Universidade Federal de São Carlos – UFSCar

Bacharelado em Ciência da Computação

Estrutura de Dados

Professor: Ferrari

Jogo: JvAsteroids

Gabrieli Santos, 726523, gabrielisantos17.gs@gmail.com

João Gabriel Barbirato, 726546, e-mail: joaobarbirato@gmail.com

Leonardo de Oliveira Peralta, 726556, leonardo98ti@gmail.com

Sumário

1-Introdução

Uma das melhores formas de absorver o conteúdo ensinado em sala de aula é realizar atividades práticas, que proporcionam desafios aos alunos. Esse tipo de atividade faz com que o aprendizado seja muito mais interessante e eficiente.

Esse projeto tem exatamente esse objetivo. Sua proposta é programar um jogo que use o conteúdo ensinado nas aulas de Estruturas de Dados, ministradas no curso de Bacharelado de Ciência da Computação, na Universidade Federal de São Carlos.

No decorrer da disciplina, foram apresentados aos alunos, os conceitos de fila, pilha, lista e árvore, e como implementá-los em situações reais. Através disso, o terceiro desafio foi escolher, em grupo, uma dessas estruturas para desenvolver um jogo.

Uma fila é uma estrutura que permite a inserção e remoção de objetos em um vetor e que segue a regra "O primeiro elemento que entra é o primeiro que sai". Uma pilha é uma estrutura que permite o empilhamento e o desempilhamento de objetos e segue a regra "O primeiro elemento que entra é o último que sai". Uma lista é uma estrutura que permite que um elemento seja inserido ou retirado em qualquer posição; é muito utilizado para guardar informações de funcionários em uma empresa. Uma árvore é um tipo de estrutura mais complexa, que tem por objetivo auxiliar na busca por dados, quando a quantidade de informação é muito grande; quando se trata de uma árvore balanceada de busca, ela segue algumas regras básicas que, se seguidas, permitem a otimização da busca por dados armazenados na mesma.

Além da estrutura do jogo, é necessário trabalhar com uma interface gráfica, para que o projeto fique mais dinâmico e divertido. Para esse projeto, a biblioteca escolhida para elaborar a interface gráfica, foi o SFML.

O programa feito pelo grupo é baseado no jogo "*Asteroids*" e adaptado para se enquadrar no ambiente e cenário do filme "*Tron: O Legado*" de Joseph Kosinski. O jogo consiste no jogador principal que controla uma nave. Essa nave roda para todos os lados, caminha por todo o campo e atira. Seu objetivo é eliminar todas as naves inimigas com seus tiros e impedir que essas naves o atinjam antes. Caso isso aconteça, o jogo termina.

As estruturas escolhidas para o jogo foram a Fila, cujo conceito é "O primeiro elemento que entra é o último que sai", e a Lista, cujo conceito é uma estrutura que permite que um elemento seja inserido ou retirado em qualquer posição.

2-Desenvolvimento

a) Prints da execução

A tela inicial do jogo apresenta as seguintes opções: Jogar, Regras e Sair.

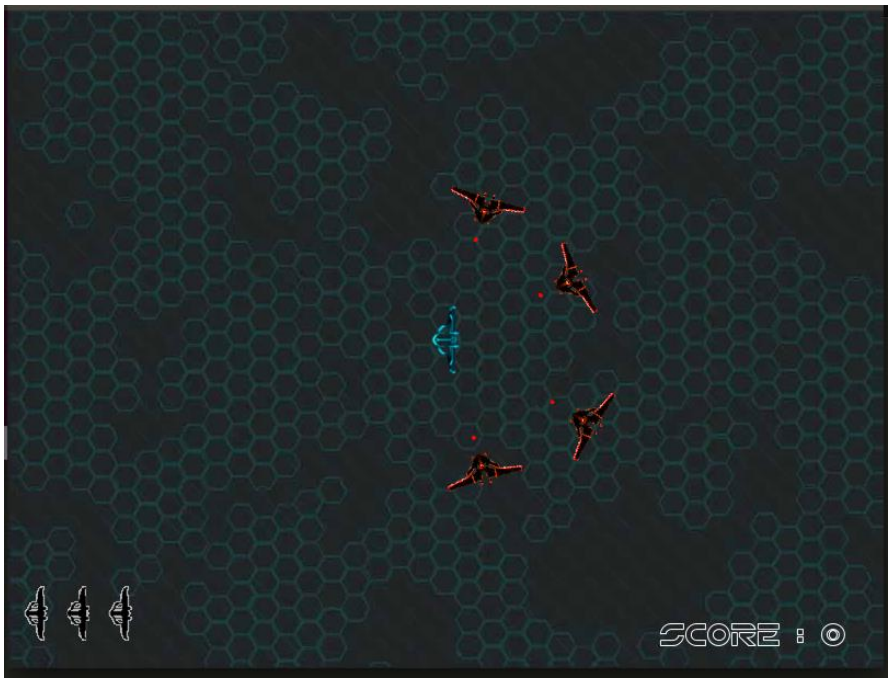
Ao clicar em regras, a tela que aparece para o jogador é a seguinte:

O jogo precisa de apenas um usuário.

