Evacuação com Agentes BDI

Relatório Final

4º ano do Mestrado Integrado em Engenharia Informática e Computação

Agentes e Inteligência Artificial Distribuída

Elementos do grupo:
João Pedro Miranda Maia - 201206047 - ei12089@fe.up.pt
Miguel Oliveira Sandim - 201201770 - miguel.sandim@fe.up.pt
Paula Cristina Teixeira Fortuna - 200600435 - ei12025@fe.up.pt

13 de Dezembro de 2015

Índice

1 Introdução

Apresenta-se neste relatório o resultado final do trabalho prático da Unidade Curricular de Agentes e Inteligência Artificial Distribuída. Será desenvolvida uma aplicação para simulação em situações de emergência recorrendo a agentes *BDI*.

Neste relatório será especificado o problema em causa, a ferramenta escolhida e a abordagem pela qual se optou.

2 Objectivo

O objetivo principal deste trabalho é a implementação de agentes *BDI* na simulação de um plano de evacuação num cenário de acidente, recorrendo-se à biblioteca *Jadex*.

No problema em causa os indivíduos encontram-se distribuídos num determinado edifício, ocupados nas suas tarefas usuais quando inesperadamente ocorre um acidente que obriga à sua evacuação. Aquando da deteção do acidente, todos os indivíduos procuram atingir uma área de segurança, que será representada pelo exterior do edifício. Portanto, cada um dos indivíduos procurará uma das saídas de emergência por forma a sair do edifício o mais rapidamente possível. Neste cenário, os indivíduos em causa serão simulados com diferentes agentes *BDI*. Esses agentes terão diversos comportamentos de emergência e poderão ser alvo de acidente.

3 Especificação da Abordagem

No presente capítulo detalham-se a abordagem adoptada na implementação do trabalho. Assim sendo, passa-se a apresentar o cenário da simulação e o modelo de comportamento implementado.

3.1 O Cenário de Simulação

No cenário de simulação desenvolvido, os indivíduos encontram-se ocupados a vaguear. Inesperadamente e de forma aleatória, gera-se um acidente, de um determinado tipo. Aquando do acidente, todos os indivíduos procuram atingir uma das saídas de emergência, o mais rapidamente possível. No decorrer do acidente podem ocorrer diversas situações: queda de um indivíduos, ferimento, obstrução de uma passagem ou parte do local.

O mundo é simulado através do *Jadex*, uma vez que este fornece a interface "ISpaceProcess", e será em duas dimensões, pelo que se pretende que este seja uma *grid*. Considera-se que neste podem ocorrer vários tipos de acidente (incêndio, inundação e ataque terrorista).

Pretende-se que o edifício, que pertencerá ao mundo e onde ocorrerá a evacuação, tenha várias saídas e obstáculos até estas.

3.2 Modelo de Comportamento de Evacuação

Com base na revisão de literatura foi desenvolvido um modelo de comportamento de evacuação. Este será o guia para o desenvolvimento dos agentes *BDI*

em *Jadex*. Apresenta-se nesta secção o modelo proposto.

3.2.1 Comportamento de Evacuação e Pertinência do Tema

A perda de vidas humanas em situação de emergência deu relevo à necessidade de planear o comportamento de evacuação. Assim, ferramentas de simulação podem ser utilizadas para desenvolver e medir a qualidade de de um plano de emergência. Neste contexto, com o trabalho desenvolvido pretende-se simular o comportamento de evacuação, por um lado individual, mas também no contexto de uma multidão, de forma similar ao que acontece no mundo real.

Para além disso, existem várias teorias sobre comportamento humano e de multidão em situação de evacuação, que foram tidas em conta neste trabalho. Contudo, tem existido pouco esforço em conectar essas várias abordagens [1], o que seria importante para melhorar a formação nesta área. Com este trabalho pretende-se também colmatar essa lacuna, integrando várias perspectivas.

