Gravidade
Unidade Curricular de Programação Concorrente

Licenciatura em Ciências da Computação
Universidade do Minho

Rúben Silva (A94633)

25 de maio de 2024

Índice

Capítulo 1

Introdução

No âmbito da unidade curricular de Programação Concorrente foi proposta pelo docente Paulo Sérgio Soares Almeida a implementação de um jogo em servidor denominado *Gravidade*, com o intuito de consolidar a aprendizagem dos dois paradigmas abordados na UC, consistindo numa aplicação distribuída com cliente e servidor.

Assim, o jogo deve permitir que vários utilizadores interajam utilizando uma aplicação cliente com interface gráfica, escrita em *Java*, através da biblioteca *Processing*, intermediados por um servidor escrito em *Erlang*. O avatar de cada jogador movimenta-se num espaço 2D. Os vários avatares interagem entre si e com o ambiente que os rodeia, segundo uma simulação efetuada pelo servidor.

Neste documento será apresentada a solução do problema, descrevendo de forma sucinta as decisões tomadas na implementação da aplicação cliente e do servidor.

Capítulo 2

Solução do Problema

2.1 Registo de utilizador

Para registar o utilizador no banco de dados do Servidor (ficheiro .txt), cada cliente após ter estabelecido a conexão TCP inicial tem acesso a 3 botões e a 2 campos para digitar (username e password). Os botões são Criar, Logar e Apagar a conta respetivamente. Quando esses botões são pressionados, o conteúdo que se encontra nos campos são entregues ao servidor e este processa os dados em questão. Se Criar conta, uma nova conta nível 1 é colocado no ficheiro, se quiser apagar a conta, é verificada a password para aprovar o apagamento da mesma. Se pressionado o botão referente ao Login, então o jogador é automaticamente colocado numa queue do lobby para Jogar.

2.2 Progressão

Sempre que dentro de uma partida, algum jogador morre, é contabilizado no ficheiro .txt que perdeu uma partida. Se ganhar, é contabilizado que ganhou. Se obtivermos os requisitos do enunciado, então o jogador desce ou sobe 1 nível.

2.3 Partidas

O lobby é responsável por esta parte da solução criando uma queue e aumentado-a sempre que algum utilizador pede ao servidor pra jogar. Sempre que existe 2 jogadores que querem jogar e este possuem o nível de diferença "<= 1", então é criada uma lista que suporta os Processos Ids dos jogadores da próxima partida a acontecer. O lobby então trata de esperar 5 segundos por mais utilizadores que respeitem o nível de diferença. Se ninguém entrar, então é criada uma thread com a partida e o lobby volta a procurar jogadores para formar uma nova partida

2.4 Espaço

O espaço é um retângulo 1280x720 sem bordas, mas se o jogador não respeitar os limites, este perderá a partida. Existe um sol fixo no meio com 150 de raio.

2.5 Planetas

É gerado aleatoriamente 6 a 12 planetas com tamanhos variados, vetores e velocidades iniciais aleatórias. Os planetas podem "nascer"em qualquer lado do mapa respeitando sempre onde o sol está e onde os jogadores potencialmente estarão no início da partida para evitar "spawn deaths".

2.6 Avatares

Os Avatares são círculos com um traço radial a representar para onde eles estão a direcionar caso queriam propulsionar as naves. Eles teem posições de nascimento pré-definidas para evitar que o jogo tenha "Spawn Disavantage". As posições sao os 4 cantos no mapa em que todas começam a apontar para o centro do Sol.

2.7 Gravidade

Para ser obtido a gravidade é incrementado a cada tick do servidor um valor às componentes de deslocamento do objeto. Para obtermos os resultados pretendidos é calculado o vetor entre o centro do sol e o centro do Planeta ou Avatar e após sabermos as componentes X e Y então é aplicado os conceitos de trigonometria a esse valor inicialmente escolhido. Assim foi obtido um efeito de aceleração pretendido.

2.8 Movimentação dos jogadores

Os jogadores possuem 3 propulsores, 2 angulares e 1 direcional. Nos propulsores angulares simplesmente foi incrementado ou decrementado o valor da rotação do objeto. Dependendo da tecla pressionada pelo utilizador, quando o propulsor direcional é ativado, é necessário decompor o vetor direcional nas suas componentes básicas e incrementar-las individualmente a partir das fórmulas fundamentais da trigonometria básica.

