1 2 9 0 FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE D COIMBRA

dei departamento de engenharia informática

Introdução à Programação e Resolução de Problemas

2024/2025 - 1º Semestre

Projeto

Pacman

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 ações 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.

Nuno Lourenço, Charles Gonçalves, João Simões, João Correia, João Macedo, Noé Godinho, Telmo Neves - 2024/2025

1 - Introdução

O Pacman (Figura 1) é um jogo lançado em 1980 inicialmente sob o nome "Puck Man". Criado pelo designer Toru Iwatani, o jogo surgiu com um propósito bastante inovador para a época: tornar os videojogos acessíveis a um público mais abrangente.

Inspirado pela visão de uma pizza com uma fatia em falta, Iwatani desenhou Pacman como uma personagem redonda com uma boca aberta, simbolizando a ideia de “comer”. O próprio nome do jogo, na sua versão original, faz alusão ao som "paku-paku" na língua japonesa, que imita o ato de mastigar. No entanto, para o lançamento nos Estados Unidos, o título foi alterado para "Pacman" para evitar interpretações erradas.

O objetivo do jogo é simples, mas desafiante: o jogador controla o Pacman, que se move num labirinto, comendo todos os pontos pequenos, ou "bolinhas", enquanto é perseguido por quatro fantasmas. Cada um dos fantasmas – Blinky (vermelho), Pinky (cor-de-rosa), Inky (azul) e Clyde (laranja) – possui uma personalidade e comportamento distintos, o que torna a sua perseguição mais imprevisível e o jogo mais emocionante. Blinky, por exemplo, é mais agressivo e tende a perseguir Pacman diretamente, Clyde alterna entre perseguição e fuga, e Pinky e Inky tentam antecipar os movimentos do jogador .

O sucesso do Pacman foi imediato e duradouro, tornando-se um fenómeno global que influenciou o mundo dos videojogos e a cultura pop. A simplicidade do seu conceito, combinada com a complexidade estratégica de escapar e capturar os fantasmas, fez de Pacman um dos jogos mais célebres e jogados de todos os tempos.

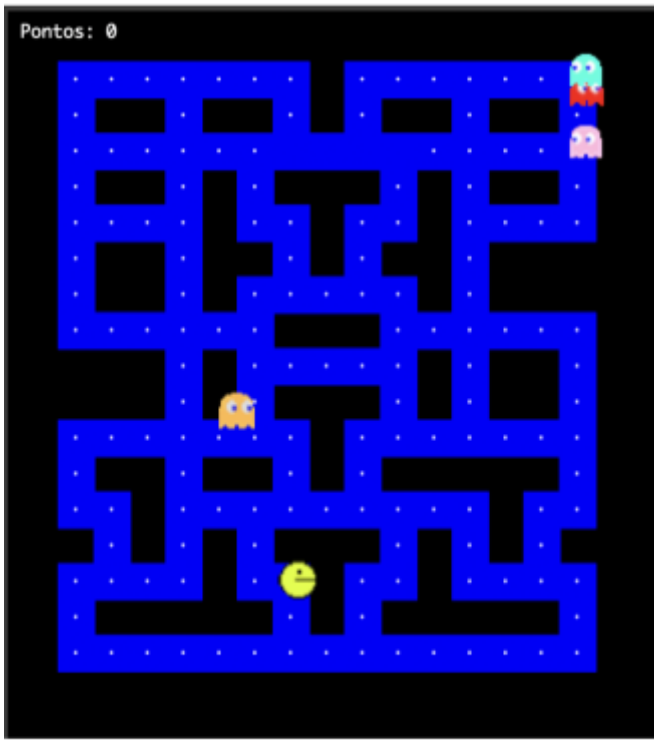

Figura 1 - Exemplo do Jogo do Pacman

2 - Enunciado e Objetivos

Este trabalho prático tem como objetivo principal a programação de algumas funcionalidades que permitam jogar o jogo. Pretende-se que os estudantes consolidem as competências adquiridas durante o semestre relacionadas com a resolução de problemas e programação em Python. Assim, os alunos terão de completar e desenvolver as funcionalidades básicas do jogo do Pacman, nomeadamente:

