Universidade Federal de São Carlos – UFSCar

Bacharelado em Ciência da Computação

Estrutura de Dados

Professor: Roberto Ferrari

Jogo: JvTRON

Nomes: Gabrieli Santos, **RA:** 726523, **e-mail:** gabrielisantos17.gs@gmail.com

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

Leonardo de Oliveira Peralta, **RA:** 726556, **e-mail:** 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 início da disciplina, foram apresentados aos alunos, os conceitos de fila e pilha, e como implementá-los em situações reais. Através disso, o primeiro desafio foi escolher, em grupo, uma dessas duas 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".

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 filme "Tron: O Legado" de Joseph Kosinski. O jogo é constituído por dois jogadores adversários que disputam no mesmo campo pela vitória. Cada jogador possui sua respectiva moto, que deixa um rastro por onde ela passa. Esse rastro aumenta conforme o jogo se desenrola, uma vez que ele "guarda" todos os locais que o velocista passou. Quando um jogador bate no rastro deixado pelo seu oponente, ou seja, quando ele cruza um caminho já percorrido pelo outro, ou bate no próprio rastro, ele perde o jogo. O objetivo é, então, correr pelo campo no máximo de espaços vazios possíveis e encurralar o adversário.

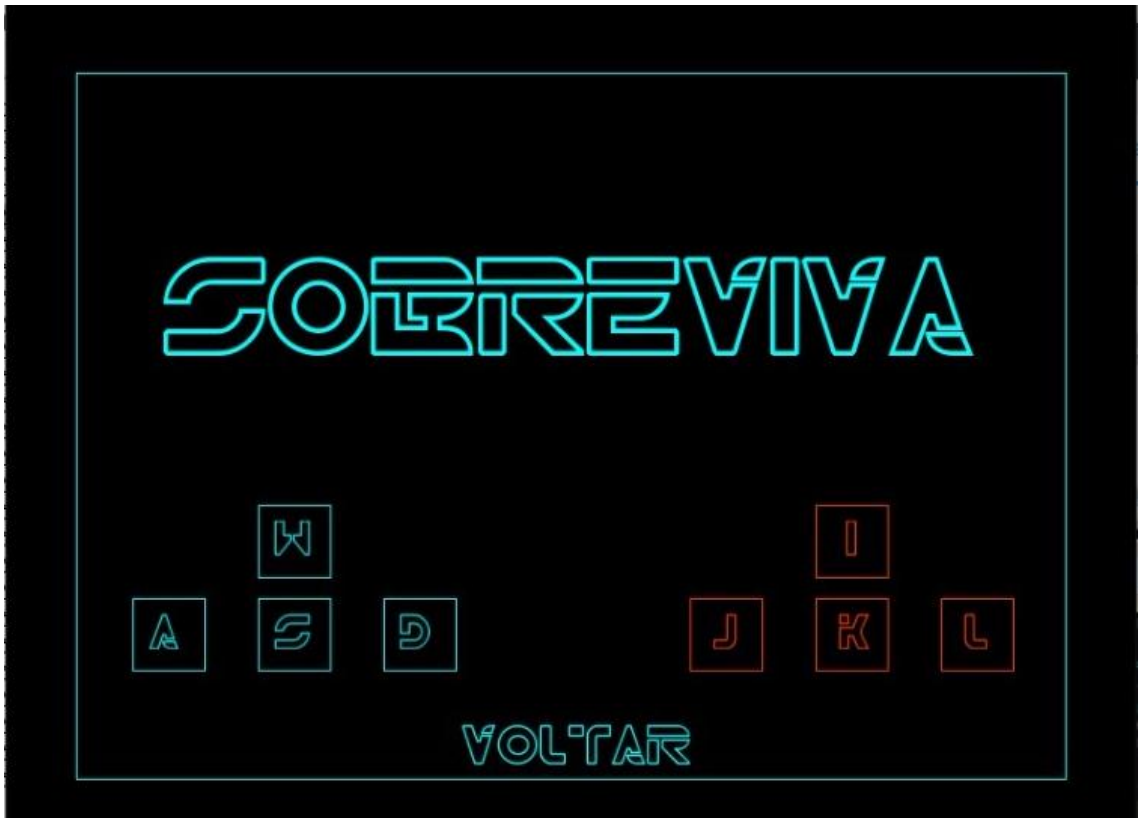
A estrutura escolhida para o jogo foi a Fila, cujo conceito é "O primeiro elemento que entra é o primeiro que sai", pois a cada novo espaço que a moto percorre, um novo elemento é inserido na fila..

