



**Universidade do Minho**

Conselho de Cursos de Engenharia

Mestrado em Engenharia Informática

## **Sistemas Inteligentes – Projeto Integrado**

Ano Letivo de 2011/2012

### **Caminho + Curto**

**Rafael Fernando Pereira Abreu**

pg20978@alunos.uminho.pt

Julho, 2012

Data de Recepção	
Responsável	
Avaliação	
Observações	

## **Caminho + Curto**

**Rafael Fernando Pereira Abreu**

pg20978@alunos.uminho.pt

Julho, 2012

## Resumo

Este relatório tem como objetivo expor o resultado do projeto realizado durante todo o ano para a Unidade Curricular do Projeto Integrado de Sistemas Inteligentes, sobre a simulação de um ambiente / terreno real, com o objetivo a determinar o caminho mais curto entre dois ou mais pontos utilizando para isso o algoritmo A\* (*A Star*). Este caminho mais curto é limitado consoante algumas restrições do ambiente em simulação.

**Área de Aplicação:** Sistemas Inteligentes

**Palavras-Chave:** Agentes, *A Star*, *Java*, *Swing*, *WADE*, *JADE*, *UML* e *AUML*.

# Índice

1. Introdução	1
1.1. Contextualização	1
1.2. Motivação e Objetivos	1
1.3. Estrutura do Relatório	2
2. Especificação	3
2.1. Descrição do Problema	3
2.2. Algoritmo A*	4
2.2.1 Área de Procura	5
2.2.2 Início da Procura	5
2.2.3 Continuar Procura	7
2.3. Diagrama de Classes	7
2.4. Ferramentas utilizadas	9
3. Desenvolvimento	10
3.1. Ontologias	10
3.2. Agentes	11
3.2.1 <i>SuperVisor</i>	11
3.2.2 <i>Caminhante</i>	11
3.3. Aplicação Desenvolvida	12
3.4. Análise de Resultados	13
4. Conclusões e Trabalho Futuro	15

## Índice de Figuras

Ilustração 1 – Exemplo dum enxerto do ficheiro com o Mapa	4
Ilustração 2 - Diagrama de Classes	8
Ilustração 3 – Ontologias	10
Ilustração 4 – Interface	12

## Índice de Tabelas

Tabela 1 - Área de procura	5
Tabela 2 - Área de procura bidimensional ( <i>PathFinding</i> )	5
Tabela 3 - Começando a Procura	6
Tabela 4 - Caminho mais curto	7
Tabela 5 – Comparação entre o Mundo Real e a simulação	13
Tabela 6 - Exemplos das funcionalidades da aplicação	14

# 1. Introdução

No âmbito do módulo de Projeto Integrado (PI) da Unidade Curricular de Especialização (UCE) do Mestrado em Engenharia Informática, propôs-me a fazer um trabalho sobre o caminho mais curto.

Este projeto consiste em criar uma aplicação, que simule o máximo de aspetos de um ambiente real e que determine com a maior exatidão o caminho mais curto.

Com este relatório pretendo ajudar o leitor, neste caso o docente, a perceber qual foi a minha visão e implementação no que diz respeito às exigências e objetivos a que me propus.

## 1.1. Contextualização

A otimização do caminho mais curto tem sido cada vez mais estudado e aprofundado em várias áreas por várias razões, muitas delas estão relacionadas com o tempo, custo ou acessibilidade.

Nesta linha propus-me a realizar uma aplicação que represente o número de movimentos de um agente, neste caso de uma pessoa, percorrendo de forma sequencial um número finito de *checkpoints* (pelo menos dois), desviando-se de um conjunto de limitações / obstáculos impostas nesse terreno.

Neste contexto a utilização de agentes inteligentes é uma boa prática visto conseguir representar todo o problema em causa de forma simples e prática.

Para a utilização de agentes inteligentes foi utilizado a *Framework Java Agent Development* (JADE) que cria o ambiente de simulação perfeito para o problema.

## 1.2. Motivação e Objetivos

A grande motivação e objetivo deste projeto era conseguir simular de forma mais detalhada as restrições que nos são deparadas no nosso dia-a-dia quando pretendemos chegar a um determinado lugar.

Outra motivação passa pela utilização de agentes inteligentes para a resolução do problema em causa, visto ser uma das melhores, se não mesmo a melhor forma simular um ambiente real.

