Universidade do Minho

Trabalho Prático da Unidade Curricular Programação Concorrente
Grupo 10

Licenciatura em Ciências da Computação

Ano Letivo de 2022/2023

André Costa (A95869)

Bruno Araújo (A97509)

Filipe Castro (A96156)

Francisco Teófilo (A93741)

Nova Arena

Introdução

No âmbito da unidade curricular de Programação Concorrente foi nos proposta a implementação de um mini-jogo onde vários utilizadores podem interagir usando uma aplicação cliente com interface gráfica, escrita em Java, intermediados por um servidor escrito em Erlang.

No mini-jogo, o avatar de cada jogador movimenta-se num espaço 2D, onde os dois avatares interagem entre si e com o ambiente que os rodeia, segundo uma simulação efetuada pelo servidor.

Neste relatório vai ser apresentado o nosso método para a resolução deste trabalho, descrevendo as nossas decisões tomadas.

Cliente

Na execução do cliente vai ser criado um socket e um tcp para comunicação com o servidor, os objetos info (Informacao), player (Jogador) e enemy (Jogador) e serão criadas duas threads (Menu e Auxiliar) que vão correr concorrentemente partilhando info, player e enemy.

- Connection_manager – Trata de enviar ou receber os pedidos tcp's feitos pelo cliente quando este clica num botão.

  Envia:

  create_account#nome pass
  Para criar uma conta .

  remove_account#nome pass
  Para remover a conta do servidor.

  login#nome pass
  Para dar login da conta.

  logout#nome pass
  Para dar logout da conta.

  leaderboard#
  Para obtermos a leaderboard, é de notar que a leaderboard é o top 5 de jogadores o cliente recebe a informação neste formato a 4|b 3|c 2 . E de seguida guarda-as em duas listas que fazem parte do objeto info, leaderboardNames e leaderboardScores.

  join#nome pass
  Para juntarmo-nos à queue de espera por um jogo, nome e pass desta operação vão servir para autenticar o utilizador.

  leave#
  Para sairmos da queue de espera, caso o utilizador não queira esperar mais um por um jogador do seu nível.

  input#W A D
  Relacionado com o comando para andar/virar.(W A e D são ou true ou false).

- Informacao – É um objeto partilhado concorrentemente entre Auxiliar e Menu que contém as seguintes variáveis:
  Um lock , duas conditions (waitAuxiliar e waitMenu) e um Enum response (varia entre nothing,done,error e switch) que terão o papel de permitir uma alteração de estados concorrente entre Auxiliar e Menu.
  Duas strings referentes ao nome do utilizador e password que o utilizador insere no seu espaço do menu específico no ecrã.
  Uma string prolongamento que tomará conta de ter apenas a palavra overtime quando tivemos num prolongamento de um jogo.
  Uma string answer que guardará as mensagens enviadas pelo servidor.
  E por fim os 2 arrays relativos á leaderboard que já foram previamente mencionados.

- Auxiliar – Classe que tem no seu run um try catch onde este dará lock ao objeto da classe Informação partilhado e esperará pelo sinal da função handleTCPState do Menu e começará a tomar conta dos envios/receções associados ao estado atual desse objeto.
  Por fim sinaliza o waitMenu associado á função previamente mencionada e tendo como sucesso o try , acaba por dar unlock ao objeto que tinha dado lock.

- Jogador – O Cliente recebe constantemente informação do servidor que será guardada em dois objetos desta class, nomeadamente em dois o enemy e o player.
  Algo de notar é que este contém uma variável array booleano de tamanho 3 que em cada posição deste terá false e true dependendo do botão que clicamos.
  Posição 0 corresponde a W , Posição 1 corresponde a A e Posição 2 corresponde a D.
  Esta depois é enviada através do Auxiliar no formato input#false false false.

