Trabalho Prático
Vaga Vermelha

Grupo: 13
Adriano Campinho - A79032
Artur Queiroz - A77136
Vasco Leitão - A79220

Curso: L.C.C.

31/05/18

1 Introdução

Implemente um mini-jogo onde utilizadores podem interagir usando uma aplicação cliente com interface gráfica, escrita em Java, intermediados por um servidor escrito em Erlang. O avatar de cada jogador movimenta-se num espaço 2D. Os avatares interagem entre si e com o ambiente que os rodeia, segundo uma simulação efectuada pelo servidor.

O objetivo deste relatorio consiste em explicar e esclarecer os passos que tomamos no desenvovimento do trabalho, demonstrar para cada problema exposto no enunciado as solucoes criadas, assim como servir como um guia de utilizacao do programa para um possivel utilizador do mini-jogo desenvolvido.

2 Implementação Geral do Problema

O problema da implementacao do jogo proposto pelo enunciado pode ser divido em duas grandes partes. Estas sao:

- Lógica do Front End:

  Nesta parte vamos ter 3 processos:

  1. responsável por desenhar para o ecrã regularmente (tratado com o draw do processing).
  2. responsável por ler constantemente o que o servidor envia para o socket.
  3. responsável por enviar para o servidor via socket, certos pedidos e informações a pedido do processo 1.

  O processo 1 e 2 vão ter acesso à informação do jogo. Dado que é um objeto partilhado entre duas threads, teremos de tratar a concorrência nesse objeto, bastando apenas usar um ”synchronized” na função de escrita e leitura do tal objeto.

- Lógica do Back End:

  Nesta parte vamos ter 5 grupos de processos:

  1. responsável por aceitar novos Sockets que se queiram connectar e responsabilizá-lo com um processo que o representa(Acceptor).
  2. responsável por representar um jogador e guardar informações locais para menor ”conversa” entre os outros processos(Player).
  3. responsável por guardar a informação de todos os jogadores inscritos no servidor e responder a pedidos sobre essa informação(Login Manager).
  4. responsável por fazer calculos 10 vezes por segundo sobre a fisica do jogo, enviando para os seus dois jogadores um novo estado do jogo(Game).
  5. responsável por representar uma vaga verde ou vermelha, usado para facilitar os calculos(Berrie).
  6. responsável por guardar todos os jogadores que estão à espera de jogar, e atribuir a 2 jogadores qualificados um processo 4 para ele jogarem(Game Manager).

  Como esta parte do código é em Erlang, a comunicação entre processos é por mensagens e cada processo tem a sua informação, não havendo tratamento de concorrência por parte do programador.

3 Front End - Java

A front end para o projeto foi desenvolvida em java. Esta consiste numa interface que permita aos jogadores registarem uma conta e fazerem login no jogo, terem acesso a informações sobre as suas estatísticas e de os três melhores(em termos de jogos vencidos) jogadores e, prioritariamente permitem ao jogador procurar uma partida e jogar com outros jogadores online; Quando uma ação e iniciada, e comunicada uma mensagem ao servidor via sockets TCP, este depois envia uma resposta de volta, e então sao feitas as ações necessárias para atualizar os valores na interface;

Esta parte do projecto foi desenvolvida totalmente em java, a parte da interface foi feita com recurso a API para gráficos sugerida no enunciado (Processing), e podemos dividi-la em 3 grupos:

Menu Pré-Login:
Este grupo consiste nas ações disponíveis ao utilizador do programa antes deste fazer login no jogo. Estas são:

1. ”Jogo Local” - Inicia um jogo local, sendo que o jogo criado funciona da mesma forma que uma partida normal, com algumas diferenças:

   - A lógica da partida e calculada localmente, dentro da aplicação da interface (sem recurso a um servidor exterior em Erlang);
   - Os resultados da partida nao contam para a estatística do jogador. Estas incluem: pontuação, experiência, vitorias e níveis;
   - Ambos os jogadores são controlados com recurso ao teclado: o jogador 1 utiliza as teclas ”W/A/D” enquanto que o jogador 2 utiliza as teclas ”Up Arrow/Left Arrow/ Right Arrow”;

2. ”Exit” - Botão que permite a saída da aplicação;

3. ”Login” - Permite ao utilizador fazer login de uma conta no servidor. Neste será pedido um utilizador e uma password, sendo este par confirmado pelo servidor. Se o login for bem sucedido, passaremos para o próximo estado do programa; se não for bem sucedido, e enviada uma mensagem de erro ao utilizador, notificando o resultado da tentativa;