### 1.3. Estrutura do Relatório

Este relatório é composto por 4 capítulos. O primeiro capítulo denominado **Introdução** expõe o problema deste projeto, as minhas motivações e objetivos para o realizar. O capítulo seguinte designado **Especificação** onde é apresentado todas as informações para uma melhor assimilação dos detalhes do projeto. O capítulo seguinte é o **Desenvolvimento** que terá todo o trabalho desenvolvido e uma explicação dos resultados obtidos. Por fim, no último capítulo **Conclusões e Trabalho Futuro** é explicado a apreciação que faço do trabalho realizado.












## 2. Especificação

### 2.1. Descrição do Problema

Como se trata de um problema de caminho mais curto tenho que definir o que é uma trajetória. A trajetória do caminhante é um caminho entre pelo menos dois pontos num mapa / terreno definido por um algoritmo, no caso o algoritmo A\* (*A Star*). Além disso ainda é permitido escolher a velocidade que o caminhante anda, o terreno / mapa e o tipo de limitações do caminhante.

O terreno / mapa é um quadrado com 80 por 80 pixéis. Cada pixel define uma posição distinta no mapa. Cada posição pode ser definida uma cor distinta que identifica restrições:

-  Branco: representa uma posição do terreno / mapa em que todos os tipos de utilizadores podem andar livremente;
-  Amarelo: representa uma posição do mapa que pertencente há trajetória realizada pelo caminhante (tem de existir pelo menos dois pontos, início e fim);
-  Preto: representa uma parede / obstáculo que é impossível trespassar pelo caminhante;
-  Azul: representa uma posição do caminho / trajetória realizada pelo tipo de caminhante escolhido (*CaminhoRealizado*);
-  Azul claro: representa água (por exemplo: fontes, lagos, rios, mar...);
-  Cinzento: representa uma rampa ou ponte;
-  Magenta: representa escadas;
-  Vermelho: representa uma área restrita ao acesso a pessoas não autorizados;
-  Verde: representa uma zona verde (por exemplo: jardins, canteiros, árvores...).

O mapa é um documento “.txt” que no topo contém a posição (x,y) inicial e final da trajetória do caminhante, seguida do número de pontos intermédios e respetiva posição como ilustra a figura seguinte.

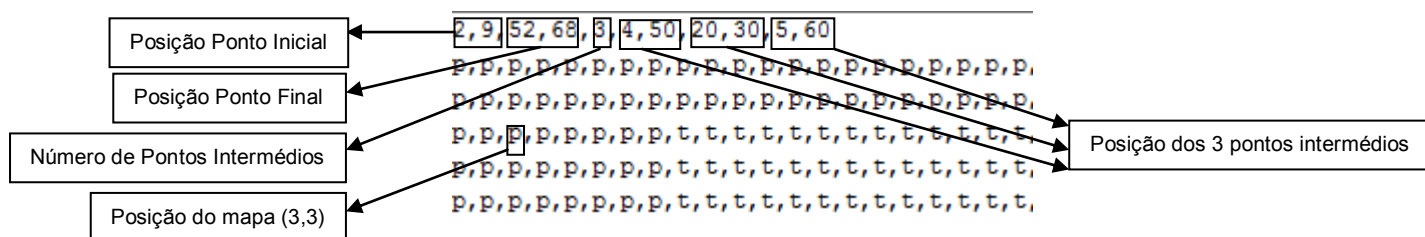


Ilustração 1 – Exemplo dum enxerto do ficheiro com o Mapa

A métrica usada para calcular a distância efetuada é o número de passos que o caminhante realizou ou seja o número de pixéis que ele percorreu desde o ponto inicial até ao final da trajetória.

Existem dois tipos de agentes os caminhantes têm um tipo e o *SuperVisor*. O primeiro é responsável por efetuar o algoritmo de caminho mais curto e enviar para o *SuperVisor*, que desenha e informa o número de passos efetuados.

Os caminhantes são divididos em quatro grupos, cada grupo com restrições diferentes:

- Pessoa sem acesso restrito: não têm acesso às zonas restritas;
- Pessoa com acesso restrito: têm acesso às zonas restritas;
- Deficiente motor com acesso restrito: pessoa com mais restrições de todas, não podem usar as escadas e não tem acesso às zonas de acesso restrito;
- Deficiente motor sem acesso restrito: não pode passar pelas escadas mas têm acesso às zonas de acesso restrito.

