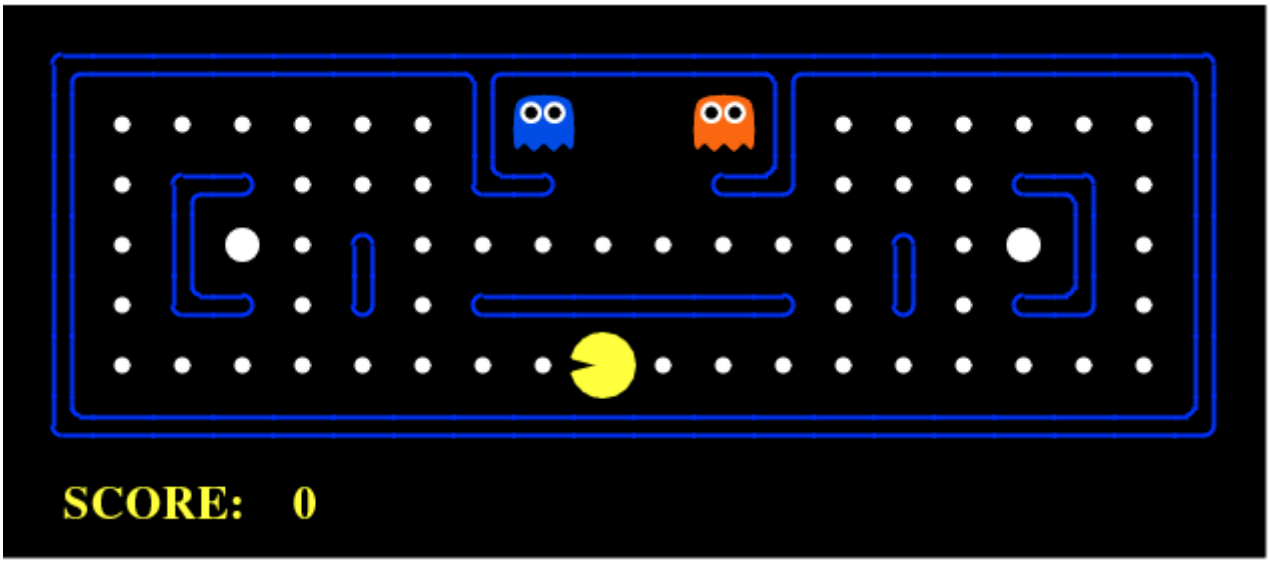
Pacman contra fantasma solitário

Projecto de IIA edição de 2019/20

Deadline: 12 de Dezembro de 2019 (1 minuto antes das 24h)



Conteúdo

Regras do jogo clássico Pacman

Topo

Objectivos

Objectivos do Pacman

O objectivo do Pacman é não ser capturado pelos fantasmas comer as pastilhas todas e obter a maior pontuação possível, tendo que fugir dos fantasmas. Os fantasmas começam sempre na base.

Objectivos dos fantasmas

Os fantasmas, por sua vez, pretendem apanhar o Pacman e querem que ele tenha o menor score possível, mesmo se não o conseguirem vencer.

Ordem de execução e acções possíveis

Em cada instante, todos os agentes, por ordem, decidem e executam as respectivas acções. O Pacman é o primeiro a executar a sua acção e os fantasmas jogam a seguir, um após o outro

Acções do Pacman

O Pacman pode mover-se 1 unidade numa das 4 direções ortogonais desde que não choque com um dos obstáculos (velocidade de 1), mas pode também ficar parado.

Acções dos fantasmas

Notem que um fantasma, se possível, nunca pode voltar para trás, tem sempre que manter a mesma direcção ao longo de um corredor e pode mudar de direcção numa encruzilhada. Mas, se chegar ao fundo de um beco sem saída, o fantasma é obrigado a inverter a direcção. Notem que podem coexistir mais do que um fantasma na mesma célula

Efeito das superpastilhas

Quando o Pacman come uma das super-pastilhas, os fantasmas ficam amedrontados e frágeis durante 40 instantes de tempo. Nesse período de tempo os fantasmas ficam vulneráveis e podem ser atacados e comidos pelo Pacman, regressado à base. Quando entram na base, perdem o medo e estão prontos para atacar o Pacman de novo. Quando estão sem medo, no modo normal, a velocidade dos fantasmas é igual à do Pacman (1), mas quando estão com medo, a velocidade é reduzida para metade (0.5).

Pontuação

Jogo manual

Topo

Para começar o ideal é jogarem manualmente. Embora o projecto seja apenas Pacman contra fantasma, podem jogar contra mais do que 1 fantasma. Para isso podem executar o comando seguinte em que jogam contra 2 fantasmas que jogam ao acaso, num dos mapas: o mediumClassic.

No caso de não especificarmos a classe python para o jogador que controla o Pacman então é porque é controlado pelo utilizador. Por omissão, o fantasma joga aleatoriamente.

Joguem com as teclas das setas!

