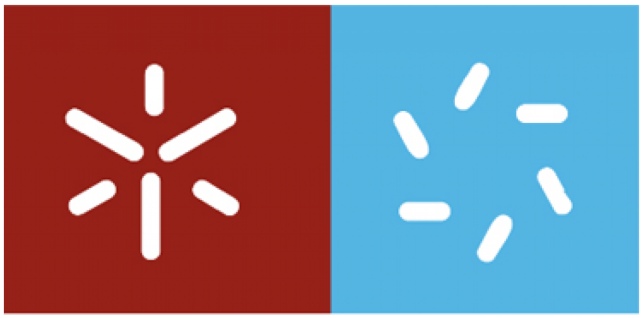

Universidade do Minho - Ciências da Computação

Programação Concorrente

01/06/2024

Relatório escrito

Trabalho Prático - Gravidade

André Silva - A100094

Paulo Freitas - A100053

Miguel Martins - A97441

Wellyson Vieira - A100078

Grupo 13

Índice

1 Introdução

Neste trabalho, no âmbito da cadeira de Programação Concorrente, foi-nos pedido a implementação de um mini-jogo chamado Gravidade onde vários utilizadores podem interagir usando uma aplicação com interface gráfica, intermediados por um servidor. O avatar de cada jogador deverá se movimentar num espaço 2D. Os vários avatares interagem entre si e com o ambiente que os rodeia, segundo uma simulação efectuada pelo servidor.

A interface gráfica será desenvolvida em *Java* e fará uso da biblioteca *Processing*. O servidor será escrito em *Erlang*, tornando assim este trabalho, num aglomerado de tudo o que foi lecionado na unidade curricular durante o presente semestre.

2 Cliente

2.1 Player

A classe *Player* é usada para organizar as informações dos jogadores, tratando assim do seu nome de utilizador, vitórias, derrotas e o seu nível.

2.2 Planet

Esta classe, à semelhança da anterior, aglomera as informações de um planeta e também trata de definir o método para o desenhar, fazendo uso do *Processing*.

2.3 Astronaut

Esta classe faz exatamente o mesmo processo que a classe *Planet*, só que para os nossos astronautas.

2.4 Jogo

A nossa classe *Jogo* faz a gestão de todos os astronautas e planetas, fazendo uso de *ArrayList's*. Possui também um método *Update* que com recurso a um *lock* permite fazer a atualização do jogo em segurança, e por fim, o método *Draw* desenha o sol, chama as funções de desenho para cada um dos planetas e astronautas e desenha, no canto superior esquerdo, a quantidade de *boost* que os jogadores ainda têm.

2.5 Connector

Aqui é onde nós fazemos a ligação ao servidor, que posteriormente vamos descrever, podendo assim ler as informações que este vai enviando para ser possível desenhar o nosso jogo.

2.6 Cliente

Por fim, esta classe trata de criar todas as funcionalidades do nosso menu, trabalha com o *Connector* de forma a interligar o servidor com o que está a acontecer no menu, como por exemplo, no momento do *login*, para sabermos se temos de referir ao utilizador que o login está errado ou que o login foi bem sucedido e este já está à procura de uma partida.

3 Servidor

3.1 Planet

Neste módulo, é feita a definição da criação de um planeta, assegurando que, por exemplo, quando maior este for, mais lento ele orbitará. Além disso, temos as definições necessárias para atualizar os planetas, fazendo-os orbitar.

3.2 Astronaut

Tal como no cliente, este módulo tambem se assemelha ao planeta, tendo definido funções de criação e atualização dos seus movimentos, só que neste caso, tivemos de fazer a gestão dos *inputs* que vão sendo recebidos.

3.3 Collision

Aqui, temos presente a função para calcular a distância entre dois pontos e cada caso de colisões que temos no nosso programa (Astronauta-Planeta, Astronauta-Sol e Astronauta-Borda), com a sua respetiva condição e uma função que coloca as colisões detetadas numa lista, para que a possamos usar posteriormente.

3.4 Conversores

Tal como o nome indica, neste módulo, temos todas as funções que usamos para nos auxiliar durante a criação do nosso programa, no que consta a converter tipos de dados.

3.5 Login_Manager

Neste módulo, temos presente as definições do que fazer quando um jogador cria ou remove uma conta e quando o mesmo ganha ou perde um jogo, nomeadamente, o algorítmo utilizado para que tenhamos os níveis e a serie de vitórias/derrotas com os valores certos.

3.6 Server

Fazendo uso das funções do módulo anterior, este gere todos os casos existentes que dizem respeito do *Login* com o autenticador e também contém a função *cicloJogo()* que continuamente vai esperando por mensagens que o cliente vai mandando.

3.7 Estado

Resumidamente, este método trata de manter um estado de jogo acertado, fazendo com que qualquer atualização seja refletida no jogo, referente a posições de jogadores, movimento de planetas e até colisões. Ainda sobre o último tópico, asseguramos que após um jogador ficar sozinho no jogo, o mesmo terá de sobreviver mais 5 segundos até que lhe seja atribuída a vitória. Neste mesmo método, temos, como por exemplo a criação de quatro salas de jogo, a lógica necessária para verificar qual sala é que o jogador vai ser atribuído e também a implementação da espera que têm de ser feita quando não há jogadores suficientes ou quando já temos o necessário para começar, mas a sala ainda não encheu.

4 Conclusão

Ao longo deste trabalho, fomos aplicando os conceitos que tinham sido lecionados durante o semestre. Apesar de termos chegado à maioria dos objetivo delineados, o grupo não pode deixar de falar sobre o que não correu tão bem. A colisão proposta entre astronautas não foi completa, apesar de termos conseguido notificar o servidor quando a mesma existia e também o facto de que, mesmo tendo os níveis implementados não temos algo que fizesse o *matchmaking* como era pedido.

Em suma, estamos satisfeitos com o resultado geral do trabalho, pois foram aplicados vários conhecimentos que demonstram o aprendizado das aulas. Contudo, uma má gestão de tempo impediu um desenvolvimento mais profundo do presente trabalho. Inicialmente, tivemos difculdades a compreender o enunciado proposto, o que tambem dificultou o grupo para um maior sucesso.

Em um possível futuro, onde retrabalharemos sobre este projeto, o grupo compromete-se a desenvolver os tópicos defasados, como por exemplo, a colisão entre astronautas e o matchmaking e o tratamento de erros e bugs que existem.