FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D COIMBRA

departamento de engenharia informática

Introdução à Programação e Resolução de Problemas

2023/2024 - 1º Semestre

Mini-Projeto:
Liga dos Últimos

Nota: A fraude denota uma grave falta de ética e constitui um comportamento inadmissível num estudante do ensino superior e futuro profissional licenciado. Qualquer tentativa de fraude levará à anulação da componente prática tanto do facilitador como do prevaricador, independentemente de acções disciplinares adicionais a que haja lugar nos termos da legislação em vigor. Caso haja recurso a material não original, as **fontes** devem estar explicitamente indicadas.

1 Introdução

Neste trabalho iremos programar um pequeno simulador de um jogo de futebol para dois jogadores. Cada jogador controla uma agente virtual cujo o objetivo é marcar golos, colocando a bola dentro da baliza adversário, evitando ao mesmo tempo que a bola entre na sua baliza. Os agentes podem movimentar-se em 4 direções (Cima, Baixo, Esquerda, Direita). Sempre que a bola toca nos limites do campo ela inverte a direção onde atingiu o limite. Quando é marcado um golo, o jogo reinicia com a bola ao centro, sendo a direção inicial aleatória. A Fig. 1 exemplifica o ambiente do jogo.

Além disso, e como estamos em Portugal, o VAR desempenha um papel preponderante para garantir a verdade desportiva. Desta forma, irá também programar as funcionalidades de um módulo que permitirá analisar todo os movimentos do jogo em modo offline.

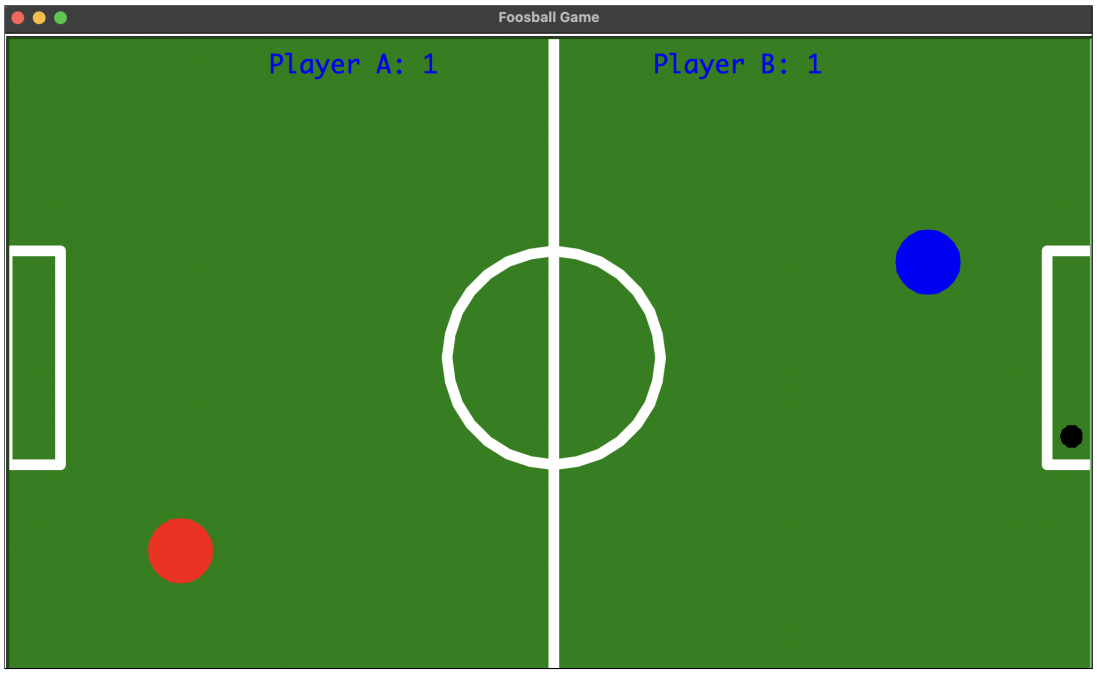

Figura 1: Exemplo do jogo que pretendemos desenvolver. Existem 2 jogadores, o vermelho e o azul, e uma bola de cor preta.

2 Enunciado

Este trabalho prático tem como objetivo principal a programação das funcionalidades que permitam jogar o jogo. Pretende-se que os estudantes consoli-

dem as competências adquirida relacionadas com a resolução de problemas e programação em Python. Assim, os alunos terão de completar e desenvolver as funcionalidades básicas do jogo, nomeadamente:

1. Inicializar os agentes;
2. Movimentar os agentes para Cima, Baixo, Esquerda, Direita usando o teclado;
3. Desenhar as linhas do campo;
4. Inicializar a bola;
5. Movimentar da bola;
6. Detetar colisões entre a bola e os limites do ambiente;
7. Detetar colisões entre a bola e os agentes;
8. Detetar se a bola entrou na baliza;
9. Programar um quadro de pontuação - Sempre que a bola entrar na baliza o jogador ganha 1 ponto. Deverá ainda criar um ficheiro que guarde o resultado histórico de todos os jogos que foram realizados. Este ficheiro deverá ser atualizado sempre que o jogo termine, i.e., o jogador pressione a tecla `ESC`;
10. Programar um mecanismo de repetição de jogadas que deram origem a um golo - Deverá ser guardada toda a informação sobre o movimento dos agentes, da bola, bem como os momentos em que ocorrem os golos.