In [49]:

```
! python pacman.py
```

O tempo entre "frames" da animação é dado pela opção ***--frameTime***, que por defeito tem o valor 0.1.

Se quiserem jogar de um modo mais rápido :-) podem desligar a animação.

In [2]:

```
! python pacman.py --frameTime 0.1
```

```
Pacman emerges victorious! Score: 2058
Average Score: 2058.0
Scores:        2058.0
Win Rate:      1/1 (1.00)
Record:        Win
```

Mas podem também precisar de mais tempo:

In [ ]:

```
! python pacman.py --frameTime 0.3
```

Random Pacman vs random fantasmas

Topo

Se quiserem assitir a um jogo entre um Pacman autónomo que joga totalmente ao acaso, e não é grande espingarda, contra dois fantasmas ao acaso, podem fazer:

In [3]:

```
## Jogar contra um fantasma mais combativo
! python pacman.py -p RandomPac --frameTime 0
```

```
Pacman died! Score: -488
Average Score: -488.0
Scores:        -488.0
Win Rate:      0/1 (0.00)
Record:        Loss
```

Opções da linha de comandos

Topo

Desligar o gráfico e mostrar o texto

Podem desligar os gráficos e ver a evolução do jogo em modo texto usando a opção -t:

In [17]:

```
! python pacman.py -p RandomPac -t
```

Modo sossegado

A opção -q serve para jogar sem imagens nem texto... sossego absoluto, nos bastidores, alguns chamam a isto o paraíso.

In [46]:

```
! python pacman.py -p RandomPac -q
```

```
Pacman died! Score: -475
Average Score: -475.0
Scores:        -475.0
Win Rate:      0/1 (0.00)
Record:        Loss
```

Limite de jogadas em cada jogo

Para que um jogo tenha sempre fim, mesmo que o Pacman teimosamente não queira comer as pastilhas nem o fantasma queira capturar o Pacman, estipulámos um limite para cada jogo que pode ser também uma opção na linha de comandos (-y). Consideramos como uma só jogada as jogadas de ambos os jogadores. Por omissão esse limite é de 10000. Se o jogo acabar devido ao limite de jogadas, esse jogo é de vitória para o Pacman porque não foi capturado. Não ecoamos o número de jogadas. Para fazer um jogo limitado a 20 jogadas duplas façam por exemplo:

In [5]:

```
! python pacman.py -p RandomPac -y 20
```

```
Pacman emerges victorious! Score: 20
Average Score: 20.0
Scores:        20.0
Win Rate:      1/1 (1.00)
Record:        Win
```

Fazendo vários jogos

Para fazerem vários jogos entre os mesmos jogadores devem usar a opção -n. Em baixo podem fazer com que se repitam 10 jogos entre os dois jogadores aleatórios. NO fim poderão observar a média das pontuações, a pontuação de cada um dos jogos, por ordem, a taxa de vitórias e a indicação se ganharam ou perderam em cada jogo, por ordem.

In [9]:

```
! python pacman.py -p RandomPac --frameTime 0 -n 10 -q
```

```
Pacman died! Score: -469
Pacman died! Score: -483
Pacman died! Score: -418
Pacman died! Score: -496
Pacman died! Score: -425
Pacman died! Score: -396
Pacman died! Score: -442
Pacman died! Score: -359
Pacman died! Score: -401
Pacman died! Score: -509
Average Score: -439.8
Scores:        -469.0, -483.0, -418.0, -496.0, -425.0, -396.0, -442.0, -35
9.0, -401.0, -509.0
Win Rate:      0/10 (0.00)
Record:        Loss, Loss, Loss, Loss, Loss, Loss, Loss, Loss, Loss, Loss
```

Fixando a RandomSeed

Para correrem o mesmo jogo, para debugging podem fixar a random seed, recorrendo à opção -f. Executem duas vezes o jogo entre dois joogadores ao acaso e verão que o jogo se repete ao segundo.

In [15]:

```
! python pacman.py -p RandomPac -f --frameTime 0 -n 10 -q
```

```
Pacman died! Score: -473
Pacman died! Score: -405
Pacman died! Score: -438
Pacman died! Score: -476
Pacman died! Score: -483
Pacman died! Score: -442
Pacman died! Score: -490
Pacman died! Score: -502
Pacman died! Score: -370
Pacman died! Score: -467
Average Score: -454.6
Scores:        -473.0, -405.0, -438.0, -476.0, -483.0, -442.0, -490.0, -50
2.0, -370.0, -467.0
Win Rate:      0/10 (0.00)
Record:        Loss, Loss, Loss, Loss, Loss, Loss, Loss, Loss, Loss, Loss
```

Mapas

Existem vários ambientes do Pacman que podem utilizar e que podem ser visualizados em modo texto e em modo gráfico. Na verdade eles estão em ficheiros em modo texto (com a extensão .lay) e se quiserem podem criar mapas novos, em particular se quiserem testar a função de avaliação para cenários especificos. Os mapas que existem são:

```
capsuleClassic
contestClasic
mediumClassic
minimaxClassic
openClassic
originalClassic
powerClassic
smallClassic
testClassic
trappedClassic
trickyClassic
```

