Mummy Maze

Martim Silva
Nº 2200681

Samuel Martins
Nº 2200674

RESUMO

O presente relatório tem como objetivo apresentar a implementação do jogo Mummy Maze realizado no âmbito de unidade curricular de Inteligência Artificial.

Ao longo deste relatório pretende-se apresentar uma breve explicação sobre os nossos métodos de implementação, o que foi implementado, a contribuição de cada membro e todos os extras desenvolvidos.

ABSTRACT

This report aims to present the implementation of the Mummy Maze game carried out within the scope of the Artificial Intelligence curricular unit.

Throughout this report is intended to present a brief explanation about our methods of implementation, what was implemented, the contribution of each member and all the developed extras.

1. INTRODUÇÃO

1.1 Mummy Maze

Lançado em 2002, o jogo *Mummy Maze* é um jogo de estratégia por turnos cujo objetivo é levar o herói até à saída do nível, sem que este seja apanhado e morto pelos perigos presentes (inimigos e/ou armadilhas).

Sendo este um jogo por turnos, começa-se por mover o herói e de seguida movem-se os inimigos.

1.2 Objetivo do projeto

Recorrendo aos algoritmos de procura (informados e não informados) lecionados nas aulas, este projeto tem como objetivo jogar o jogo Mummy Maze de forma automática, para que seja possível realizar um estudo comparativo do desempenho dos vários algoritmos e heurísticas implementadas no projeto.

2. REPRESENTAÇÃO DOS ESTADOS

2.1 Níveis

Na base dos diferentes níveis disponíveis está uma matriz 13x13, lida a partir de um ficheiro de texto (.txt), que contém as diferentes entidades e estados. Estes elementos são representados por caracteres que constituem o nível.

2.2 Entidades

As entidades correspondem aos elementos posicionados nas células principais (chão) da matriz. Estes elementos podem-se mover de célula a célula se existirem condições para o fazer.

Tabela 1. Entidades e seus identificadores

Nome	Identificador	Caractere representativo
Herói	HERO	H
Chão	FLOOR	.
Saída	EXIT	S
Múmia Branca	WHITE_MUMMY	M
Múmia Vermelha	RED_MUMMY	V
Escorpião	SCORPION	E
Armadilha	TRAP	A
Chave	KEY	C

2.3 Estados Secundários

Estes estados são aqueles que se encontram entre as posições das entidades e representam elementos que não se movem. No entanto, alguns podem sofrer alterações (como é o caso das portas).

Tabela 2. Estados secundários e seus identificadores

Nome	Identificador	Caractere representativo
Porta Vertical aberta	VERT_DOOR_OPEN	)
Porta Vertical fechada	VERT_DOOR_CLOSED	“
Porta Horizontal aberta	HORIZ_DOOR_OPEN	_
Porta Horizontal fechada	HORIZ_DOOR_CLOSED	=
Parede Vertical	VERT_WALL	|
Parede Horizontal	HORIZ_WALL	-

3. ALGORITMOS DE PROCURA

Neste projeto foram implementados os algoritmos de procura estudados nas aulas práticas.

Algoritmos de procura não informados:

- Procura em Largura (*Breadth First Search*);
- Procura Uniforme (*Uniform Cost Search*);
- Procura em Profundidade (*Depth First Search*);
- Procura em Profundidade (*Depth Limited Search*);
- Procura por Aprofundamento Progressivo (*Iterative Deepening Search*).

Algoritmo de procura informados:

- Procura Sôfrega (*Greedy Best First Search*);
- Procura em Feixe (*Beam Search*);

- A* (*AStarSearch*);
- IDA* (*IDAStarSearch*).

4. IMPLEMENTAÇÃO

Ao longo do projeto, foram tomadas várias decisões que não constavam no enunciado do projeto. Neste capítulo são descritas essas decisões.

4.1 Classes Cell e Agent

Para uma melhor gestão dos elementos do jogo, foram criadas duas novas classes: *Cell* e *Agent*.

A classe *Cell* é responsável por: armazenar o caracter que representa o elemento e a posição do mesmo na matriz (linha e coluna).

A classe *Agent* foi desenvolvida para ser utilizada na criação do herói e dos inimigos. Esta classe estende da classe *Cell* e conta com uma variável booleana para verificar se o herói (ou inimigo) está vivo. Esta variável é utilizada mais tarde para remover inimigos das suas listas caso, tenham morrido em lutas.

4.2 Movimentação dos inimigos e herói

Para a ordem de movimentação dos inimigos, optámos por: primeiro moverem-se as múmias brancas, seguido das múmias vermelhas e por fim os escorpiões.

Por cada movimentação, é verificado se existem chaves ou armadilhas que necessitam de ser repostas na matriz, caso a *Cell* não esteja a ser ocupada por um inimigo ou herói. Outra verificação que é feita após cada movimentação, é a verificação se o inimigo ou herói está em cima de uma chave, para assim proceder à mudança de estado (abrir ou fechar) de todas as portas do nível.

