Programação Concorrente

Trabalho Prático – Buraco Negro

Trabalho Realizado Por:

- Isac Meira A70069
- José Brandão A68704
- Miguel Bogas A68699

Índice

Introdução

No âmbito da Unidade Curricular de Programação Concorrente, inserida no 3º ano da Licenciatura em Ciências da Computação, foi nos proposta a realização do Trabalho Prático – "Buraco Negro", elemento de avaliação da Unidade Curricular.

O trabalho consiste na implementação de um mini-jogo onde utilizadores podem interagir usando uma aplicação cliente com interface gráfica, escrita em Java, intermediados por um servidor escrito em *Erlang*.

A dinâmica do jogo é dada pelo avatar de cada jogador movimentando-se num espaço 2D. Os avatares interagem entre si e com o meio, por uma simulação efetuada pelo servidor.

Intermediados pela ferramenta *Processing* para o desenvolvimento da interface gráfica na parte cliente e por Java na comunicação com o servidor via *sockets TCP* e por *Erlang* na escrita do servidor mantendo em memória a informação relevante para fazer a simulação do cenário descrito, recebendo conexões e input dos clientes bem como fazendo chegar a estes a informação relevante para a atualização da interface gráfica.

O servidor suporta várias partidas a decorrer em simultâneo, em que o espaço 2D é retangular, vazio, sem limites nos quatro lados, existindo alguns planetas, de diferentes massas, dispostos num cenário predefinido e no centro deste espaço encontra-se a existência de um buraco negro, fixo, com massa bastante maior que a dos planetas.

Os avatares são em forma de círculo, com o tamanho de acordo com a massa do avatar, exceto o buraco negro, que tem um tamanho pequeno em relação à sua massa.

Os dois jogadores têm massa igual e muito menor do que os planetas. Possuem uma direção para onde estão voltados, sinalizada graficamente.

Cada jogador e planeta deslocam-se pelas regras da mecânica clássica, considerando a massa concentrada no seu centro. Os jogadores têm ainda velocidade angular, que não afeta esta simulação, que deverá ser feita através de 3 teclas (esquerda, direita, frente) que controlam 3 propulsores, provocando aceleração enquanto estão a ser premidas; as duas primeiras provocando aceleração angular, na direção respetiva, e a terceira aceleração linear na direção para onde está voltado o jogador.

Cada propulsor gasta combustível enquanto esta a ser utilizado. Os jogadores começam com a mesma quantidade de combustível, que quando esgotado, os propulsores deixam de funcionar.

O primeiro jogador a entrar no buraco negro ou sair do retângulo de jogo perde.

Cliente - Processing

No cliente existem duas t*hreads*, uma de receção e uma de desenho.

A *thread* de receção recebe mensagens provenientes do *erlang*. Esta encontra-se no *setup* e está constantemente a atualizar a posição dos planetas, a posição das naves e seu combustível remanescente e se as naves se encontram no ecrã ou se caíram no buraco negro.

A *thread draw* está implícita no *processing*. Esta desenha a cena e os seus objetos (planetas e naves).

De salientar o facto de quando a *thread* de receção está a atualizar os valores das variáveis, o *draw* não mexe nas variáveis.

Os cálculos necessários ao correto funcionamento da aplicação são todos realizados no servidor escrito em *Erlang*. O *Processing* apenas desenha conforme os valores que recebe, realizando o *parsing* dessas mesmas mensagens.

No processo de comunicação com o servidor, o cliente envia os seguintes comandos ao servidor:

- **F** – simboliza um movimento em frente;
- **E** – simboliza um movimento à esquerda;
- **D** – simboliza um movimento à direita;
- **Frame** – a cada frame requisita ao servidor que envie mensagem com as posições dos planetas.

É enviado também um registo de um utilizador ao servidor, como também tentativas de *login*.

Servidor – Erlang

Ao executar-se o servidor, este cria listas de espera e gestor de logins, invocando então método *acceptor*. Este por sua vez, espera que alguma *socket* se conecte e invoca outro *acceptor* passando então o atual a ser *user*.