Para a programação do jogo, é fornecido um ficheiro com o nome `fosball.py`, que os alunos deverão descarregar da UCTeacher. Este ficheiro já contém o código necessário para inicializar o ambiente de jogo, nomeadamente no que toca a criação da janela, a criação do ambiente de jogo, a criação do quadro de pontuação e deteção do carregar nas teclas `W,A,S,D` e $\leftarrow,\rightarrow,\uparrow,\downarrow$. A Fig. 2 mostra o ambiente inicial.

As restantes funcionalidades deverão ser programadas pelos alunos, de forma a permitir ter um jogo completamente funcional. O ficheiro contém algumas funções definidas cujo código deverá ser completado. No entanto, isto não significa que sejam apenas necessárias as funções fornecidas. Assim, deverá criar todas as funções adicionais que ache necessárias para resolver o problema.

Relativamente às funções que estão definidas e que deverá completar, elas são as seguintes:

- `jogador_[direção](estado_jogo, jogador)` - Funções responsáveis pelo movimento dos jogadores no ambiente. O número de unidades que o jogador se pode movimentar é definida pela constante `PIXEIS_MOVIMENTO`. As funções recebem um dicionário que contém o estado do jogo e o jogador que se está a movimentar.

- `desenha_linhas_campo()` - Função responsável por desenhar as linhas do campo, nomeadamente a linha de meio campo, o círculo central, e as balizas.

- `criar_bola()` - Função responsável pela criação da bola. Deverá considerar que esta tem uma forma redonda, é de cor preta, começa na posição `BOLA_START_POS` com uma direção aleatória. Deverá ter em conta que a velocidade da bola deverá ser superior à dos jogadores. A função deverá devolver um dicionário contendo 4 elementos: o objeto bola, a sua direção no eixo dos xx, a sua direção no eixo dos yy, e um elemento inicialmente a `None` que corresponde à posição anterior da mesma.

- `movimenta_bola(estado_jogo)` - Função responsável pelo movimento da bola que deverá ser feito tendo em conta a posição atual da bola e a direção em xx e yy.

- `cria_jogador(x_pos_inicial, y_pos_inicial, cor)` - Função responsável por criar e devolver o objeto que corresponde a um jogador. A função recebe 3 argumentos que correspondem às coordenadas da posição inicial em xx e yy, e a cor do jogador. A forma dos jogadores deverá ser um círculo, cujo seu tamanho deverá ser definido através da função `shapesize` do módulo `turtle`, usando os seguintes parâmetros: `stretch_wid=DEFAULT_TURTLE_SCALE,` `stretch_len=DEFAULT_TURTLE_SCALE`.

- `verifica_colisoes_ambiente(estado_jogo)` - Função responsável por verificar se há colisões com os limites do ambiente, atualizando a direção da bola. Não se esqueça de considerar que nas laterais, fora da zona das balizas, a bola deverá inverter a direção onde atingiu o limite.

- `verifica_golo_jogador_[cor](estado_jogo)` - Função responsável por verificar se um determinado jogador marcou golo. Para fazer esta verificação poderá fazer uso das constantes: `LADO_MAIOR_AREA` e `START_POS_BALIZAS`. Note que sempre que há um golo, deverá atualizar a pontuação do jogador, criar um ficheiro que permita fazer a análise da jogada pelo VAR, e reiniciar o jogo com a bola ao centro. O ficheiro para o VAR deverá

conter todas as informações necessárias para repetir a jogada, usando as informações disponíveis no objeto `estado_jogo['var']`. O ficheiro deverá ter o nome "replay_golo_jv_[TotalGolosJogadorVermelho]_ja_[TotalGolosJogadorAzul].txt" onde `[TotalGolosJogadorVermelho]`, `[TotalGolosJogadorAzul]` deverão ser substituídos pelo número de golos marcados pelo jogador vermelho e azul, respectivamente. Este ficheiro deverá conter 3 linhas, estruturadas da seguinte forma:

  - Linha 1 - coordenadas da bola;
  - Linha 2 - coordenadas do jogador vermelho;
  - Linha 3 - coordenadas do jogador azul;

  Em cada linha, os valores de xx e yy das coordenadas são separados por uma ",", e cada coordenada é separada por um ";". Um exemplo de como o ficheiro deverá ser organizado pode ser consultado na Listagem 1.

- `verifica_toque_jogador_[cor](estado_jogo)` - Função responsável por verificar se o jogador tocou na bola. Sempre que um jogador toca na bola, deverá mudar a direção desta.

- `terminar_jogo(estado_jogo)` - Função responsável por terminar o jogo. Nesta função, deverá atualizar o ficheiro "historico_resultados.csv" com o número total de jogos até ao momento, e o resultado final do jogo. Caso o ficheiro não exista, ele deverá ser criado com o seguinte cabeçalho: `NJogo,JogadorVermelho,JogadorAzul`.

```
1 5.000,5.000;1.000,1.000;-3.000,-3.000
2 -350.000,0.000;-350.000,0.000;-350.000,0.000
3 350.000,0.000;350.000,0.000;350.000,0.000
```

Listing 1: Exemplo do ficheiro de replay de jogos.