Por defeito, o mapa é mediumClassic, mas experimentem os vários mapas, por exemplo, o originalClassic:

In [ ]:

```
! python pacman.py -l originalClassic
```

Fazer zoom

Se quiserem fazer zoom sobre o gráfico podem usar a opção -z. Passarão para o dobro se colcoarem 2

In [ ]:

```
! python pacman.py -z 2
```

Manipular o número de fantasmas

O Layout define o número máximo de fantasmas para um certo mapa, mas podemos jogar com menos fantasmas. Por exemplo se quisermos jogar no mapa mediumClassic apenas com um fantasma poderemos usar -k:

In [58]:

```
! python pacman.py -k 1
```

```
Pacman died! Score: -103
Average Score: -103.0
Scores:        -103.0
Win Rate:      0/1 (0.00)
Record:        Loss
```

Arquivar um jogo

Podemos arquivar um jogo usando na linha de comandos -r. O nome do ficheiro é gerado automaticamente com base no Pacman e Fantasmas utilizados e colocado na directoria *results*.

In [66]:

```
! python pacman.py -r -k 2
```

```
Pacman died! Score: -45
Average Score: -45.0
Scores:        -45.0
Win Rate:      0/1 (0.00)
Record:        Loss
```

Correr um jogo arquivado

Podem procurar na directoria *results* por um ficheiro que começa em "recorde-game-". Então para o correr de novo executem o comando seguinte em que subsitutem o nome do ficheiro pelo que foi gerado:

In [69]:

```
! python pacman.py --replay results/recorded-game-PacKeyboardAgent-FantRandomGhost-1-11
-26-22-33-17
```

```
Replaying recorded game results/recorded-game-PacKeyboardAgent-FantRandomG
host-1-11-26-22-33-17.
Pacman died! Score: -45
```

Objectivos do projecto

Topo

Fantasma contra Pacman

Neste projecto terão de desenvolver dois jogadores, um para controlar o Pacman e o outro para controlar um fantasma apenas. Mesmo que o mapa tenha mais do que um fantasma, os fantasmas em excesso serão ignorados através da opção -k 1. Todos os jogadores vão utilizar o mesmo algoritmo alfabeta limitado à mesma profundidade e terão que desenvolver duas funções de avaliação, uma para jogarem como Pacman e outra para jogarem como fantasma.

Função de avaliação para o Pacman e para o Fantasma

Para que possamos organizar mais facilmente o torneio é conveniente formatar os nomes das funções de avaliação que conterão o número dos grupos.

Assim, qualquer função de avaliação do Pacman será chamada de ***pac_X*** em que X é o número do grupo. O mesmo para o fantasma que será ***fant_X***.

O grupo 1 desenvolverá as funções ***pac_1()*** e ***fant_1()***. Atenção que o código auxiliar deve também estar etiquetado com o número do grupo (variáveis, métodos e classes) para que não hajam grupos a correrem código do vizinho do lado.

Notem que essas funções recebem dois argumentos: o estado do jogo e o jogador.

Há dois jogadores: 'Pacman' e 'Ghost'

Vamos descrever o estado do jogo mais abaixo.

Torneio

No final do projecto, vamos realizar um torneio em que todos os Pacmans jogam com todos os Fantasmas dos outros grupos em pelo menos dois mapas e várias vezes em cada mapa. Os mapas são os mesmos para todos e à partida ninguém sabe em que mapa irá decorrer o torneio. Pelo menos um dos mapas é completamente novo, desconhecido dos jogadores.

Para cada Pacman:

1. Vamos contar o número de vitórias
2. Vamos obter o score acumulado

Para cada Fantasmas:

1. Vamos contar o número de derrotas do adversário
2. Vamos obter o score acumulado Desta tabela de valores sairá a nota final do projecto.

Limite de profundidade da árvore de procura

Iremos em princípio utilizar o limite de profundidade de 8 no torneio tanto para o agente Pacman como para o fantasma. Uma profundidade de 8 quer dizer que 4 jogadas alternadas de cada um. Pode acontecer que por razões de tempo tenhamos que diminuir esse limite para 6.

pacmanX e fantasmaX

Nós vamos fornecer dois jogadores básicos: pac_X e fant_X. Notem que todos os fantasmas irão também jogar contra o pac_X e que todos os pacmans irão jogar contra o fant_X. É conveniente que obtenham um desempenho superior aos nossos jogadores base X.

Estado do jogo Pacman

Topo

O estado do jogo vai ser formado por um triplo que indica:

```
to_move   - o jogador que vai jogar a seguir
board  - toda a informação sobre o jogo
extra - campo livre para poderem colocar a informação que acharem interessante p
ara a construção da função de avaliação. É um dicionário que começa vazio: {}
```

O estado é definido como um namedtuple no ficheiro *multiagents.py*

```
ClassicPac = namedtuple('ClassicPac', 'to_move, board, extra',defaults=[None])
```

O campo **to_move** pode ter dois valores: 'Pacman' ou 'Ghosts'

O campo **extra** serve para guardarem o que quiserem para formar o estado do jogo, para além da informação dada pelo campo **board**. É como se aumentassem o estado do jogo de modo livre, guardando informação relevante.