Por fim, depois do herói e todos os inimigos terminarem os seus turnos, é removido das listas de inimigos, todos aqueles que perderam lutas.

4.3 Luta entre inimigos

Não havendo regras explicitas no enunciado, decidiu-se que cada inimigo pode matar qualquer outro tipo de inimigo. Para além disso, o sistema de luta foi implementado de forma a matar o inimigo que se está a mover para cima de outro inimigo.

5. HEURÍSTICAS

5.1 Distância ao objetivo

Esta primeira heurística avalia a distância a que o herói se encontra da saída. Sendo que, quanto mais próximo o herói se encontrar da saída, melhor o resultado.

A implementação da heurística consiste na soma da diferença entre a linha do herói com a linha da saída na matriz e da coluna do herói com a coluna da saída na matriz.

5.2 Distância aos inimigos

Esta heurística calcula a distância entre o herói e o inimigo mais próximo. Para esta heurística, foi tomada em consideração que quanto mais perto o inimigo estiver do herói, pior o resultado.

A implementação da heurística consiste em calcular a distância do inimigo mais próximo do herói. Para isso, são percorridas todas as listas de inimigos, armazenando a menor distância encontrada numa variável auxiliar. Finalmente, é subtraído da distância máxima possível na matriz, o valor armazenado na variável auxiliar. Caso não existam inimigos, ou o herói estiver na saída, a heurística devolve zero.

5.3 Distância ao objetivo e aos inimigos

Esta heurística consiste na soma das duas heurísticas mencionadas anteriormente. Assim, com o reaproveitamento destas heurísticas, é possível avaliar a distância ao objetivo e a distância aos inimigos simultaneamente.

6. RESULTADOS OBTIDOS

Testando os níveis isoladamente, através da funcionalidade extra de testes, criaram-se gráficos com base nos resultados obtidos nos ficheiros Excel. Foi possível efetuar estes testes em todos os níveis fornecidos pelos docentes à exceção do nível “nivel21_v1”, onde não foi obtida solução.

Ao longo deste capítulo serão abordados os diferentes resultados obtidos. Por isso, recomenda-se uma análise dos gráficos presentes na secção dos anexos.

6.1 Análise da média de custos

Analisando os resultados dos custos, consegue-se chegar à conclusão de que os algoritmos testados não aparentam grande discrepância nos valores de custos das soluções encontradas, com exceção do algoritmo não informado *Depth First Search* (Procura em Profundidade), que apresentou valores maiores (Figura 4). Esta análise permite concluir quais são os algoritmos que, em média, encontram soluções com menos movimentações do herói até chegar à saída.

6.2 Análise do número de estados gerados

No que diz respeito à média da quantidade de estados gerados por algoritmo, verificou-se que os algoritmos *Depth First Search* (Procura em Profundidade), *IDA** e o *Iterative Deepening Search* (Procura por Aprofundamento Progressivo) apresentam médias bastante superiores em comparação aos restantes algoritmos (Figura 5).

Note-se que, no gráfico do estudo dos estados gerados não está incluído o *Iterative Deepening Search* devido a este apresentar uma média muito elevada de estados gerados (18783190 estados).

6.3 Desempenho (tempo de execução)

Em termos do desempenho e processamento dos algoritmos, é possível concluir que o algoritmo não informado *Iterative Deepening Search* (Procura por Aprofundamento Progressivo) é o algoritmo mais oneroso, demorando mais tempo a ser processado e testado. No entanto este algoritmo tem a capacidade de retornar, em média, boas soluções. No que toca ao tempo de execução, os restantes algoritmos são relativamente semelhantes, sendo que as maiores diferenças estão nos custos das soluções encontradas.

Por fim, é importante salientar que a quantidade de estados gerados e percorridos está diretamente relacionada com o tempo de execução de cada algoritmo.

7. CONTRIBUIÇÃO DE CADA ELEMENTO

7.1 Martim Silva

Movimentação das múmias

Lutas entre inimigos

Implementação das portas

7.2 Samuel Martins

Movimentação dos escorpiões

Reposição de elementos na matriz (chaves e armadilhas)

7.3 Em conjunto

Leitura do estado inicial do nível

Heurísticas

Algoritmo IDA*

Otimização dos algoritmos: *A** e *GreedyBestFirstSearch*

Teste automático de níveis

Relatório

8. EXTRAS

8.1 Algoritmo IDA* (*IDAStarSearch*)

Dado que este algoritmo não foi implementado nas aulas, e de forma a completar o estudo, foi decidido desenvolver o algoritmo *IDA** (*IDAStarSearch*).

8.2 Melhoria de alguns algoritmos

Foram feitas algumas alterações aos algoritmos *A** e *GreedyBestFirstSearch*, tornando-os mais eficientes.

8.3 Teste de um nível

De modo a simplificar o estudo do desempenho dos algoritmos, desenvolveu-se uma funcionalidade que possibilita testar um nível de forma automática. Esta funcionalidade, ao carregar num botão, permite testar o nível selecionado com todos os métodos de pesquisa e heurísticas (no caso dos algoritmos de pesquisa informados), guardando os resultados num ficheiro Excel (.csv) na diretoria "tests", também gerada automaticamente, com todas as estatísticas relevantes ao estudo.

