Documentação do Jogo - Corrida pelo Tesouro

Este projeto implementa um jogo multiplayer focado em uma corrida para encontrar os tesouros espalhados pelo mapa e pelas salas de tesouro, até que todos eles sejam reunidos e, assim, um vencedor seja estabelecido.

O desenvolvimento do jogo foi organizado em dois elementos fundamentais: servidor e cliente. O servidor tem a função de coordenar, administrar as ligações e tratar as solicitações feitas pelos clientes. Em contrapartida, o cliente é encarregado de interagir diretamente com o usuário, disponibilizando a interface do jogo, registrando as ações dos jogadores e encaminhando pedidos ao servidor.

Vídeo do Projeto

Servidor

Bibliotecas Necessárias

O código utiliza as seguintes bibliotecas:

- `socket`: Para comunicação em rede.
- `random`: Para gerar posições aleatórias no mapa.
- `json`: Para manipular dados no formato JSON.
- `time`: Para temporização.
- `threading`: Para controle de concorrência (semaforos e threads).

Constantes

- Ícones usados no mapa:
  - `WALL`: Paredes do mapa.
  - `EMPTY`: Espaços vazios.
  - `TREASURE`: Tesouros comuns.
  - `PORTAL`: Portais para salas de tesouro.
  - `PLAYER_1` e `PLAYER_2`: Representam os dois jogadores.
  - `GREAT_TREASURE`: Tesouros especiais.
- Tamanhos:
  - `MAIN_MAP_SIZE`: 16x16 para o mapa principal.
  - `TREASURE_ROOM_SIZE`: 8x8 para as salas de tesouro.

Estruturas de Dados

- **`game_main_map`**: Mapa principal do jogo, representado como uma matriz 16x16.
- **`treasure_rooms`**: Lista de dicionários representando salas de tesouro, cada uma com:
  - `room_id`: Identificador único da sala.
  - `room`: Matriz 8x8 representando a sala.
  - `position`: Coordenadas do portal no mapa principal.
  - `stopwatch`: Temporizador para eventos futuros.
- **`clients`**: Lista de dicionários de jogadores conectados ao servidor, cada um com:

    - `name`: Nome do jogador.
    - `id`: Identificador do jogador.
    - `connection`: Conexão do socket.
    - `position`: Posição atual no mapa.
    - `score`: Pontuação do jogador.
    - `current_map`: Mapa atual em que o jogador está.
    - `map_state`: Estado atual do mapa.
- `treasure_room_semaphores`: Lista de semáforos para controle de acesso às salas de tesouro.
- `map_semaphores`: Matriz de semáforos para controle de acesso às posições do mapa principal.

Variáveis Globais

- `treasure_semaphore`: Semáforo para controle de acesso aos tesouros.
- `remaining_treasures`: Contador de tesouros restantes no jogo.

Funções

Configuração Inicial

1. `set_player_position()`
    - Define as posições iniciais dos jogadores no mapa principal.
2. `set_treasure_position()`
    - Posiciona aleatoriamente 5 tesouros no mapa principal.
3. `set_portal_position()`
    - Posiciona dois portais no mapa principal e vincula cada um a uma sala de tesouro.
4. `set_treasure_room_treasures_position()`
    - Posiciona 5 tesouros (incluindo tesouros especiais) em cada sala de tesouro.

Monitoramento e Interação

1. `monitoring_requests(clients)`
    - Monitora as requisições de movimento enviadas pelos jogadores.
2. `evaluate_move_request(client, message)`
    - Processa os comandos de movimento (`UP`, `DOWN`, `LEFT`, `RIGHT`) e executa ações correspondentes, como:
        - Movimentar o jogador.
        - Coletar tesouros.
        - Entrar em portais.

Movimentação

1. `move_player(client, x, y)`
    - Move o jogador para uma nova posição no mapa.
2. `move_player_to_portal(client, x, y)`
    - Remove o jogador do mapa principal antes de entrar em uma sala de tesouro.
3. `goto_treasure_room(client, x, y)`
    - Adiciona o jogador à fila de uma sala de tesouro usando semáforos.

Coleta de Tesouros

1. `collect_treasure(move_x, move_y, is_great=False)`
    - Coleta um tesouro na posição especificada, atualizando a pontuação do jogador e o contador de tesouros restantes.