Percepção do risco

Uma das principais abordagens que guiou o modelo desenvolvido tem por base a ideia de que a percepção do risco desempenha um papel relevante no comportamento de evacuação [2]. Assim, é referido que no contexto de uma evacuação com fogo (os autores assumem que noutras situações de emergência este processo é similar) um maior risco percebido leva as pessoas a acções de protecção [2]. Neste âmbito, percepção de risco é a percepção de uma ameaça eminente à própria vida e saúde. Portanto, corresponde à probabilidade percepcionada de o indivíduo ser afectado por um evento indesejável [2]. Existem uma série de factores que contribuem para a percepção de risco:

- Género – Homens percepcionam menos risco que as mulheres;
- Impulsividade – Ocupantes mais impulsivos percebem maior risco nas mesmas situações;
- Confiança nas autoridades - Maior confiança nas autoridades implica menor risco percebido;
- Proximidade – Mais proximidade do acidente implica maior risco percebido;
- Ansiedade Traço e Estado – Maiores níveis de ansiedade implica maior risco percebido;
- Medo Traço e Estado – Maiores níveis de medo implica maior risco percebido;
- Raiva Traço e Estado – Maiores níveis de raiva implica menor risco percebido;
- Locus de controlo Traço e Estado – Se interno menor risco percebido; se externo maior risco percebido (o ambiente influência o locus de controlo).

Tal como foi referido, existem ainda outras teorias, sendo que algumas se centram menos nos processos e mais nos comportamentos que ocorrem em situação de evacuação.

Ações frequentes em evacuação

Outros autores citam alguma ações frequentes em situação de evacuação [3], nomeadamente:

1. As pessoas vão adoptar acções de procura de informação, especialmente se as pistas forem ambíguas e inconsistentes;
2. As pessoas agem de forma racional e altruista;
3. As pessoas adoptam frequentemente acções preparatórias antes da resposta de evacuação (por exemplo, colocam um lenço junto à face);
4. Na situação de emergência as pessoas tendem a mover-se para o familiar.
5. Em incerteza, o primeiro instinto é sentir segurança face ao ambiente;

Ainda os mesmos autores, referem um conjunto de regras comportamentais onde o comportamento é simulado com base em condições específicas (if-then):

1. Se o ocupante está na presença de fumo de uma certa densidade então vai procurar outra rota.
2. Se o ocupante está na presença de fumo de uma certa densidade então vai parar de explorar.
3. Se o ocupante está na presença de fumo de uma certa densidade então pode ficar ferido e reduzir a sua velocidade.
4. Se o ocupante está na presença de fumo de uma certa densidade e a partir de uma dada altura então pode optar por rastejar.

Ainda relativamente a outros pessoas na situação de emergência:

1. Se existir um determinado limite de pessoas numa saída, então a pessoa redireccionará o seu plano para outra saída.
2. Um ocupante ferido, ou em pior condição seguirá os outros para a saída.

Para além das teorias já referidas, são também enumerados vários tipos de agente em evacuação.

Tipos de agente

São citados três tipos de agentes em situação de evacuação [2]:

- Activos – observam activamente o ambiente para encontrarem a rota mais rápida. Os resultados mostram que a presença destes elementos pode ter um efeito significativo e positivo no tempo de evacuação de uma multidão.
- Conservativos – Seguem rotas que lhes são previamente familiares, frequentemente verificados em multidões reais.
- *Herding* – Seguem os restantes agentes, frequentemente verificados em multidões reais.

O mesmo autor também refere a dimensão da paciência [2]:

- Pacientes – Agentes pacientes esperam pela sua vez na fila, quando se encontram a sair.
- Impacientes – Comportam-se agressivamente para tentar tirar vantagem face aos outros agentes.

Para além disso, é ainda de salientar outras teorias que explicam o comportamento de massas [1].

Pânico em massa

A teoria de Pânico em Massa refere-se a um pânico excessivo e a uma resposta intensa de medo e fuga. Neste contexto a multidão é vista como menos inteligente e mais reactiva que cada indivíduo isoladamente. Contudo, posteriormente outros autores [1] vieram defender que esta situação apenas ocorrerá se:

- Existir a percepção de elevado risco físico.
- Existir um sentimento de estar encurralado.
- Existir um sentimento de incapacidade colectiva e isolamento individual.

