UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E COMPUTAÇÃO
DEPARTAMENTO DE SISTEMAS DE COMPUTAÇÃO

SSC0140 - Sistemas Operacionais I

Projeto Final - Confecção de um Jogo

Guilherme Jun Yazaki Grillo - 11208350
Gustavo Lelli Guirao - 11918182
João Francisco Caprioli Barbosa Camargo de Pinho - 10748500
Leonardo Chagas Pizzo - 10288511

Profª. Drª. Kalinka Regina Lucas Jaquie Castelo Branco
2º Semestre 2021

SÃO CARLOS
2021

SUMÁRIO

1. INTRODUÇÃO

1.1. Propósito

Neste projeto serão mostrados o desenvolvimento da ideia e da implementação do jogo proposto na disciplina de Sistemas Operacionais I. A proposta do projeto é a confecção de um jogo feito em C, C++ ou C# que implemente os conceitos de semáforos e *threads* vistos em aula.

1.2. Organização

Na primeira parte do documento está a explicação sobre a biblioteca *Raylib*, que foi a biblioteca chave para a confecção do jogo, já que ela foi feita para programar jogos em mais de 50 linguagens diferentes, e possui diversas funções de processamento gráfico úteis.

Na segunda e terceira partes são explicadas as implementações propostas pelo projeto, a do semáforo e das *threads*. Esses conceitos foram implementados para realizar a movimentação e detectar a colisão entre os objetos do jogo.

2. RAYLIB

Por ser uma ferramenta muito versátil, a biblioteca *Raylib* foi utilizada em diversas áreas do jogo. Toda a parte gráfica, desde a página de início e a tela de jogo, até o controle do personagem, foram feitos usando funções e classes disponibilizadas pela biblioteca. Além dos exemplos de uso disponibilizados no site da biblioteca, que mostram diversas possibilidades que ela oferece e como implementá-las.

2.1. Como Jogar

Figura 1: Página inicial do jogo *SOgger*

Como a página de início mostra, o personagem é controlado usando as setas, e o objetivo do jogo é desviar dos carros que ficam cada vez mais rápidos, atravessar a avenida para cima e para baixo, a fim de obter a maior quantidade de pontos.

