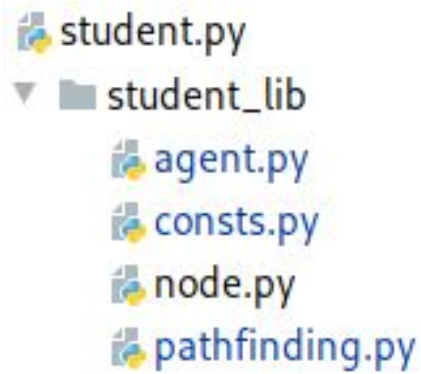
Desenvolvimento de um Agente autónomo para o jogo **Bomberman**

Relatório

Trabalho Prático de Grupo
Introdução à Inteligência Artificial
2019/2020

Rodrigo Rosmaninho - 88802
Daniel Correia - 88753

Organização do código fonte



- **student.py**
  Comunicação com o servidor, Inicialização do Agent a cada novo nível
- **agent.py**
  Lógica principal do agente (determinação do estado, lógica dos diferentes estados)
- **consts.py**
  Constantes e Enumerados
- **node.py**
  Estrutura de Dados para os nós da pesquisa
- **pathfinding.py**
  Funções de pesquisa (A*, fuga de bombas, cálculo de distâncias, cálculo de posição ótima para matar inimigos Smart.LOW, etc…)

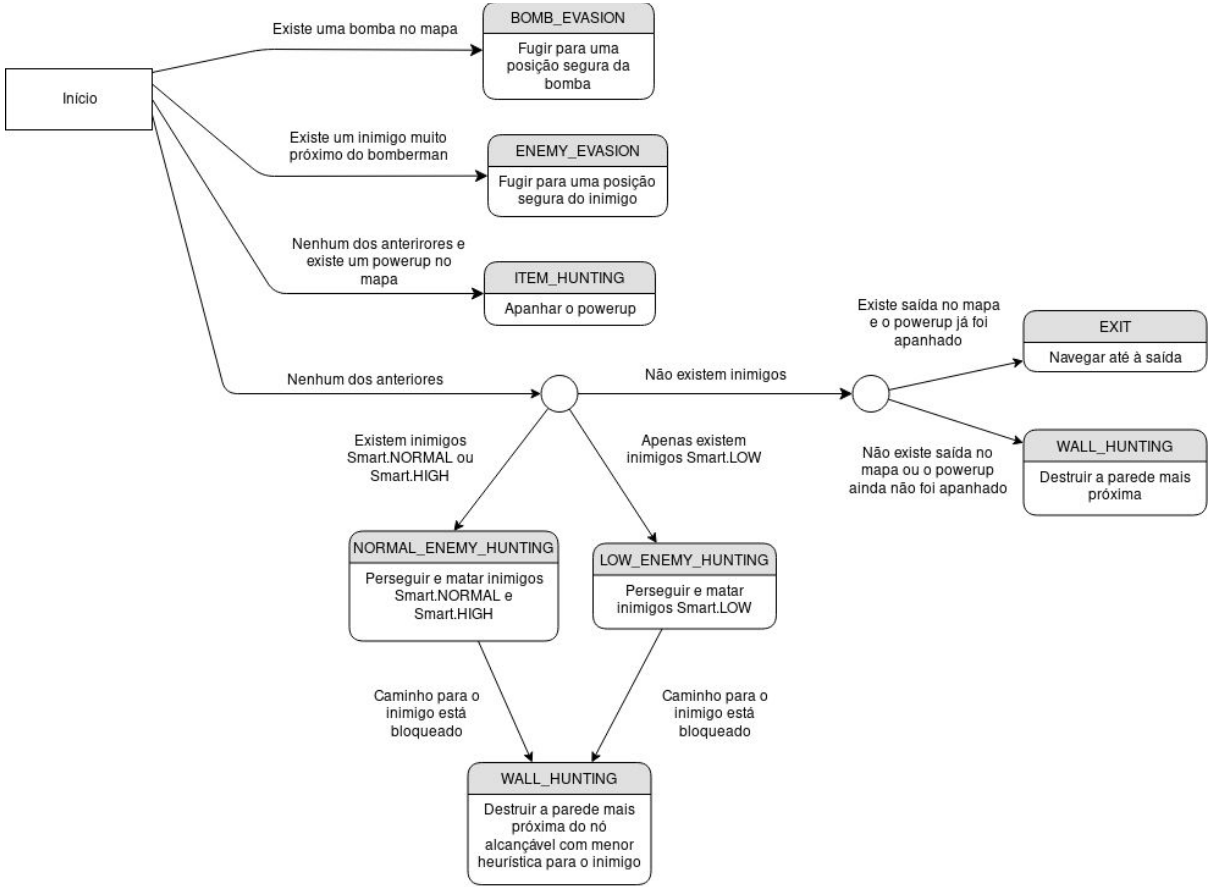
Repositório

Algoritmo

O diagrama representa uma visão simplificada do algoritmo de determinação do estado do agente. É executado sempre que um *state* é recebido.

