ISEL – LEIM

Matemática Discreta e Programação

Projeto Final

Interface do motor do jogo *4 em linha*

(6 de outubro de 2021 17:41:31)

1 Objetivo

Descrever o interface do motor do jogo *4 em linha*. Descrever o interface de um agente autónomo que joga o jogo *4 em linha*.

2 Introdução

Pretende-se desenvolver o jogo *4 em linha* separando o desenvolvimento do aspeto visual do jogo, do desenvolvimento da lógica do jogo. Neste contexto, chama-se ao aspeto visual do jogo, a parte gráfica, e à lógica do jogo, o motor do jogo.

Tipicamente a parte gráfica é dependente do dispositivo onde o programa vai ser executado (por exemplo, computador (jogar com o mouse) ou telemóvel (jogar com os dedos)). É a parte gráfica que faz a ligação com o utilizador. Por outro lado a lógica do jogo é independente do dispositivo, é sempre a mesma em qualquer dispositivo.

A versão em modo de texto do jogo *4 em linha*, disponível no moodle, usa um motor de jogo que disponibiliza o interface descrito neste documento. O interface é constituído por funções. Toda a gestão da lógica do jogo é feita pelo motor do jogo. O objetivo do projeto final é desenvolver o motor do jogo *4 em linha*, disponibilizando o interface descrito neste documento. No final, deverá ser possível substituir o motor do jogo usado pela versão em modo de texto, disponível no moodle, pelo motor desenvolvido pelo aluno.

3 Aspetos gerais

3.1 Números de linhas e colunas

Nesta especificação os números de linhas e de colunas iniciam-se em 1. A contagem das linhas faz-se de cima para baixo. A contagem das colunas faz-se da esquerda para a direita.

3.2 Operação de jogar

Na operação de jogar não se indica qual é o jogador que está a jogar porque os jogadores jogam alternadamente. Por isso não é necessário indicar qual é o jogador que está a jogar. Se o jogo está vazio é porque é o primeiro jogador a jogar. Se o primeiro jogador jogou então é porque é o segundo a jogar. E assim sucessivamente.

3.3 Valores armazenados na grelha de jogo

Há sempre um primeiro jogador a jogar e um segundo jogador a jogar. Assim optou-se por armazenar na grelha do jogo o número inteiro 1 para representar uma peça do primeiro jogador a jogar, o número inteiro 2 para representar uma peça do segundo jogador a jogar, e o número inteiro 0 para representar uma posição vazia.

3.4 Estrutura de dados usada para armazenar um jogo

À partida, a única estrutura de dados que é necessário armazenar para gerir a lógica do *4 em linha*, é a grelha do jogo. A partir da grelha do jogo consegue-se saber: qual é o próximo jogador a jogar [1], se o jogo chegou ao fim, se alguém ganhou, qual é a linha que originou uma vitória, etc.

No entanto, o armazenamento apenas da grelha do jogo conduziria à necessidade de repetir certas operações em certas situações. Por exemplo. Depois de cada jogada averiguar-se se o jogo terminou. Só no final do jogo se averigua se alguém ganho ou se o jogo ficou empatado. Ora se só se armazenasse a grelha de jogo, para se testar se o jogo terminou ter-se-ia que percorrer a grelha. Depois, para mostrar a informação de vitória/derrota/empate, ter-se-ia que percorrer a grelha novamente. Estas duas informações podem ser obtidas num único percurso pela grelha do jogo. Neste único percurso

[1] Ou o jogo está vazio e é o primeiro jogador a jogar, ou já tem peças. Se tem peças, ou há tantas de um jogador como do outro e é o primeiro jogador a jogar, ou há mais uma peça do primeiro jogador a jogar e é o segundo jogador a jogar.

armazena-se, desde logo, todas as informações relevantes, como o facto do jogo poder ter terminado e alguém poder ter ganho e, nesse caso, as coordenadas da linha vitoriosa. Por esta razão o motor de jogo usado pela versão em modo de texto, disponível no moodle, usa a estrutura de dados seguinte para armazenar um jogo *4 em linha*:

```
jogo = (grelha, fim, vencedor, jogador, linha_vitoria)
```

Nesta estrutura de dados:

`grelha`: É uma lista com 6 linhas. Cada linha é uma lista com 7 elementos (um elemento para cada coluna). Cada elemento é o número inteiro 0, 1 ou 2.

`fim`: É `True` se o jogo terminou (porque um dos jogadores ganhou ou porque os jogadores empataram) e `False` se o jogo não terminou.

`vencedor`: É `None`, o número inteiro `1` ou o número inteiro `2`.