9. ANEXOS

Level	Search Algorithm	Heuristic	Solution F	Solution Cost	Num of Expanded Nodes	Max Frontier Size	Num of Generated States
nivel1.txt	Breadth first search	null	Yes	11.0	64	13	172
nivel1.txt	Uniform cost search	null	Yes	11.0	68	12	183
nivel1.txt	Depth first search	null	Yes	15.0	347	29	511
nivel1.txt	Iterative deepening search	null	Yes	11.0	4754	14	9951
nivel1.txt	Greedy best first search	Distance to exit position	Yes	11.0	22	30	82
nivel1.txt	Greedy best first search	Distance to enemy position	Yes	12.0	23	13	88
nivel1.txt	Greedy best first search	Distance to exit and enemy position	Yes	12.0	25	30	94
nivel1.txt	A* search	Distance to exit position	Yes	11.0	16	26	62
nivel1.txt	A* search	Distance to enemy position	Yes	12.0	45	26	165
nivel1.txt	A* search	Distance to exit and enemy position	Yes	11.0	23	28	84
nivel1.txt	IDA* search	Distance to exit position	Yes	11.0	16	9	62
nivel1.txt	IDA* search	Distance to enemy position	Yes	11.0	1345	24	5061
nivel1.txt	IDA* search	Distance to exit and enemy position	Yes	11.0	74	11	271

Figura 1. Resultado de um teste sobre o nível 1

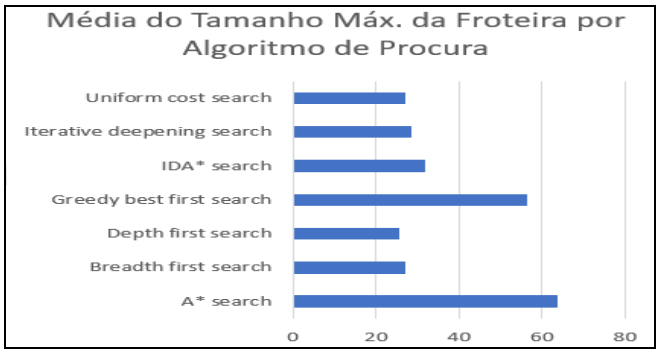

Figura 2. Média do tamanho da fronteira por algoritmo

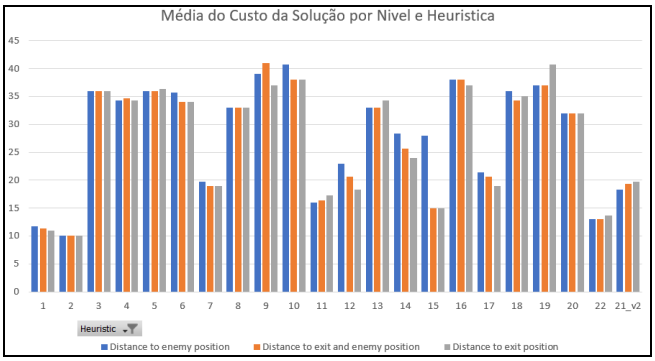

Figura 3. Média do custo da solução por nível e heurística

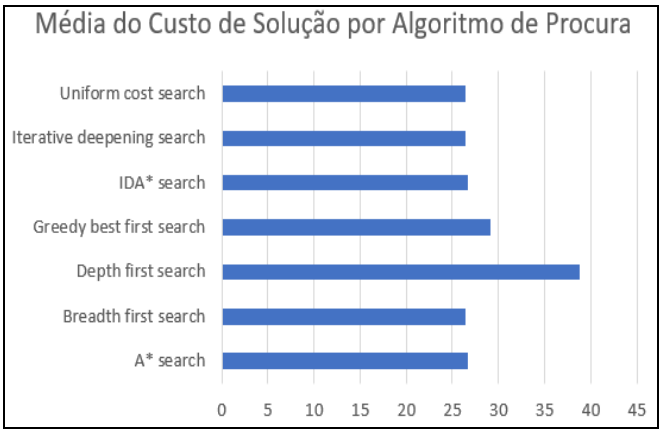

Figura 4. Média do custo de solução por algoritmo de procura

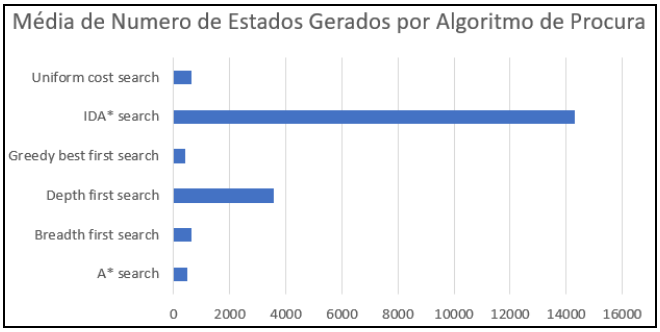

Figura 5. Média de número de estados gerados por algoritmo de procura