4. ”Register” - Permite ao utilizador fazer registo de uma conta no servidor. Neste será pedido um utilizador e uma password, sendo este par confirmado pelo servidor. Se o registo for bem sucedido, e criada um novo utilizador com as estatísticas iniciais; Se não for bem sucedido devido a existência de um jogador previamente registado com o mesmo utilizador, e enviada uma mensagem de erro ao utilizador notificando o erro;

Menu Pós-Login:
Este grupo consiste nas ações disponiblizadas ao utilizador do programa depois deste fazer login no jogo. Estas sao:

1. ”Procurar Partida” - Permite ao jogador indicar ao servidor que pretende jogar uma partida. Quando forem encontrados dois jogadores onde o requisito dos níveis identificado no enunciado for satisfeito, a partida será iniciada;

2. ”Reload Ranks” - Faz um pedido de informação ao servidor para atualizar os dois top 3 player ranks exibidos na interface;

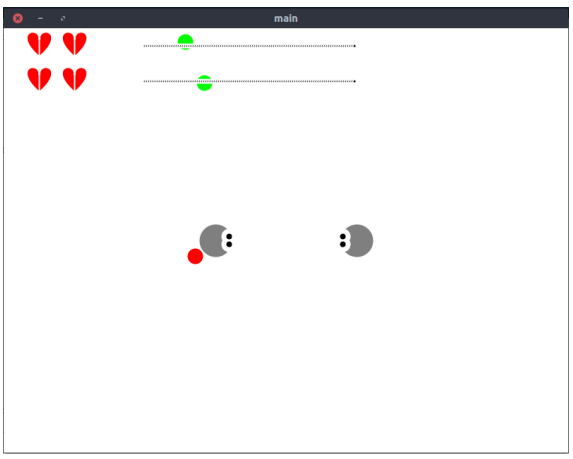
3. ”LogOut” - Permite ao utilizador fazer logout da conta, voltando a interface para o seu estado inicial (pré-login);

4. ”Jogo Local” - Descrito acima;

5. ”Exit” - Descrito acima;

Partida:
Esta consiste na interface de jogo. A janela define a área de jogo, mostrando a posição de ambos os jogadores, assim como também são representadas todas as criaturas vivas. No topo é demonstrada informação sobre os jogadores como a vida e a energia.

Figure 1: Exemplo de uma partida a decorrer



4 Back End - Erlang

O servidor foi desenvolvido em Erlang. Este guarda toda a informacao sobre os utilizadores registados, assim como permite o login e registo de novos utilizadores, fazendo ainda o processamento em simultâneo de todas as partidas a decorrer pelos jogadores. Esta parte do projecto pode ser divida nos seguintes grupos de processos:

1. Berries:
   Este grupo de processos representa todas as vagas de uma partida.
   Contendo um loop principal responsável por atualizar as vagas existentes e também para adicionar novas na tela.

2. Champion:
   Este grupo de processos representa os jogadores de uma partida.
   Contendo um loop principal responsável por atualizar os jogadores.

3. Game Manager:
   Este processo representa o criador de jogos. Pedindo um jogo e havendo um jogador à ”altura” ele ”dá” um jogo para ambos os jogadores jogarem, senão fica à espera.
   Este contem um loop principal responsável por receber inscrições de jogadores.

4. Game:
   Este processo representa um jogo entre duas pessoas.
   Contem um loop principal responsável por receber input dos jogadores, informação se um jogador morreu e pedido de update de informação (esse pedido é recebido 10 vezes por segundo).

5. Login Manager:
   Este processo representa a memória da aplicação, tendo toda a informação de cada jogador, para quando um jogador fizer login, pedir as suas informações.
   Contendo um loop principal responsável por receber pedidos dos jogadores.

6. Server:
   Este processo representa a ”receção” da aplicação. Sendo responsável por atribuir processos ao jogador conectado por um socket.

5 Conclusao

Dando por encerrado este trabalho, não estamos completamente satisfeitos com o resultado, porque faltam fazer muitos refinamentos na apresentacao da interface, bem como melhorias na propia jogabilidade do projeto desenvolvido.
Em geral, os principais objetivos do trabalho foram cumpridos mas a falta de tempo nao nos permitiu apresentar o resultado que gostariamos.