Programação Concorrente
Trabalho Prático - Gravidade

Paulo Alexandre Azevedo Ferreira
(A74816)

Vitor Hugo Gonçalves Dias
(A76531)

11 de Junho de 2017

Conteúdo

Capítulo 1

Introducão

Neste trabalho foi desenvolvido um jogo multiplayer onde o objetivo de cada jogador é sobreviver o máximo de tempo possível aos obstáculos encontrados (planetas). O trabalho é constituído pela estrutura clássica cliente servidor, o servidor é desenvolvido em Erlang que comunica através de Sockets TCP com o cliente desenvolvido em Java. Este relatório está separado pelas várias funcionalidades do jogo e em cada ponto será explicada de uma forma resumida a estratégia utilizada para resolver o problema.

Capítulo 2

Servidor

2.0.1 Registo de utilizador

Como em qualquer outra aplicação/jogo é possível efetuar um registo dos dados do utilizador para posteriormente efetuar o login. Da mesma forma que é possível se registar, também há a possibilidade de eliminar esse registo. Esta funcionalidade faz com que cada jogador para entrar na área de jogo, tenha de ser previamente autenticado pelo servidor. Para suportar esta funcionalidade foi desenvolvido um módulo "LoginManager"que trata da gestão dos registos, bem como logins e remoção de contas. Quando um utilizador efetua um registo, este é adicionado a um Map, juntamente com a sua password. Quando posteriormente efetua o login, o servidor verifica se está presente e da permissão para a sua entrada.

2.0.2 Participação no jogo

O jogo suporta um máximo de 4 jogadores em simultâneo e sempre que um novo jogador entra em jogo quando este limite é atingido o jogador recém entrado terá de esperar até que alguém saia do jogo ou acabe por morrer. Para tal utilizamos um Map com o Username como chave e um tuplo com as informações do avatar como valor apenas com os utilizadores a jogar e uma queue com o username dos utilizadores à espera para jogar, quando o tamanho do Map for 4, coloca-se o username do utilizador recém entrado na queue. Sempre que um utilizador que estava no Map sai do jogo, retira-se o Username que estava à espera à mais tempo na queue para entrar em jogo.

2.0.3 Espaço

O área de jogo é um espaço em 2D, com forma retangular, vazio e sem limites nos seus quatro lados. Nesta aréa posteriormente estarão os avatares dos jogadores e planetas, bem como algumas informações relevantes.

2.0.4 Planetas

Foram criados 3 planetas com tamanhos diferentes e velocidades aleatórias, quando o servidor é iniciado este gera a posição dos 3 planetas e sempre que algum utilizador entra em jogo as suas coordenadas são enviadas para o cliente o desenhar.

2.0.5 Avatares

Cada jogador e planeta tem associado um avatar. Este avatares são formas circulares, sendo a sua área de acordo com a massa do avatar. Os jogadores possuem, entre eles, massa igual e menor quando comparados com os planetas. Os jogadores têm também uma direção para onde estão voltados. De forma a concretizar esta funcionalidade, é gerado um avatar para cada jogador que entre na área de jogo com algumas características comums e outras diferentes. Este avatar gerado é associado num Map ao utilizador sendo Username,Avatar. A direção, é sinalizada graficamente utilizando uma simples primitiva do processing para a desenhar.

2.0.6 Simulação de movimento

Para a movimentação dos planetas foi criado um processo que a cada 100 milissegundos atualiza a sua posição e envia esta informação ao cliente para o desenhar na sua nova posição.

2.0.7 Movimentação dos Jogadores

A movimentação dos jogadores é feita através de 3 teclas (esquerda, direita, frente) que provocam uma rotação/movimento, respetivamente. Porém estes movimentos têm a si associdos uma bateria, que conforme o avatar se movimenta, descarrega a sua energia. Como tal, quando uma destas teclas é pressionada, o servidor recebe uma mensagem com a tecla em questão bem como o utilizador que a pressionou, para que desta forma seja calculada a sua nova posição ou direção. Esta informação é depois enviada de volta para o cliente para ser atualizada a sua interface gráfica. A energia das baterias está associada a estes movimentos e como tal quando estes se verificam, a energia do propulsor correspondente é decrementada. A bateria é incrementada por um processo que sempre que passam 4000 milisegundos incrementa a sua energia.

2.0.8 Escudo

Cada jogador tem um escudo que repele outros jogadores que eventualmente se encontrem no caminho, como tal, sempre que há um movimento de um avatar avalia-se a sua nova posição e posição de todos os outros avatares presentes em jogo, caso a distância entre os mesmos seja menor do que a soma do seu raio com a soma do raio do avatar atingido entao o avatar atingido é repelido para uma nova posição.

2.0.9 Colisões

O objetivo do jogo é sobreviver o mais tempo possível e isso é feito evitando colições com planetas, ou evitar sair da área de jogo. Como tal, as colisões entre jogadores ou entre planetas são ignoradas, porém colisões entre jogadores e planetas fazem com que o jogador perca. Em cada momento do jogo, o servidor verifica a posição relativa do jogador em relação a cada um dos planetas, se a distancia entre o centro do avatar do jogador e do planeta for menor que a soma dos seus raios, então eles estao em contacto, pelo que esse jogador perde.

2.0.10 Pontuação

A pontuação dos jogadores é o numero de segundos que o jogador conseguiu sobreviver, para a pontuação foi utilizado um processo idêntico ao movimento dos jogadores, onde um processo a cada segundo incrementa um contador associado a um determinado jogador e envia esta informação ao cliente.

2.0.11 Listagem das Pontuações

No ambiente do jogo é demonstrada uma listagem das pontuações dos jogadores atualmente em jogo, esta listagem é obtida sempre que a pontuação é atualizada tal informação é também enviada para o cliente para este a imprimir

Capítulo 3

Cliente

O Cliente é constituído por duas threads, a thread principal que é responsável por desenhar o ambiente gráfico, a outra thread é responsável por receber a informação do servidor e passá-la à thread principal responsável pelo desenho. Um jogador inicia o cliente, conecta-se ao servidor e é iniciada a thread que ficará à espera das mensagens vindas do Servidor. Quando uma mensagem é enviada, a thread que leu a mensagem do Socket, escreve a informação num objeto para depois a thread responsável por desenhar o ambiente gráfico ler tal informação e desenhar a mesma. Por estarem duas threads constantemente a aceder ao objeto que contém a informação, uma das threads a ler outra a escrever, foi necessário fazer controlo de concorrência para tal utilizamos um ReentrantLock e sempre que o objeto é acedido, a thread obtém o lock do objeto libertando-o mais tarde quando não precisar do mesmo.

Capítulo 4

Conclusão

Apesar da dimensão reduzida, quando comparado com outros projetos de jogos da mesma natureza, este projeto deu-nos umas percepção mais concreta de como é tratada toda a concorrência e atualização do estado neste tipo de jogos/aplicações onde estão várias pessoas ligadas a um servidor comum. Concretamente neste trabalho, uma das dificuldade encontradas inicialmente foi estabelecer toda a coneção existente entre processing(interface gŕafica), java(receber e enviar informações para o servidor) e servidor(que trata de praticamente tudo, desde colisões, atualização de quem estão a jogar, bem como as suas posições no mapa. Depois de termos conseguido ultrapassar esta dificuldade inicial, fomos construindo como que uma casa(fazendo uma analogia). De modo geral conseguimos concretizar, de forma mais ou menos satisfatória, todos as funcionalidades pedidas para este trabalho prático.