Algoritmo A* no ambiente Agent0

André Sousa (20182451@uac.pt), Bruno Viveiros (20182149@uac.pt), Gonçalo Almeida (20182448@uac.pt)

Resumo

O objetivo deste projeto foi o de implementar o algoritmo de pesquisa A* sobre o código base do Agent0, usando a distância de Manhattan como heurística. Foram desenvolvidas duas diferentes instâncias – uma em que o agente conhece todos os obstáculos presentes no ambiente, outra em que são colocados obstáculos "invisíveis" – na segunda, ao encontrar o obstáculo, o agente pára e re-mapea o ambiente, voltando a precorrer, desta vez, o caminho re-calculado.

Introdução

Para o projeto, foi utilizado o Agent0_ver2, o qual foi fornecido pelo professor. Este programa consiste num "mapa" criado no código (Python) e num "agente" que possui, à priori, diversas funcionalidades de interação com o ambiente. Toda a interação é feita num ambiente Cliente/Servidor.
O Agent0 já nos foi providenciado com a estutura base, preparada para que fossem criados e testados diversos algoritmos de pesquisa, como por exemplo a "Pesquisa em Largura" e a "Pesquisa em Profundidade", experiências as quais nos ajudaram a melhor compreender e interpretar as diversas vertentes do complexo tema que é a Inteligência artificial.
Finalmente, e de acordo com os requerimentos do projeto, foi implementado o algoritmo de pesquisa A*.

GitHub

https://github.com/brun9/IA_Agent0_ASTAR

Descrição do Algoritmo ou Algoritmos e outro contexto de aplicação

O algoritmo de pesquisa utilizado, foi o A* (A-star), conhecido por ser o algoritmo de pesquisa "best-first" mais utilizado. É parecido com o algoritmo de custos uniformes, mas com capacidade tomar decisões mais bem informadas.
O algoritmo pode ser ótimo, caso se garanta que a heurística utilizada h(n) seja admissível e consistente, sendo que a função [f(n) = g(n) + h(n)], avalia o nó combinado g(n) e o custo para atingir o nó h(n), de modo a calcular o custo para atingir o objetivo f(n), escolhendo sempre o nó com menor custo total.
Caso o algoritmo seja ótimo, f(n) é sempre crescente, e o mesmo só pára a sua execuçao, caso o nó a ser expandido seja o nó objetivo.

Referências bibliográficas:

https://en.wikipedia.org/wiki/A*_search_algorithm

Experiências Realizadas ou Exemplos de Aplicação

Nota: Foram utilizadas as funções **markVisited()**, **markFrontier()** e **markMove()** para marcar as posições durante o mapeamento e cáculo do percurso. **markVisited()** marca os nós já visitados a **Verde, markFrontier()** marca a **Vermelho** os nós fronteira em relação aos nós já visitados, e finalmente, a **Azul**, e utilizando a função **markMove()** são marcados os nós presentes no percurso traçado pelo algoritmo.

Situação 1: Mapear o mapa de modo a calcular todos os caminhos possíveis até ao objetivo, obtendo assim a solução ótima (caminho mais curto).

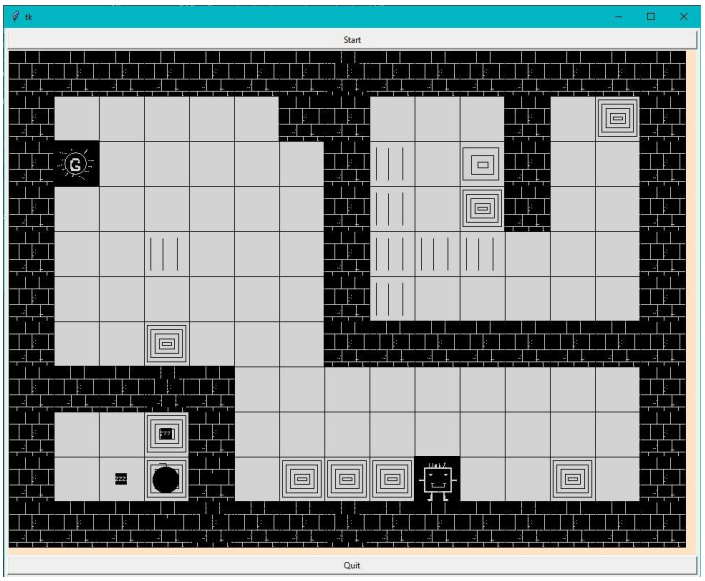

Figura 1- Ambiente s/ obstáculos invisíveis

1ª etapa: Utilizar o algoritmo A* para percorrer os nós e calcular o melhor percurso até ao objetivo (Figura 2).