A seguir iremos explicar o que consta no campo **board** do estado, que é o componente do estado que é o mais importante e que é utilizado pelo próprio motor do jogo para executar o jogo.

Informação sobre o tabuleiro de jogo

Topo

Toda a informação relevante sobre o jogo, i.e., sobre as pastilhas, as super-pastilhas, o estado do Pacman, dos fantasmas e o score, está na representada na classe **GameState**, que se situa no ficheiro *pacman.py*.

As instâncias de **GameState** são utilizados pelo próprio controlador do jogo para capturarem o estado actual do jogo e vai ser utilizado pelos agentes para raciocinarem acerca do jogo.

A maior parte da informação está guardada num objecto da classe **GameStateData**. Recomenda-se que acedam aos dados através des métodos de leitura de dados ("accessors") e não directamente aos atributos dos objectos **GameStateData**.

É importante saber que o index do Pacman é o 0 e os fantasmas correspondem ao index de 1 até n, no caso de termos n fantasmas, mas para o torneio teremos apenas dois agentes, o de índice 0 e de índice 1.

Eis alguns métodos da classe:

```
initialize(layout, numGhostAgents): Cria um estado inicial do jogo com base num
 mapa

getLegalActions(agentIndex): Devolve a lista de acções dado o index do agente

getLegalPacmanActions(): Devolve a lista de acções do Pacman (agente com index
 0)

generateSuccessor(agentIndex, action): O estado do jogo que resulta do action de
agentIndex

generatePacmanSuccessor(action): O estado do jogo que resulta do action do agent
Index 0 (Pacman)

getPacmanState(): Devolve o estado do agente (objecto AgentState, em game.py) pa
ra o Pacman
    state.pos devolve a posição actual
    state.direction devolve o vector de movimento

getPacmanPosition(): Devolve a posição actual do Pacman

getGhostStates(): Devolve os estados dos fantasmas

getGhostState(agentIndex): Devolve o estado do fantasma com índice agentIndex
```

Convertendo o mapa (layout) no estado inicial do jogo

Vamos converter o ficheiro layout num objecto da classe layout que está definida em layout.py. Notem que no modo texto os mapas usam os símbolos:

```
%  -- Obstáculo
P  -- Pacman
G  -- Fantasma
.  -- Pastilha
o  -- Super-pastilha
```

Os pontos do mapa possuem coordenadas cartesianas em que o ponto (0,0) corresponde ao canto mais à esquerda, no fundo do mapa.

Para fazer essa conversão iremos criar um object da classe **Layout()** e usar o método ***initialize()***. Podemos usar um dos mapas mais pequenos, por exemplo o smallClassic

In [6]:

```
import layout
import math
lay=layout.getLayout("smallClassic.lay")
print(lay)
```

```
%%%%%%%%%%%%%%%%%%%%
%......%G  G%......%
%.%%...%%  %%...%%.%
%.%o.%........%.o%.%
%.%%.%.%%%%%%.%.%%.%
%........P........%
%%%%%%%%%%%%%%%%%%%%
```

Temos de converter este layout num objecto GameState, criando uma instância dessa classe e depois aplicando o método ***initialize()***. Ao imprimir esse objecto podemos vizualizar o jogo em modo texto, juntamente com a pontuação. O pacman tem o símbolo de '>', todos os outros símbolos são os mesmos que no ficheiro de layout. Estes comandos serão importantes para mais tarde testarem as vossas funções de avaliação. Poderão criar os cenários de teste em modo texto e depois inicializarem um jogo, acedendo ao estado e podendo depois executar as funções de avaliação.

In [7]:

```
from  pacman import *
g = GameState()
g.initialize(lay,2)
print(g)
```

```
%%%%%%%%%%%%%%%%%%%%
%......%G  G%......%
%.%%...%%  %%...%%.%
%.%o.%........%.o%.%
%.%%.%.%%%%%%.%.%%.%
%........<........%
%%%%%%%%%%%%%%%%%%%%
Score: 0
```

Pontuação, paredes, pastilhas e super-pastilhas

Vamos obter a pontuação, as coordenadas das super-pastilhas e as pastilhas

In [57]:

```
print("O score:",g.getScore())
print('Pastilhões:',g.getCapsules())
print('Comidinha:')
print(g.getFood())
```

```
O score: 0.0
Pastilhões: [(3, 3), (16, 3)]
Comidinha:
FFFFFFFFFFFFFFFFFFF
FTTTTTTFFFFFFTTTTTF
FTFFTTTFFFFFFTTTFFF
FTFFTFTTTTTTTTFTFFF
FTFFTFTFFFFFFTFTFFF
FTTTTTTTTFTTTTTTTTF
FFFFFFFFFFFFFFFFFFF
```