Temporizador

1. `treasure_room_timer(client, room_index)`
    - Temporizador que expulsa o jogador da sala de tesouro após 10 segundos.
2. `return_to_main_map(client)`
    - Retorna o jogador ao mapa principal após ser expulso da sala de tesouro.

Finalização

1. `set_winner()`
    - Define o vencedor com base na pontuação dos jogadores.
2. `start_server()`
    - Inicia o servidor e gerencia as conexões durante o jogo.

Observações

- **Semáforos:** As salas de tesouro utilizam semáforos para limitar o acesso simultâneo de jogadores.
- **Mapas:** O sistema utiliza múltiplas matrizes para representar mapas e salas de tesouro.
- **Mensagens:** O servidor exibe mensagens informativas no console, como pontuação e ações realizadas pelos jogadores.

Cliente

Bibliotecas Necessárias

O código utiliza as seguintes bibliotecas:

- `socket`: Para comunicação via sockets.
- `json`: Para manipulação de dados em formato JSON.
- `time`: Para controle de tempo e espera.
- `os`: Para execução de comandos no sistema operacional.
- `pynput.keyboard`: Para detecção de eventos de teclado.
- `threading`: Para execução de threads paralelas.

Constantes

- `TOTAL_POINTS`: Representa o total de pontos disponíveis no jogo. Valor definido como `3100`.

Variáveis Globais

- `player`: Variável que armazena o identificador do jogador atual, recebida do servidor.
- `stopwatch_state`: Indica se o cronômetro está ativo. Inicialmente `False`.
- `stopwatch`: Define o tempo inicial do cronômetro em segundos. Valor inicial: `11`.

Funções

Cronômetro

1. `start_stopwatch()`
    - Inicia um cronômetro decrescente de 11 segundos enquanto o jogador está na sala do tesouro.

Renderização

1. `render_map(game_map)`
    - Renderiza o mapa do jogo no terminal. Caso o jogador esteja na sala do tesouro, ativa o cronômetro.

Conexão

1. `start_client()`
    - Gerencia a conexão do cliente com o servidor:
        - Conecta ao servidor.
        - Recebe mensagens do servidor e executa ações com base no conteúdo das mensagens.
        - Monitora entradas do teclado para movimentação do jogador.
        - Renderiza o mapa recebido do servidor.
        - Envia requisições de movimento e mensagens de status ao servidor.

Entrada do Usuário

1. `on_press(key)`
    - Detecta teclas pressionadas (`w`, `a`, `s`, `d`) e envia a requisição de movimento correspondente ao servidor.
2. `request_move(moviment)`
    - Envia a movimentação do jogador ao servidor utilizando o socket cliente.

Observações

1. O cliente recebe mensagens do servidor para determinar o estado do jogo:
    - **START**: Inicia o jogo.
    - **GAME_OVER**: Encerra o jogo.
    - **WINNER**, **LOSER**, **DRAW**: Exibe o resultado final com os pontos dos jogadores.
2. **Mensagens adicionais**:
    - **READY**: Enviada pelo cliente ao servidor indicando que está pronto para iniciar o jogo.
    - **POINTS**: Enviada pelo cliente ao servidor indicando que está enviando sua pontuação final.
    - **CLOSE**: Enviada pelo cliente ao servidor indicando que está fechando a conexão.

Conceitos

Semáforos

- **Definição**: Um semáforo é uma variável ou um tipo de dado abstrato usado para controlar o acesso a um recurso comum em um ambiente de computação concorrente.
- **Uso no Jogo**:
    - **map_semaphores**: Controla o acesso a cada posição do mapa principal.
    - **treasure_room_semaphores**: Controla o acesso às salas de tesouro.
    - **treasure_semaphore**: Controla o acesso aos tesouros para evitar que dois jogadores peguem o mesmo tesouro simultaneamente.

Exclusão Mútua

- **Definição**: Exclusão mútua é um conceito de controle de concorrência que garante que apenas um processo ou thread possa acessar um recurso compartilhado por vez.
- **Implementação no Jogo**:
    - **Movimentação dos Jogadores**: Semáforos garantem que apenas um jogador possa ocupar uma posição no mapa por vez.
    - **Coleta de Tesouros**: Semáforos garantem que apenas um jogador possa coletar um tesouro por vez.
    - **Entrada nas Salas de Tesouro**: Semáforos garantem que apenas um jogador possa entrar em uma sala de tesouro por vez.