: primeiro instancia-se um objeto da classe Tela, que contra as funções de imprimir mensagens e limpar a janela do jogo. Então, um objeto da classe Canhao é instanciado, ele será utilizado pelo jogador para derrotar os inimigos. Depois, cria-se um *ArrayList* onde são armazenados os 55 objetos da classe Invasor, que servirão como os inimigos do jogo. Por fim, os conjuntos de objetos da classe Barreira são armazenados em mais um vetor do tipo *ArrayList*.

Após a finalização do método IniciaGame, cria-se uma *InnerClass* do tipo *AnimationTimer* onde será executado o loop principal do jogo. Esse loop consiste em: limpar a janela do jogo, verificar as teclas pressionadas pelo jogador, movimentar as entidades por meio da classe MoveEntities da classe Game, desenhar as entidades na janela por meio do método DesenhaEntidades, verificar colisões entre as entidades pelo método Colisao e, por fim, mostrar as informações atuais do jogo pelo método MostraInfo.

Além disso, dentro do loop é chama a função VerificaFim para chegar se alguma das condições de fim-de-jogo foram satisfeitas. Caso isso tenha acontecido, é chamado o método GameOver e o jogo termina.

4. Entidades

Todas as entidades utilizadas no jogo herdam características da classe mãe “Entity.java”. Nessa classe é onde estão os atributos comuns a todas as outras entidades, como posição (tanto na coordenada X quanto na Y) e velocidade. Além disso, essa classe possui um atributo da classe Image do JavaFx, que é utilizada para guardar a imagem .png que representa cada entidade no jogo. Em complemento, a classe Entity implementa três métodos muito importantes para o funcionamento da interface gráfica.

O primeiro deles é o método render, responsável por imprimir a imagem na tela do jogo em uma determinada coordenada. Ele utiliza um método da classe Image chamado drawImage. Além disso, ele é chamado pelo método DesenhaEntidades da classe Game na hora de desenhar as imagens na tela.

O segundo método é o update, que atualiza a posição das entidades tomando como referência sua posição atual e sua velocidade. Ele é chamado pelo método MoveEntidades da classe Game.

Por fim, o método intersects é utilizado para verificar se houve a colisão de uma entidade com a outra. O método Colisao da classe Game chama o intersects para todas as entidades presentes no jogo.

4.1. Invasores

Os invasores são a classe filha de entidades mais complexa. Existem três tipos de invasores que são diferenciados por um atributo do tipo inteiro presente na classe. Cada tipo de invasor possui sua própria imagem .png. Assim, no construtor da classe é passado o tipo do invasor sendo criado e seleciona-se automaticamente o nome da imagem que será usado na criação do atributo do tipo Image presente na superclasse. O tipo do invasor também determina a pontuação que é atribuída ao jogador quando este atinge um inimigo.

Além disso, essa classe possui um método chamado move que determina a movimentação dos invasores. É passado um booleano para essa classe especificando se o movimento deve ser horizontal ou vertical. Caso o movimento seja vertical, o método modifica a velocidade da entidade nos eixos coordenados de forma que ela apenas se mova no eixo Y. Caso o movimento seja horizontal, um atributo de controle chamado “Direcao” determina se o invasor deve se mover para a direita ou para a esquerda.

Essa classe também possui um atributo booleano chamado Morto que determina se o objeto deve ser impresso na tela do jogo. Caso o jogador tenha atingido o invasor, esse atributo se torna verdadeiro, impedindo que ele seja impresso.

5. Classe Game

A classe Game presente no *package* *Engine* é a mais completa dentre todas as classes do código. Ela controla a grande maioria das funções do jogo. É nela onde são instanciadas as entidades e onde são contabilizados os pontos. Como previamente citado, a classe GUI instancia um objeto dessa classe. Assim, por meio da classe Game, é possível controlar a criação, movimentação, colisão, disparo e renderização das entidades do jogo. A criação é feita pelos métodos do tipo "load", a movimentação pelos métodos do tipo "move", a colisão pelo método Colisao, os disparos pelos métodos do tipo "atira" e a renderização pelos métodos do tipo "desenha".

Além disso, essa classe possui métodos auxiliares que são utilizados em várias partes do código como por exemplo o método LimpaTiro, responsável por excluir todos os objetos do vetor de tiros quando o jogador é atingido.

Por fim, essa classe também é utilizada para mostrar informações pertinentes para o jogador, como a contagem de pontos e o número de vidas restantes, utilizando o método MostraInfo e verificar as condições de fim de jogo pelo método VerificaFim.