**Instituto Superior de Engenharia**

Politécnico de Coimbra

Departamento de Engenharia Informática e de Sistemas

**Classic Frogger Game**

Joaquim Rodrigo Rodrigues Milheiro – 2020131794

Lucas Ribeiro Caetano – 2020132564

**Ano Letivo**

**2022/2023**

Uma imagem contendo Texto

Descrição gerada automaticamente

**Licenciatura em Engenharia Informática**

**Sistemas Operativos 2**

**RELATÓRIO DO TRABALHO PRÁTICO DO**

**JOGO FROGGER**



Joaquim Rodrigo Rodrigues Milheiro

Lucas Ribeiro Caetano

**Índice**

[1. INTRODUÇÃO 4](#_Toc105531665)

[2. FUNCIONAMENTO DO JOGO 5](#_Toc105531665)

[3. ESTRUTURAS DE DADOS 7](#_Toc105531665)

[3.1. Servidor 10](#_Toc105531667)

[3.2. Operador 11](#_Toc105531670)

[3.3. DLL 12](#_Toc105531670)

[3.4 Registry 13](#_Toc105531670)

[4.1. MECANISMOS DO JOGO 9](#_Toc105531680)

[4.1. Mecanismos de Comunicação 10](#_Toc105531667)

[4.2. Mecanismos de Sincronização 11](#_Toc105531670)

[5. CONCLUSÃO 14](#_Toc105531692)

# 1 INTRODUÇÃO

O jogo Frogger é um clássico que desafia os jogadores a guiarem uma pequena rã através de um perigoso ambiente repleto de obstáculos, com o objetivo de alcançar a segurança do outro lado da tela. O jogo, lançado originalmente em 1981, tornou-se imensamente popular devido à sua jogabilidade desafiadora e viciante.

Neste trabalho da disciplina de Sistemas Operativos 2, iremos explorar o desenvolvimento de uma versão do jogo Frogger, aplicando conceitos e técnicas relacionadas a sistemas operativos, com ênfase nos mecanismos de comunicação e sincronização entre os elementos do jogo.

**CAPÍTULO 2**

**FUNCIONAMENTO DO JOGO**

## 

Na nossa implementação do jogo *Frogger* existem 3 programas principais: O servidor, o operador e o sapo.

O servidor é o programa que recebe os comandos e os executa, sendo também nele realizada toda a validação de input.

O operador trata-se de um programa para configuração/alteração do jogo principal que corre no servidor por via de comandos.

O programa sapo são os jogadores, sendo cada jogador uma instância deste programa.

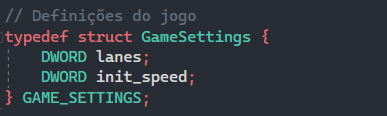
**CAPÍTULO 3**

**ESTRUTURAS DE DADOS**

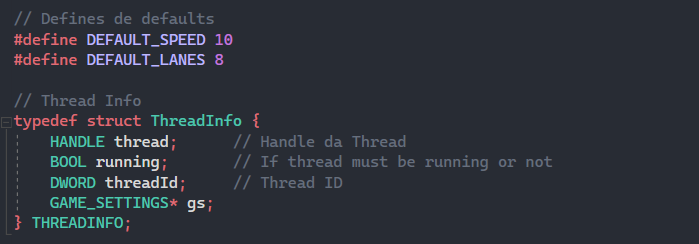
# 3.1. SERVIDOR



No servidor temos uma estrutura de dados que armazena as definições do jogo:



Definimos também uma estrutura para armazenar a informação da thread como o seu handle, se está a correr ou não, o seu ID e um ponteiro para uma estrutura onde estão guardadas as configurações.

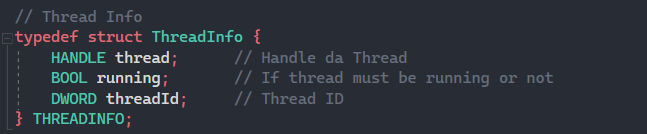


# 3.2. OPERADOR

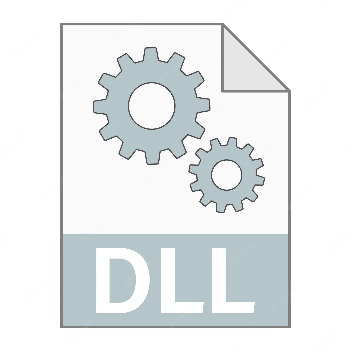


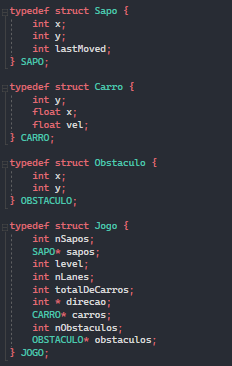
No Operador, temos apenas uma estrutura de dados relevante que é a estrutura *Thread\_Info* já referida no servidor.

