NOVA ARENA

Programação Concorrente 22/23

Trabalho Realizado por:
Bárbara Faria A85774
José Luís Pires A84552
Tiago Lima A85126

1. INTRODUÇÃO

No âmbito da Unidade Curricular de Programação Concorrente, foi-nos proposta a criação de um pequeno jogo, ao qual apelidamos de “Nova Arena”, com o objetivo principal de pôr em prática os conhecimentos adquiridos ao longo do semestre.

O programa consiste num servidor que é capaz de comunicar com vários clientes concorrentemente e armazenar a informação necessária para a criação e decorrer de partidas. Estas comunicações ocorrem através da troca de mensagens utilizando *sockets TCP*.

2. CLIENTE

Desenvolvido em Java, o Cliente é uma entidade capaz de interagir com o Servidor, recebendo mensagens através do *socket* e traduzindo-as visualmente para os utilizadores. Cada jogador terá associado um Cliente.

Ao iniciar a aplicação, o cliente terá a opção de se inscrever, enviando os seus dados para o Servidor. Este ficará encarregado de validar essa informação, verificando se o nome do utilizador e a respetiva palavra-passe não existem nos registos. Em caso de sucesso, os dados do cliente serão armazenados, para em futuras utilizações não ser necessário novo registo.

Numa situação de *login*, o Servidor fará a mesma tarefa de validação de dados: caso o nome de utilizador e a respetiva palavra-passe existam nos registos, então o *login* é efetuado.

Após um *login*, o utilizador terá a opção para jogar, para fazer *logout*, regressando ao menu principal, ou ver a lista dos jogadores que se encontram *online*.

Ao selecionar a opção para jogar, o utilizador ficará à espera de um oponente. Encontrado esse oponente, a partida começará. O Cliente cria uma *thread* que lerá a informação enviada pelo Servidor para poder "desenhar" o jogo. Deste modo é possível não só enviar inputs como também receber nova informação, de modo assíncrono.

Os avatares dos jogadores são círculos com uma linha interior a indicar a direção destes (cor azul para o "meu" círculo, cor vermelha para círculo "inimigo"). Estes podem movimentar-se para a frente ou rodar para a esquerda/direita (teclas "W" e "A"/"D", respetivamente). A informação do movimento é enviada pelo Cliente para o Servidor através de um *socket TCP*. O Servidor calculará as novas posições que serão enviadas para o Cliente "redesenhar" o jogo.

Na janela do jogo (um quadrado 500px) serão também apresentados seis círculos "bónus", dois de cada cor (azul, verde e vermelho), que modificam os atributos dos jogadores, quando consumidos. Os azuis e verdes aumentam a velocidade linear e a velocidade angular, respetivamente. Os vermelhos removem os bónus, repondo os atributos ao seu estado inicial. Quando consumidos, estes reaparecem num ponto aleatório da janela do jogo.

O objetivo de cada jogador é “apunhalar” o oponente pelas “costas” o maior número de vezes possível dentro de dois minutos. Quando um jogador é atacado, ambos os avatares serão colocados nas suas posições iniciais (canto inferior esquerdo e canto superior direito). Não há colisão caso os jogadores embatam de frente. Ganha aquele que consegue “apunhalar” mais vezes. Caso haja um empate, o jogo prossegue até haver um ataque final. Um jogador é desqualificado se metade do seu avatar sair da área do jogo.

No fim de uma partida, um ecrã com as pontuações é apresentado, dando a opção a cada jogador de fazer *logout* ou de voltar ao menu do jogo.

Toda a informação previamente mencionada é guardada com a ajuda das classes *GameState* (estado da partida), *Player* (informação de um jogador) e *Bonus* (informação dos objetos “consumíveis”).

3. CONEXÃO

A conexão Cliente/Servidor é assegurada por duas variáveis - *toSocket* e *fromSocket* - e pela classe *Reader*. A variável *toSocket* envia ao Servidor toda a informação acerca dos *inputs* dos utilizadores (teclado e rato). Por sua vez, a variável *fromSocket* encarrega-se de ler a informação que o Servidor envia para o Cliente. A classe *Reader* recebe a informação de uma partida e atualiza o estado do jogo.

4. SERVIDOR

O servidor, desenvolvido em Erlang, tem como principal função gerir utilizadores e partidas. Ao ser iniciado são criados dois processos: *login_manager* e *match_manager*. O primeiro tratará da autenticação dos utilizadores (criar/remover contas, login, logout) recorrendo a funções que se encontram em *login_manager.erl*. Toda essa informação será guardada num mapa. O segundo gere os jogadores que se encontram *online* (menu do jogo) e que querem jogar, alocando-os em várias "salas" de jogo.

Sempre que um Cliente se liga ao Servidor, é criado um processo que ficará encarregado da comunicação entre os dois. Recorremos às funções que se encontram em *user_manager.erl*, para o efeito.

Quando um Cliente desejar jogar, o Servidor procura outro que deseje o mesmo. Mal encontre, é criado um processo que vai gerir a informação dessa partida. Com este método, o Servidor consegue gerir várias partidas concorrentemente. Este processo define o estado inicial da partida (posições dos avatares, valores das velocidades, posições dos objetos "consumíveis", …) e envia-o aos Clientes. Por sua vez, estes transmitem mensagens (*input*, …) e o processo calcula o novo estado do jogo, isto é, se um jogador colide com o oponente, se saiu dos limites da janela, se consumiu um bónus, etc.

No início de uma partida, é criado um processo que envia uma mensagem de fim de jogo, ao fim de dois minutos, ao processo responsável pelo duelo. Caso o jogo se encontre empatado, este continua até haver um ataque final. No final da partida, o Servidor recebe a informação da escolha dos utilizadores: se fazem *logout*, serão reencaminhados para o menu principal; se quiserem continuar *logged* in regressam ao menu do jogo).

Para efeitos de otimização, o Servidor apenas envia informação ao Cliente caso haja mudança no estado do jogo, por exemplo, um avatar ter-se movimentado.

5. CONCLUSÃO

Em geral, o jogo obedece à maior parte dos requisitos propostos: a ligação e comunicação Servidor-Cliente são estabelecidas com sucesso; os utilizadores podem registar-se, anular uma conta, fazer *login* e *logout*; os jogadores conseguem colidir; há um sistema de pontuação; existe um limite temporal da partida com possível prolongamento; e há um sistema de modificação de atributos.

No entanto, algumas condições não foram programadas: não existe um sistema de *matchmaking,* isto é, os jogadores não têm um nível associado e, consequentemente, não há progressão; a movimentação não é uma aceleração linear, é constante; as melhorias de atributos não desaparecem com o tempo; e não foi implementada uma listagem de vitórias.

Em relação ao reaparecimento de um jogador que é atingido por outro, foi tomada a decisão de ambos os jogadores voltarem para a posição de origem.