De seguida, é executado o método do estado determinado.

O caminho para o objetivo atual é sempre recalculado nas iterações seguintes, de forma a refletir mudanças na posição dos adversários.



Pathfinding

A pesquisa de caminhos é realizada através do algoritmo *A-star (A*)* utilizando a Distância de Manhattan ao objetivo como heurística.
A lista de nós abertos é guardada numa Min PriorityQueue através da estrutura de dados *heapq*. Desta forma os nós são inseridos por ordem crescente da soma do seu custo e heurística.

Ao construir um caminho, são apenas utilizados os nós que não estejam ocupados por paredes ou inimigos, nem se encontrem num raio de duas células de um dado inimigo *Smart.HIGH* ou *Smart.NORMAL*.
No início do algoritmo do agente é compilado um dicionário com as direções atuais de todos os adversários *Smart.LOW*, o que permite ao algoritmo de pesquisa utilizar todos os nós que estejam dentro do raio do inimigo mas nas restantes direções.

De forma a evitar a expansão de demasiados nós quando não existe caminho possível para o objetivo, este processo está limitado a 400 nós.
No caso deste limite ser excedido assume-se que foi impossível encontrar um caminho e de entre os nós expandidos durante o processo é devolvido o nó com menor heurística para o objetivo. Desta forma, o agente pode calcular as paredes mais próximas desse nó e proceder à sua destruição para desbloquear o caminho de forma rápida.

Inimigos

Os inimigos são divididos em dois tipos, aos quais são aplicadas estratégias distintas.

- **Normal Enemies**

Os inimigos com inteligência igual a *Smart.HIGH* ou *Smart.NORMAL* distanciam-se do bomberman, logo não são necessárias tantas precauções para evitar ir de encontro a estes e morrer.

Estes adversários são eliminados antes dos restantes, de forma a manter uma densidade suficiente de paredes destrutíveis no mapa para possibilitar que estes fiquem encurralados.

A estratégia consiste em navegar até à posição do inimigo até ser possível colocar uma bomba cujo raio o inclua. É, no entanto, mantida uma distância de segurança de duas células.

- **Low Enemies**

Os inimigos com inteligência igual a *Smart.LOW* constituem um perigo maior já que, ao contrário dos restantes, não se distanciam do agente.

A forma mais eficiente de eliminar estes adversários é colocar uma bomba no caminho que estes estão a tomar. Para tal, é utilizado o dicionário das direções atuais deste tipo de inimigos (atualizado sempre que o algoritmo inicia) para calcular a posição que este irá ocupar a uma distância de 5 células da sua posição atual. Eventuais mudanças de direção por existência de um obstáculo também são levadas em conta neste cálculo.

Observações adicionais

A fuga de bombas e inimigos consiste em calcular todos os caminhos cujo fim é uma posição segura e com custo inferior a uma dada constante. A lista resultante é filtrada para incluir apenas os caminhos de menor custo e é retornado um aleatoriamente. Assim é possível evitar que o agente fique sincronizado com o movimento do inimigo e não o consiga matar.
Antes de colocar qualquer bomba o agente simula uma fuga para determinar se há caminhos seguros. Em caso de não haver, a bomba não é colocada.

Se o agente estiver a perseguir o mesmo adversário durante mais do que 100 steps, abandona temporariamente essa tarefa para destruir a parede mais próxima. Desta forma há a possibilidade de desbloquear um novo caminho. Se este processo ocorrer mais do que 4 vezes, o agente destrói a parede mais longínqua e recalcula qual o inimigo alvo.

Se um inimigo estiver dentro do raio especificado, o agente coloca bombas mesmo que exista uma parede destrutível entre eles, visto que o raio da bomba não é afetado por este tipo de paredes.

Todos os powerups descobertos são capturados. No entanto, o agente apenas procura ativamente os powerups *Flames*, *Speed*, *Detonator*, e *Wallpass*.

Ocasionalmente, ao tentar matar um inimigo *Smart.LOW*, o agente encurrala-se entre o inimigo e paredes.