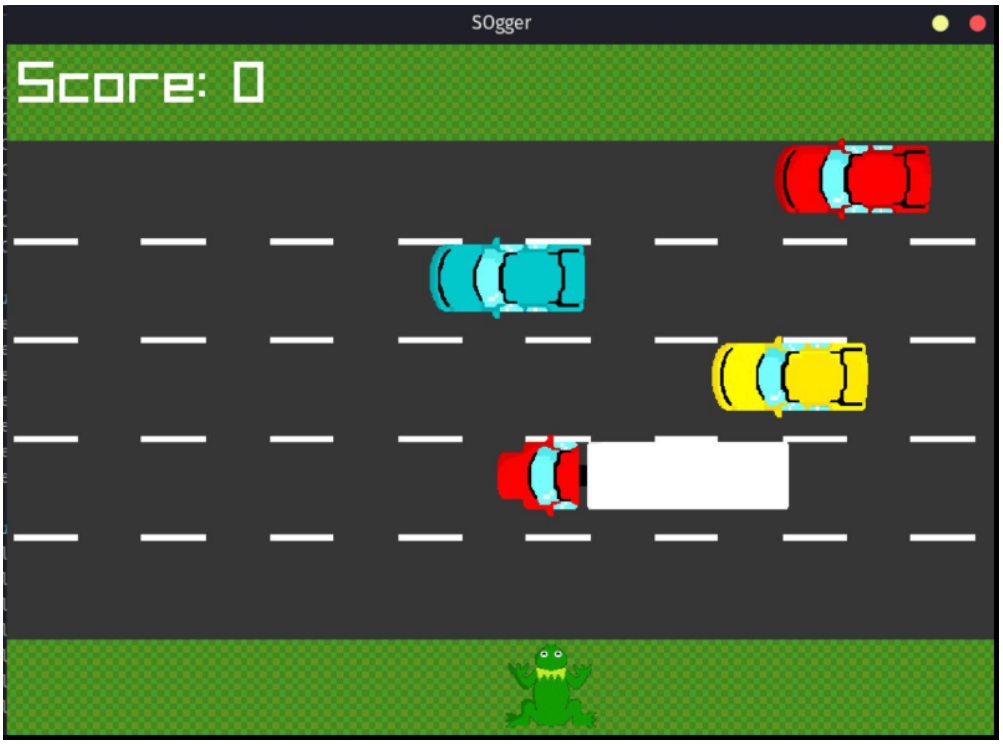

Figura 2: Início do *SOgger*

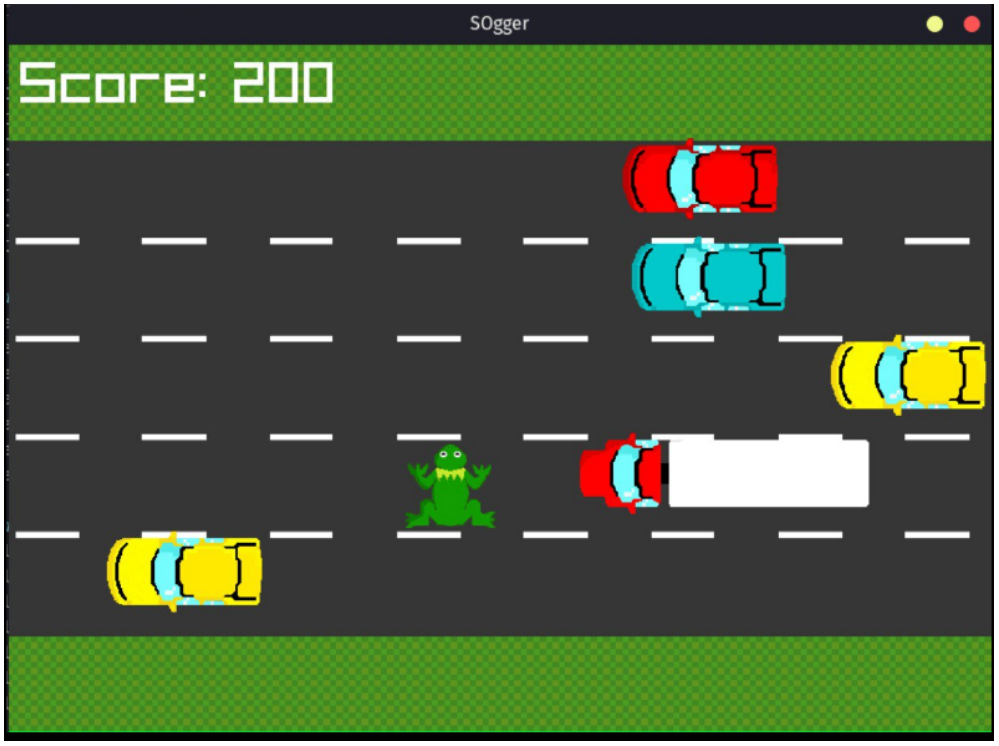

Figura 3: *SOgger* funcionando

Caso o jogador falhe em desviar dos carros, o personagem é atropelado e o jogo acaba.