O *user* fica à espera de mensagens do cliente (registar e login) e envia-as ao gestor de logins. Após o login confirmado o *user* passa a *user login*, enviando previamente uma mensagem à lista de espera para o colocar lá.

A lista de espera quando verifica 2K jogadores cria K jogos e envia uma mensagem a cada jogador, passando assim o servidor para o estado *user jogo*.

Este estado é responsável pela troca de mensagens entre o cliente e o jogo, sendo o jogo responsável por gerir todas as posições e dados da partida.

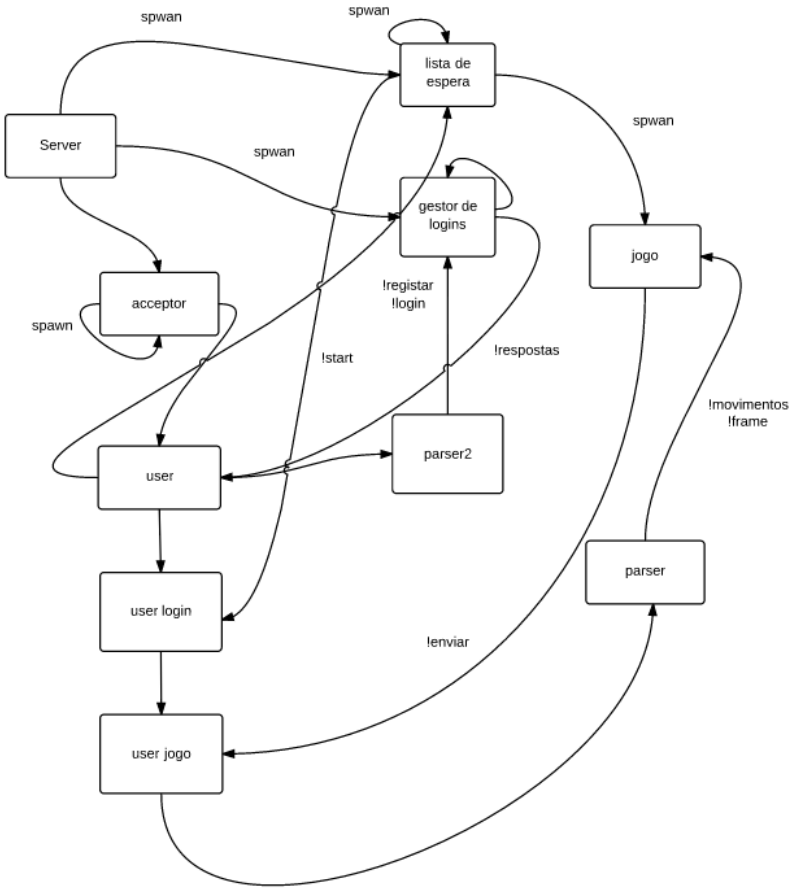

Figura 1 - Esquema de Modelação do Servidor em Erlang

Conclusão

A nossa proposta de solução ao trabalho prático de Programação Concorrente, apesar de funcional, poderia ser refinada, acrescentando *scores* aos jogadores nas suas partidas, bem como um ranking de scores, tal como fora proposto no guião do trabalho prático e a correção de alguns *bug's* que se manifestaram ao longo do período de testes ao código desenvolvido.

Não obstante, à falta destes pormenores, o trabalho desenvolvido pelo grupo, consuma os conhecimentos lecionados ao longo do ano na unidade curricular.

Desta feita, aumentamos a nossa destreza no uso de mecanismos de controlo de concorrência, aumentámos a nossa perceção do modelo cliente-servidor, bem como aprendemos a utilizar novas ferramentas como o *Processing* no uso da interface gráfica e a ferramenta *Erlang* na escrita do servidor.

Temos em nós que o trabalho realizado pelo grupo cumpre as expectativas de avaliação delineadas pelo professor a quando da formulação do mesmo e esperámos que tenha desfrutado da solução ao problema idealizada por nós.