As teclas de controle são:

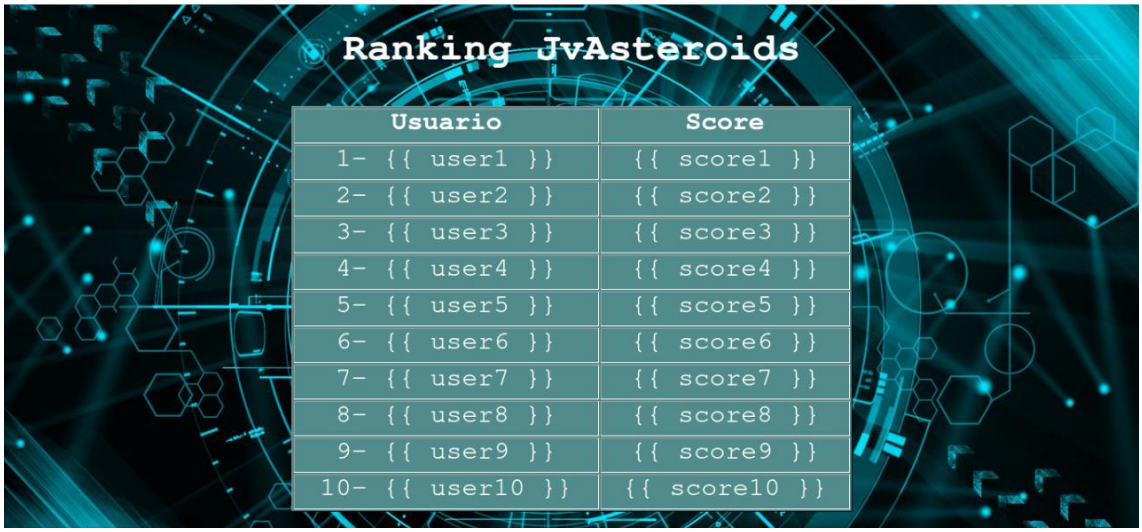
- Tecla “*Up*” do teclado: é responsável por fazer toda a movimentação da nave pelo cenário. A nave não se limita aos lados da tela, podendo aparecer do lado oposto quando atingir uma das “paredes”.
- Tecla “*Left*”: é responsável por girar a nave no sentido anti horário.
- Tecla “*Right*” é responsável por girar a nave no sentido horário.
- Tecla “*Space*” é responsável por atirar.

Ao voltar para o Menu inicial, o jogador pode selecionar a opção “Jogar” e iniciar a partida. A partida se inicia com a seguinte interface:



O jogador já começa enfrentando quatro naves inimigas. Se o tiro de uma delas acertar a nave do jogador, ela perde uma “vida”. Se ela perder três “vidas”, o jogo termina e a imagem que aparece está representada abaixo. Para ter controle sobre a quantidade de vidas restantes, basta olhar para o canto inferior esquerdo da tela.

Onde o jogador pode enviar seus resultados para um site que mostra o Ranking dos melhores jogadores.



Cada vez que o jogador mata todas as naves inimigas, novos quatro inimigos aparecem.

Todos os inimigos perseguem a nave do jogador, eles se aproximam toda vez que o jogador movimenta sua nave pela tela de jogo.

Para sair do Jogo, basta clicar na opção “Sair” no menu.

b) Estrutura da Lista

As estruturas TAD utilizadas para a implementação do jogo são a Pilha e a Lista.