Os quatro grupos não podem passar pelas paredes, água, e zonas verdes, obrigando sempre a desviar-se dessas zonas.

## 2.2. Algoritmo A\*

Existem vários algoritmos de caminho mais curto (*Shortest Paths*), entre eles o A\* (lê-se A-Star) que de todos foi o que melhor satisfazia, de uma forma aceitável, o objetivo principal do projeto.

Em 1968 era descrito pela primeira vez por *Peter Hart, Nils Nilsson e Bertram Raphael* numa publicação. O algoritmo é uma combinação de heurísticas como o algoritmo *Best-first Search* e da formalidade do algoritmo *Dijkstra*.

Este algoritmo é muito usado em rotas (deslocações de um ponto A para um ponto B), resolução de quebra-cabeças e em jogos.

### 2.2.1 Área de Procura

O algoritmo A\* começa por descobrir o ponto inicial e final. Após identificar estes pontos identifica a área de procura no mapa.

Assumindo que queremos ir de um ponto A até um ponto B e o que separa esses dois pontos é uma parede (a preto) como ilustra a Tabela 1.

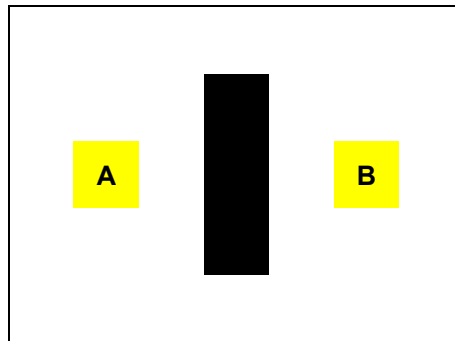


Tabela 1 - Área de procura

A primeira coisa que se faz é dividir a área de procura numa grade quadrada bidimensional (designado *PathFinding*), que simplifica a complexidade do problema (Tabela 2).

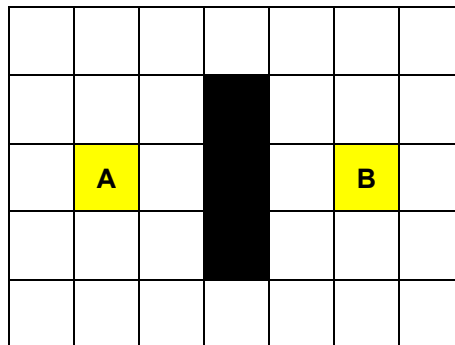


Tabela 2 - Área de procura bidimensional (*PathFinding*)

O caminho mais curto é achado encontrando os quadrados brancos que no levam do ponto A até ao ponto B. Após achar este caminho basta mover a nossa entidade A do centro do quadrado para o próximo e assim sucessivamente até o objetivo seja alcançado. Aos centros dos pontos é chamado Nós.

### 2.2.2 Início da Procura

Agora depois de simplificar a área de procura temos de passar para a procura:

- Como se pode ver na Tabela 3 cada quadrado a cinza faz parte dos quadrados pertencentes há lista aberta no qual o quadrado A é o seu pai.

	A			B		

Em seguida prosseguimos com o algoritmo escolhendo na lista aberta os quadrados com menor custo, adjacentes ao ponto A e repetimos o processo.

$$F = G + H$$

- **G** - é o custo do movimento para se mover do ponto de início até o quadrado determinado na malha seguindo o caminho criado para chegar lá (por exemplo: 10 para movimentos para quadrados horizontais e verticais e 14 para movimentos para os quadrados nas diagonais)
- **H** - é o custo estimado do movimento para mover daquele quadrado determinado até o destino final, ponto B. Isto é frequentemente referido como a heurística. Não é possível determinar a distância real devido a todas as restrições existentes (por exemplo: distância de *Manhattan*).