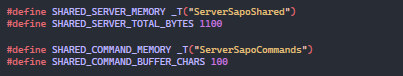
Esta estrutura serve para os mesmos propósitos previamente referidos.



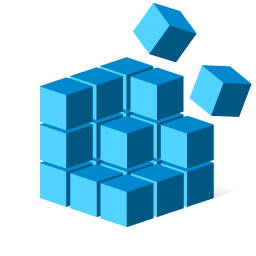
# 3.3. DLL



Na DLL, temos um conjunto de estruturas que armazenam informação sobre a posição dos objetos e jogadores e também o número total de carros, etc.

Temos também definidas macros para ambas as secções de memória partilhada.

# 3.4. REGISTRY



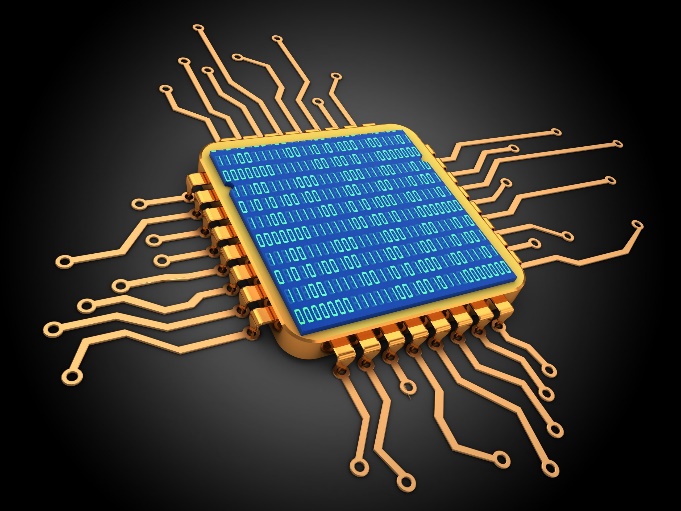
# Com o intuito de manipulação e gestão da registry, achamos por melhor criar 3 funções:

1. **createOptions():** Esta função destina-se à criação das chaves e atribuição de valores às mesmas através de outra função setOptions();
2. **loadOptions():** Esta função destina-se apenas a carregar as opções já criadas.
3. **setOptions():** Esta função tem como principal objetivo atribuir valores às chaves sendo usada na função createOptions().

## 

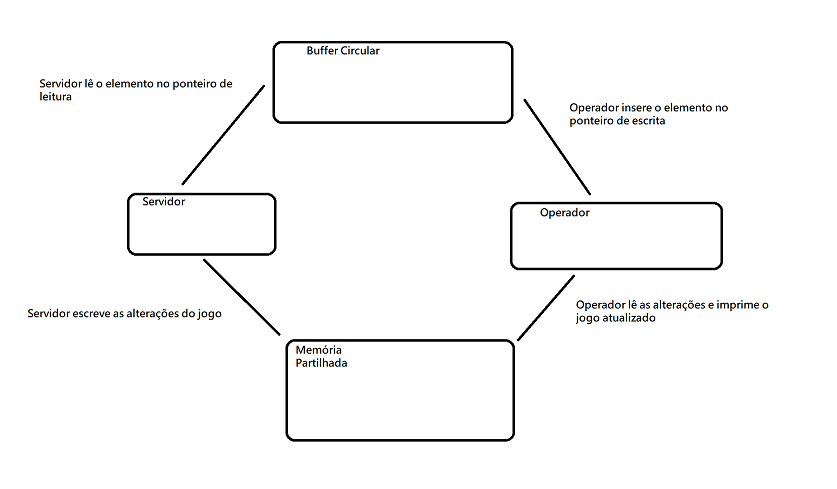
**CAPÍTULO 4**

**MECANISMOS DO JOGO**

**4.1. MECANISMOS DE COMUNICAÇÃO**

Relativamente à comunicação entre o servidor e o operador utilizamos dois mecanismos:

1. **Memória Partilhada:** O servidor escreve na memória as alterações do jogo e o operador lê as alterações e imprime no ecrã.
2. **Buffer Circular:** O comando insere os comandos no buffer circular e o servidor lê, de forma cíclica.

****

**4.2. MECANISMOS DE SINCRONISMO**

# Sincronía. – Burbujitas de Esperanza

# Para garantir a sincronização adequada, empregamos dois mecanismos distintos. Utilizamos mutexes, que impedem a execução simultânea de múltiplos servidores, assegurando a exclusividade de acesso. Além disso, empregamos eventos, que servem para notificar o servidor sobre a necessidade de ler um comando ou informar ao operador sobre atualizações no jogo que requerem intervenção.

**CAPÍTULO 5**

**CONCLUSÃO**

# 5 CONCLUSÃO

Em conclusão, neste trabalho de Sistemas Operativos 2, utilizamos a linguagem de programação C juntamente com a API do Windows para implementar o jogo Frogger. Durante o processo, aplicamos uma variedade de conceitos e técnicas, incluindo o uso de threads, mutexes, DLLs e manipulação do registro do sistema.