A classe da Fila pode ser representada pela imagem a seguir:

```
#ifndef LISTA_H
#define LISTA_H
#include <SFML/Graphics.hpp>
#include <iostream>
#include "Tiro.hpp"
#include "NaveInimigo.hpp"
// Estrutura da Lista
template <class Gen> struct Node{
    Gen info;
    struct Node<Gen> *dir;
    struct Node<Gen> *esq;
};

template<class Gen>
class Lista{
    private:
        int quant;
        struct Node<Gen> *Primeiro;
        struct Node<Gen> *Atual;
        struct Node<Gen> *header;
    public:
        Lista();
        ~Lista();
        void PegaOProximo(Gen&,bool&);
        void PegaOPrimeiro(Gen&,bool&);
        void cria();
        int getQuant();
        void insere(Gen&,bool&);
        bool estaNaLista(Gen& x);
        void insereAEsquerdaDeP(Gen&,bool&);
        void removeP(Gen&,bool&);
        void remove(Gen& x,bool& deuCerto);
        void atualizaP(Gen& x,bool& deuCerto);
};
```

A implementação dos métodos da Pilha pode ser vista na imagem a seguir:

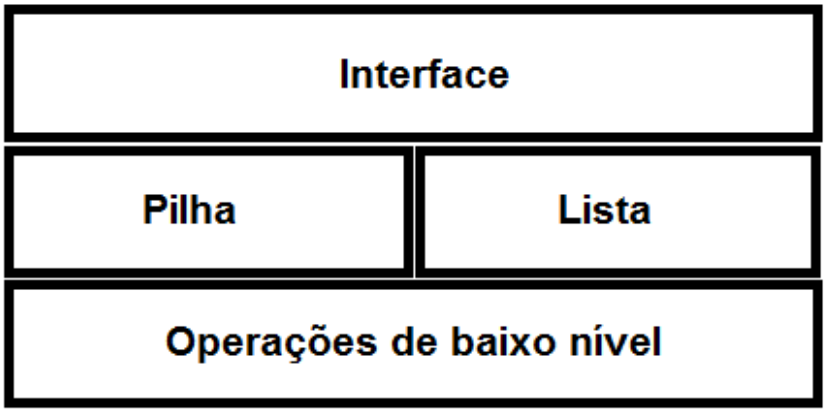
```
1  #include <iostream>
2
3  template<class Gen> struct NodePilha{
4    Gen Info;
5    struct NodePilha<Gen> *Next;
6  };
7
8  template<class Gen>
9  class Pilha
10 {
11     public:
12         Pilha();
13         virtual ~Pilha();
14         void Empilha(const Gen & X, bool & DeuCerto);
15         void Desempilha(Gen & X, bool & DeuCerto);
16         bool Vazia();
17         bool Cheia();
18         Gen getTopo();
19     private:
20         struct NodePilha<Gen> *  P_Topo;
21 };
```

O código do jogo responsável por controlar as teclas que irão movimentar a nave pode ser vista no arquivo Jogo.h. Esse trecho do código é fundamental, pois controla toda situação do jogo enquanto o jogador está tentando sobreviver. O trecho do código responsável por isso pode ser visto a seguir:

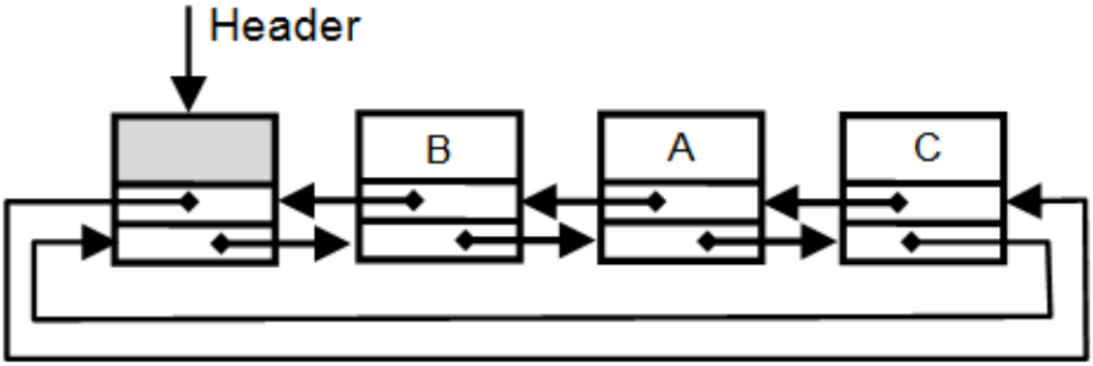
```
165
166     // comecando no meio
167     nave->setPosition(sf::Vector2f(largura/2,altura/2));
168     deuCerto = true;
169     //  Aqui q vai tudo do jogo.
170     sf::Event evento; // eventos de jogo
171     bool executando = true;
172     while (executando){ // loop da tela
173         // Verificação de eventos
174         while (App.pollEvent(evento)){ // loop de eventos
175             if (evento.type == sf::Event::Closed){
176                 return (-1);
177             }
178             if(sf::Keyboard::isKeyPressed(sf::Keyboard::Up)){
179                 nave->andaFrente(2);
180                 if(!atirou){
181                     tiroAuxSpace.setPosition(nave->getFrente());
182                     tiroAuxSpace.setDirecao(nave->getDirecao());
183                 }
184                         // estrutura do toroide para a nave heroi
185                 obedecerToroide(nave, largura, altura);
186             }else{
187                 if(sf::Keyboard::isKeyPressed(sf::Keyboard::Left)){
188
189                         nave->rodaAntiHorario();
190                         if(!atirou){
191                             tiroAuxSpace.setPosition(nave->getFrente());
192                             tiroAuxSpace.setDirecao(nave->getDirecao());
193                         }
194                 }else{
195                     if(sf::Keyboard::isKeyPressed(sf::Keyboard::Right)){
196                         nave->rodaHorario();
197                         if(!atirou){
198                             tiroAuxSpace.setPosition(nave->getFrente());
199                             tiroAuxSpace.setDirecao(nave->getDirecao());
200                         }
201                     }else{
202                         if(sf::Keyboard::isKeyPressed(sf::Keyboard::Space)){
203                             adicionarTiro = true;
204                             tiroAuxSpace.setPosition(nave->getFrente());
205                             tiroAuxSpace.setDirecao(nave->getDirecao());
206                         }
207                     }
208                 }
209             }
```