### 2.2.3 Continuar Procura

Ao continuar a procura escolhesse o quadrado que contém o valor de **F** mais baixo de todos os quadrados que estão na lista aberta. Depois de escolhido o quadrado:

1. Retira-se da lista aberta e acrescentasse à lista fechada.
2. Confirmam-se todos os quadrados adjacentes. Ignorando os que estão na lista fechada ou não passáveis (terreno com paredes, água, ou outra posição ilegal), acrescentam-se os quadrados à lista aberta, se eles já não estiverem na lista aberta. Acrescenta-se o quadrado atual como pai dos novos quadrados.
3. Se um quadrado adjacente já estiver na lista aberta, identifica-se o melhor caminho (**G** mais baixo).
4. Se o **G** for mais baixo troca-se o “pai” do quadrado adjacente para o quadrado selecionado e finalmente, recalcula-se o **F** e **G** do novo quadrado.

Desta forma e repetindo este processo podemos chegar ao caminho mais curto entre os quadrados A e B como representa a Tabela 4:

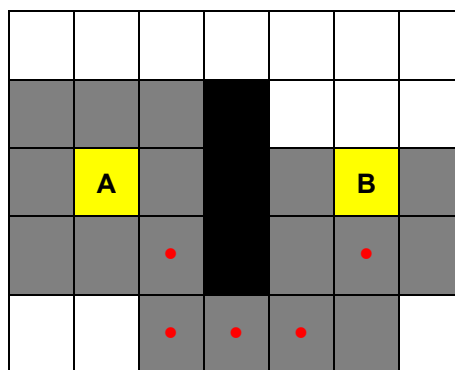


Tabela 4 - Caminho mais curto

### 2.3. Diagrama de Classes

De forma a esclarecer a estrutura e relação entre as várias entidades em causa decidi criar um diagrama de classes. Optei por não colocar os atributos nem os métodos em cada classe para simplificar o respetivo diagrama.

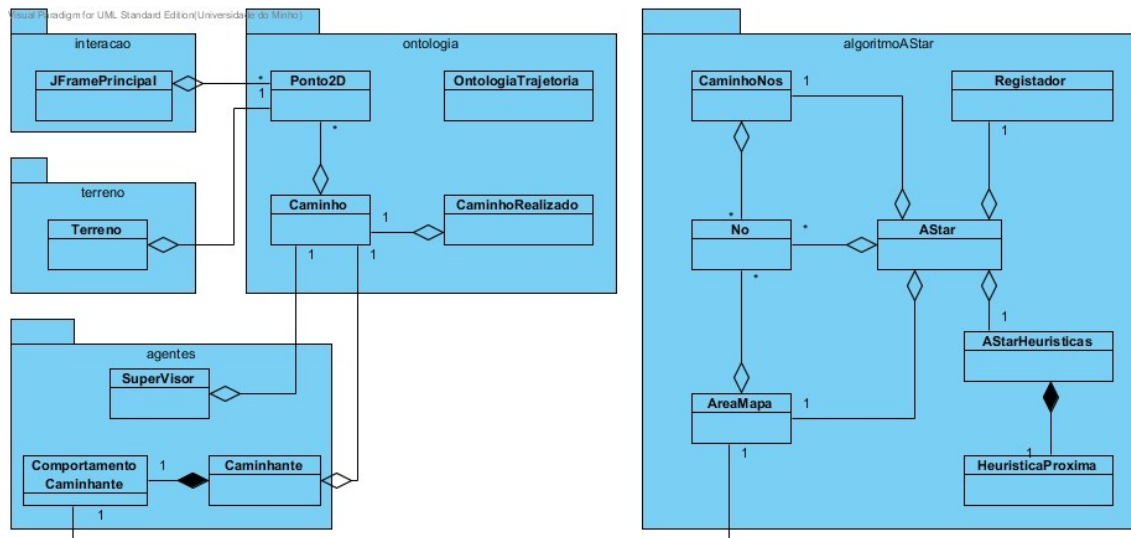


Ilustração 2 - Diagrama de Classes

O diagrama foi dividido em 5 *packages* distintos como se pode ver na imagem anterior, de seguida podemos ver uma descrição de cada um:

- Interação: tem a classe *JFramePrincipal* que inicializa a interface *SWING* da aplicação e é também responsável por inicializar os agentes (*SuperVisor* e *Caminhante*);
- Ontologia: contém todas as classes que servem de ontologia, permitindo definir a “linguagem” que todos os agentes falam permitindo a interação e troca de informação entre os agentes;
- Terreno: contém a classe *Terreno* que é responsável por ler, criar e editar as posições do mapa;
- Agentes: contém as classes dos agentes (*SuperVisor* e *Caminhante*) do sistema e a classe responsável pelo *behaviour* do caminhante (*ComportamentoCaminhante*);
- AlgoritmoAStar: contém todas as classes necessárias para a implementação e utilização do algoritmo A\*;

## 2.4. Ferramentas utilizadas

Esta secção serve para detalhar as ferramentas usadas tanto na criação como na edição da aplicação *software* como no auxílio à elaboração de todos os elementos que são expostos neste documento.

