Fundamentos de Processamento Paralelo e Distribuído

Relatório Trabalho 2: Implementação de Jogo Multiplayer Usando Go e RPC

Nomes: Fabrine Drescher Machado, João Pedro Carneiro, Marcelo Nascimento

Introdução

Este relatório aborda a implementação de um sistema multijogador onde outro cliente se conectam a um servidor central para jogar em um mapa compartilhado. Cada jogador interage com o jogo através de um cliente, visualizando o estado atual do jogo em sua própria interface gráfica. O servidor é responsável por gerenciar o estado do jogo e coordenar a comunicação entre os clientes.

Arquitetura do Sistema

O sistema consiste em três componentes principais:

1. **Servidor de Jogo (server_jogo)**: Responsável por gerenciar o estado do jogo, armazenar o mapa e a posição dos elementos dinâmicos, e coordenar a comunicação entre os clientes.
2. **Cliente do Jogo (client)**: Interface gráfica onde o jogador interage com o jogo. Conecta-se ao servidor para obter atualizações do estado do jogo e enviar comandos de movimento e interação.
3. **Interfaces RPC**: Utilizadas para definir os métodos que o servidor deve implementar e que os clientes podem chamar para interagir com o jogo.

Implementação Detalhada

Estrutura de Dados e Elementos do Jogo

- **Elemento**: Estrutura que define os elementos do jogo, como jogadores, inimigos, itens, etc. Cada elemento possui características como posição, símbolo, cor, etc.

- **Mapa**: Representado por uma matriz de elementos. O servidor mantém e atualiza o estado do mapa, incluindo a posição dos jogadores e outros elementos dinâmicos.

Servidor de Jogo (server_jogo)

- **Funcionalidades**:
    - **Gerenciamento de Clientes**: Registra novos clientes, mantém uma lista de clientes conectados.
    - **Atualização do Estado do Jogo**: Processa comandos dos clientes (movimento, interação) para atualizar o estado do jogo.
    - **Comunicação**: Responde às solicitações de estado do jogo feitas pelos clientes através de métodos RPC.
    - **Idempotência**: Garante que comandos repetidos dos clientes (com identificadores únicos) sejam tratados de forma idempotente para manter a consistência do estado do jogo.
- **Implementação**:
    - Utiliza goroutines para tratar conexões concorrentes de múltiplos clientes.
    - Mantém uma representação do estado do jogo em memória.
    - Utiliza RPC (Remote Procedure Call) para definir métodos como RegisterClient, SendCommand e GetGameState.

Cliente do Jogo (client)

- **Funcionalidades**:
    - **Registro e Conexão**: Conecta-se ao servidor RPC, registra-se como cliente.
    - **Interface Gráfica**: Renderiza o estado do jogo em sua própria interface gráfica.
    - **Envio de Comandos**: Envia comandos de movimento e interação para o servidor.

    - **Atualização do Estado**: Periodicamente solicita atualizações do estado do jogo ao servidor e atualiza sua interface gráfica com base nas respostas.
- **Implementação**:
    - Usa goroutines para gerenciar a interface gráfica e a comunicação com o servidor de forma assíncrona.
    - Utiliza termbox-go ou similar para renderização da interface gráfica.

Conclusão

Este relatório apresentou uma análise detalhada da implementação de um sistema multijogador baseado em processamento distribuído e comunicação entre processos, utilizando chamadas de procedimento remoto (RPC). A arquitetura do sistema foi estruturada de maneira a garantir eficiência na gestão do estado do jogo e na interação dos jogadores por meio de um servidor centralizado.

O processamento distribuído foi fundamental na concepção do servidor de jogo (server_jogo), que emprega goroutines para lidar com conexões concorrentes de clientes. Isso permite que múltiplos jogadores interajam simultaneamente com o ambiente compartilhado, garantindo sincronização adequada das ações e atualizações do estado do jogo. A utilização de uma representação em memória para o estado do jogo e a aplicação de técnicas de idempotência asseguram consistência e integridade dos dados, mesmo diante de comandos repetidos ou conflitantes.

A comunicação entre processos é facilitada pelo uso de RPC, onde métodos como RegisterClient, SendCommand e GetGameState são definidos para permitir que os clientes se registrem no servidor, enviem comandos e solicitem atualizações do estado do jogo de forma eficiente e segura. Essa abordagem não só simplifica a interação entre os componentes do sistema, mas também promove uma comunicação assíncrona e responsiva, essencial para a fluidez da experiência do usuário.

Além disso, a implementação do cliente do jogo (client) demonstra a aplicação prática de chamadas de procedimento remoto, conectando-se ao servidor RPC para receber atualizações do estado do jogo e enviar comandos de interação. O uso de

goroutines na gestão da interface gráfica garante que a interação do jogador seja contínua e sem bloqueios perceptíveis, mantendo a experiência imersiva.

Em síntese, o projeto não apenas explorou os fundamentos de processamento distribuído e comunicação entre processos, mas também exemplificou sua aplicação prática em um ambiente de jogo multijogador. A integração harmoniosa desses conceitos proporciona um sistema robusto, escalável e capaz de oferecer uma experiência de usuário satisfatória e dinâmica.