- Carregar um mapa a partir de um ficheiro de texto;
- Movimentar o Pacman para cima, baixo, esquerda, direita usando o teclado;
- Movimentar os fantasmas automaticamente tendo em conta as suas diferentes personalidades;
- Atualizar a pontuação sempre que o jogador come um ponto pequeno;
- Verificar se o jogador foi apanhado por um fantasma;
- Guardar o estado do jogo actual sempre que o jogador pressionar a tecla ‘s’;

Para a programação do jogo, deverá descarregar o ficheiro ZIP disponível na UCTeacher, e que contém dois ficheiros:

- pacman.py - 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, dos agentes (pac man e fantasmas) a criação do quadro de pontuação e detecção do pressionar das teclas ⬅️, ➡️, ⬆️, ⬇️.

- principal_alunos.py - Ficheiro onde os alunos deverão programar 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 achar necessárias para resolver o problema.

No código fornecido, existe um objeto Python chamado `estado_jogo`, que é um dicionário que armazena diversas informações relacionadas com o estado atual do jogo. Este dicionário é composto pelas seguintes chaves:

1. `estado_jogo['pacman']`: O valor associado a esta chave é, por sua vez, outro dicionário que armazena informações específicas sobre o Pacman:
    - `'objeto'`: Este campo tem o valor `None` por defeito, e destina-se a armazenar o objeto `Turtle` que representa o Pacman no jogo.
    - `'direcao_atual'`: Também com valor inicial `None`, este campo armazena um tuplo com a direção em que o Pacman está atualmente a mover-se.

2. `estado_jogo['fantasmas']`: Esta chave contém um dicionário de fantasmas, em que cada fantasma é identificado por um número (neste caso, de 3 a 6). Cada um destes fantasmas tem as seguintes propriedades:
    - `'objeto'`: Inicia com o valor `None` e destina-se a armazenar o objeto `Turtle` que representa o fantasma no jogo.
    - `'direcao_atual'`: Também com valor `None`, que será atualizado com tuplo com a direção atual de cada fantasma.
3. `estado_jogo['mapa']`: Esta chave tem inicialmente o valor `None`, mas será usada para armazenar uma lista com mapa jogo.
4. `estado_jogo['score']`: Armazena a pontuação atual do jogador, começando com o valor `0`.
5. `estado_jogo['marcador']`: Inicia-se como `None` e destina-se a armazenar um objeto `Turtle` cujo objetivo é desenhar o mundo.
6. `estado_jogo[quadro]`: Inicia-se como `None` e destina-se a armazenar um objeto `Turtle` que representa o quadro das pontuações.

O `estado_jogo` permite que o estado do jogo seja mantido de forma organizada e acessível, facilitando a atualização e monitorização dos diferentes elementos do jogo, como Pacman, os fantasmas, o mapa, a pontuação e as componentes visuais.

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

- `carrega_jogo(estado_jogo, nome_ficheiro)` é responsável por carregar o estado de um jogo previamente gravado num ficheiro de texto. Esta função permite iniciar um jogo, ou retomar o progresso de um jogo que foi guardado, atualizando o conteúdo `estado_jogo['mapa']` com os dados armazenados no ficheiro. É fornecido um ficheiro com o nome `mapa_inicial.txt` que contém a configuração inicial do jogo. Este ficheiro deverá ser transformado numa lista com 400 elementos que correspondem à configuração do mapa. Estes ficheiros contêm a seguinte informação:
    - `0` - representa uma barreira;
    - `1` - representa uma peça de comida;
    - `2` - representa o Pacman;
    - `[3..6]` - representam os 4 fantasmas;
    - `7`- representa um peça de comida ingerida pelo Pacman

- `pacman_direita(estado_jogo)` atualiza a direção do Pacman no jogo, definindo-a de modo que ele se mova 5 unidades para a direita. Esta função deverá atualizar o dicionário `estado_jogo`,, nomeadamente o campo `estado_jogo['pacman']['direcao_atual']`. Note que o valor associado à chave `'direcao_atual'` deverá ser um tuplo que será posteriormente usado para atualizar a posição do Pacman.