Para além disso, outros autores [1] vieram ainda defender que este comportamento de pânico em massa é raro, verificando-se que:

- Comportamentos anti-sociais e egoístas são raros e se ocorrerem não se transmitem aos outros indivíduos;
- Comportamentos de entreajuda são frequentes;
- Em vez de antissocial, o comportamento de multidão é racional e socialmente estruturado.

Apresenta-se, de seguida, o modelo BDI desenvolvido, adaptado das várias teorias analisadas.

3.2.2 Adaptação do Modelo à Arquitectura BDI

Para implementação do modelo de comportamento de evacuação anteriormente apresentado, utilizou-se a biblioteca Jadex, uma vez que esta permite implementar directamente a arquitectura BDI em agentes. Para aumentar o isolamento das features implementadas, optou-se por se fazerem vários módulos BDI, com relativa autonomia entre si. Nestas classes de agentes foram alocados *Beliefs*, *Desires* e *Intentions* para comportamentos distintos. Relativamente ao funcionamento dos agentes, seguiu-se a arquitectura que se apresenta na *Figura 1.*

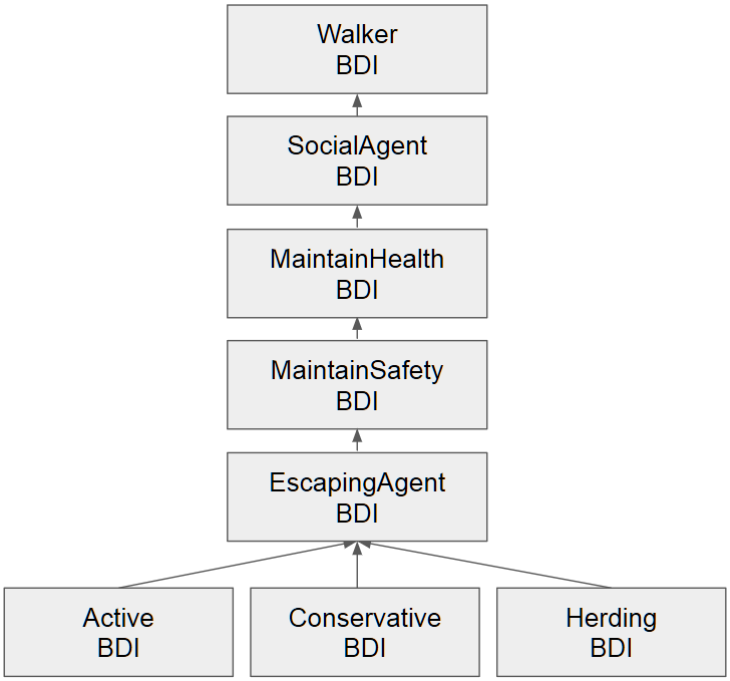

Figura 1: Arquitetura dos agentes sob a forma de classes UML.

Módulo WalkerBDI

Funcionalidade Módulo responsável pela movimentação do agente no mundo. Esta classe foi considerada a mais genérica, pois todas as seguintes necessitam desta funcionalidade para se deslocarem no mundo.

Beliefs

- nextPosition: próxima posição desejada, que pode ser atualizada por qualquer classe descendente.
- currentPosition: posição atual, guardada para facilidade de cálculos.
- speed: velocidade do agente, iniciada a 1 célula por segundo.

***Desires* (*Goals*)**

- WanderGoal: objectivo do agente antes de se dar o incidente.

***Intentions* (*Plans*)**

- WanderPlan: plano despoletado pelo WanderGoal. Este consiste em procurar uma nova posição aleatória no espaço.
- GoPlan: é despoletado pela alteração da variável nextPosition. Caso esta variável mude, e, caso não haja colisões no mundo, então o agente muda de posição.

Módulo SocialAgentBDI

Funcionalidade Módulo responsável pelos comportamentos sociais e pela personalidade do agente.

Constantes

- DISTANCE_TO_HELP: Distância em células até à qual o agente é capaz de ouvir e ajudar outro.