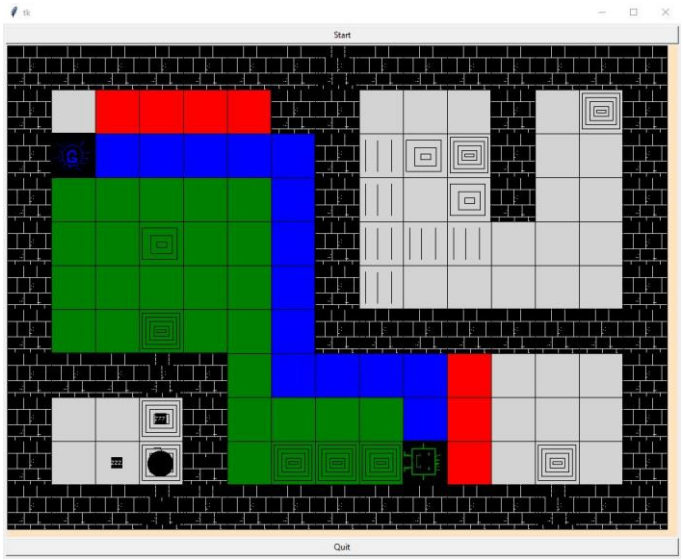

Figura 2 - Cálculo do percurso ótimo

2ª etapa: Percorrer o caminho calculado até ao objetivo (Figura 3).

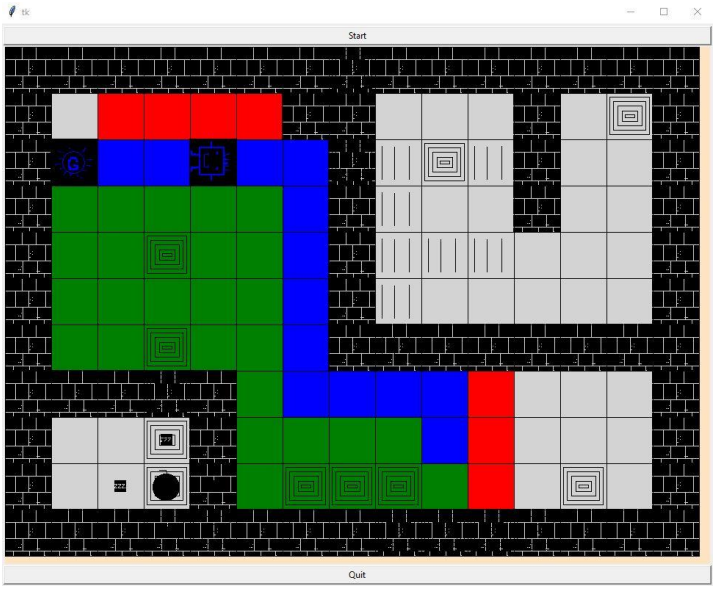

Figura 3 - O agente percorre o percurso ótimo

Situação 2: Colocar obstáculos "invisíveis" no mapa – os obstáculos "normais" já são conhecidos no início, é calculado o percurso mais curto até ao objetivo, utilizando o algoritmo A*, e o agente então dá inicio à sua deslocação – caso embata de frente num dos obstáculos "invisíveis", adiciona-o à sua estrutura que armazena os obstáculos, re-calculando então um novo percurso e dando início à sua deslocação de novo.

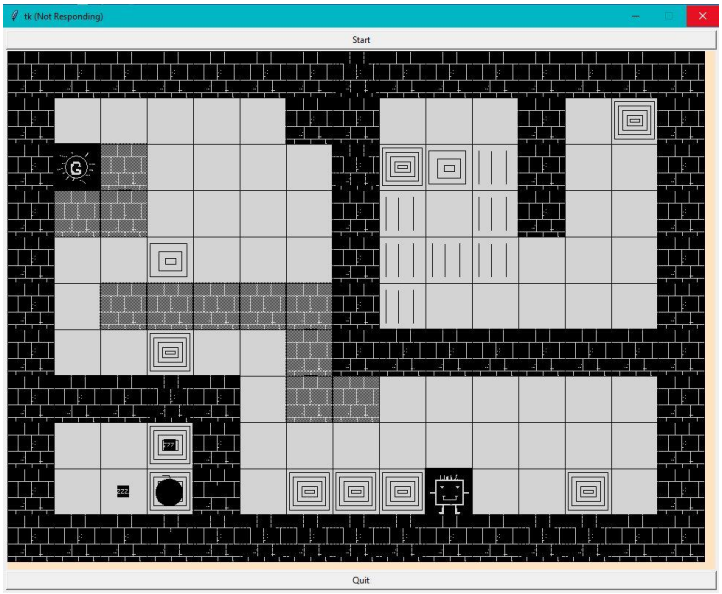

Figura 4 - Ambiente c/ obstáculos invisíveis

1ª etapa: Utilizar o algoritmo A* para percorrer os nós e calcular o melhor percurso até ao objetivo (Figura 5).

