# Programação Concorrente
# Trabalho Prático
# Gravidade

Grupo de Sistemas Distribuídos
Universidade do Minho

4 de abril de 2024

## Informações gerais

- Cada grupo deve ser constituído por até (e de preferência) quatro elementos.
- O trabalho deve ser entregue até 25 de maio de 2024 às 23h59;
- Deve ser entregue o código fonte e ainda um relatório até 6 páginas no formato pdf.
- A apresentação do trabalho será entre 27 e 29 de maio de 2024.

## 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 $n$ partidas consecutivas (antes de mudar). Desce de nível se perder $\lceil n/2 \rceil$ partidas consecutivas (antes de mudar).
- Partidas: cada partida é constituida por de 2 a 4 jogadores, com diferença máxima de um nível entre eles; quando um jogador pede para jogar tem que aguardar até tal ser possível. Caso ainda não estejam 4 jogadores para uma partida, o servidor deve esperar mais 5 segundos, desde o momento em que chegou o último dos jogadores, para ver se chega mais algum. O servidor deve permitir que em cada momento possam estar a decorrer várias partidas em simultâneo.
- Espaço: o espaço é 2D, rectangular, vazio, sem limites nos quatro lados. No centro do espaço (e écran) existe um sol, fixo.

- Planetas: existem alguns (poucos) planetas, de diferentes tamanhos, dispostos aleatoriamente no espaço, com velocidades iniciais aleatórias. Escolha, gamas dos valores aleatórios de modo a que os planetas normalmente estejam visíveis a maior parte do tempo.
- Avatares: os avatares (jogadores, planetas, sol) são em forma de círculo. Os jogadores têm uma direcção para onde estão voltados (um ângulo), sinalizada graficamente.
- Gravidade: cada jogador e criaturas deslocam-se pelas regras da mecânica clássica, considerando a massa concentrada no seu centro. Simplifique, considerando todas as outras massas insignificantes comparadas com o sol (calcule apenas a atração pelo sol).
- Movimentação dos jogadores: deverá ser feita através de 3 teclas: (esquerda, direita, frente) que controlam 3 propulsores, provocando aceleração enquanto estão a ser premidas; as duas primeiras provocando aceleração angular, na direção respectiva, e a terceira aceleração linear na direção para onde está voltado o jogador. Cada jogador começa com a mesma quantidade de combustível; quando esgotado, os propulsores deixam de funcionar.
- Colisões: uma colisão entre um jogador e um planeta ou o sol faz com que o jogador perca, saindo de jogo. Colisões entre planetas ou entre estes e o sol são ignoradas. Uma colisão entre jogadores deve ser perfeitamente elástica, sem atrito (preservando as velocidades angulares).
- Pontuação: ganha o jogador que, quando todos os outros tiverem perdido, sobreviver durante mais 5 segundos. Caso contrário considera-se que não houve vencedor, sendo o(s) outros jogador(es) perdedor(es). Um jogador também perde se sair da área visível no écran.
- Listagem do *top* 10 de jogadores: ordenada primeiro por nível e depois pela última série de vitórias/derrotas consecutivas.

## 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.

## 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.