Programação Concorrente - Projeto Final

Aluno

Nome: Artur Filgueiras Scheiba Zorron

Matrícula: 180013696

Para executar

```
make
```

Introdução

Existe um jogo de tira para console chamado "Call of Duty: Black Ops" o qual possui um modo de jogo chamado "Gun Game".

"Gun Game" se baseia em um jogo de tiro no estilo "Free for All" (todos contra todos) onde 6 jogadores em um mapa fechado se enfrentam.

Dentro do "Gun Game" existem 20 armas a serem conquistadas, onde a cada morte a seu favor, você ganha a próxima arma. Vence quem conseguir matar alguém com a última arma a ser conquistada, ou seja, conseguir matar com todas as armas disponíveis do "Gun Game".

Caso você morra por facada de outro jogador, você é "humilhado" e volta uma arma na sequência.

Para este trabalho, desenvolvi o modo de jogo "Black Ops: Gun Game" no mapa "Nuketown" onde os personagens do jogo são meus amigos que jogavam comigo este jogo na época que eu ainda jogava.

Formalização do Problema Proposto

Para este trabalho desenvolvi um algoritmo utilizando conceitos de programação concorrente e da biblioteca POSIX Pthreads a fim de poder gerar a experiência de estar lendo os logs de uma partida real de "Black Ops: Gun Game - Nuketown".

Neste projeto cada personagem, que são 6 por partida, serão threads do meu programa onde, dentro da função do jogador, poderão:

- Nascer no mapa ao início do jogo
- Renascer no mapa após ter morrido
- Se locomover entre diferentes áreas do mapa
- Entrar em combate com outros personagens

O jogo possui:

- Uma tela de início
- Uma listagem inicial dos jogadores escolhidos para a partida
- Uma distribuição inicial dos jogadores pelo mapa
- Log dos eventos do jogo (movimentações, renascimentos, mortes, humilhações)

- Ranking final da partida

Struct de cada jogador

Esta struct armazena o nome do jogador, o indice da arma que ele possui atualmente, o indice da area do mapa que ele se encontra e seu status de vivo ou morto.

```
typedef struct {
    char name[12];
    int gun;
    int area;
    int is_dead;
} player;
```

Seu id é a sua posição na array.

```
player players[6];
```

Struct de cada área do mapa

Esta struct armazena o nome de cada área e uma lista de conexões desta área com outras a partir dos seus ids.

```
typedef struct {
    char name[21];
    int connections[6];
} nuketown_area;
```

O id de cada área é dado pelo seu índice na array.

```
nuketown_area nuketown[9];
```

Lista dos possíveis jogadores da 102FR:

```
char all_players[16][12] = {
    "AUGUSTO",
    "BERNARDES",
    "FILLIPO",
    "GOIAS",
    "JOHN JOHN",
    "LEO ARANTES",
    "LEOZINHO",
    "LUIZ MITO",
```

```
    "MATEUS",
    "MICHEL",
    "PAGANO",
    "PZ",
    "SANTISTA",
    "TUCA",
    "VITAO",
    "VITINHO",
};
```

Lista de armas disponíveis:

```
char guns[20][22] = {
  "PYTHON SPEED RELOADER",
  "MAKAROV DUAL WIELD",
  "SPAS-12",
  "STAKEOUT",
  "MP5K",
  "SKORPION DUAL WIELD",
  "AK74U",
  "M14",
  "M16",
  "FAMAS",
  "AUG",
  "HK21",
  "M60",
  "L96A1",
  "WA2000",
  "GRIM REAPER",
  "M72 LAW",
  "CHINA LAKE",
  "CROSSBOW",
  "BALLISTIC KNIFE",
};
```

Logs do programa

Morte:

```
MORTE!
LEO ARANTES (CROSSBOW) MATEUS
LEO ARANTES: CROSSBOW => BALLISTIC KNIFE
```

Isso significa que "LEO ARANTES" matou "MATEUS" usando a arma "CROSSBOW".

Com essa morte, "LEO ARANTES" sai da arma "CROSSBOW" e passa para a arma "BALLISTIC KNIFE".

Humilhação:

```
HUMILHAÇÃO!
MATEUS humilhou TUCA
TUCA: M72 LAW => GRIM REAPER
```

Isso significa que "MATEUS" matou "TUCA" usando uma faca. Com isso "TUCA" volta de arma, sainda da "M72 LAW" e indo para a "GRIM REAPER". "MATEUS" não avança de arma.

Movimentação:

```
VITAO entrou em CASA VERDE JARDIM
```

Isso significa que "VITAO" entrou na área chamada "CASA VERDE JARDIM".

Nascimento:

```
MICHEL nasceu em RUA
```

Isso significa que "MICHEL" nasceu na área "RUA".

Ranking Final

```
RANKING FINAL

1º: LEO ARANTES
2º: MICHEL
3º: VITAO
4º: VITINHO
5º: TUCA
6º: MATEUS
```

Isso mostra qual o ranking que os jogadores ficaram ao final da partida. Vence quem conseguiu pelo menos uma morte com cada arma do jogo.

Descrição do Algoritmo Desenvolvido para Solução do Problema Proposto

Para resolver esse problema eu utilizei o conceito de:

- Threads
- Mutex
- Barrier

Sobre threads, cada thread do meu programa é um jogador que realiza ações já comentadas anteriormente. Dentro de suas ações existem área críticas onde é necessário que cada jogador realize uma ação por vez.

É neste momento que entra o mutex. O mutex foi utilizado dentro do código para estabelecer ordem às ações dos jogadores e não permitir que eles mexam em regiões críticas ao mesmo tempo.

E o barrier foi utilizado antes de entrar no loop principal dos jogadores apenas para garantir que todos entrariam no jogo ao mesmo tempo e que não teriam pessoas vagando pelo mapa sozinhas.

Conclusão

Este projeto foi um projeto bem desafiador e muito divertido pois tive a missão de aplicar conceitos vistos nas aulas e nos estudos dirigidos de programação concorrente dentro de um jogo que me traz boas lembranças da infância. Aprendi muito durante o desenvolvimento deste projeto sobre boas práticas utilizando threads e conceitos de comunicação entre processos.

Referências

Não utilizei referências externas sobre código para este projeto. Apenas meus estudos dirigidos já me deram uma boa base de como aplicar os conceitos da biblioteca POSIX Pthreads.

Porém sobre o jogo utilizei esse site como fonte de informações:

https://callofduty.fandom.com/wiki/Gun_Game