Bomberman – Projeto de IIA 2019

- O nosso agente divide-se principalmente em dois ficheiros, o “search.py” e o “students.py”.
- O “search.py” é em todo muito parecido com o módulo implementado nas aulas práticas “tree_search.py”. Contém os métodos abstratos precisos para definir um domínio, a inicialização de um problema dado um ponto de partida e um objetivo final, a criação dos nós específicos para a árvore de pesquisa, e por fim a implementação da nossa árvore de pesquisa que se rege por uma pesquisa do tipo Greedy.
- O “students.py” é quem irá fazer a maior parte do trabalho. É lá que são implementados os métodos para definir o domínio, criar um domínio e alternar entre os estados; “fugir da bomba”, “matar inimigos”, “destruir paredes”, “procurar powerup” e “procurar a saída”.

João Tomás 88930
Martim Neves 88904
Tiago Dias 88896

Search.py

SearchDomain()

- Contém os métodos necessários para implementar um domínio. Todos os métodos desta classe são abstratos e serão mais tarde implementados no ficheiro “student.py”.

SearchProblem

- Esta classe é capaz de criar um problema, recebendo como argumento o domínio em questão, o ponto de partida e um objetivo final.

BombermanNode

- Cria e implementa os nós necessários e específicos para a nossa árvore de pesquisa (BombermanTree).

BombermanTree

- Cria e implementa a nossa árvore de pesquisa utilizando a estratégia de pesquisa denominada de Greedy.

Student.py

BombermanDomain

init()	Recebe o mapa do jogo e um estado. Inicializa assim o nosso domínio com a posição do bomberman, as walls, bombs, enemies, powerups, portal e level.
actions()	Recebe um estado como argumento. Retorna então a tecla disponível que leva o bomberman a ir para uma posição livre.
result()	Dado um estado e uma ação recebidos como argumentos, retorna a nova posição consequente da ação passada.
cost()	Recebe um state e uma action como argumentos. Define o custo de cada ação como 1.
heuristic()	Dado um estado e um objetivo como argumentos, vai calcular a distância entre o estado e o objetivo recebidos.

Student.py - estados

Procurar Powerup

- Inicialmente verifica se a lista dos powerup's se encontra vazia. Caso seja verdade, cria um problema e devolve o caminho para o powerup em questão.
Existe uma variável "powerupsCount" que serve para garantir que o bomberman não avança para o nível seguinte sem ter apanhado o powerup.

Procurar Inimigo

- Verifica se a lista de inimigos não se encontra vazia. Caso seja verdade, percorre esta lista de forma a encontrar o inimigo mais próximo, através do cálculo da heuristic. Quando o inimigo mais próximo é localizado, o caminho é calculado e o bomberman irá persegui-lo de forma a tentar matá-lo.

Procurar Paredes

- Vai percorrer a lista de paredes de forma a encontrar a parede mais próxima, através do cálculo da heuristic. Após saber a parede mais próxima, calcula o caminho até ela e planta a bomba de forma a destruí-la.

Student.py - estados

Fugir da Bomba

- A função avoidBomb é chamada e a posição do bomberman é passada como argumento (local onde a bomba é plantada). A função avoidBomb tem acesso ao bomb radius, e com esta informação vai calcular a casa que está livre e fora do alcance da bomba, devolvendo o caminho até ela.

Procurar Saída

- Verifica se a lista do portal não está vazia, se não há mais inimigos para matar e se o powerup foi apanhado. Caso todas estas condições se verifiquem o caminho para o portal de saída é calculado.