Relatório de Projeto Final
Paradigmas de Programação
UFABC - Universidade Federal do ABC
2021
Gabriel de Mello Fiali
RA - 11201721286

Link do vídeo:
https://drive.google.com/file/d/1QUNA4tNgwBZ7CsJ9cdH7CzvILUvLhs8s/view?usp=sharing

1. Introdução

Este projeto se baseou na adaptação e elaboração do jogo clássico Space Invaders na linguagem funcional Haskell, levando em conta suas particularidades para ser implementado.

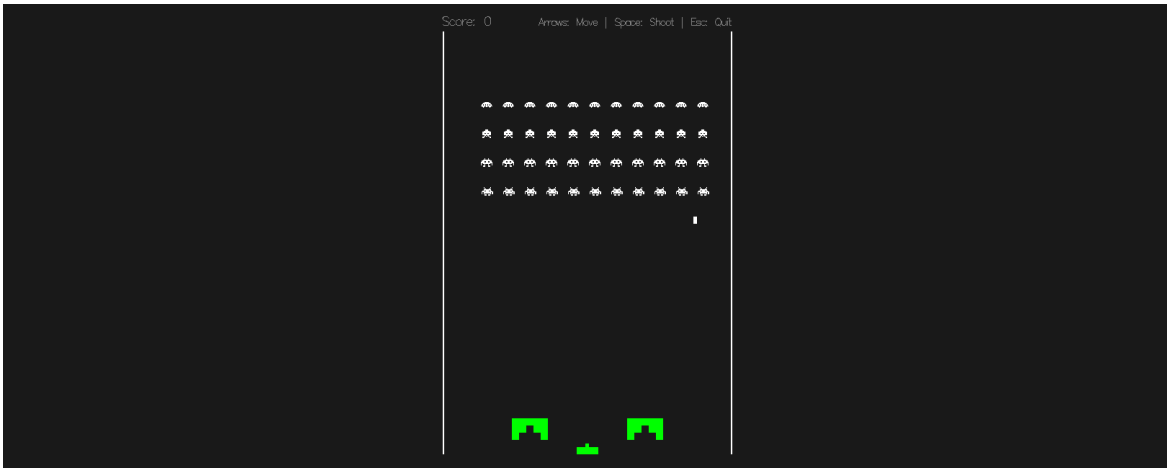

Figura 1: *Captura de tela apresentando o jogo implementado.*

2. Space Invaders

O jogo consiste em linhas de inimigos que se movimentam em um padrão “S”, movendo-se horizontalmente até encontrarem uma borda do espaço de jogo, em seguida descendo verticalmente e repetindo o movimento lateral, agora invertendo sua direção. Esse padrão se repete até um patamar, que culmina com um inimigo “pousando” e assim definindo o fim de jogo.

O objetivo do jogador então é não permitir que esses inimigos alcancem esse patamar, dispondo da capacidade de se movimentar horizontalmente e de atirar um projétil por vez, que se move verticalmente para cima e é capaz de eliminar os mesmos. O jogo é vencido com a eliminação de todos os inimigos da tela.

Outra característica dos mesmos, entretanto, é sua capacidade de também atirar projéteis verticalmente, mas para baixo, que podem eliminar o jogador e definir o fim do jogo caso entrem em contato com o mesmo. Como anteriormente, limitam-se a um projétil atirado por vez, mas com a escolha do inimigo que irá dispará-lo sendo aleatória.

Assim, como ferramenta extra para evitar os disparos, o jogador conta com o auxílio de duas barreiras entre o mesmo e os inimigos para servirem de proteção. Estas, entretanto, podem ser destruídas parcial ou totalmente com os disparos de ambas as entidades.

O jogo também conta com dificuldade dinâmica e um placar. O primeiro consiste no aumento da velocidade de movimento dos inimigos conforme vão sendo eliminados, com níveis de incremento diferentes dependendo dessa quantidade e o segundo consiste num incremento de 10 pontos para cada inimigo eliminado, com esse valor sendo cumulativo entre partidas vencidas ou reiniciando em caso de fim de jogo.

3. Detalhes da Implementação

A implementação em Haskell foi feita com o uso da ferramenta *Stack* e da biblioteca *gloss*, para elaborar os gráficos vetoriais.

A estruturação do jogo em si consiste em um Data Type *WorldState*, responsável por armazenar os parâmetros de jogo: modelos dos alienígenas recebidos a partir de ".bmp"s salvos; alienígenas em tela armazenados em uma estrutura *Zipper*; índice de linha de alienígenas atual a ser atualizada (como forma de replicar o comportamento do jogo original); jogador; barreiras, também em uma estrutura *Zipper*; projéteis disparados pelo jogador; projéteis disparados pelos alienígenas, numa tupla com o tipo *StdGen* para simular a aleatoriedade nos disparos; velocidade do jogo; placar; flags de fim de jogo e condição de vitória; e índice de rodadas.

A cada ciclo de execução, esses valores vão sendo atualizados conforme as especificações da função "*play"* do *gloss*, através de uma função de atualização que retorna a nova definição de *WorldState*. Esse processo gira em torno de transladar os elementos móveis na forma de inimigos e projéteis, conferir colisões entre os mesmos e barreiras, definir o placar de acordo e fazer a checagem das condições de fim de jogo e vitória. Isso é feito por funções individuais que se responsabilizam por checar cada um desses casos. Como destaques há a geração de números aleatórios através da biblioteca *System.Random*, para a definição do inimigo que irá efetuar o disparo do projétil e o retorno do novo seeder para a operação futura; e o modelo de colisões baseado em tuplas com booleanos, inspirado pelo projeto *Haskeroids* de Artur Henrique Allen Santos, como forma de marcar os elementos a serem removidos após seu contato. Este último se mostrou a maior dificuldade inicialmente, pois a mudança de ciclo não estava permitindo manter a referência dos elementos em contato que deveriam ser eliminados e o sistema de marcação se mostrou uma solução para esse problema.

Após a definição inicial e cada atualização subsequente, os modelos são renderizados em tela com uma função para converter os mesmos no tipo *Picture*. Isso é feito com o auxílio de um conjunto de outras funções, no qual cada uma responsável por renderizar esses elementos, que são concatenados em um array na função principal antes da conversão.

Finalmente, conta com uma função para capturar a interação do usuário com o jogo, na forma do movimento do jogador, disparo de projéteis e reinício de jogo. Isso é feito com um manipulador de eventos, que se aproveita de condicionais e dos parâmetros atuais para realizar o retorno de um novo *WorldState* para ser traduzido e renderizado.

4. Execução do jogo

Após realizar o build do projeto com o Stack, executar `stack run`.

As **setas** definem o movimento horizontal, a **barra de espaço** efetua os disparos, a tecla **Enter** é utilizada para reinício do jogo ao final e **Esc** encerra o mesmo.