Atributos

- personState: objecto da classe PersonState, responsável pelas características de personalidade do agente, apresentadas posteriormente.
- cureSet: Estrutura onde são guardados os objectos de cura para os outos agentes.
- pushSet: Estrutura onde são guardados os objectos de empurrar outros agentes.

Estes dois permitem ao agente eliminar estes objectos quando deixou de realizar estas actividades, bem como certificar-se que o agente não se cura nem empurra a si próprio.

Beliefs

- patience: booleano, se for *true* o agente será paciente e aguardará a sua vez para sair numa porta; se estiver a *false* o agente será impaciente e empurrará os outros na saída.

***Desires* (*Goals*)**

- HelpOthersGoal: objectivo do agente que fica activo se existir um objecto HURT_AGENT dentro da distância limite.
- PushOthersGoal: objectivo que fica activo se existir outro agente na mesma posição que o agente e se este estiver em pânico e sem paciência para esperar.
- ReceivePushOthersGoal - objectivo do agente, que fica activo se existir um objecto PUSH_AGENT na currentPosition e atualiza o valor da condição física.

***Intentions* (*Plans*)**

- HelpOthersPlan: plano despoletado pelo HelpOthersGoal. Este consiste em procurar o objecto HURT_AGENT mais próximo e deslocar-se até lá. Quando o agente está numa célula adjacente, então gera um objecto CURE_AGENT que o ferido deve procurar para ficar curado.
- PushOthersPlan: é despoletado pelo PushOthersGoal. Cria um objecto PUSH_AGENT na posição onde se encontra.

MaintainHealthBDI

Funcionalidade Módulo responsável pela condição física do agente.

Beliefs

- condition: inteiro que indica a condição física do agente. Varia no intervalo [0-100]
- isDead: booleano que é verdadeiro quando a condition é 0.
- isHurt: booleano que é verdadeiro quando a condition é <50.

***Desires* (*Goals*)**

- MantainHealthGoal: objectivo do agente que fica activo quando o agente está ferido.

***Intentions* (*Plans*)**

- MantainHealthPlan: plano despoletado peloMantainHealthGoal. Este consiste em procurar objectos CURE_AGENT na currentPosition e, se existir algum, incrementar a condição física.
- DeadPlan: plano a implementar quando o *belief* isDead é verdadeiro. Consiste em eliminar a *thread* do agente, bem como os objectos que este possa ter pendentes.
- HurtedPlan: plano a implementar quando o *belief* isHurt é verdadeiro. Consiste em criar o objecto HURT_AGENT, que sinaliza um agente ferido.
- RecoveredPlan: plano a implementar quando o *belief* isHurt é falso e o agente saiu do estado ferido. Consiste em eliminar o objecto HURT_AGENT.

MaintainSafetyBDI

Funcionalidade Módulo responsável pela manutenção da percepção de segurança do agente.

Beliefs

- riskPerception: inteiro que corresponde à percepção de risco do agente. Varia no intervalo [0-100]. É actualizado periodicamente a uma rate de 500 milissegundos. O seu valor é calculado recorrendo ao método evaluateAgentSituation. O funcionamento deste método será explicitado no final deste capítulo.
- inPanic: booleano que é verdadeiro, quando riskPerception >90.

***Desires* (*Goals*)**

- MantainSafetyGoal: objectivo do agente, que fica activo quando o agente percepciona risco.
- IncreaseDistanceFromDangerGoal: objectivo de aumentar a distância do perigo.
- FindExitGoal: objectivo de aumentar a distância, encontrando a saída.

***Intentions* (*Plans*)**

- MantainSafetyPlan: para manter a segurança o plano consiste em incrementar a distância do perigo, através da chamada do objectivo IncreaseDistanceFromDangerGoal.
- IncreaseDistanceFromDangerPlan: para aumentar a distância do perigo, como o cenário de evacuação é dentro de um edifício, inicia-se o objectivo encontrar a saída FindExitGoal.

EscapingAgentBDI

Funcionalidade Módulo responsável pelo plano de fuga do qual vão herdar todos as diferentes classes de agentes. Foi criada esta classe para evidenciar que cada um dos tipos de agente (Activo, Conservativo e *Herding*) consiste numa estratégia de escape em particular.

