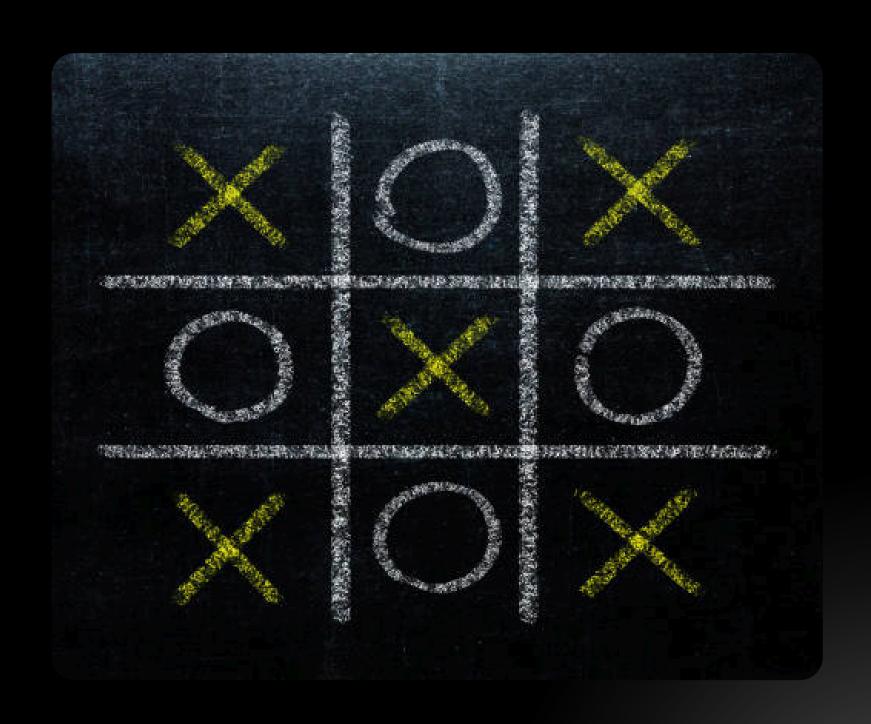


CONCORRÊNCIA DE THREADS

DOCENTE: HEBERT ROCHA
DISCENTES ARTHUS RAMOS E YAN TEIXEIRA

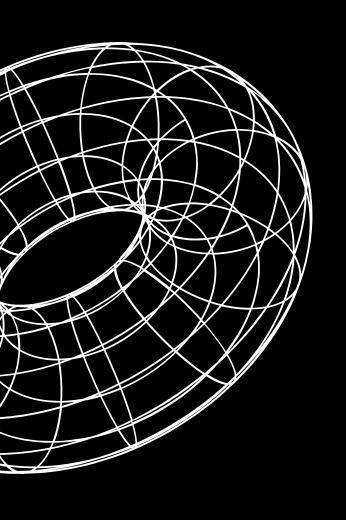
SÚMARIO

- Introdução;
- Ferramentas utilizadas;
- Procedimento realizado;
- Prévia das funções;
- Implementação dos códigos



INTRODUÇÃO

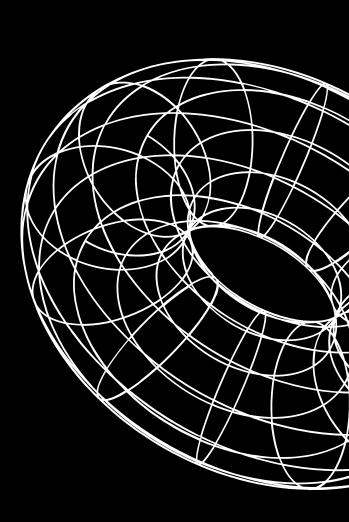
O presente trabalho tem como objetivo o desenvolvimento de uma versão automatizada do clássico jogo da velha (também conhecido como "tic-tac-toe"). Este projeto simula a partida entre dois jogadores artificiais, representados por threads distintas que interagem simultaneamente com um mesmo tabuleiro 3x3. Por se tratar de um recurso compartilhado, é fundamental que cada thread respeite sua vez de jogar, garantindo a integridade das regras do jogo e evitando conflitos de concorrência. Assim, este projeto também serve como uma aplicação prática de conceitos de programação concorrente e sincronização de threads, oferecendo uma abordagem didática e interativa para esses temas.



FERRAMENTAS UTILIZADAS







PROCEDIMENTO REALIZADO

- Criado um tabuleiro de matriz 3x3, dois jogadores alternando jogadas (X e O).
- Threads: execução paralela de tarefas.
- Recurso Compartilhado: o tabuleiro precisa ser acessado por ambos os jogadores.
- Sincronização: garantir que um jogador jogue por vez (controle de acesso).
- Controle de turnos entre os jogadores.
- Verificação de jogadas válidas e fim do jogo.

PRÉVIA DAS FUNÇÕES

```
GameBoard* create_board();
void free_board(GameBoard* board);
void display_board(const GameBoard* board);
bool is_valid_move(const GameBoard* board, int row, int col);
bool make_move(GameBoard* board, int row, int col, char player);
bool check_winner(const GameBoard* board, char player);
bool is_board_full(const GameBoard* board);
bool game_ended(const GameBoard* board);
#endif
```

IMPLEMENTAÇÃO DOS CÓDIGOS