Devolve um objeto do tipo ***Grid()***. Temos de o converter numa lista, usando o método ***asList()***:

In [9]:

```
print('Comidinha:',g.getFood().asList())
```

```
Comidinha: [(1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (2, 1), (2, 5), (3,
1), (3, 5), (4, 1), (4, 2), (4, 3), (4, 4), (4, 5), (5, 1), (5, 4), (5,
5), (6, 1), (6, 2), (6, 3), (6, 4), (6, 5), (7, 1), (7, 3), (8, 1), (8,
3), (9, 3), (10, 1), (10, 3), (11, 1), (11, 3), (12, 1), (12, 3), (13, 1),
(13, 2), (13, 3), (13, 4), (13, 5), (14, 1), (14, 4), (14, 5), (15, 1), (1
5, 2), (15, 3), (15, 4), (15, 5), (16, 1), (16, 5), (17, 1), (17, 5), (18,
1), (18, 2), (18, 3), (18, 4), (18, 5)]
```

O mesmo com as paredes...

In [10]:

```
print('Blocos:',g.getWalls().asList())
```

```
Blocos: [(0, 0), (0, 1), (0, 2), (0, 3), (0, 4), (0, 5), (0, 6), (1, 0),
(1, 6), (2, 0), (2, 2), (2, 3), (2, 4), (2, 6), (3, 0), (3, 2), (3, 4),
(3, 6), (4, 0), (4, 6), (5, 0), (5, 2), (5, 3), (5, 6), (6, 0), (6, 6),
(7, 0), (7, 2), (7, 4), (7, 5), (7, 6), (8, 0), (8, 2), (8, 4), (8, 6),
(9, 0), (9, 2), (9, 6), (10, 0), (10, 2), (10, 6), (11, 0), (11, 2), (11,
4), (11, 6), (12, 0), (12, 2), (12, 4), (12, 5), (12, 6), (13, 0), (13,
6), (14, 0), (14, 2), (14, 3), (14, 6), (15, 0), (15, 6), (16, 0), (16,
2), (16, 4), (16, 6), (17, 0), (17, 2), (17, 3), (17, 4), (17, 6), (18,
0), (18, 6), (19, 0), (19, 1), (19, 2), (19, 3), (19, 4), (19, 5), (19,
6)]
```

Podemos querer saber se uma determinada célula ou posição é uma parede ou uma pastilha.

In [11]:

```
print('(1,1) é comida?',g.hasFood(1,1))
print('(0,5) é uma parede?',g.hasWall(0,5))
```

```
(1,1) é comida? True
(0,5) é uma parede? True
```

Informação sobre o Pacman e os fantasmas

Pacman

Vamos então obter a posição do Pacman

In [12]:

```
print('Pacman:',g.getPacmanPosition())
```

```
Pacman: (9, 1)
```

Podemos pedir o estado do Pacman que é um objecto da classe **AgentState** (em *game.py*) e podemos usar o método ***getPosition()***

In [14]:

```
g.getPacmanState().getPosition()
```

Out[14]:

```
(9, 1)
```

e o método ***getDirection()***, que neste momento é 'Stop' (Estava sem direcção)

In [15]:

```
g.getPacmanState().getDirection()
```

Out[15]:

```
'Stop'
```

Se imprimirmos o objecto AgentState podemos ver a posição e a direcção

In [16]:

```
print(g.getPacmanState())
```

```
Pacman: (x,y)=(9, 1), Stop
```

Fantasmas

Podemos obter a lista das posições dos fantasmas

In [17]:

```
print('Posições dos Fantasmas:',g.getGhostPositions())
```

```
Posições dos Fantasmas: [(8, 5), (11, 5)]
```

Podemos imprimir a direcção e posição de cada um dos fantasmas:

In [18]:

```
print(g.getGhostState(1))
print(g.getGhostState(2))
```

```
Ghost: (x,y)=(8, 5), Stop
Ghost: (x,y)=(11, 5), Stop
```

O Fantasma 1 é o que está mais à esquerda no mapa, neste caso. Podemos também obter a informação da posição e direcção separadamente:

In [19]:

```
print('Fantasma 1:','Pos:',g.getGhostState(1).getPosition(),'Dir:',g.getGhostState(1).g
etDirection())
```

```
Fantasma 1: Pos: (8, 5) Dir: Stop
```

Também poderemos saber o tempo que falta até esgotar o seu medo, usando o atributo ***scaredTime***. Cada fantasma começa sem medo e esse atributo estará a 0.

In [20]:

```
print('Medo do fantasma 1:',g.getGhostState(1).scaredTimer)
```

```
Medo do fantasma 1: 0
```

In [30]:

```
g.getPacmanState().start.getPosition()
```

Out[30]:

```
(9, 1)
```

Acções dos agentes

In [ ]:

```
print(g)
print('Acções do agent de índice 0 (Pacman):')
g.getLegalActions()
```

Vejemos quais as acções possíveis de cada um dos fantasmas.