As ferramentas utilizadas foram as seguintes:

- Eclipse: o código da aplicação foi todo elaborado em linguagem *JAVA* no IDE *Open Source Eclipse*;
- WADE: como o trabalho foi elaborado com agentes inteligentes, utilizei a plataforma *WADE* (**W**orkflow and **A**gents **D**evelopment **E**nvironment). A plataforma *WADE* é uma extensão famosa da popular *Framework Open Source* para desenvolvimento de aplicações orientadas a agentes, designada por *JADE*. As duas principais características da plataforma *WADE*, fluxos de trabalho (*Workflows*) e suporte na administração, são bastantes distintas, tornando possível fazer-se desde simples aplicações em *JADE*, onde algumas das tarefas dos agentes (*Behaviours*) são definidos como *Workflows* ou, então, podendo-se fazer aplicações estruturadas de acordo com a arquitetura *WADE*, explorando todas as características que ela tem para oferecer;
- WOLF: de forma a utilizar a plataforma baseada em *WADE*, foi necessário usar o *plugin WOLF* do *Eclipse* que agrupa um ambiente gráfico à referida plataforma;
- UML: para a elaboração deste documento, e de modo a representar as relações entre as várias entidades de uma forma mais perceptível para o leitor, decidi utilizar a linguagem de modelação padrão de sistemas, designada *UML* (**U**nified **M**odeling **L**anguage);
- AUML: é uma variação / extensão do *UML* para modelação de atividades relativas aos agentes.

## 3. Desenvolvimento

### 3.1. Ontologias

As ontologias definem um conjunto de conceitos (*Concept*) que permitem esclarecer a informação necessária, para a comunicação dos agentes. Assim sendo as ontologias definem o universo de discurso.

Como se pode ver na imagem a seguir o conceito que foi utilizado no sistema foi o *Caminho*, que consiste num conjunto de *Pontos2D* (*Predicate*). Enquanto o *CaminhoRealizado* (*Predicate*) é o caminho realizado pelo *Caminhante*.

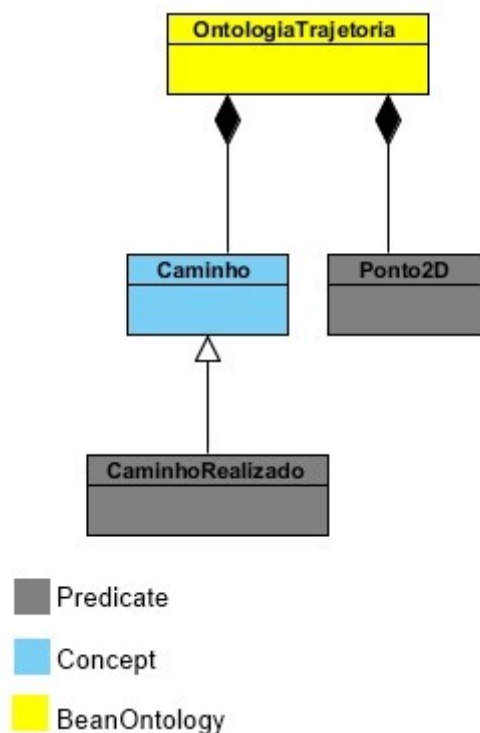


Ilustração 3 – Ontologias

Para definir e representar todos os predicados e conceitos defini uma ontologia designada *OntologiaTrajetoria*.



## 3.2. Agentes

Foi necessária a utilização de multi-agentes, apesar de não se tratar de um sistema muito complexo, mas é a melhor maneira de simular um sistema real permitindo a interação entre os agentes, a partir das ontologias definidas e já explicadas neste documento.