Existe ainda um objeto com o nome `estado_jogo` que contem todas as informações sobre o estado atual do jogo.

1. `estado_jogo['bola']` - Contém o objeto turtle que corresponde à bola;
2. `estado_jogo['jogador_vermelho']` - Contém o objeto turtle que corresponde ao jogador vermelho;
3. `estado_jogo['jogador_azul']` - Contém o objeto turtle que corresponde ao jogador azul;

4. `estado_jogo['var']` - Contém um objeto onde serão guardadas as posições da bola e dos jogadores;

5. `estado_jogo['pontuacao_jogador_vermelho']` - Golos marcados pelo jogador vermelho;

6. `estado_jogo['pontuacao_jogador_azul']` - Golos marcados pelo jogador azul

Para a programação do VAR, é fornecido um ficheiro com o nome `var.py`. Este ficheiro já contém o código necessário para inicializar o ambiente de jogo, utilizando o código desenvolvido no ficheiro `fosball.py`. Deverá completar o código disponibilizado de forma a que seja possível visualizar a repetição de um jogo. Assim, e depois de ter programado as funcionalidades do jogo, deverá completar, a seguinte função:

- le_replay(nome_ficheiro) - Função que recebe o nome de um ficheiro contendo um replay, e que deverá retornar um dicionário com as seguintes chaves:
  - `bola` - lista contendo tuplos com as coordenadas xx e yy da bola
  - `jogador_vermelho` - lista contendo tuplos com as coordenadas xx e yy da do jogador_vermelho
  - `jogador_azul` - lista contendo tuplos com as coordenadas xx e yy da do jogador_azul

Finalmente, e após terminar a programação de todas as funcionalidades pedidas, poderá ainda acrescentar novas funcionalidades que serão consideradas como bónus, permitindo compensar algumas perdas de pontuação nas funcionalidades obrigatórias.

3 Datas e Modo de Entrega

O trabalho deverá ser realizado parcialmente durante as aulas e por grupos com uma dimensão máxima de 2 alunos. Irá existir uma defesa obrigatória, em que todos os elementos do grupo têm de estar presentes.

Modo de Entrega:

O trabalho deverá ser entregue eletronicamente através do Inforestudante.

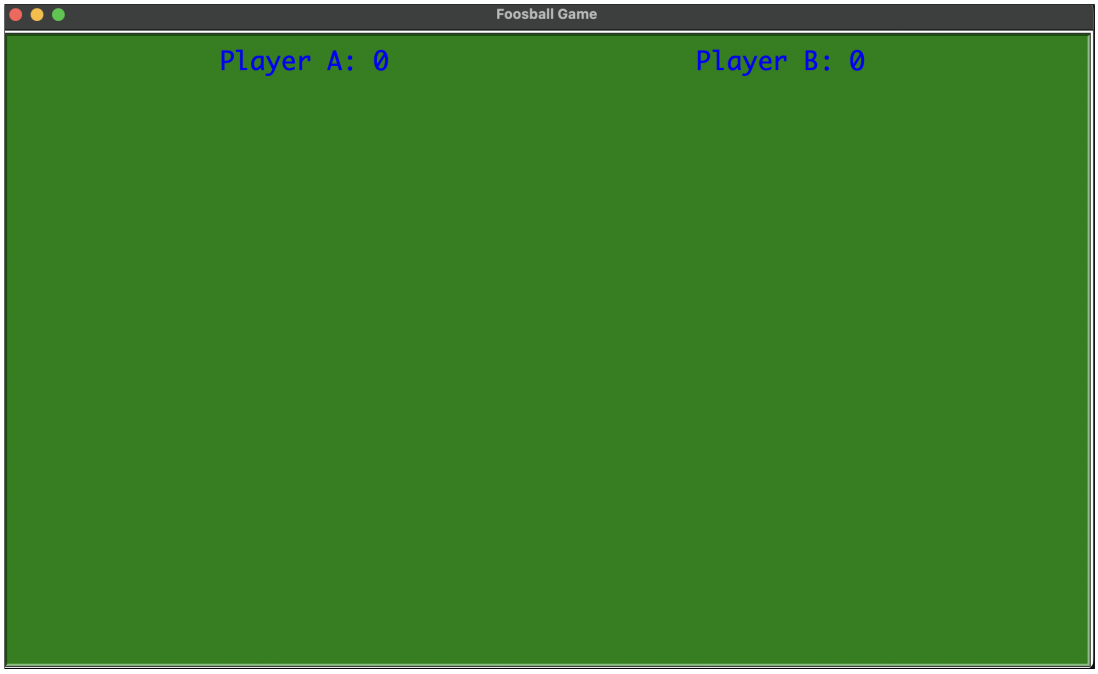

Figura 2: Ambiente Inicial

Data Limite: **23h59 do dia 01 de Dezembro de 2023**