Programação Concorrente
Trabalho Prático
Battle Royale

Grupo de Sistemas Distribuídos
Universidade do Minho

3 de Maio de 2022

Informações gerais

- Cada grupo deve ser constituído por até, e de preferência, quatro elementos.
- O trabalho deve ser entregue até 12 de Junho de 2022 às 23h59;
- Deve ser entregue o código fonte e ainda um relatório até 6 páginas no formato pdf.
- A apresentação do trabalho será em data a confirmar, possivelmente no dia 21 ou 22 de Junho de 2022.

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. O registo poderá ser feito por um cliente específico (independente do cliente gráfico principal) que guarde em ficheiro a informação do registo.
- Partidas: cada partida é constituida por um mínimo de três jogadores e um máximo de 8. Quando um jogador pede para jogar tem que aguardar até tal ser possível. O servidor deve permitir que em cada momento possam estar a decorrer até 4 partidas em simultâneo. Cada partida decorre até restar só um jogador sobrevivente.
- Espaço: é 2D, rectangular, vazio, sem limites nos quatro lados.

- Avatares: todos os avatares (jogadores e cristais) são em forma de círculo. No caso dos jogadores, o tamanho varia entre um mínimo e um máximo pré-definido, com área correspondente à massa do avatar. Cada jogador vê o seu avatar destacado (por exemplo com uma circunferência de cor diferente do interior do círculo).
- Movimentação dos avatares: o avatar do jogador movimenta-se tendencialmente na direcção do cursor, a uma velocidade normalmente constante (uma velocidade base pré-defimida). Para tal, o avatar sofre de uma aceleração na direcção do cursor, enquanto a velocidade, segundo essa direcção, for menor do que a base. Um jogador pode activar um *boost* de velocidade, premindo o botão do rato. Nesse caso a velocidade sofre uma variação instantânea, na direcção do cursor, e vai tendendo com o tempo para o valor base (sempre que for maior do que este, sofre uma aceleração em sentido contrário). Activar um *boost* causa ligeira perda de massa (reflectida na área do círculo), e só pode ser feito se o avatar não estiver com a massa mínima pré-definida. A velocidade máxima depende do tamanho, sendo tanto menor quanto maior o avatar.
- Cristais: deverão ir aparecendo de vez em quando, ao longo do tempo, aleatoriamente no mapa, cristais, pequenos, até um limite máximo em simultâneo. Os cristais permanecem fixos. Quando um jogador captura (colide com) um cristal, ganha massa e fica com a cor do cristal, fazendo o cristal desaparecer.
- Regras das cores: jogadores e cristais aparecem aleatoriamente com uma de três cores possíveis: vermelho, verde e azul. Vermelho ganha a verde, verde ganha a azul e azul ganha a vermelho.
- Colisões entre jogadores: Uma colisão entre jogadores deve ser perfeitamente elástica. Quando ocorre, quem ganhar pela regra das cores, ou for maior caso tenham a mesma cor, ganha alguma da massa, que é subtraida do perdedor.
- Um jogador perde (saindo do jogo) quando, estando a sua massa no mínimo, é o perdedor numa colisão. Quando só resta um jogador, esse é o vencedor da partida.
- Listagem de victórias: deverá ser mostrado, enquanto um jogador não está numa partida, 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, nomeadamente visualizar o avatar de cada jogador e os cristais. 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.

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 actualização da interface gráfica, de acordo com a simulação que efectua.