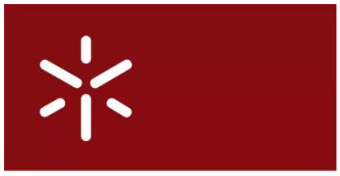

DUELO

Unidade Curricular de Programação Concorrente

Licenciatura em Ciências da Computação

Universidade do Minho

18 de maio 2025

Grupo 10, composto por:

José Loureiro Nº 96467

José Ferreira Nº 96798

Pedro Gonçalves Nº 101250

Hugo Ferreira Nº 100082

Índice

1 – Introdução

No âmbito da unidade curricular de Programação Concorrente, foi proposto pelo docente Paulo Sérgio Soares Almeida a implementação de um jogo em servidor denominado $DUELO.$ Esta proposta foi efetuada com o intuito de consolidar a aprendizagem do paradigma abordado 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*, intermediado 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 relatório será apresentada a solução do problema, descrevendo de forma sucinta as decisões tomadas na implementação da aplicação cliente e do servidor.

2 - Desenvolvimento e Desconstrução do Trabalho

2.1 – Autenticação do utilizador

Para registar um utilizador na Estrutura de dados do Servidor (ficheiro.txt), cada cliente após ter estabelecido a conexão TCP inicial tem acesso a 3 botões, um para se registar, um para efetuar l*ogin e um para eliminar a conta de um Utilizador. Ao interagir com o* no botão de registar, aparecem duas caixas de texto para colocar *username* e *password*, após clicar no botão *Continue* essas informações guardadas no ficheiro mencionado anteriormente.

2.2 – Progressão

Um Utilizador após a sua criação, é inicializado com o nível 1, ou seja, n = 1, com um sistema de progressão que consiste em aumentar de nível n para o nível n + 1 depois de ganhar n partidas consecutivas (antes de mudar). Desce de nível se perder [n/2] partidas consecutivas (antes de mudar), até chegar ao limite inferior respetivamente, n = 1.

2.3 – Lobby / Inicialização de Partidas

O processo do lobby é responsável por esta parte da solução criando uma *queue,* e aumentando-a sempre que algum utilizador pede ao servidor para jogar. Sempre que existe 2 jogadores que entram no lobby e possuem uma diferença máxima de apenas um nível, o lobby cria um processo para efetuar a partida entre os dois jogadores, disponibilizando o *pid* e o *username* de cada jogador. O lobby elimina então estes dois jogadores da sua *queue* e fica à ficara que mais utilizadores peçam para jogar.

2.4 Espaço

O espaço de jogo de uma partida é baseado num formato 2D, retangular de 1000x600, com um campo de jogo inicial vazio, delimitado por paredes nos quatro lados.

2.5 Avatares e Movimento dos jogadores

Os avatares (jogadores) são em forma de círculo, o movimento dos mesmos deverá ser feito através de quatro teclas (w, a, s, d), que controlam a aceleração. Os avatares têm inércia, ou seja, acelerar para um dos lados aplicará uma força que irá aumentar a velocidade do jogador progressivamente para esse lado, até atingir uma velocidade máxima pré-definida. Já acelerar para o lado oposto irá progressivamente desacelerar o movimento atual, invertendo eventualmente a direção.

2.6 Sistemas de Colisões

Para a realização do nosso jogo, assim como é pedido pelo docente, adicionamos a mecânica de colisões, que consiste na consequência do embate entre certos objetos presentes numa partida com regras distintas entre os mesmos, sendo elas, as seguintes:

Uma colisão entre um jogador e um projétil resulta na adição de 1 ponto na pontuação do outro jogador, bem como na eliminação do projétil.

Uma colisão entre um projétil e uma borda elimina o projétil.

Uma colisão entre um jogador e uma das bordas do mapa resulta na adição de 2 pontos na pontuação do outro jogador, e recoloca os dois jogadores nas suas posições iniciais.

Uma colisão entre um jogador e um modificador aplica o respetivo efeito nesse jogador e remove o modificador.

Uma colisão entre jogadores ou entre projéteis e modificadores é ignorada.