c) Diagrama e arquitetura do software

Para a implementação do T3, foi utilizada a seguinte arquitetura de software. A interface é toda a imagem gráfica do jogo, que foi criada utilizando a biblioteca SFML. A TAD Pilha e Fila são as estruturas que dão vida ao jogo e as operações de baixo nível são todas as outras classes utilizadas, que somadas, representam toda a usabilidade do jogo, como por exemplo, as classes Jogo, Tiro, Nave, etc.



d) Implementação da Lista



Trata-se de uma lista generalizada, duplamente encadeada, com nó header, com quantidade de elementos, *template* e operações de baixo nível. No jogo, instanciam-se algumas listas, como a lista de tiros do jogador, a lista de tiros dos inimigos e a lista de inimigos que o jogador precisa enfrentar.

e) Implementação

A linguagem de programação utilizada foi C++ e a biblioteca de interface utilizada foi SFML. Todo o código foi dividido em arquivos menores para visar organização.

Os arquivos do projeto são:

- Fila.hpp
- Jogo.hpp
- main.cpp
- Lista.hpp
- Menu.hpp
- Nave.hpp
- NaveInimigo.hpp
- Pilha.hpp
- Ranking.hpp
- Regras.hpp
- Request.hpp
- Tela.hpp
- Tiro.hpp
- TodasTelas.hpp

Além disso, alguns outros arquivos estão na mesma pasta, que são os arquivos de imagem, músicas e sons. Todos os métodos estão disponíveis no https://github.com/gabrielissantos/JvTron-3.0.

3-Conclusão

Uma das maiores dificuldades encontradas durante o desenvolvimento do projeto foi a simultaneidade do desenvolvimento de dois jogos. Somado a isso, houve a complexidade da lógica dos jogos escolhidos pelo grupo, que atrapalhou o andamento do projeto.

No início do desenvolvimento do jogo, a equipe dividiu as tarefas do seguinte modo: O João ficaria responsável pelo desenvolvimento do T2 e a Gabrieli ficaria responsável pelo desenvolvimento do T3, enquanto que o Leonardo acompanharia o desenvolvimento dos dois. Logo em seguida, todos se uniriam e desenvolveriam os dois jogos. Contudo, essa subdivisão não prevaleceu durante todo o processo e cada membro da equipe fez um pouco de cada parte.

Essa foi a segunda oportunidade que o grupo teve de realizar um projeto de grande escala, implementando conceitos de linguagem de programação orientada a objetos, trabalhar com controle de versão e usar a criatividade. Em relação ao T1, foi possível notar uma grande diferença em muitos aspectos. Contudo, é importante dizer que ainda há pontos a serem melhorados.

Outro problema importante a ser ressaltado é a dificuldade em gerar um executável que rode em todos os computadores, nos diversos sistemas operacionais.

Tivemos também, a oportunidade de trabalhar na prática com conceitos teóricos apresentados em sala de aula, como implementação das TADs vistas em classe, zelando sempre pela portabilidade e usabilidade do código. Além disso, tivemos oportunidade de aprender conceitos muito importantes com essa disciplina, que fará com que usemos diversos conteúdos em disciplinas futuras.