Universidade do Minho

Licenciatura em Ciências da Computação

Programação Concorrente - Trabalho Prático

Grupo 16

Breno Fernando G. Marrão (A97768)
Tales Andre M. Rovaris (A96314) Tiago Passos Rodrigues (A96414)

2022/2023

Conteúdo

Capítulo 1

Introdução

Este projeto consistiu na criação de um mini-jogo em 2D com interface gráfica em Java que fizemos através do processing e com um servidor em Erlang.

Capítulo 2

Servidor

Para inicializar o servidor é necessário a porta, e o arquivo que contém a informação referente aos usuários, na inicialização do servidor começamos a correr concorrentemente a função acceptor e a função rm.

A função acceptor sempre que receber um novo pedido de conexão, aceita a conexão, corre um novo processo concorrente acceptor que irá esperar novas conexões, e depois encaminha para a função user que irá "ouvir"as informações enviadas pelo cliente e comunicar com o rm, e enviar informações para o cliente.

A função rm é o loop principal do servidor, "toma conta"dos dados, é através desta função que será feita a consulta e a alteração dos dados (criação de conta, login, fechar conta), comunicação com os usuários, procura de partidas e criação das funções concorrentes gameRoom que vai controlar cada jogo.

A função gameRoom tem o papel de guardar os pontos feitos por cada usuário, comunicação com o rm e com a engine e de gerir o tempo da partida. Esta função existe em duas formas ,

uma de inicialização, na qual iniciamos a engine e os timers para o jogo e para a criação dos objetos, estes três correm concorrentemente.

A função engine guarda e calcula as posições de cada jogador, quais teclas estão a ser apertadas, a aceleração, o ângulo, informação sobre o boost, verifica se houve colisões, se um jogador saiu do campo, etc. Também faz a comunicação com os jogadores, enviando as novas posições e novos objetos.

Capítulo 3

Cliente

Fizemos o cliente em processing , pois era mt simples a componente gráfica, e assim podiamos focar mais em outros aspetos como a concorrência.

O cliente quando inicializa o jogo tem várias escolhas a fazer, tais como, login, criar conta, eliminar conta e ver a tabela de líderes. Uma vez que um utilizador faça tanto login como criação de conta, estes mandam a informação para o servidor e serão conectados caso tenham sucesso. Daí vai para um ecra novo onde podem carregar na tecla SPACE para tentar encontrar jogo, onde irão ter a um novo ecrã que dá a informação que estão à procura de um adversário. Uma vez encontrado este, o jogo começa e podem usar as teclas "W"fazer movimento para frente e "E", "Q"para rodar.

Ganha a pessoa que tiver mais pontos ou que não sair fora do campo de jogo. Uma vez que o jogo acaba, o utilizador tem a oportunidade de voltar a entrar no matchmaking.

Para as opções do login, criação de conta e eliminar conta mandamos para o Servidor a informação dos utilizadores e este processa-a. Tabela de líderes também é feito um pedido ao Servidor dos 5 jogadores com o maior número de vitórias. Quando um jogador está no ecrã de matchmaking, uma vez que carrega SPACE, o servidor recebe uma mensagem de procura de jogo e tenta encontrar um adversário do mesmo nível. O jogo em si é todo processado pelo servidor onde o cliente atualiza as variáveis consoante recebe as mensagens.

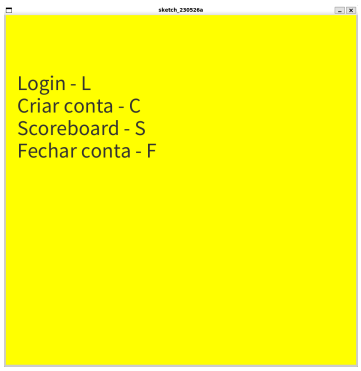
Alguns problemas que tivemos em relação ao cliente foi que como tinhamos uma thread a ler a informação do servidor enquanto outra desenhava as posições e objetos, quando o servidor dizia para o cliente retirar um objeto porque teve uma colisão , tinhamos uma thread a

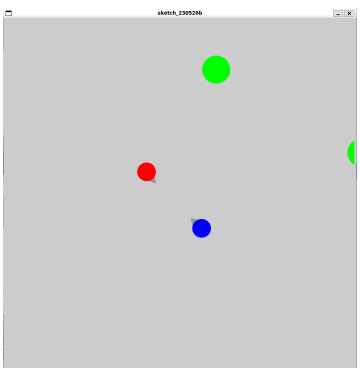
percorrer a lista de objetos e outra a remover um objeto da lista. A nossa solução foi um reentrant lock para quando formos percorrer e remover um objeto da lista.

3.1 Connection Manager

Criamos a class Connection Manager onde ele faz o controlo com o "BufferedReader"e "PrintWriter"das mensagens entre o servidor e cliente.

Este envia as mensagens ao usar o PrintWriter. Para receber uma mensagem ainda fazemos o parsing e adicionamos esta a um HashMap, caso esta seja o type (passado como argumento da função) desejado de quem chamou a função, notificamos as outras threads, senão fico num ciclo até que alguma outra thread tenha adicionado uma mensagem do type desejado.





Capítulo 4

Conclusão

Este projeto foi do mais essencial para compreender e consolidar os conhecimentos aprendidos nas aulas práticas sobre concorreência como criação de threads e comunicação entre processos,.

Sentimos que conseguimos responder eficazmente em relação a todos os desafios propostos, porém tem coisas que poderiam ser melhoradas. A organização do código, a questão gráfica feita no Processing que não é a mais bonita e também possui algumas inconveniências, tais como, não poder ocorrer erros na escrita pois não é possível apagar o que foi escrito no login, criação da conta e fecho da conta, criar a conta e fechar a conta.