In [ ]:

```
print('Acções do agent de índice 1 (fantasma):',g.getLegalActions(1))

print('Acções do agent de índice 2 (fantasma):',g.getLegalActions(1))
g.getLegalActions(2)
```

Transições entre estados: resultantes de jogadas

In [ ]:

```
g1=g.generateSuccessor(1,'East')
print(g1)
g2=g1.generateSuccessor(2,'West')
print(g2)
g3=g2.generateSuccessor(1,'East')
print(g3)
g3.getLegalActions(1)
```

Agora, façamos o Pacman avançar 4 passos para leste, e vamos imprimir a evolução do mapa, da pontuação e das acções possíveis.

In [ ]:

```
g4=g3.generateSuccessor(0,'East')
print(g4)
print('Acções do Pacman:',g4.getLegalActions(0))
g5=g4.generateSuccessor(0,'East')
print(g5)
print('Acções do Pacman:',g5.getLegalActions(0))
g6=g5.generateSuccessor(0,'East')
print(g6)
print('Acções do Pacman:',g6.getLegalActions(0))
g7=g6.generateSuccessor(0,'East')
print(g7)
print('Acções do Pacman:',g7.getLegalActions(0))
```

Globais

Topo

Existem 3 variáveis em *pacman.py* que indicam a duração máxima do efeito das super-pastilhas sobre os fantasmas:

```
SCARED_TIME = 40    # Moves ghosts are scared
```

A distância que o Pacman precisa de estar de um fantasma para ser considerado capturado

```
COLLISION_TOLERANCE = 0.7  # How close ghosts must be to Pacman to kill
```

A penalidade na pontuação por cada unidade de tempo

```
TIME_PENALTY = 1  # Number of points lost each round
```

As variáveis que indicam a velocidade do Pacman e dos fantasmas estão nas variáveis seguintes, em que a primeira pertence à class **PacmanRules** e a segunda à classe **GhostRules**.

```
PACMAN_SPEED = 1   na classe PacmanRules em pacman.py

GHOST_SPEED  1.0     está na classe GhostRules em pacman.py
```

Noten que a velocidade de um fantasma com medo é sempre igual GhostRules.GHOST_SPEED.

A classe Agent

Todos os jogadores são instâncias da classe **Agent** que tem o índice do agente (0 para o Pacman e os restates para os fantasmas) e o método ***getAction()*** que vai devolver a acção em cada momento.

```
class Agent:
    """
    An agent must define a getAction method, but may also define the
    following methods which will be called if they exist:

    def registerInitialState(self, state): # inspects the starting state
    """

    def __init__(self, index=0):
        self.index = index

    def getAction(self, state):
        """
        The Agent will receive a GameState and must return an action from Direct
ions.{North, South, East, West, Stop}
        """
        raiseNotDefined()

    def myName(self):
        """O identificador que aparece no ficheiro que memoriza um jogo
        quando um objecto desta classe joga"""
        return self.__class__.__name__
```

A classe RandomPac (subclasse de Agent)

O Pacman ao acaso escolhe uma das acções possíveis com igual probabilidade.

```
class RandomPac(Agent):
    """ A pacman that chooses its actions in a random way"""

    def getAction(self, state):
        # Generate candidate actions
        legal = state.getLegalPacmanActions()
        return random.choice(legal)
```

A classe RandomGhost (subclasse de Agent)

Um **GhostAgent** decide a sua acção ao acaso

```
class RandomGhost(Agent):
    """ A ghost that chooses its actions in a random way"""

    def getAction(self, state):
        # Generate candidate actions
        legal = state.getLegalActions(self.index)
        return random.choice(legal)
```

Jogadores que usam o alfabeta

Topo

Pacman que maximiza o score contra fantasma ao acaso

O jogador Pacman que utiliza a variante alfabeta do minimax tem de ser invocado com a opção -p igual a **AlphaBetaAgent** e depois na opção -a vão os dois parâmetros: a profundidade e a função de avaliação. Notem que é também necessário invocar a opção -k 1, para que o jogador Ghost controle apenas um fantasma. Atenção que os jogadores alfabeta não funcionarão correctamente com mais do que um fantasma.

Vejemos o exemplo de um jogo entre o Pacman alfabeta à profundidade 2 que utiliza a pontuação como função de avaliação contra um fantasma que joga completamente ao acaso. A função de avaliação que pega num estado avalia esse estado em função do Pacman é ***so_score()***.