2-Desenvolvimento

a- Print-screens da execução

A tela inicial do jogo tem as seguintes opções para o usuário: Jogar, Regras e Sair.

Ao clicar em Regras, a tela que aparece para o usuário é a seguinte:



Cuja única regra é sobreviver!

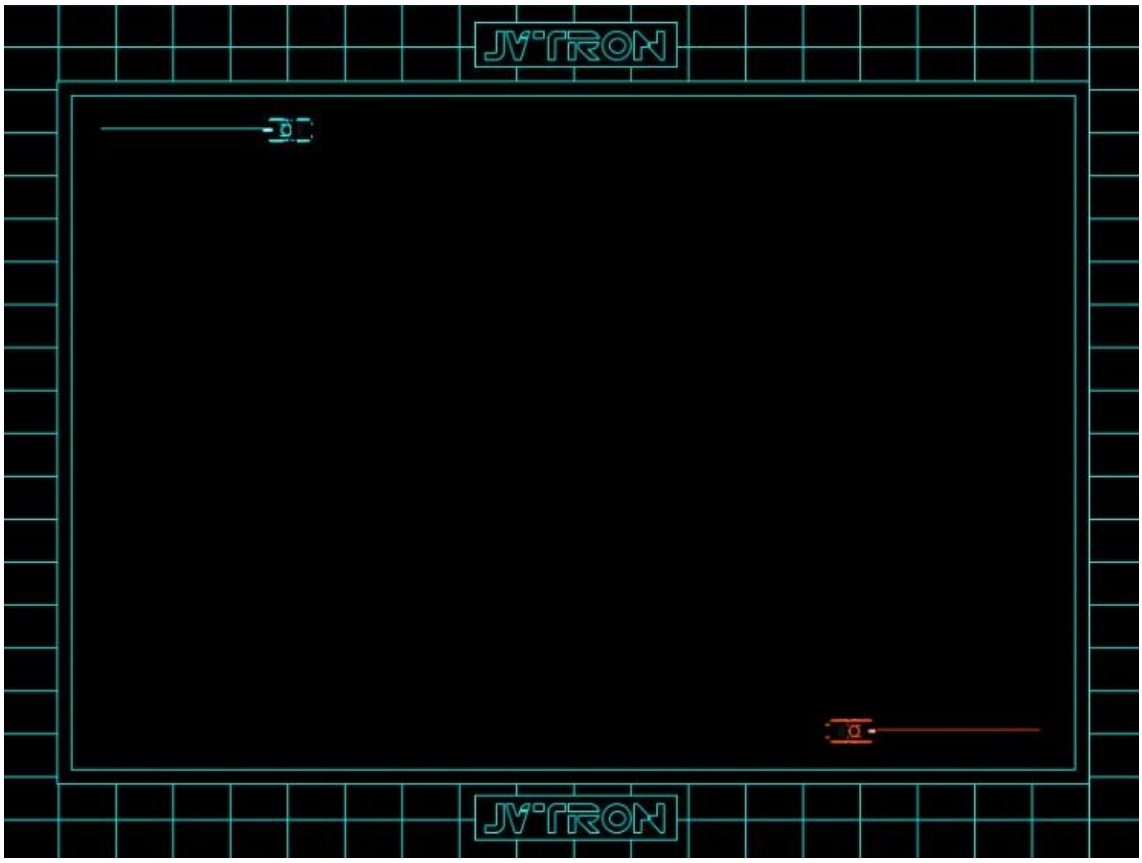
O jogo precisa de dois usuários. Assim, o jogador de cor azul usará as teclas:

- A – virar para direita
- S – ir para baixo
- D – virar para a esquerda
- W – ir para cima

  E o jogador da cor laranja usará as teclas:

- J – virar para a direita
- K – ir para baixo
- L – virar para a esquerda
- I – ir para cima

Ao voltar para o menu principal e selecionar a opção “Jogar”, o jogo se inicia e os jogadores podem começar a jogar.