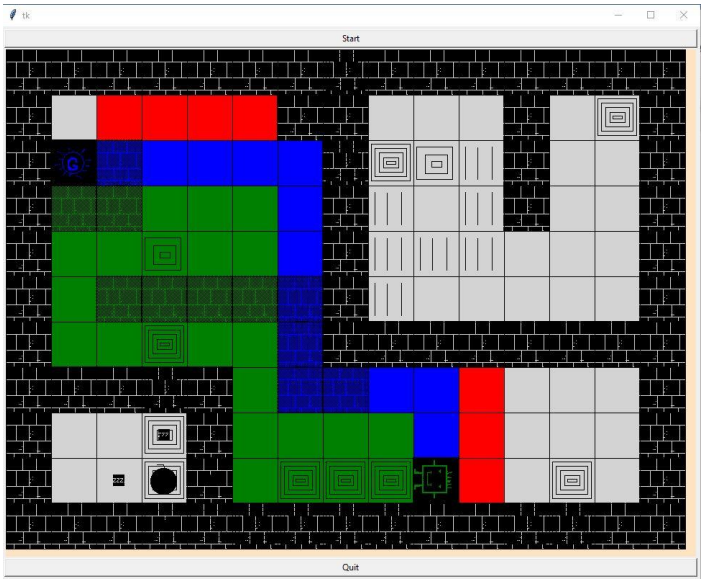

Figura 5 - Cálculo do percurso ótimo

2ª etapa: Percorrer o caminho calculado até ao objetivo. Neste cenário, ao embater de frente num dos obstáculos invisíveis (Figura 6), o agente pára, o obstáculo é então adicionado à lista de obstáculos na memória do agente. O algoritmo A* é então utilizado de novo para que o novo percurso seja calculado.

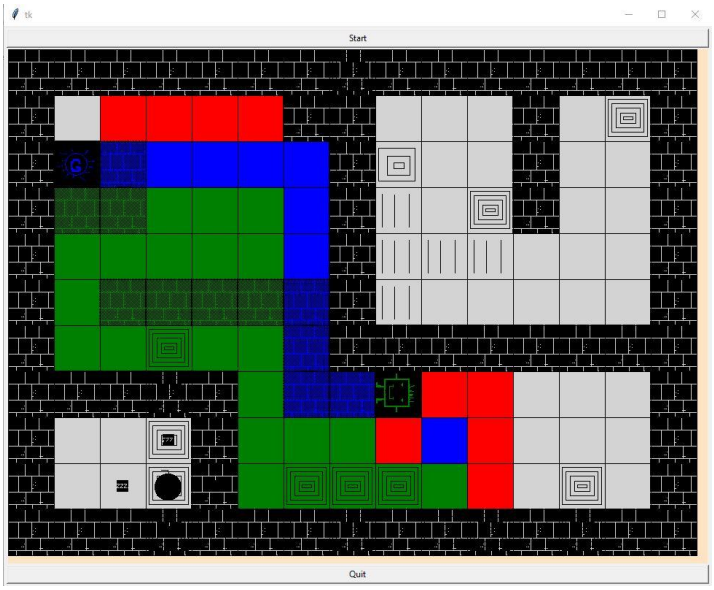

Figura 6 - Embate num dos objs. invisíveis

3ª etapa: Após terem sido adicionados todos os obstáculos “invisíveis” presentes no caminho até ao objetivo, o agente consegue obter a informação necessária para o conseguir atingir (Figura 7).