In [50]:

```
! python pacman.py  -r -p AlphaBetaAgent -a depth=2,evalFn=so_score --frameTime 0 -k 1
 -l smallClassic
```

```
Pacman emerges victorious! Score: 329
Ending graphics raised an exception: 0
Average Score: 329.0
Scores:        329.0
Win Rate:      1/1 (1.00)
Record:        Win
```

Fantasma que maximiza o simétrico do score contra Pacman aleatório

O jogador Fantasma que utiliza a variante alfabeta do minimax tem de ser invocado com a opção -g igual a **AlphaBetaGhost** e a opção -b encarrega-se de passar os dois parâmetros importantes: a profundidade e a função de avaliação. Vamos usar também a profundidade 2. A função de avaliação que pega num estado avalia esse estado em função do Ghost é ***anti_score()***

In [51]:

```
! python pacman.py -g AlphaBetaGhost -b depth=2,evalFn=anti_score -p RandomPac --frameT
ime 0.2 -k 1 -l mediumClassic
```

...agora contra Pacman que maximiza score

In [52]:

```
! python pacman.py -g AlphaBetaGhost -b depth=8,evalFn=anti_score -p AlphaBetaAgent -a
depth=8,evalFn=so_score --frameTime 0 -k 1 -l mediumClassic
```

```
Pacman emerges victorious! Score: 676
Ending graphics raised an exception: 0
Average Score: 676.0
Scores:        676.0
Win Rate:      1/1 (1.00)
Record:        Win
```

Notem que quando não há pastilhas para comer à profundidade desejada, o score é o mesmo para todas as folhas da árvore de procura, o Pacman funciona como um jogador aleatório. Mas, se o fantasma se aproximar, até lhe dá uma ajudinha. Deixa de andar às voltas, porque começas a fugir e a comer as pastilhas se estiverem ao seu alcance.

A classe MultiAgentSearchAgent

Todos os agentes que usam o minimax alfabeta são subclasses da classe **MultiAgentSearchAgent**, no ficheiro *MultiAgents.py*. Por omissão a função de avaliação é a **so_score()** e a profundidade é 2.

```
class MultiAgentSearchAgent(Agent):
    """
     This class provides the constructor with additional three attributes: dept
h, evaluationFunction and a update the extra part of the state. It is a default
 name to
    every subclass
    """

   # def __init__(self, evalFn = 'scoreEvaluationFunction', depth = '2'):
    def __init__(self, index=0,evalFn = 'so_score', extraFn='identity',depth =
'2'):
        self.index = index  #0 # Pacman is always agent index 0
        self.evaluationFunction = util.lookup(evalFn, globals())
        self.extra_fn = util.lookup(extraFn,globals())
        self.depth = int(depth)
        self.extra = {}


    def myName(self):
        # This will be used for the identification of this agent in the record f
ile
        #return self.__class__.__name__ + self.evaluationFunction.__name__
        return self.evaluationFunction.__name__
```

A classe AlphaBetaAgent

Esta é a classe de todos os Pacmans que usam o alfabeta. É uma subclasse da classe anterior e por isso o limite da profundidade, a função de avaliação e a função de update do extra do estado, estão guardadas nos atributos herdados.

```
class AlphaBetaAgent(MultiAgentSearchAgent):
    """
     The alpha-beta pruning Pacman (question 3)
    """



    def getAction(self, gameState):
        """
        Returns the minimax action using self.depth and self.evaluationFunction
        """
        #print("Lets decide with Extra do Pac",self.extra_fn)
        #self.extra=self.extra_fn(gameState,self.extra)
        #print('Current extra:',self.extra)
        currentState = ClassicPac(to_move="Pacman",board=gameState,extra=self.ex
tra)
        thisGame = ClassicPacman(currentState)
        #print("UAU. here goes alpha beta")
        return alphabeta_cutoff_search(currentState, thisGame, d=self.depth, eva
l_fn=self.evaluationFunction, extra_fn=self.extra_fn)
```

A função de avaliação so_score

Esta é a função de avaliação que devolve o score do jogo e que é usada pelo Pacman: ele gosta de maximizar o seu score. Notem que as funções de avaliação recebem sempre um estado do jogo e um jogador.

```
def so_score(gState,player):
    return gState.board.getScore()
```

A classe AlphaBetaGhost

Esta é a classe de todos os fantasmas que utilizam o minimax alfabeta.

```
class AlphaBetaGhost(MultiAgentSearchAgent):
    """
    Your minimax ghost with alpha-beta pruning
    """
    def getAction(self, gameState):
        """
        Returns the alfabeta action using self.depth and self.evaluationFunction
        """
        #print('Jogada dupla:',gameState.numMoves())
        #print("Lets decide with Extra do Ghost",self.extra_fn)
        self.extra=self.extra_fn(gameState,self.extra)
        #print('Current extra:',self.extra)
        currentState = ClassicPac(to_move="Ghosts",board=gameState,extra=self.ex
tra)
        thisGame = ClassicPacman(currentState)
        #print("UAU. here goes alpha beta")
        return alphabeta_cutoff_search(currentState, thisGame, d=self.depth, eva
l_fn=self.evaluationFunction,extra_fn=self.extra_fn)
```

A função de avaliação anti_score

Esta função é para o fantasma: ele pretende maximizar o simétrico da pontuação do jogo.

```
def anti_score(gState,player):
    return -gState.board.getScore()
```

Melhorando a função de avaliação: a função scooore

Vamos melhorar o desempenho do nosso Pacman. Ele prefere aumentar o seu score mas também gosta de se aproximar da pastilha mais próxima. Notem que ele não calcula a distância real, mas usa uma estimativa da distância, a nossa conhecida distância de manhatan. Vamos fazer um jogo entre o ***anti_score*** e o ***scooore***.