Quando o jogador laranja ganha, a tela que aparece para ele é:

Quando o jogador azul ganha, a tela que aparece para ele é:

b- Estrutura da Fila

A Fila utilizada nesse jogo foi implementada num arquivo chamado Fila.hpp, e pode ser vista na figura a seguir. Os métodos utilizados foram Vazia, Cheia, Insere, Retira, Cria e Destroi.

```
8   // Estrutura da fila
9   template<class Gen>
10  class Fila{
11  private: //atributos privados
12          int primeiro;
13          int ultimo;
14          int nElementos;
15      Gen elementos[20000];
16  public: //atributos publicos
17          Fila();
18          ~Fila();
19          void Insere( Gen &, bool &);
20          void Retira(Gen  &, bool &);
21          bool Vazia() const;
22      bool Cheia() const;
23      int getNElementos();
24      Gen* getDesenhoRastro();
25  };
```

A cada espaço que o velocista percorre, é inserido (Insere(Gen &, bool &)) um elemento ao final da fila. Dessa forma, o ultimo elemento está mais próximo da moto. Quando o adversário, ou o próprio jogador, bate em seu rastro, todos os elementos da fila são retirados até o ponto da colisão. Quando o elemento da colisão se torna o primeiro da fila, o jogo termina e o jogador que promoveu a colisão perde.

Além disso, já que implementamos as caudas das motos e as paredes que elas formam como dois entes diferentes, há uma fila para paredes. Assim, a cada determinada quantidade de unidades que a moto se move, insere-se, em uma outra fila, uma unidade de parede contendo sua localização.

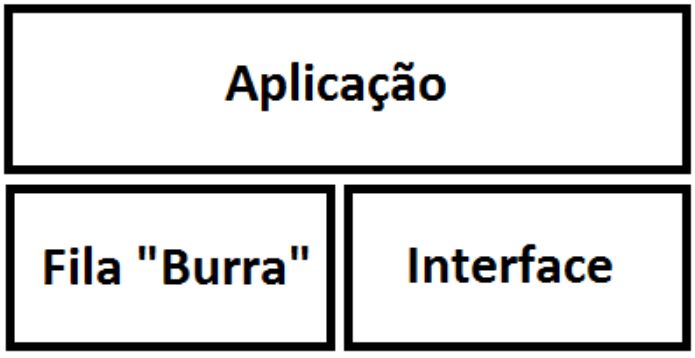
A seguir, algumas aplicações das filas em código:

```
273
274 // Verificações de choques nas filas de paredes (onde e por qual moto)
275         for(j = 0; j < paredesTron.getNElementos() - 2 ; j++){ // para paredes na moto azul
276                 paredesTron.Retira(tempRetangulo, ok);
277                 paredesTron.Insere(tempRetangulo, ok);
278                 if(tron.getForma().getGlobalBounds().intersects(tempRetangulo.getGlobalBounds())){
279                     return (3); // a laranja ganha (TELA 3: Ganhou em laranja)
280                     break;
281                 }
282                 if(rinz.getForma().getGlobalBounds().intersects(tempRetangulo.getGlobalBounds())){
283                     return (2); // a azul ganha (TELA 2: Ganhou em azul)
284                     break;
285                 }
286                 semChoqueT = true;
287         }
```

```
244         // Mudou de orientação ou não, insere-se na fila de exibição
245         //  Inserir na fila de exibição da moto azul
246         tempCauda.position = tron.getForma().getPosition() + tron.getFimCauda();
247         tempCauda.color = sf::Color(0,255,255);
248         FETron.Insere(tempCauda,ok);
249
250         //  Inserir na fila de exibição da moto laranja
251         tempCauda.position = rinz.getForma().getPosition() + rinz.getFimCauda();
252         tempCauda.color = sf::Color(255,60,0);
253         FERinz.Insere(tempCauda,ok);
```