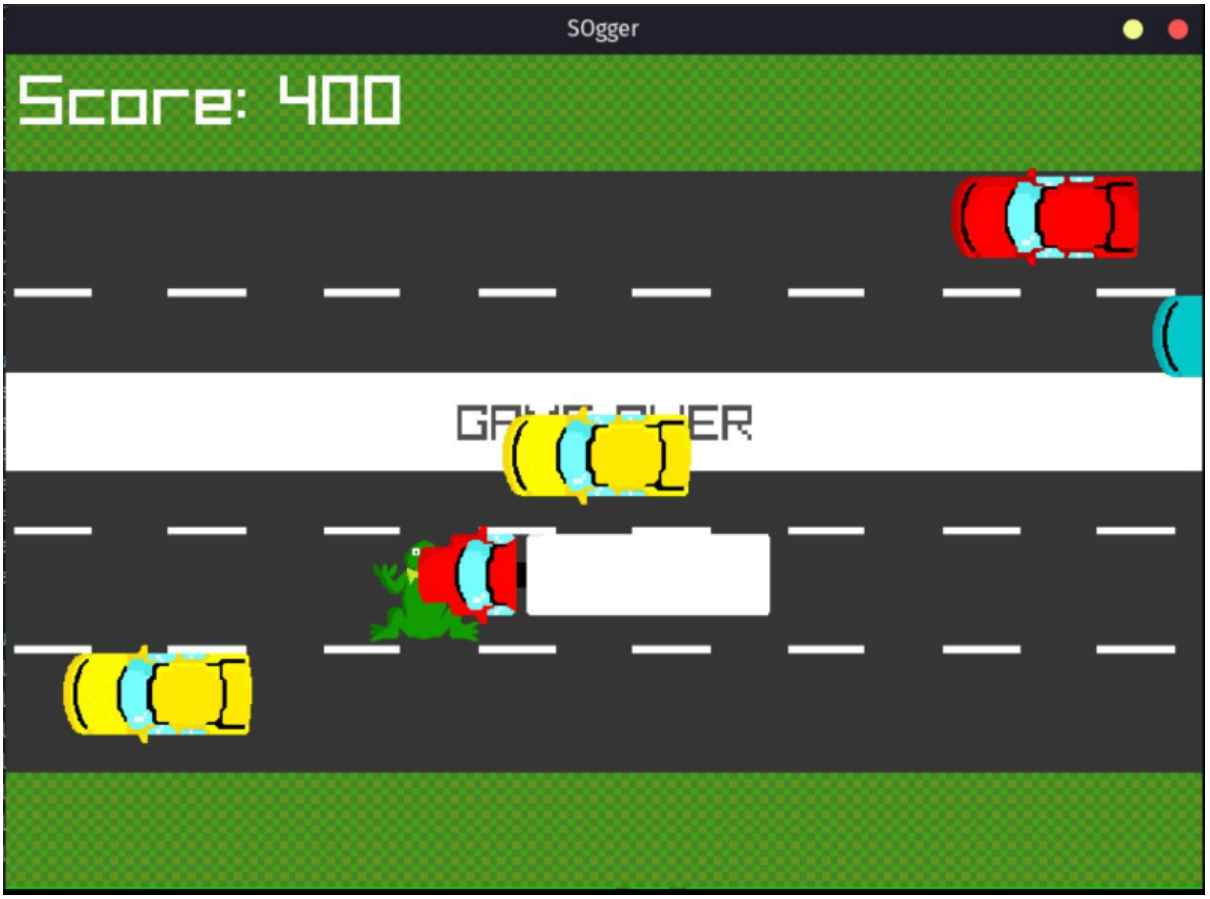

Figura 4: Exemplo de *Game Over*

2.2. Instalação do Jogo

Não há necessidade de instalar nada para rodar o jogo. Apenas seguir as instruções da seção seguinte.

2.3. Execução do Jogo

Em um terminal do *Linux* usar o comando:

```
$ make
```

para gerar o arquivo executável.

E em seguida:

```
$ ./main
```

para executar o jogo.

3. IMPLEMENTAÇÃO DE SEMÁFOROS

3.1. O Que São Semáforos

Semáforos são mecanismos que resolvem o problema de exclusão mútua. Um semáforo pode ser visto como um objeto que pode sofrer dois tipos de operação sobre ele: trancando e destrancando a execução de instruções. Semáforos são implementados no sistema operacional e são considerados uma forma de IPC.

3.2. Implementado no Jogo

Inicialmente, a ideia era que o semáforo cuidaria da movimentação dos inimigos, já que todos são *threads*, o semáforo se fecharia quando houvesse uma colisão, o que não funcionou.

A ideia final então seria o semáforo controlar duas *threads* diferentes: uma que estaria executando durante o jogo e outra que estaria executando no fim de jogo. A proposta é que o jogo estaria executando com a primeira *thread*, e quando ocorresse uma colisão, o semáforo se fecharia para a primeira *thread* e se abriria para a segunda, iniciando o processo de fim de jogo.