- É `None` se nenhum jogador ganhou: se o jogo terminou é porque terminou empatado. Nenhum jogador fez *4 em linha* e já não há posições vazias.
- É o número inteiro `1` se o primeiro jogador a jogar ganhou (fez *4 em linha*).
- É o número inteiro `2` se o segundo jogador a jogar ganhou (fez *4 em linha*).

`jogador`: É o número inteiro 1 se o próximo jogador a jogar é o primeiro jogador a jogar e é o número inteiro 2 se o próximo jogador a jogar é o segundo jogador a jogar.

`linha_vitoria`: Se o jogou terminou e algum jogador ganhou, é uma lista com 4 outras listas. Cada uma das 4 listas contém o número da linha e o número da coluna de cada uma das posições do um *4 em linha* vitorioso. Se nenhum jogador ganhou é `None`.

Note-se que a definição da estrutura de dados usada para armazenar o jogo é um detalhe de implementação do motor do jogo, que a parte gráfica não tem que conhecer, e como tal pode ser definida livremente pelo programador do motor da forma que achar mais conveniente. A estrutura apresentada aqui constitui apenas uma sugestão.

4 Especificação do interface do jogo *4 em linha*

Função `novo_jogo`

Argumentos Não tem argumentos.

Retorno A estrutura de dados usada para armazenar um jogo, inicializada com um jogo vazio.

Descrição Permite criar um jogo vazio, pronto para se começar a jogar.

Função `ha_espaco`

Argumentos

1. `jogo`: A estrutura de dados que armazena o jogo.
2. `coluna`: O número da coluna onde se pretende saber se há espaço para jogar.

Retorno `True` se há espaço para jogar na coluna indicada e `False` caso contrário.

Descrição Permite saber se há espaço numa coluna. Serve para não deixar os jogadores jogarem em colunas cheias (no jogo físico real também não se pode jogar numa coluna cheia).

Função `jogar`

Argumentos

1. `jogo`: A estrutura de dados que armazena o jogo.
2. `coluna`: O número da coluna onde se pretende jogar.

Retorno A estrutura de dados que armazena o jogo atualizada com a jogada efetuada.

Descrição Permite jogar numa coluna. Assume-se que a jogada é sempre efetuado numa coluna que tem espaço (posições vazias). MUITO IMPORTANTE: é da responsabilidade do utilizador do motor, ou seja, da parte gráfica, garantir que as jogadas são sempre efetuadas em colunas com espaço (com posições vazias).

Função `valor`

Argumentos

1. `jogo`: A estrutura de dados que armazena o jogo.
2. `linha`: O número da linha da posição da grelha, da qual se pretende obter o valor.
3. `coluna`: O número da coluna da posição da grelha, da qual se pretende obter o valor.

Retorno O número inteiro 0, 1 ou 2.

Descrição Permite obter o valor que está numa dada posição do tabuleiro do jogo, para que se possa representar graficamente esse valor (por exemplo, com um espaço vazio, com uma peça vermelha ou uma peça amarela).

Função `terminou`

Argumentos

1. `jogo`: A estrutura de dados que armazena o jogo.

Retorno O elemento no índice `1` do argumento `jogo`.

Descrição Permite saber se o jogo já terminou para, por exemplo, poder dar uma mensagem ao utilizador ou mudar para um ecrã de fim do jogo.

Função `quem_ganhou`

Argumentos

1. `jogo`: A estrutura de dados que armazena o jogo.

Retorno O elemento no índice `2` do argumento `jogo`.

Descrição Permite saber se o jogo terminou empatado ou, caso algum jogador tenha ganho, qual foi. Esta função deve ser usada depois de se saber que o jogo terminou.

Função `get_linha_vitoria`

Argumentos

1. `jogo`: A estrutura de dados que armazena o jogo.

Retorno O elemento no índice `4` do argumento `jogo`.

Descrição Permite obter as posições onde estão as peças de uma linha de quatro peças seguidas que originou a vitória do último jogador que jogou.

Função `proximo_a_jogar`

Argumentos

1. `jogo`: A estrutura de dados que armazena o jogo.

Retorno O elemento no índice `3` do argumento `jogo`.

Descrição Permite saber-se qual é o próximo jogador a jogar.

5 Especificação do agente do jogo *4 em linha*

Função `jogada_agente`

Argumentos

1. `jogo`: A estrutura de dados que armazena o jogo.
2. `jogador`: O jogador pelo qual o agente vai jogar.

Retorno O número da coluna onde o agente pretende jogar.

Descrição Permite que um agente eletrónico (um programa Python) possa jogar. O agente pode analisar o estado do jogo, sabe qual é o jogador pelo qual está a jogar, e jogar de acordo com uma estratégia definida. A estratégia mais simples poderá ser jogar à sorte (numa das colunas com espaço). Note-se que esta função só deve ser usada quando o jogo ainda não terminou.