c- Diagrama e arquitetura do software

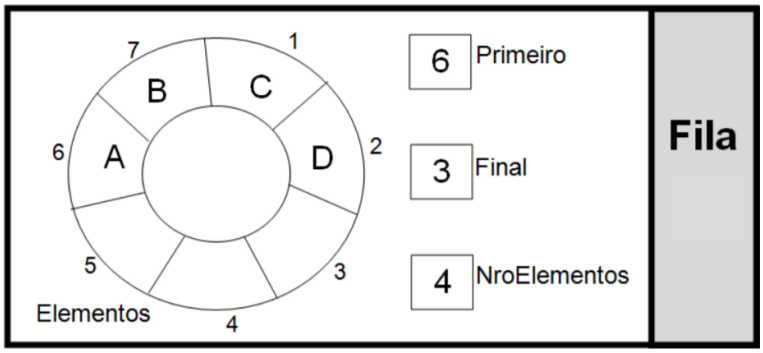
A fila utilizada nesse projeto não faz qualquer verificação sobre os elementos que entram ou saem da fila, por isso, ela é chamada de fila “burra”. Além da aplicação do jogo, e da fila “burra”, temos também a interface. Esses três pilares compõe a arquitetura de software do jogo. O diagrama pode ser visto a seguir:



d- Implementação da fila

A fila utilizada pode ser representada pela figura abaixo. Trata-se de uma fila circular com número de elementos. O primeiro elemento da fila corresponde à posição da moto, ou seja, o primeiro local que ela passou. Os elementos seguintes correspondem à sequência de movimento do jogador, e por fim, o último elemento corresponde ao último local que a moto passou. Ou ainda, os elementos seguintes correspondem às paredes da cauda.

Se um dos dois jogadores trombarem em um dos rastros, na posição C, por exemplo, a operação Remove() remove todos os elementos da fila até chegar na posição F. Desse modo, o elemento A é o primeiro a sair, logo em seguida é o B e por fim, C. Após isso, o jogo termina.



e- Implementação

A linguagem de programação utilizada foi C++ e a biblioteca para interface utilizada foi o SFML. Todo o código foi dividido em aquivos.hpp para torna-lo mais organizado e legível. O arquivo principal do código, ou seja, o main é o que define as propriedades da tela e a música.

Os arquivos do projeto são:

- Campo.hpp
- Fila.hpp
- Ganhou.hpp
- Menu.hpp
- Moto.hpp
- Regras.hpp
- Tela.hpp
- main.cpp
- Screens.hpp

Estão documentados os métodos específicos de cada função em cada código.

Todos os códigos estão disponíveis abertamente num arquivo com extensão rar junto com esse documento e no git https://github.com/joaobarbirato/JvTron

3-Conclusão

Uma das maiores dificuldades encontradas durante o desenvolvimento do projeto foi a incompatibilidade da biblioteca gráfica SFML com sistemas operacionais, como o Windows. Além disso, a falta de experiência com a biblioteca atrasou um pouco o cronograma. A dificuldade com controle de versão também esteve presente, uma vez que os integrantes do grupo ainda não tinham experiência em programar em equipe.

Somado a isso, houve a complexidade da lógica do jogo escolhido pelo grupo, que atrapalhou o andamento do projeto, em detrimento da simplicidade da proposta de se utilizar um TAD fila.

No início do desenvolvimento do jogo, a equipe dividiu as tarefas do seguinte modo: O João e a Gabrieli ficariam responsáveis pela *backend* (funcionamento) (funcionamento), e o Leonardo pelo *frontend* (exibição). Contudo, com as dificuldades encontradas durante as fases do projeto, essa subdivisão não prevaleceu, uma vez que a biblioteca utilizada para a interface gráfica depende fortemente da implementação. Assim, cada membro da equipe fez um pouco de cada parte.

O grupo todo percebeu que essa atividade proporcionou um desafio inédito para os alunos, uma vez que essa foi a primeira situação em que tivemos a oportunidade 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.

Tivemos também, a oportunidade de trabalhar na prática com conceitos teóricos apresentados em sala de aula. Por exemplo, um dos desafios notados durante o desenvolvimento foi a necessidade de alterar as propriedades da fila, de sequencial para estática, sem alterar suas chamadas de métodos, garantindo sua portabilidade.