- Menu – É nesta classe que temos a interface gráfica relativa ao jogo
  O draw e os clickables variam dependendo do estado que nos encontramos.
  Uma função importante que se encontra neste que já foi mencionada anteriormente, é o handleTCPState, este recebe o estado que vem a seguir e um estado caso ocorra algum erro. Este tem no seu try um lock á variável partilhada , de seguida altera o estado da variável partilhada e sinaliza o sinal Auxiliar, depois entra num ciclo de espera que termina após o auxiliar terminar o que tem a fazer naquele estado.
  Por fim dependendo da resposta ele altera o estado atual ou o da variável partilhada.
  Um exemplo que mostra a utilidade desta função, seria quando o utilizador clica em login, caso não existe uma conta criada com aquele username e password , temos um caso de erro então iremos para o estado MENU em vez do LOGGED.

Servidor

Na inicialização do servidor criamos 2 processos, um acceptor que terá a socket para o envio da informação para o cliente e um processo party que tomará da queue dos jogadores.

O processo acceptor para cada cliente que se liga ao servidor, criar um processo acceptor. Este depois torna-se no processo client onde após receber os pedidos do cliente este reencaminha para uma função que irá processar a mensagem e encaminhar para funções que irão trabalhar/modificar um mapa dos users. Além disso para o comando join este processo caso a autenticação seja feita com sucesso este irá comunicar com o processo party tornar-se-á no processo clientGame.

O processo party coloca na queue o cliente e mantém-no á espera até este encontrar outro jogador do mesmo nível, quando isto acontecer ele cria o Jogo processo envolvendo só estes dois depois remove-os da queue. Durante a espera o cliente é livre de sair da queue quando quiser,se assim o fizer o processo clienteGame torna-se no processo client se não o fizer e realmente conseguir encontrar outro jogador este torna-se no processo clienteGameLoop.

O jogo começa na função game que tratará de inicializar o jogo, criar temporizadores para diferentes mensagens que envia ao gameLoop e por fim entra em gameTimer, onde temos o tickrate e diferentes chamadas de funções auxiliares que tratarão de diferentes componentes do jogo, colisões etc. Por fim é chamado o gameLoop.

O processo gameLoop tem varias chamadas disponíveis que tratam de diferentes partes da partida, damos mais importância ao timeout que fará a chamada do gameTimer criando um tickrate e ao Info que processará os inputs recebidos dos clientes e fará os cálculos necessários.

Enquanto ocorre uma partida temos uma constante troca de informação entre o servidor e o cliente, cliente envia os inputs com a indicar as teclas que este está a clicar e o servidor envia diferentes dados importantes para o desenho do jogo (posições dos jogadores e dos powerups etc.).

Quando a partida termina o processo clienteGameLoop torna-se no processo client, de forma a permitir que o cliente possa fazer novamente pedidos de logout etc.

Para que o mapa que é usado pelo handle seja atualizado depois de um jogo terminar, o servidor manda juntamente com as mensagens de vitoria e derrota uma frase deste formato:

update#nome nível numerodepartidasganhas npartidas

Depois o cliente devolve-a para o servidor e este invocará uma função para atualizar o mapa. A variável usada frequentemente chamada npartidas é usada para transitar de nível, se 2*nível= npartidas+1 (sendo este 1 a vitoria da ultima) , npartidas será colocada a 0. Fazendo com que para transitar do nível 1 para o 2 sejam precisas 2 vitorias, do 2 para o 3 seja precisas 4 vitorias etc.

Conclusão

Este projeto fez-nos consolidar melhor a matéria dada durante as aulas afinando os nossos dotes de programação concorrente em erlang e java. Além disso permitiu-nos aprender novos conceitos nomeadamente o processing que era uma ferramenta que nunca tínhamos usado .

Foi um trabalho bastante interessante, mas muito exaustivo, tivemos de investir muito tempo a corrigir bugs que por vezes requeriam alterar várias coisas ou que simplesmente eram frustrantes de perceber porque aconteciam. Gostamos do projeto e estamos satisfeitos com o que conseguimos fazer, quando o jogo começou lentamente a formar-se trouxe-nos bastante motivação para continuar a batalhar para terminarmos com sucesso.

Acabamos com um projeto com todas as funcionalidades do jogo mencionadas no enunciado , presentes exceto o envio do type nas mensagens enviadas pelo Connection manager. Ainda acrescentamos algumas capacidades extras que achamos interessantes, música , diferentes mapas/fundos do jogo associados a um nível específico entre outras.