In [ ]:

```
! python pacman.py  -g AlphaBetaGhost -b depth=8,evalFn=anti_score -p AlphaBetaAgent -a
depth=8,evalFn=scooore -k 1
```

Função de avaliação scooore

```
def manhatanDist(p1,p2):
        p1x,p1y=p1
        p2x,p2y=p2
        return abs(p1x-p2x)+abs(p1y-p2y)


def scooore(gState,player):
    foodList = gState.board.getFood().asList()
    minDistance = 0
    # Compute distance to the nearest food
    if len(foodList) > 0: # This should always be True,  but better safe than so
rry
        myPos = gState.board.getPacmanPosition()

        minDistance = min([manhatanDist(myPos, food) for food in foodList])
    return gState.board.getScore() * 100 - minDistance
```

Extra Extra

Por omissão a função que faz o update do extra é a função ***identity()*** que mantém o extra intacto.

Vamos dar dois exemplos de jogadores, um que controla o Pacman e outro que controla o Fantasma que memorizam as posições do Pacman e do Fantasma ao longo de todo o jogo. Notem que cada jogador tem um atributo ***extra*** onde é actualizado esse extra estado do jogo e que neste caso vai ser um dicinário com a lista de posições do Pacman e do Fantasma. Por outro lado, o estado que o algoritmo alfabeta tem como input transfere o valor do atributo extra para o campo extra, e assim a árvore alfabeta terá estados com o extra actualizado e preparado para ser usado com a função de avaliação.

Notem que os extras só são actualizados antes do jogador respectivo fazer a sua jogada, tanto durante o jogo como durante a árvore de procura do alfabeta. O fantasma começará com as a posição actual e original do Pacman e a sua própria. O Pacman começará com as posições iniciais dos dois.

Podem ver que as função de actualização do campo extra é feita no método ***resultado()*** da classe **ClassicPacman** como no ***getAction()*** de ambas as classes **alphaBetaAgent** e **AlphaBetaGhost**

Este exemplo é dado apenas para ilustração porque não temos nenhuma função de avaliação que faça uso dessa informação. Vejam as duas funções ***extra_mem()*** e ***extra_memF()***. Se não usarem esses campos na linhas de comandos a função ***identity()*** será utilizada.

```
    def extra_mem(gState,extra):
        """ memorizes the positions of Pacman and ghosts
        """

        if extra == {}:
            n_extra ={'Pacman':[gState.getPacmanPosition()],'Ghosts':[gState.getGhos
    tPosition(1)]}
            return n_extra
        else:
            n_extra=extra.copy()
            n_extra['Pacman']=[gState.getPacmanPosition()]+n_extra['Pacman']
            n_extra['Ghosts']=[gState.getGhostPosition(1)]+n_extra['Ghosts']
        return n_extra

    def extra_memF(gState,extra):
        """ memorizes the positions of Pacman and ghosts
        """
        if extra == {}:
            n_extra ={'Pacman':[gState.getPacmanPosition(),gState.getPacmanState().s
    tart.getPosition()],'Ghosts':[gState.getGhostPosition(1)]}
            return n_extra
        else:
            n_extra=extra.copy()
            n_extra['Pacman']=[gState.getPacmanPosition()]+n_extra['Pacman']
            n_extra['Ghosts']=[gState.getGhostPosition(1)]+n_extra['Ghosts']
        return n_extra
```

Para usarmos as funções de update do extra usamos o campo extraF nas opções -a e -g, respectivamente:

In [70]:

```
! python pacman.py  -y 4 -g AlphaBetaGhost -b depth=8,evalFn=anti_score,extraFn=extra_m
emF -p AlphaBetaAgent -a depth=8,evalFn=scooore,extraFn=extra_mem -k 1
```

```
Pacman emerges victorious! Score: 36
Average Score: 36.0
Scores:        36.0
Win Rate:      1/1 (1.00)
Record:        Win
```

Grupos

Os grupos para o projecto são formados por no máximo 3 alunos. Recomendamos a formação de grupos de 3 mas não impedimos que hajam grupos de dois elementos ou até alunos a solo.

Submissão do projecto

Coloquem na pasta projIIA_X, em que X é o identificador do grupo, um ficheiro projIIA_X.py com pelo menos as duas funções pac_X e fant_X e um pequeno report a explicar o que fazem as duas funções de avaliação. No caso de utilizarem o campo extra do estado é necessário incluir também as duas funções de actualização do campo parao Pacman: extraP_X e para o fantasma: extraF_X, que actualizam o campo extra com as jogadas reais e com as jogadas pensadas e antecipadas através do algoritmo alfabeta.Zipem essa pasta para projIIA_X.zip e submetam esse ficheiro.