No caso deste projeto foram necessários dois agentes, um *SuperVisor* que é o chefe das operações sendo responsável por atuar sempre que os agentes *Caminhantes* acabam as suas tarefas. O agente *Caminhante* executa um conjunto de passos consoante um algoritmo e por fim comunica o caminho realizado ao *SuperVisor*.

### 3.2.1 SuperVisor

Todo o processo pesquisa do caminho mais curto é controlado pelo *SuperVisor* que interage com o agente *Caminhante*, recebendo deste o caminho realizado. Após receber o caminho realizado analisa e informa a distância mínima para o conjunto de pontos que o *Caminhante* realizou.

Este agente tem o seguinte comportamento:

- CyclicBehaviour: Comportamento responsável por gerir toda a informação proveniente da comunicação com os agentes *Caminhantes*.

### 3.2.2 Caminhante

Para a simulação de trajetórias foi utilizado o Agente *Caminhante* que é responsável por originar as trajetórias e, desta forma, simular um movimento de um qualquer tipo de pessoa.

O movimento a realizar pelo agente *Caminhante* será num terreno previamente carregado de um ficheiro e que tem como objetivo representar um terreno real através de uma escala. Definido por um conjunto de pontos (pelo menos dois) que têm de ser executados sequencialmente.

O *Caminhante* tem a tarefa de percorrer uma trajetória entre os ditos pontos, desviando-se dos potenciais obstáculos seguindo um algoritmo. Neste caso e por falta de tempo apenas se utilizou o algoritmo A\*, mas o sistema está preparado para suportar mais de que um agente *Caminhante* a efetuar algoritmos diferentes.

Para a execução deste agente, existe o seguinte comportamento:

- WalkBehaviour (TickerBehaviour): Este comportamento determina que o Caminhante caminhe aos poucos, marca a sua presença no sistema de simulação, e deixe o registo da sua trajetória no final de sua atuação. Este comportamento deverá ser executado periodicamente, enviando a sua posição até que chegue ao ponto final. O caminho realizado é enviado para o agente *SuperVisor*, facto que levou à escolha de um *TickerBehaviour*.

### 3.3. Aplicação Desenvolvida

Depois de definidos os agentes necessários para a utilização neste projeto, é necessário uma interface gráfica para demonstrar as funcionalidades da aplicação. A interface é intuitiva e agradável aos comuns utilizadores. A interface implementada é a seguinte:

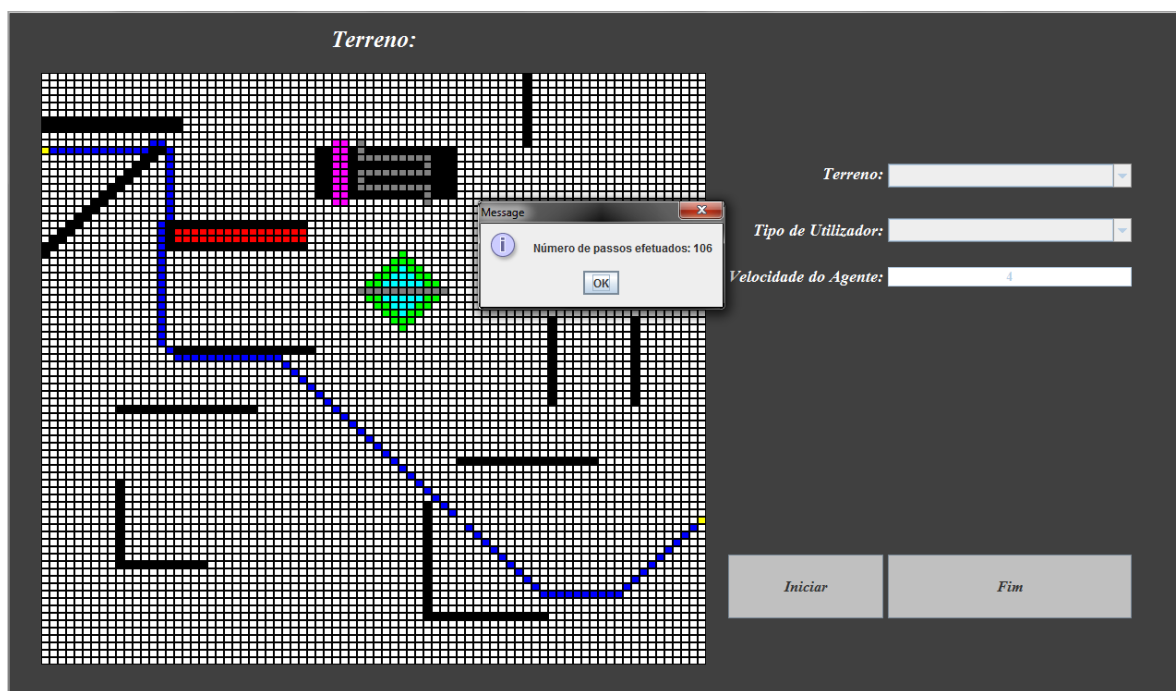


Ilustração 4 – Interface

Do lado esquerdo temos uma matriz (80x80) que representa num conjunto de cores o mapa desejado que o agente *Caminhante* tem de percorrer. Esta matriz encontra-se em conformidade com uma variável pública que instancia a classe *Terreno*, variável esta que possui um *Observer* de modo a reagir a qualquer alteração com a consequente atualização da matriz desenhada na janela.

Do lado direito temos de escolher o terreno numa *ComboBox* que lista e todos ficheiros “.txt” com os mapas / terrenos existentes no sistema. Em baixo podemos escolher o tipo de utilizador noutra *ComboBox* que lista os tipos de utilizadores que restringem o resultado do

caminho mais curto. Mais em baixo podemos ainda definir a velocidade (1 até 5) do agente *Caminhante na caixa de texto*.

O botão *Iniciar* após a leitura das configurações necessárias (terreno, velocidade e tipo de utilizador) cria o agente *SuperVisor* e o agente *Caminhante* que automaticamente dá início à sua caminhada.

O último botão, *Fim*, irá reiniciar todo o programa, coloca os valores padrão nas respetivas caixas de texto, reinicia o mapa e mata todos os agentes que se encontram ativos no *Container*.

### 3.4. Análise de Resultados

No fim da elaboração dos agentes e *interface*, como era esperado passei há fase de testes de forma a testar exaustivamente a aplicação. No início deparei-me com vários erros que foram corrigidos de imediato.

Após esta fase todos os testes mostraram-se bastante satisfatórios, simulando perfeitamente a realidade e calculando efetivamente o caminho mais curto.

Na tabela seguinte mostro alguns exemplos de comparações entre o simulado pela aplicação e a realidade:

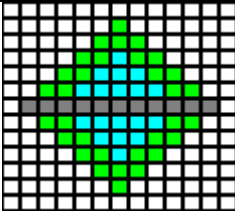
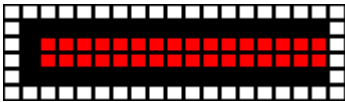
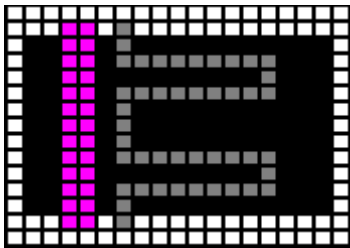
Descrição de elementos do Mundo Real	Excerto do mapa da simulação criada
Lago (azul claro) delimitado por árvores / arbustos (verde) com uma ponte (cinzento) para atravessar o lago de uma ponta a outra	
Zona restrita (vermelho) delimitada por três paredes (preto), apenas com uma entrada por uma porta que só entra quem tiver acesso (por exemplo: cartão magnético, identificador, cracha...);	
Uma passagem de nível com uma rampa (cinzento) em formato de S para facilitar o acesso a deficientes motores. E com acesso através de umas escadas para pessoas sem problemas de mobilidade	

Tabela 5 – Comparação entre o Mundo Real e a simulação

Na tabela seguinte pretendo demonstrar as funcionalidades da aplicação criada obdecendo às restrições impostas e desviando-se dos obstáculos encontrados, ilustrando com um exemplo:

