MANUAL DO JOGO JOSIEL ADVENTURES - SISTEMAS OPERACIONAIS

Eduardo Rodrigues Amaral	NUSP 11735021
Fernando Henrique Paes Generich	NUSP 11795342
Gabriela Rodrigues do Prado	NUSP 11892917

Instalação

Inicialmente, para poder jogar o jogo Josiel Adventures, criado para a disciplina de Sistemas Operacionais, é preciso realizar o download do jogo. Após o download completo, basta clicar duas vezes sobre o executável que está dentro da pasta 'bin'. Após alguns segundos, o jogo já estará disponível para ser jogado.

Para abrir o código-fonte da aplicação, basta realizar o download da versão Mono do Godot Engine e abrir o projeto "project.godot" presente na pasta raiz do projeto.

Como Jogar

Nesse jogo, o jogador controla o personagem Josiel, que comanda um cruzamento. Neste cruzamento, os carros passam em dois sentidos: de sul ao norte, e de leste para oeste. O objetivo do jogo é conseguir a maior pontuação possível mantendo corretamente o cruzamento sem causar nenhum acidente com a colisão dos carros. Para isso, Josiel possui a capacidade de ordenar a parada de qualquer uma das duas direções para que o(s) veículo(s) na outra direção possa passar e continuar seu trajeto. Vale lembrar que para que uma via tenha sua movimentação interrompida, Josiel precisa estar olhando para essa via.

Entretanto, ao solicitar a parada de uma das direções do cruzamento, o jogador possui um tempo de apenas 7 segundos para manter a via congestionada. Caso esse tempo seja excedido, o jogo é finalizado por conta de inanição (aqui utilizamos o conceito de Starvation, ou Inanição).

Para iniciar o jogo, basta apertar a tecla Enter. Assim que o jogo se inicia, vão surgindo os carros aleatoriamente e em ambas as direções. As setas direcionais para baixo e para a direita do teclado controlam a direção da visão de Josiel. Para parar a movimentação da via para a qual Josiel aponta, basta apertar a tecla Espaço. Para voltar a permitir a movimentação da via interrompida, basta apertar a tecla Espaço novamente.
Caso ocorra o fim do jogo e o jogador queira jogar novamente, basta apertar a tecla Enter.

Descrição de partes estudadas no Jogo

Nesse jogo, foram abordados os conceitos de Semáforos e Threads aprendidos na disciplina de Sistemas Operacionais realizadas no segundo semestre de 2021. Foi abordado também o conceito de Mutex. A implementação da engine Godot de tais conceitos foi utilizada, ao invés da implementação nativa de C#, devido à maior integração com as API's da engine.

As threads foram utilizadas na liberação das instâncias (*free*) dos veículos que estavam fora da tela. Dessa forma, foram utilizadas duas threads: uma no sentido sul e uma à direita da tela do jogo. Cada thread é responsável por realizar a limpeza dos veículos que já haviam passado pela tela. Assim, foram iniciadas duas threads:

Figura 1: Início das Threads

```
static private void ClearRightCars()
{
  while (GameRunning)
  {

    if (RCars.Count == 0)
    {
      GD.Print("Waiting R");
      CarCleanerSemaphoreR.Wait();
      if (!GameRunning)
      {
        GD.Print("GAME WAS NOT RUNNING");
        return;
      }
      GD.Print("Woken up R");
    }

    Car car = RCars[0];

    if (car != null)
    {

      if (car.Position.x < -100)
      {

        car.QueueFree();
        RCars.Remove(car);
        CarsSafelyCrossedMutex.Lock();
        CarsSafelyCrossed++;
        CarsSafelyCrossedMutex.Unlock();
      }
    }
  }
}
```

Figura 2: Rotina de limpeza de carros executada nas Threads

A utilização dessas threads que verificam se um carro está ou não na tela ativa de jogo e realiza a liberação dos que não estão ajuda a otimizar a performance do jogo.

O conceito de semáforo foi utilizado para realizar o controle das threads de limpeza, ou seja, quando não existe nenhum veículo para ser liberado, o semáforo entra em espera e só é reativado quando adiciona-se um novo veículo na thread principal. Vale lembrar que o semáforo utilizado foi um do tipo binário.

```
if (RCars.Count == 0)
{
  GD.Print("Waiting R");
  CarCleanerSemaphoreR.Wait();
  if (!GameRunning)
  {
    GD.Print("GAME WAS NOT RUNNING");
    return;
  }
  GD.Print("Woken up R");
}
```

Figura 3: Trecho de espera do semáforo por mais carros para limpar

```
if (car.GetSpawnSelect() == "right")
{
  RCars.Add(car);
  if (RCars.Count == 1)
  {
    CarCleanerSemaphoreR.Post();
  }
}
```

Figura 4: Trecho de retomada do semáforo

Por fim, o Mutex foi abordado através da variável *CarsSafelyCrossed*. Essa variável é incrementada toda vez que um veículo realiza sua passagem através da tela do jogo e é liberado pelo sistema de threads. Como existem duas threads funcionando, a variável Mutex é utilizada para que não haja nenhuma inconsistência ao ser alterada. Vale ressaltar que o Mutex é responsável pelo placar do jogo.

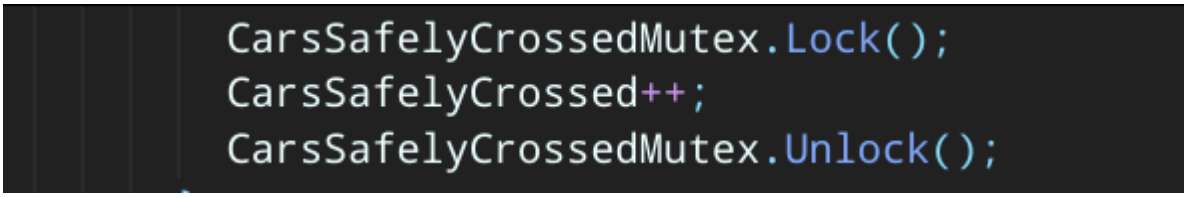

Figura 5: Trecho de Utilização do Mutex para incrementação segura da variável.

Estes foram os conceitos abordados na disciplina de Sistemas Operacionais que embasaram a criação do jogo **JosielAdventures**.

Links Úteis

Link do repositório Github do projeto:
https://github.com/FerHPGene/josielAdventures

Assets do Josiel:
https://deadmadman.itch.io/the-quaken-assets

Documentação do Godot para multi-threading:
https://docs.godotengine.org/en/stable/tutorials/threads/using_multiple_threads.html