2.9 Colisões

A cada tick do servidor, todos os objetos presentes no espaço são calculados com o jogador, isto é, primeiramente é calculado o vetor entre o jogador e o objeto em questão, e se a normal desse vetor for menor que o raio do jogador, então este perde ou provoca a "colisão elástica"(troca os valores das velocidades X e Y um com o outro, preservando as velocidades angulares).

2.10 Pontuação

Quando só sobrar 1 jogador na partida, é ativado uma thread com o contador de 5 segundos, se esta thread não for comunicada que o jogador colidiu com algum obstáculo, então o jogador ganha, Se a thread for avisada, então o jogador empata.

2.11 Listagem do top 10 de jogadores

Logo após a coneção TCP feita entre o servidor e o cliente, este último recebe os 10 melhores jogadores para serem processados posteriormente no canto esquerdo superior do cliente.

O servidor quando é inicializado regista um processo principal com o nome do módulo e cria um *socket* de *TCP* e dois processos, o *acceptor* e o *party*. O servidor vai também ler o ficheiro de registos onde está armazenada a informação dos clientes já registados e guarda a informação num mapa.

Capítulo 3

Cliente-Servidor

3.1 Estruturação do Cliente

O cliente feito em Processing (java) é composto por um Main, clientConnection, Login, Mob (planetas), Player e Sun. Estas classes (excluindo o Main) são responsáveis por todo o funcionamento gráfico do Jogo.

Ao ser iniciado o Cliente, a conexão TCP é feita e é criada uma thread inicial que será responsável por estar constantemente a ouvir o que servidor tem a dizer. Os Planetas e os Players estão em sincronizados durante a sua renderização e registo no cliente para evitar problemas futuros.

Em suma, o Cliente simplesmente mostra graficamente o que acontece no servidor e diz ao servidor quais as teclas que são pressionadas.

3.2 Estruturação do Servidor

O Servidor é inicialmente inicializado com 2 threads, 1 relacionado com o jogo em si e a outra relacionado com o comunicador do servidor para o cliente.

Quando alguém estabelece conexão com o servidor, é criado uma nova thread para alocar um novo jogador e assim sempre infinitamente. Esta thread envia uma mensagem ao lobby com os dados do utilizador recém-logado, e posteriormente, esta mesmo, vira torna-se a thread que entregará os dados ao comunicador e posteriormente ao Utilizador o que está a acontecer na simulação do jogo em questão. Esta thread mesmo antes de se tornar somente uma thread de escrita, é criada uma nova thread para ser a escuta do Cliente e o servidor.

O Lobby é uma parte do programa que está constantemente a procurar jogadores e caso que as condições necessárias sejam satisfeitas cria partidas entre 2 a 4 jogadores. Estas partidas são threads individuais que são responsáveis pela simulação toda (o jogo em si).

A thread responsável pelo jogo cálcula toda a simulação. Quando só existe 1 jogador em campo, é ativado uma nova thread responsável por ser o timer da final da partida.

Capítulo 4

Conclusão

Durante a realização deste trabalho aplicamos todos os conhecimentos adquiridos durante a decorrência da UC.

Consideramos que foi alcançado os objetivos esperados, sendo que foi sentida uma crescente experiência ao longo do projeto, desde de comunicação cliente servidor, até à gestão interna de processos.

Infelizmente, o mau planeamento da solução levou a um aumento da complexidade superior ao desejável. O facto de ter sido a primeira experiência com uma linguagem funcional num desafio desta complexidade. Foi tomada algumas decisões que atrasaram o percurso e potencialmente não seriam as mais viáveis a nível de performance em si.

Todo este projeto levou-nos também a conhecer melhor a ferramenta `Processing` e a linguagem Erlang que demonstrou-se ser uma Linguagem muito poderosa para este tipo de tarefas

Por fim, podemos garantir que este trabalho fundamentou as nossas bases para que futuramente consigamos tirar um maior proveito do que foi lecionado durante toda esta UC.