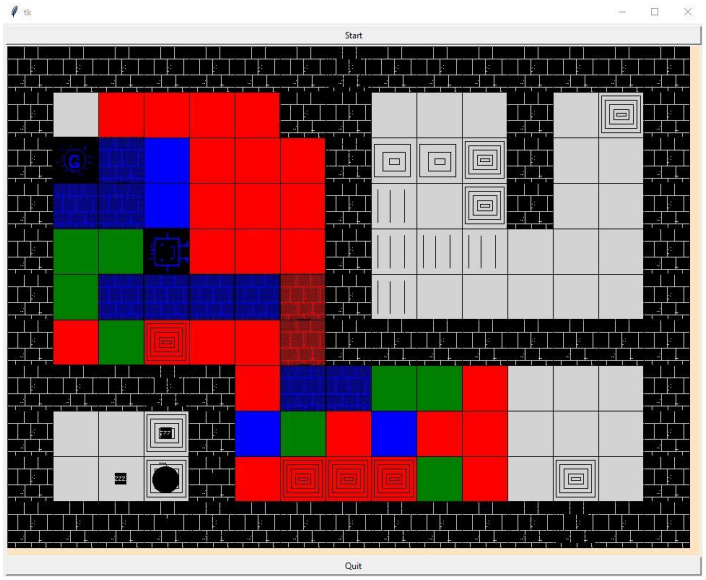

Figura 7 - Percurso final obtido

Situação 3: Circundar a posição em que o agente é iniciado por obstáculo “invisíveis”, de modo a analisar o comportamento do mesmo.

1ª etapa: Mapear os obstáculos – Nesta fase, o agente não conhece os obstáculos invisíveis, exemplificados na (Figura 8) como os obstáculos cinzentos.

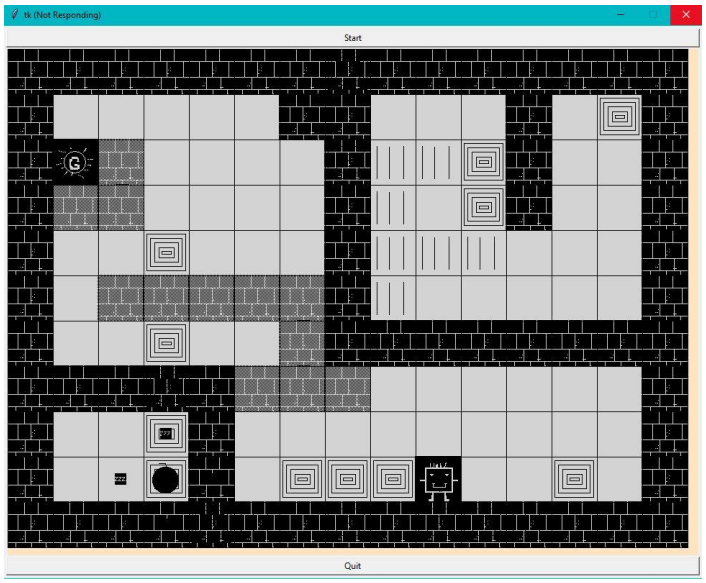

Figura 8 - Mapa com agente "cercado"

2ª etapa: Como na situação anterior, o agente calcula o melhor percurso até ao objetivo, inconsciente da presença de obstáculos pelo caminho (Figura 9).

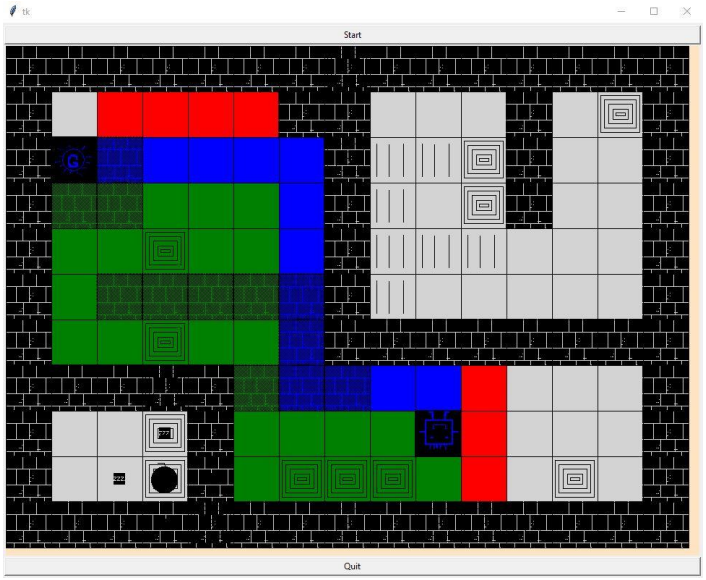

Figura 9 - Cálculo do percurso ótimo, s/ conhecimento de obstáculos presentes

3ª etapa: O agente embate nos 3 primeiros obstáculos “invisíveis presentes em cada ocasião da execução do algoritmo A* (Figura 10).