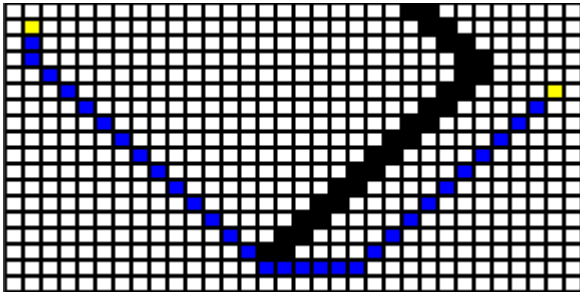
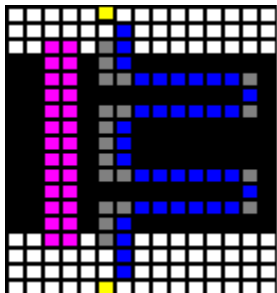
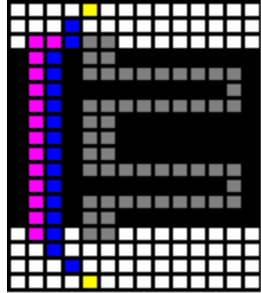
Descrição	Exemplo
<p>Como podemos ver no exemplo o algoritmo A* consegue determinar que vai encontrar uma parede e começa logo por se desviar dela encontrando o caminho mais curto entre os dois pontos</p>	
<p>O exemplo seguinte trata-se de um tipo de utilizador que é um deficiente motor em que o caminho mais curto entre os dois pontos passa por passar pela ponto / rampa em S</p>	
<p>Agora é o mesmo exemplo anterior só que o tipo de utilizador neste caso é um utilizador normal</p>	

Tabela 6 - Exemplos das funcionalidades da aplicação

## 4. Conclusões e Trabalho Futuro

Após a realização deste trabalho posso afirmar que o modelo de simulação criado era o esperado e é bastante satisfatório, visto representar bem um sistema real.

A utilização de Agentes Inteligentes foi uma mais-valia na integração do projeto, visto trazer uma autonomia e independência de todo o sistema. Outra grande valia é a utilização de multi-agentes de forma fácil e rápida, o que permite uma interação mais coerente e parecida com a realidade do dia-a-dia.

Sobre o algoritmo A\* usado neste projeto, posso dizer que apesar de ter demorado muito para conseguir integrar na aplicação, valeu a pena, porque o seu funcionamento é rápido e retorna sempre o caminho mais curto entre os pontos.

Como trabalho futuro, fica a implementação de mais algoritmos para posterior extração de conhecimento de forma a poder tirar relações sobre cada algoritmo, de forma a tentar melhorar a execução do algoritmo. Também era interessante integrar esta aplicação com um sistema de base de dados permitindo, o registo de um utilizador, e consoante os seus dados (preferências e limitações), o *software* adaptaria a pesquisa do caminho mais curto. Também gostaria de integrar este sistema com um sistema GPS integrado dentro dos edifícios.

Assim, efetuando um balanço honesto do projeto, sou da opinião que o resultado final obtido é bastante promissor e que ainda existe bastante motivação na continuação deste, pois existem muitas possibilidades para o trabalho futuro, tal como referido.

## Bibliografia

- [1] Wooldrige M.: An Introduction to MultiAgent .  
John Wiley & Sons, 2002.
- [2] J. Machado, J. Neves e V. Alves.: Sistemas Multiagente.  
2003.
- [3] F. Bellifemine, G. Caire, D. Greenwood.: Developing multi-agent systems with JADE.  
John Wiley & Sons, 2007.
- [4] Hart, P. E.; Nilsson, N. J.; Raphael, B. (1972). "Correction to "A Formal Basis for the  
Heuristic Determination of Minimum Cost Paths"". [\*SIGART Newsletter\* 37](#): 28–29.

## Referências WWW

- [1] A\* *Pathfinding* para Iniciantes  
[http://www.policyalmanac.org/games/aStarTutorial\\_port.htm](http://www.policyalmanac.org/games/aStarTutorial_port.htm);
- [2] Amit's *Game Programming Information*  
<http://www-cs-students.stanford.edu/~amitp/gameprog.html#paths>
- [3] *Memoization - A-Star Algorithm in Java*  
<http://memoization.com/2008/11/30/a-star-algorithm-in-java/>
- [4] A\* *Demonstration by James Macgill*  
<http://www.vision.ee.ethz.ch/~cvcourse/astar/AStar.html>
- [5] Shortest Path  
<http://harablog.wordpress.com/2011/09/07/jump-point-search/>
- [6] Heyes-Jones - A\* Algorithm Tutorial  
<http://www.heyese-jones.com/astar.html>