2.7 Projéteis

Cada jogador durante a realização de uma partida, tem acesso à mecânica de disparar um projétil num intervalo de tempo de 300 milissegundos, em formato de círculo e com uma velocidade de 20 milissegundos, enquanto que a direção é controlada pela posição do cursor em relação ao avatar do jogador.

2.8 Modificadores

Os modificadores são aparecendo aleatoriamente, ao longo do tempo, com quatro tipos e com cores diferentes, até um dado máximo (por tipo) de modificadores presentes. Possuem forma de círculo, com um tamanho entre o dos projéteis e o dos jogadores, estando sempre parados durante a realização de uma partida. Quando consumidos, isto é, quando um jogador passa e colide contra eles, desaparecem e aplicam as seguintes modificações ao respetivo jogador:

- Verdes ou Laranjas: Aumentam ou diminuem, respetivamente, a velocidade de saída de projéteis subsequentes. A velocidade de saída voltar progressivamente ao valor base.

– Azuis ou Vermelhos: Diminuem ou aumentam, respetivamente, o intervalo de tempo entre disparos. O intervalo de tempo volta progressivamente ao valor base.

2.9 Pontuação

O sistema de pontuação é bastante simples, consta na vitoria do jogador que, ao fim de 2 minutos, tiver a maior pontuação. Caso tiverem os mesmos pontos no final da duração do jogo, assume-se o empate, sendo a partida ignorada para efeitos de nível e de top 10, ou seja, como se não tivesse ocorrido.

2.10 Listagem do top 10 de jogadores

Ao aceder ao top 10, surge uma tabela com o nome dos utilizadores (*Username*), nível (*Level*) e sequencia (*Sequence*) em que, é ordenada primeiro por nível e depois pela melhor última série de vitórias/derrotas consecutivas entre os utilizadores.

3 – Cliente e Servidor

3.1 Estrutura do cliente

O cliente feito em *Processing* (java) é composto por um Button, *ConnectionManager*, Game, *InputBox*, *Player*, Item e Shot, estas classes (excluindo o *ConnectionManager*) são responsáveis por todo o funcionamento gráfico do Jogo, enquanto o *ConnectionManager* é responsável pela comunicação entre o cliente e o servidor.

Ao ser iniciado o Cliente, a conexão TCP havendo assim comunicação entre o cliente e o servidor consoante as opções escolhidas pelo utilizador. Quando é encontrada uma partida é criada uma *thread* que será responsável por estar constantemente a receber e interpretar as mensagens do servidor atualizando assim as posições dos nossos objetos para os jogadores, tiros, itens, assim como o tempo atual da partida e por fim termina ao receber a mensagem com o resultado da partida. Enquanto isso a *thread* principal do *processing* desenha todos os elementos do jogo e envia para o servidor as teclas pressionadas.

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

3.2 Estrutura do Servidor

O Servidor é inicializado com número da *port* para a ligação e começa por criar uma *listening socket*, caso seja criada com sucesso, são extraídas as contas dos utilizadores existentes no nosso ficheiro que as armazena, assim como iniciar um processo dedicado a correr uma função do lobby e outro processo dedicado à gestão, autenticação e atualização dos utilizadores.

Quando alguém estabelece conexão com o servidor, é criado um processo com a nossa *listening socket* de modo a receber a conexão de mais clientes e o processo atual chama a função *comunicator* responsável por comunicar com o cliente através da *socket*. É aqui que é feita a comunicação para o registo, login e eliminação de utilizadores, assim como o pedido para ver o Top 10 e entrar no *lobby*. Caso o *lobby* encontre uma partida o processo criado para a partida manda mensagem ao servidor a dizer que foi encontrada uma partida e a função *match_comunicator* é responsável por enviar ao processo do jogo toda a informação relacionada a botões pressionados, largados e tiros disparados. A função *match_comunicator* é também responsável por receber e extrair a informação enviada pelo processo da partida e de seguida enviar essa informação para o cliente de modo que este atualize a interface gráfica do mesmo.

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 comunicação cliente servidor, até à gestão interna de processos.

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.