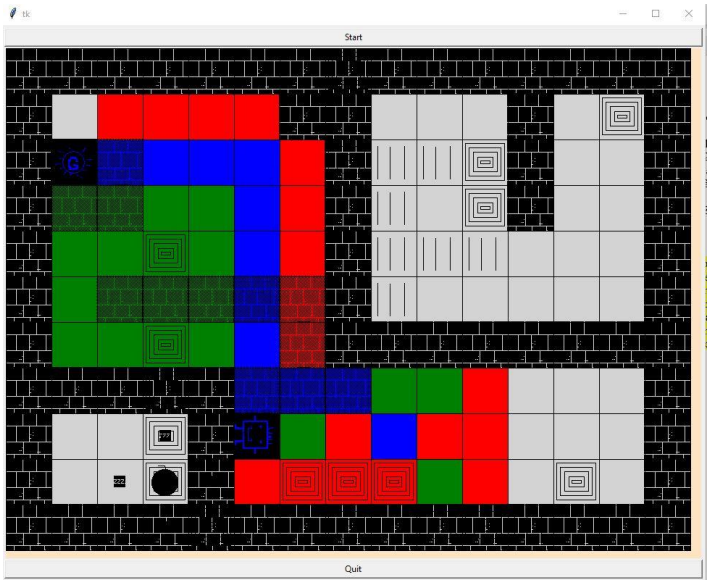

Figura 10 - Agente posicionado na posição "mais perto" do objetivo

4ª etapa: O agente então fica encostado à zona Oeste do mapa (Figura 11) – para onde está presente o objetivo – mapeando, no entanto, a Este da sua posição, pois é conhecedor da presença de obstáculos a Oeste. Após alguns minutos de execução, o programa termina a sua execução, e o agente não consegue alcançar o objetivo.

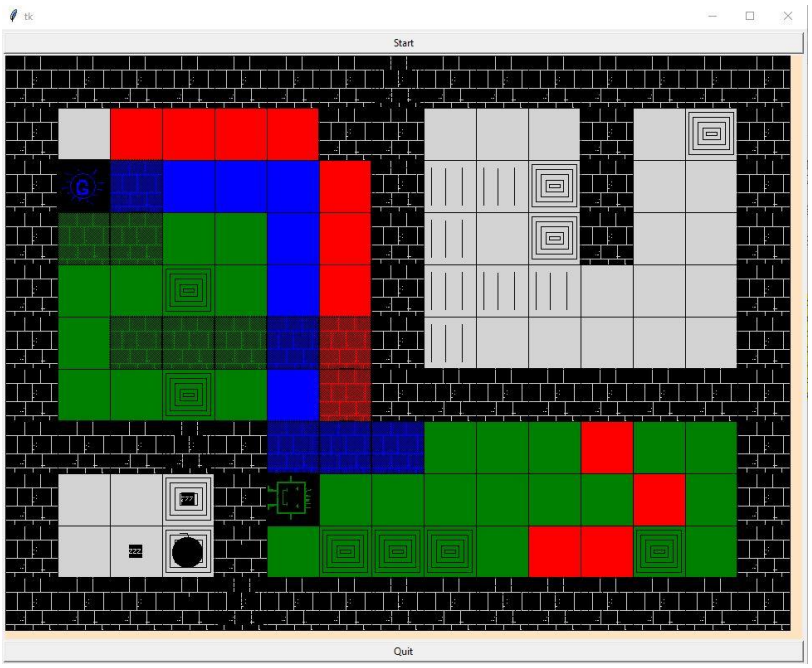

Figura 11 - O agente não tem para onde ir

Discussão e Conclusão

De acordo com as simulações efetuadas para as três diferentes situações especificadas, foram criados três problemas com diferentes níveis de complexidade, tanto no espaço, como no tempo. Na primeira situação, como o agente já conhece à partida a posição, tanto do objetivo, como dos obstáculos, tem uma tarefa considerada um pouco mais "facilitada", sendo esta uma situação de menor complexidade tanto no espaço, como no tempo, quando comparada com as que se seguiram. Na segunda situação, dado que são colocados obstáculos, os quais o agente desconhece à partida, a complexidade do problema é incrementada, principalmente em termos de tempo, pois o algoritmo A* tem de ser executado a cada novo obstáculo encontrado pelo agente.
Na terceira e última situação, é feito um enclausuramento do agente num dos "cantos" do ambiente, impossibilitando assim, qualquer cenário em consiga atingir o objetivo. Neste caso, ambas as complexidades são incrementadas, pois são "abertos" mais nós do que na segunda situação, o que por sua vez implica um maior tempo de execução, tornando-se, por fim, redundante contínuar a procurar.

Bibliografia

Artificial Intelligence -A Modern Approach (3rd Edition) – Stuart Russell, Peter Norvig

Heuristics - http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html

Ambiente de simulação - Agent0 - http://github.com/josecascalho/Agent0_ver2

Material de apoio da disciplina de Inteligência Artificial – Prof. José Cascalho, UAC