Universidade de São Paulo-USP
Instituto de Ciências Matemáticas e de Computação-ICMC

Space Invaders: Manual do sistema

Mireli Damaceno Barbosa NºUSP: 11200293

São Carlos, 20 de Dezembro de 2020

1. Introdução

Neste manual será abordado o funcionamento interno do jogo Space Invaders desenvolvido como projeto trabalho da disciplina Programação Orientada a Objetos. Ou seja, como o código fonte foi estruturado, quais são as classes e como elas funcionam, qual foi a estratégia empregada para o movimento dos invasores etc. Para isso, o manual foi dividido em tópicos:

- Estruturação geral: A classe Elemento
- Movimentação dos inimigos
- Barreira em pixels
- Menus e Imagens
- Som

2. Estruturação geral: A classe Elemento

Um jogo é formado, dentre outras coisas, por elementos que têm aspectos em comum como posição, velocidade, tamanho e ações. Pensando nisso, foi criada uma classe especial no jogo chamada Elemento. Ela é a classe mãe das classes Canhão, Invasor, Barreira e Tiro, ou seja, essas classes estendem as características da Elemento. Assim, além de reaproveitarmos os comportamentos e atributos em comum, muitos métodos precisarão simplesmente saber que nosso objeto é um Elemento.

3. Movimentação dos inimigos

Cada linha tem um tipo de inimigo, usamos o array **tipoPorLinha**, que armazena os valores da enum definida na classe Invasor para controlar isso. Os inimigos são posicionados um ao lado do outro e o posicionamento horizontal leva em conta a largura de cada mais um espaçamento adicional. O mesmo é feito para posicioná-los em linha, deixando espaço para escrevermos a pontuação, a quantidade de vidas do canhão e a fase corrente no topo da tela. Utilizamos um contador para controlar a marcha dos invasores. Quanto menor for o número de inimigos, menor será o tempo de espera para a movimentação deles: a cada loop do jogo (frame), incrementamos a variável contador e, quando ela for maior que **contadorVelocidade**, que foi definido como sendo o quádruplo do total de invasores somado de 60 subtraindo a vigésima parte do total de inimigos destruídos e o dobro da fase, movemos os invasores. Então, a cada avanço de nível, o jogo fica mais difícil. Essa escolha foi feita depois de alguns testes na velocidade com que os inimigos se movem e pensando, principalmente, nas chances de o jogador passar de fase.

Na hora de mover os inimigos há algo importante a ser considerado: o sentido do movimento. Por isso, é necessário fazer a verificação se o movimento será na vertical (nova linha) ou na horizontal para esquerda ou para direita dependendo do valor de uma variável criada na classe principal, a classe Jogo, chamada **direcao**. Essa variável pode assumir o valor 1 para o movimento para direita e -1 para o movimento para esquerda. Caso não seja uma nova linha e a colisão com as bordas da tela ainda

não tenha sido detectada, verificamos se o próximo movimento resultará em uma colisão.

Por fim, verificamos se houve movimentação com nova linha, ou movimentação e colisão. Para movimentação com nova linha, invertemos a direção com valor positivo vai da esquerda para direita e, com valor negativo, da direita para esquerda) e desativamos a nova linha. Caso a segunda verificação seja verdadeira, a próxima movimentação será para uma nova linha

4. Barreiras em pixels

Para que os bloquinhos das barreiras possam ser destruídos aos poucos pelos tiros dos invasores ou pelos tiros do canhão foi criada uma matriz de 0 e 1 que representa o formato de cada barreira. Onde há 0, é preenchido com preto (a cor original da tela) e onde há 1, é preenchido com verde, a cor definida para a barreira. São 4 barreiras no total, distribuídas pela tela. Elas são nós de Group. Há um método para defini-las chamado **defineBarreiras** logo no começo da classe principal e na classe Barreira há um método chamado **destroiBlocos** para deletar cada pixel caso haja uma colisão com uma delas.

5. Menus e Imagens

Os menus foram implementados para o jogo ficar mais ilustrativo, favorecendo a interação com o usuário. Eles fornecem opções a serem tomadas como começar o jogo, sair do jogo ou jogar novamente. Essas opções são na verdade botões que executarão uma determinada ação, dependendo da escolha do jogador.

Além disso, há labels para que o usuário sempre saiba como está o andamento do jogo: qual a pontuação feita até um determinado momento, quantas vidas restam, qual a fase atingida, estado de game over etc.

6. Som

A classe som foi feita como um adicional para deixar o jogo mais dinâmico, mais divertido e para simular melhor uma luta no espaço contra inimigos que querem invadir a Terra. No jogo há uma música geral que é tocada assim que o jogo inicia. Ela foi escolhida pensando em um contexto de guerra em jogos de videogame. Há também um som de tiro que é executado toda vez que o canhão atira. Assim, além de o jogador ver o tiro subindo na vertical, ele pode ouvir uma simulação da ação.