Na imagem a seguir temos um exemplo de onde o semáforo foi usado:

```
if (!*end) {
  // Realiza o movimento do jogador
  updatePlayer(frogg);

  // Realiza o movimento dos 'inimigos'
  updateEnemies(enemies);

  // Verifica e realiza a colisao do jogador com os 'inimigos'
  checkCollision(*end, frogg->position.x, frogg->position.y, enemies);

  // Quando ha uma colisao do jogador com um 'inimigo', o jogo finaliza pelo semaforo
  //  o semaforo a principicio e desbloqueado para liberar a thread de fim de jogo que
  //  foi bloqueada, e na linha abaixo o semaforo e bloqueado nessa thread, impedindo que o jogo continue
  //  ate que a thread de fim de jogo reinicie todos os objetos para a posicao inicial
  if (*end) {
    sem_post(sem);  // Realiza o desbloqueio do semaforo
    sem_wait(sem);  // Realiza o bloqueio do semaforo
  }
```

Figura 5: Semáforo checando a colisão do sapo

4. IMPLEMENTAÇÃO DE *THREADS*

4.1. O Que São *Threads*

As *threads* são uma forma de um processo dividir a si mesmo em 2 ou mais tarefas que podem ser executadas concorrencialmente, compartilhando o mesmo endereço de memória.

4.2. Implementado no Jogo

No nosso jogo, as threads foram implementadas em nível de usuário, através da biblioteca *thread.h*. Uma funcionalidade foi de simular o comportamento de atualização e movimentação dos veículos (inimigos) que passam horizontalmente na tela. Outra funcionalidade do jogo que utiliza threads foi no paralelismo que aplicamos na tarefa de executar o jogo, e uma tarefa para o finalizar quando o personagem morre.

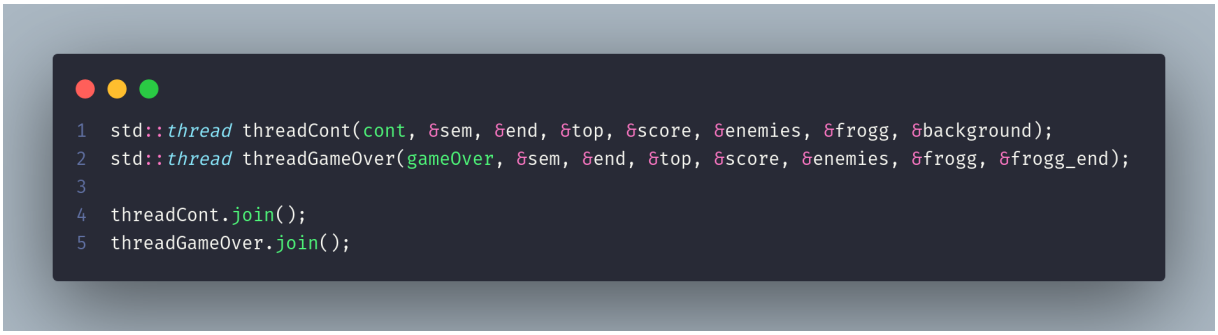

Figura 6: Página inicial do jogo *SOgger*

Na Figura 6, pode se observar a implementação de duas *threads*: de continuação do jogo (linha 1) e uma de fim de jogo (linha 2), os dois utilizando um ponteiro de função para as funções '*cont()*' e '*gameover()*', além de passar todos os parâmetros necessários em seguida.

Uma vez que uma *thread* é iniciada, precisamos esperar para que ela termine antes de executar outra ação. Para isso, utilizamos a função *.join()* da biblioteca *thread.h*, que espera a *thread* finalizar.

5. CONCLUSÃO

Com o fim da idealização e implementação do jogo *SOgger*, conseguimos aplicar os conceitos da disciplina de Sistemas Operacionais I de forma prática, que, com exceção da biblioteca utilizada, foi toda realizada pelos integrantes do grupo.

O jogo, implementado em C++ com a biblioteca *Raylib,* é apenas um dos inúmeros exemplos de programas que utilizam *threads* e semáforos. No processo de criação e implementação do jogo, pudemos ver como esses conceitos são importantes e vantajosos para programas complexos.