Bomberman

Introdução à Inteligência Artificial

Dário Matos (89288), Pedro Almeida (89250) e Samuel Duarte (89222)

Arquitetura

O projeto em questão consiste na construção de um agente de inteligência artifical com o objetivo de completar o jogo “*Bomberman*”.

O funcionamento deste assenta maioritariamente em 3 ficheiros:

- student.py
- bomb.py
- path.py

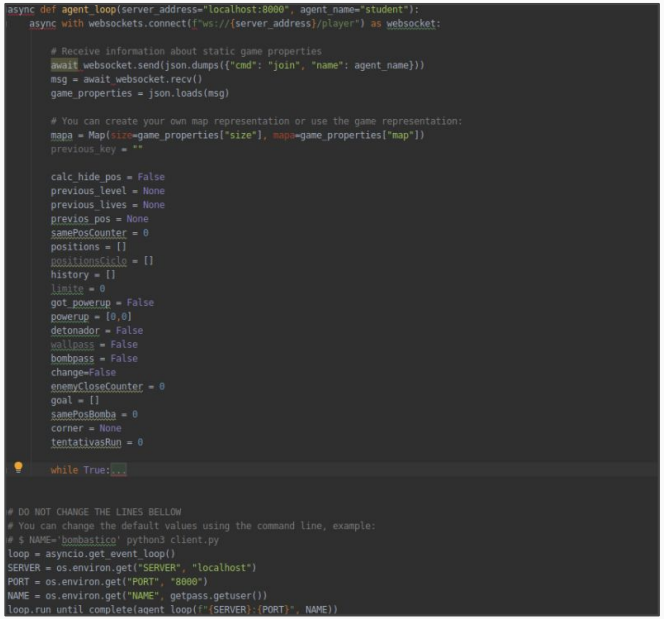
sendo o primeiro responsável por enviar dados ao servidor, o segundo por armazenar funções auxiliares, e o terceiro para métodos de fuga da bomba.

student.py

O posicionamento de bombas é efetuado aquando da **proximidade imediata** de uma **parede** através de pesquisa, ou através da **proximidade (5 posições)** de um **inimigo**.

O algoritmo de fuga ocorre **apenas** ao encontrar uma bomba, visto que estar perto de um inimigo resulta sempre no posicionamento de outra, e a fuga é feita tendo em conta o posicionamento de ambos

A estratégia para completar o jogo consiste em, quando possível, tentar eliminar os inimigos antes das paredes, de forma a que estes possuam menos caminhos possíveis de fuga.



```
async def agent_loop(server_address="localhost:8000", agent_name="student"):
    async with websockets.connect(f"ws://{server_address}/player") as websocket:

        # Receive information about static game properties
        await websocket.send(json.dumps({"cmd": "join", "name": agent_name}))
        msg = await websocket.recv()
        game_properties = json.loads(msg)

        # You can create your own map representation or use the game representation:
        mapa = Map(size=game_properties["size"], mapa=game_properties["map"])
        previous_key = ""

        calc_hide_pos = False
        previous_level = None
        previous_lives = None
        previos_pos = None
        samePosCounter = 0
        positions = []
        positionsCiclo = []
        history = []
        limite = 0
        got_powerup = False
        powerup = [0,0]
        detonador = False
        wallpass = False
        bombpass = False
        change=False
        enemyCloseCounter = 0
        goal = []
        samePosBomba = 0
        corner = None
        tentativasRun = 0

        while True:...


# DO NOT CHANGE THE LINES BELLOW
# You can change the default values using the command line, example:
# $ NAME='bombastico' python3 client.py
loop = asyncio.get_event_loop()
SERVER = os.environ.get("SERVER", "localhost")
PORT = os.environ.get("PORT", "8000")
NAME = os.environ.get("NAME", getpass.getuser())
loop.run_until_complete(agent_loop(f"{SERVER}:{PORT}", NAME))
```

path.py

O ficheiro *path.py* contém funções auxiliares de cálculo de posições livres (determinadas ou aleatórias), e funções de fuga.

choose_random_move	choose_move	choose_key	findPath	pathToEnemy
Escolha aleatória da próxima direção a mover	Escolha da próxima direção do Bomberman com base no custo até ao objetivo	Escolha da próxima direção com base no custo dos objetivos, utilizando o algoritmo A*, já tendo o caminho completo	Cálculo de caminho até a um determinado objetivo utilizando A*, e armazenamento do mesmo	Devolução imediata do caminho para um inimigo utilizando A*

path.py

```python
# retorna a key para um inimigo ou '' caso nao encontre
def pathToEnemy(mapa, my_pos, enemy_pos):
    # procura caminho para inimigo
    if enemy_pos is not None:
        # procura caminho para o inimigo
        positions = astar(mapa.map, my_pos, enemy_pos, mapa, False)
        print('positions to enemie: ' + str(positions))

        if positions != [] and positions is not None:
            if len(positions) == 1:
                return 'B'

            # se a posicao seguinte for igual à posicao atual
            # tira essa posição da lista
            while dist_to(my_pos, positions[0]) == 0:
                print('my_pos == next_pos')
                print('apagar posições inuteis')
                print('positions' + str(positions))
                positions.pop(0)

            return goToPosition(my_pos, positions[0])

        # nao encontrou caminho
        return ''
```

Tirando a deslocação para uma posição segura após posicionamento da bomba, todos os restantes movimentos são calculados através de um algoritmo A*. Contudo, e devido a imperfeições no mesmo, um possível bloqueio na resolução do nível (resultante em imobilizações) é resolvido com a alteração do objetivo a encontrar.

O algoritmo foi baseado no encontrado em https://medium.com/@nicholas.w.swift, e possui um limite de abertura de 300 nós, tentando maximizar a probabilidade de encontrar soluções, sem comprometer a eficiência do programa.