Programação Concorrente
Trabalho Prático
Nova Arena

Grupo de Sistemas Distribuídos
Universidade do Minho

26 de abril de 2023

Informações gerais

- Cada grupo deve ser constituído por quatro elementos.
- O trabalho deve ser entregue até 27 de maio de 2023 às 23h59;
- Deve ser entregue o código fonte e ainda um relatório até 6 páginas no formato pdf.
- A apresentação do trabalho será dia 29 de maio de 2023.

Resumo

Implemente um mini-jogo onde vários utilizadores podem interagir usando uma aplicação cliente com interface gráfica, escrita em Java, 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 efectuada pelo servidor.

Funcionalidade

Este jogo deverá suportar as seguintes funcionalidades:

- Registo de utilizador: dado *username* e *password*; deverá ainda ser possível um utilizador cancelar o registo. Sempre que um jogador quer entrar no jogo deverá ser autenticado pelo servidor. Um jogador começa no *nível* 1 quando se regista. O registo poderá ser feito por um cliente específico (independente do cliente gráfico principal) que guarde em ficheiro a informação do registo.
- Progressão: um jogador passa do nível n para o nível $n + 1$ depois de ganhar $2 \times n$ partidas.
- Partidas: cada partida é constituida por dois jogadores do mesmo nível. Um jogador pode ter que esperar até tal ser possível. O servidor deve permitir que em cada momento possam estar a decorrer várias partidas em simultâneo. Cada partida demora dois minutos (com possível extensão).
- Espaço: é 2D, quadrado (assuma uma dimensão normalizada, e.g., 1mx1m), vazio, sem paredes. Todo o espaço de jogo deve ser visualizado de mesmo modo por todos os jogadores. Relativamente às cores dos jogadores, existe a cor do "meu" avatar e a cor do "outro".

- Avatares: todos os avatares (jogadores e objetos) são em forma de círculo. Os jogadores têm uma direção para onde estão voltados (um ângulo, e.g., 1.23 rad), sinalizada graficamente.
- Movimentação: Deverá ser feita através de 3 teclas: rodar para a esquerda ou direita, e aceleração na direção para onde estão voltados. Várias teclas podem estar premidas ao mesmo tempo, e.g., direita e aceleração. Os jogadores possuem 2 parâmetros que em cada momento definem o movimento aplicado enquanto estão a ser premidas as teclas de movimentação: velocidade angular (e.g., 2.13 rad/s) e aceleração linear (e.g., 0.23 m/s^2).
- Objetos: deverão ir aparecendo aleatoriamente, ao longo do tempo, três tipos de objetos (verdes, azuis e vermelhas), que estão sempre parados. Os dois primeiros quando consumidos (via colisão), dão bónus ao parâmetro da velocidade angular e aceleração linear, respectivamente. O bónus é dado até ser atingido um dado limite (e.g., 5 vezes o valor base; sugestão: use ganhos decrescentes), e deverá ir desaparecendo com o tempo, fazendo o parâmetro convergir para o valor base. O consumo de um objeto vermelho remove os bónus, repondo os parâmetros no valor base.
- Pontuação: um jogador ganha um ponto quando colide com outro por trás (direção de incidência fazer um ângulo menor do que $\pi/2$ com a direção da vítima). O jogador atingido reaparece num local aleatório, e voltam ambos aos valores base dos parâmetros de movimento. Outras colisões entere jogadores são ignoradas, continuando estes o seu movimento.
- Vitória: ganha a partida o jogador com mais pontos ao fim do tempo de jogo. Em caso de empate, a partida prolonga-se até ser obtido o próximo ponto. Caso um jogador saia do espaço de jogo perde imediatamente a partida.
- Listagem de vitórias: deverá ser mostrado o top dos jogadores com mais partidas ganhas, desde que o servidor foi arrancado.

Cliente

Deverá ser disponibilizado um cliente com interface gráfica que permita suportar a funcionalidade descrita acima. Este cliente deverá ser escrito em Java e comunicar com o servidor via sockets TCP. Sugestão: utilize Processing (`http://processing.org`) para a interface gráfica.

Deve ser utilizado um protocolo orientado à linha de texto, com uma conexão por cliente. No cliente, para facilitar o uso de várias threads a aceder ao socket, deve ser implementada uma classe:

```
public class ConnectionManager implements AutoCloseable {
    public static ConnectionManager start(Socket socket) throws IOException { ... }
    public void send(String type, String message) throws IOException { ... }
    public String receive(String type) throws IOException { ... }
    public void close() throws IOException { ... }
}
```

que deve gerir o controlo de concorrência no uso do socket por várias threds. Deve permitir: o envio e receção concorrente; o envio de mensagens etiquetadas por um *type* (uma linha de texto resultante da concatenação `type + ":" + message`); uma thread bloquear até chegar uma mensagem de determinado *type*, devolvendo-a.

Servidor

O servidor deverá ser escrito em Erlang, mantendo em memória a informação relevante para fazer a simulação do cenário descrito, receber conexões e input dos clientes bem como fazer chegar a estes a informação relevante para a atualização da interface gráfica, de acordo com a simulação que efetua.