Programação Concorrente

Vaga Vermelha

Relatóorio

Luís Albuquerque	Rafael Fernandes	Rafaela de Pinho
A79010	A78242	A77293

31 de Maio de 2018

Conteúdo

Capítulo 1

Introdução

Neste relatório apresentamos o trabalho da unidade curricular "Progamção Concorrente". Este consiste em implementar um mini-jogo onde os utilizadores podem intergir utilizando um cliente com interface gráfica escrita em Java e são intermediados por um servidor em Erlang. Os avatares interagem entre si e com o ambiente que os rodeia, segundo uma simulação efectuada pelo servidor.

O cliente deverá comunicar com o servidor via sockets TCP. Para a interface gráfica do cliente seguimos a sugestão do professor, uasar Processing já que não estavamos familiarizados com nenhuma API para gráficos. O servidor mantendo em memória a informação relevante para fazer a simulação do jogo, receber conexões e input dos clientes bem como fazer chegar a estes a informação relevante para a actualização da interface gráfica.

Este relatório está dividido em 2 partes. Na primeira explicamos o cliente e na segunda parte explicamos o servidor.

Capítulo 2

Jogo-Interface

Quando abrimos o jogo, este abre num menu onde podemos escolher a opção de "Login", "Criar Conta", "Remover Conta"e "Sair". Escolhendo a primeira opção se o "login"for efectuado com sucesso este passa para o segundo menu. Se não for efectuado com sucesso continua lá até ter sucesso, ou poderá escolher a opção de recuperar conta ou voltar ao primeiro menu. Se escolher "Criar Conta"e criar conta for efectuado com sucesso este passa para o segundo menu. Se não continua lá onde pode voltar ao primeiro menu.Se escolher "Remover Conta"e remve conta for efectuado com sucesso. Se não continua lá onde pode voltar ao primeiro menu.

Quando está no segundo menu pode ver o seu nível, as vitórias e o seu score. Pode escolher jogar, ver o ranking, ajuda ou sair. Se quiser jogar fica a espera que encontre um jogador para depois passar para o jogo. No final do jogo volta a ver o ranking e pode sair para o menu ou jogar de novo. Se escolher o ranking mostra o top 3 dos jogadoes.

Capítulo 3

Cliente em Java

Para fazermos o cliente em java criamos 11 classes.

Começamos por explicar **Obj**, esta é uma super classe para todos os objetos que se movem durante o jogo. As classes **Player, Enemy e Energy** extendem a classe Obj. A classe **Player** desenha o "player"e como o servidor sozinho não seria capaz de emular um movimento suave, calculamos também a proxima posição do "player", colisões com outros objetos e se perdeu de alguma maneira. **Enemy** similar a classe "player", **Enemy** calcula a posição da bola vermelha e desenha-a. A **Energy** apenas desenha a bola verde na sua posição.

A classe **Variaveis** contém toda a informação sobre o jogo, pode ser considerada o "estado"do jogo.

A classe **Receiver** cria uma "stream"que recebe as mensagens do servidor, que será utilizada por uma "thread"que se encontra na classe principal.

Sender analoga á classe **Receiver**, cria uma "stream"que enviae as mensagens do cliente ao servidor, que será utilizada por uma "thread"que se encontra na classe principal.

A classe principal **Cliente**, é toda a aplicação. Esta chama a função "setup()"para iniciar a interface gráfica e a "draw()"para desenhar todos os estados, também tem funções para controlar o teclado e o rato. todas as outras classes estão contidas nesta.

As outras classes como **TEXTBOX, TEXTBOXP, Caixas**, são classes auxilias para ter caixas onde se pode escrever texto, caixas onde o texto que se escreve é alterado para o "$*$"e criar caixas com texto.

Capítulo 4

Servidor em Erlang

O servidor em erlang é responsável pela criação e ligação de todas as coneções, guardar informações relativas aos seus utilizadores e todos os cálculos necessários para o jogo funcionar.

O servidor implementado em Erlang é inicializado pela chamada da função start que recebe como argumento a porta.

Loop é o processo que faz a gestao principal do jogo.É um processo que está sempre a correr e que vai atualizando os dados do jogo, e dos jogadores. O **Loop** também controla a criação de conta, os "logins", o que permite fechar uma conta e o que sobe o nivel ao jogaor assim que ele ganhe um jogo.
O loop recebe 4 argumentos o "Map"onde se guarda toda a informaçã (User => { Password,online,level,Pontos,nvitorias}),
o "Level"onde se encontram os jogadores que estão à espera de parceiro para jogar (LEVEL => [Jogador1,Jogador2...Jogadorn]),
p "List"que é o top "3"de jogadores com mais nivel e por último receve "Pids"que associa a cada Pid o número de utilizador.

A função **player** recebe como argumentos o "socket"de onde vai ler, a informação do jogador(Username e Password), uma flag que diz se a informação está completa e recebe por fim o pid do controlador do jogo desse player. Esta gera a comunicção entre o Loop e o jogo.

O servidor tem funções analogas ao java, e são funções auxiliares.

Capítulo 5

Conclusão

Este trabalho abrangeu toda a matéria lecionada ao longo do semestre. Através do conhecimento adquirido criamos um cliente em Java e o servidor em Erlang.

Quando saiu o enunciado do trabalho ficamos muito interessados em aprendar a trabalhar com o Processing e a parte do cliente em Java e o servidor em Erlang ficou um pouco para trás.

Com as dificuldades a cosntruir o servidor e o cliente, não conseguimos acabar o trabalho a tempo.Deveriamos de ter acabado mais cedo, para testar melhor o jogo para encntar eventuais "bugs".

Como trabalho futuro gostaríamos de acrescentar a opções como mudar o tamanho da janela, o jogador poder escolher o seu "background"e o seu avatar. Também queríamos colocar a opção de email para recuperar conta, criar conta e remover conta. Com essa opção poderiamos colocar o servidor a mandar email's para o jogador com os seus dados. Poderiamos também acrescentar um Ranking semanal onde os melhores classificados possam desbloquear "background's"e avatares.