- `pacman_esquerda(estado_jogo)` atualiza a direção do Pacman no jogo, definindo-a de modo que ele se mova 5 unidades para a esquerda. Esta função deverá atualizar o dicionário `estado_jogo`, nomeadamente o campo `estado_jogo['pacman']['direcao_atual']`. Note que o valor associado à chave `'direcao_atual'` deverá ser um tuplo, que será posteriormente usado para atualizar a posição de Pacman.

- `pacman_cima(estado_jogo)` atualiza a direção do Pacman no jogo, definindo-a de modo que ele se mova 5 unidades para cima. Esta função deverá atualizar o dicionário `estado_jogo`, nomeadamente o campo `estado_jogo['pacman']['direcao_atual']`. Note que o valor associado à chave `'direcao_atual'` deverá ser um tuplo, que será posteriormente usado para atualizar a posição de Pacman.

- `pacman_baixo(estado_jogo)` atualiza a direção do Pacman no jogo, definindo-a de modo que ele se mova 5 unidades para baixo. Esta função deverá atualizar o dicionário `estado_jogo`, nomeadamente o campo `estado_jogo['pacman']['direcao_atual']`. Note que o valor associado à chave `'direcao_atual'` deverá ser um tuplo, que será posteriormente usado para atualizar a posição de Pacman.

- `movimenta_inky(estado_jogo)` é responsável por mover o fantasma "Inky" de forma aleatória dentro do jogo. Poderá utilizar a variável `DIRECOES_POSSIVEIS` que representa uma lista de tuplos com as direções possíveis.

- `movimenta_blinky(estado_jogo)` é responsável por mover o fantasma "Blinky" de forma aleatória dentro do jogo. Poderá utilizar a variável `DIRECOES_POSSIVEIS` que representa uma lista de tuplos com as direções possíveis.

- `movimenta_pinky(estado_jogo)` é responsável por mover o fantasma "Pinky" em direção ao Pacman, criando uma perseguição direta. O objetivo é fazer com que o Pinky se aproxime continuamente do Pacman. Para isso deverá ter em conta o seguinte:
    1. **Obter a posição atual de Pacman**: A função começa por ler a posição de Pacman a partir de `estado_jogo`, identificando as coordenadas onde Pacman se encontra no labirinto.
    2. **Obter a posição atual de Pinky**: Da mesma forma, a função recupera a posição atual de Pinky, representada no dicionário `estado_jogo['fantasmas'][PINKY_OBJECT]` (assumindo que Pinky é identificado pelo nome `PINKY_OBJECT`).
    3. **Calcula a direção**: Com base nas posições do Pacman e do Pinky, a função calcula a direção em que Pinky deve mover-se para se aproximar de Pacman. Por exemplo:
        - Se Pacman está acima de Pinky, a direção será "para cima".
        - Se Pacman está à direita de Pinky, a direção será "para a direita".
    4. **Movimento Válido**: Verifique se o movimento do Pinky é válido[1]. Caso não seja, deverá gerar um direção que seja válida, e que permita que o Pinky se movimente.
    5. **Retorna a direção**: A função retorna a direção calculada, indicando o movimento de Pinky para outras partes do programa.

- `movimenta_clyde(estado_jogo)` movimenta o fantasma Clyde, alternando entre um comportamento de dispersão e perseguição em direção ao Pacman. Clyde é

[1] Poderá utilizar a função `movimento_valido(ponto, estado_jogo)` que recebe um tuplo com as coordenadas (x,y) e verifica se o movimento é válido

conhecido por mudar de comportamento dependendo da distância que o separa de Pacman: quando está longe, ele persegue o Pacman, mas quando se aproxima demais, afasta-se, criando uma movimentação mais imprevisível. Para programar este comportamento deverá ter em conta o seguinte:

1. **Obtém as posições de Pacman e Clyde**: A função começa por ler as posições de Pacman e de Clyde a partir de `estado_jogo`, identificando as coordenadas de ambos no labirinto.
2. **Calcula a distância entre Clyde e Pacman**: A função calcula a distância entre Clyde e Pacman para decidir se Clyde deve dispersar-se ou perseguir Pacman. Geralmente, essa distância é calculada como a distância euclidiana ou apenas a diferença nos eixos x e y.
3. **Threshold (limiar) de distância**: Um valor limite é usado para definir o ponto em que Clyde altera o seu comportamento, e é definido pela variável `scatter_distance_threshold`. Se a distância entre Clyde e Pacman for superior ao threshold,o Clyde persegue Pacman; caso contrário, o Clyde afasta-se em direção ao `scatter_corner`.
4. **Determina a direção de movimento**:
    - **Perseguição**: Se Clyde está a uma distância superior ao threshold, ele move-se na direção de Pacman, aproximando-se.
    - **Dispersão**: Se Clyde está próximo de Pacman (abaixo do threshold), ele move-se em sentido oposto, dispersando-se para aumentar a distância.
5. **Movimento Válido**: Verifique se o movimento do Clyde é válido. Caso não seja, deverá gerar um direção que seja válida, e que permita que o Clyde se movimente.
6. **Retorna a direção**: A função retorna a direção calculada, indicando o movimento de Clyde para outras partes do programa.

- `perdeu_jogo(estado_jogo)` verifica se o jogador perdeu o jogo, detectando uma colisão entre Pacman e qualquer um dos fantasmas. Se houver uma colisão, a função deverá chamar a função `terminar_jogo(estado_jogo)`

- `guarda_jogo(estado_jogo)` tem como objetivo gravar o estado atual do mapa do jogo num ficheiro de texto, usando um formato específico para armazenar a informação sobre a posição de cada elemento do jogo (Pacman, fantasmas, pontos, paredes, etc.). O formato definido é uma matriz de 20x20, onde cada linha do mapa é representada por

uma sequência de números separados por vírgulas, e termina com uma vírgula após o último elemento (ver Figura 2). Este mapa é gravado no ficheiro de forma contínua, linha por linha. Poderá consultar o exemplo do ficheiro `'save.txt'`.

```
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,1,1,1,1,1,1,4,0,1,1,1,1,1,1,1,0,0,0,0,
0,1,0,0,1,0,0,1,0,1,0,0,1,0,0,1,0,0,0,0,
0,3,1,1,1,1,1,7,7,7,7,1,1,1,1,1,0,0,0,0,
0,1,0,0,1,0,1,0,0,0,1,0,1,0,0,1,0,0,0,0,
0,1,1,1,1,0,1,1,0,1,1,0,1,1,1,1,0,0,0,0,
0,1,0,0,1,0,0,1,0,1,0,0,1,0,0,0,0,0,0,0,
0,1,0,0,1,0,7,7,7,7,7,0,7,0,0,0,0,0,0,0,
0,1,1,1,1,1,7,0,0,0,7,7,7,7,7,7,0,0,0,0,
0,0,0,0,1,0,7,1,1,1,1,0,7,0,0,7,0,0,0,0,
0,0,0,0,1,0,7,0,0,0,1,0,7,0,0,7,0,0,0,0,
0,1,1,1,1,1,2,1,0,7,7,7,7,7,7,5,0,0,0,0,
0,1,0,0,1,0,0,1,0,7,0,0,0,0,0,1,0,0,0,0,
0,1,1,0,1,1,7,7,7,7,1,1,1,0,1,6,0,0,0,0,
0,0,1,0,1,0,7,0,0,0,1,0,1,0,1,0,0,0,0,0,
0,1,1,1,1,0,7,7,0,1,1,0,1,1,1,1,0,0,0,0,
0,1,0,0,0,0,0,1,0,1,0,0,0,0,0,1,0,0,0,0,
0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
```

Figura 2 - Exemplo do conteúdo do ficheiro resultante da função `guarda_jogo`

3 - Datas e Modo de Entrega

O trabalho deverá ser realizado parcialmente durante as aulas TP e por grupos com uma dimensão máxima de 2 alunos pertencentes à mesma turma.

O trabalho está sujeito a uma defesa obrigatória, em que todos os elementos do grupo têm de estar presentes. A não realização da defesa resultará numa classificação de 0 valores no trabalho.

Modo de Entrega

O trabalho deverá ser entregue eletronicamente através do Inforestudante.

Data Limite

23h59 do dia 01 de Dezembro de 2024