Beliefs Não tem.

***Desires* (*Goals*)** Não tem.

***Intentions* (*Plans*)**

- FindExitPlan: chama o método para encontrar uma saída, findExit. Este método vai ser *Override* em cada uma das subclasses.

ActiveBDI

Funcionalidade Módulo responsável pelo plano de fuga dos agentes tipo Activo.

Beliefs Não tem.

***Desires* (*Goals*)** Não tem.

***Intentions* (*Plans*)**

- findExit: consiste em utilizar o algoritmo A* para encontrar o melhor caminho até à saída. Caso a saída se encontre ocupada com outros agentes, este procurará uma posição aleatória no mapa, para vaguear até ela e assim descobrir novas possíveis saídas.

ConservativeBDI

Funcionalidade Módulo responsável pelo plano de fuga dos agentes tipo conservativo.

Beliefs Não tem.

***Desires* (*Goals*)** Não tem.

***Intentions* (*Plans*)** findExit: consiste em utilizar o algoritmo A* para encontrar o melhor caminho até à saída. Este objetivo permanecerá conservado, ao contrário dos activos.

HerdingBDI

Funcionalidade Módulo responsável pelo plano de fuga dos agentes tipo *Herding*.

Constantes

- DISTANCE_TO_HERDING: distância a que o agente consegue encontrar outro, para segui-lo.

Beliefs Não tem.

***Desires* (*Goals*)** Não tem.

Intentions (Plans)

- findExit: consiste em seguir outros agentes até à saída. Esses agentes são do tipo conservativo ou activo.

Encadeamento dos diferentes módulos

O encadeamento dos vários módulos é integrado no método *evaluateAgentSituation()* do módulo *MaintainSafetyBDI*. Este método é chamado periodicamente, por forma a garantir a atualização das crenças (*beliefs*) do agente perante o seu meio. É também no *evaluateAgentSituation()* onde são criados os vários objectivos (*Goals*), mediante certas condições. Apresenta-se no *Algoritmo 1* o seu pseudocódigo.

Algoritmo 1 Pseudocódigo do algoritmo de avaliação do meio e atualização das crenças do agente.

```
1:  function EVALUATEAGENTSITUATION
2:      if agent.goals = {} then
3:          agent.goals ← agent.goals ∪ ReceivePushOthersGoal
4:          if ¬ incident started then
5:              agent.goals ← agent.goals ∪ WanderGoal
6:          else
7:              if agent.isNotHurt ∧ agent.inPanic ∧ agent.isNotPatient then
8:                  if world.anotherAgentInMyCell() then
9:                      agent.goals ← agent.goals ∪ PushOthersGoal
10:                 else
11:                     agent.goals ← agent.goals ∪ MaintainSafetyGoal
12:             else if agent.isNotHurt and world.someoneNeedsHelp then
13:                 agent.goals ← agent.goals ∪ HelpOthersGoal
14:             else if agent.isNotHurt then
15:                 agent.goals ← agent.goals ∪ MaintainSafetyGoal
16:             else if agent.isHurt then
17:                 agent.goals ← agent.goals ∪ MaintainHealthGoal
```

4 Desenvolvimento

4.1 Descrição da Ferramenta

A linguagem utilizada para implementar este cenário foi Java, mais em concreto a biblioteca *Jadex*. Esta foi escolhida devido às suas características, pois permite implementar agentes *BDI* directamente. A versão mais recente do motor *BDI* é o *bdiv3* [4]. O acrónimo BDI [5] neste contexto refere-se a agentes racionais que têm *Beliefs*, *Desires* e *Intentions*:

- *Beliefs* (crenças) é o conhecimento que o agente tem, correspondente a uma base de dados, ou a um conjunto de estados. A alteração nas crenças pode despoletar eventos.
- *Desires* (desejos) representam objectivos que o agente gostaria de atingir. No Jadex esta componente tem a designação de Goals.
- *Intentions* (intenções) representam o estado deliberativo do agente. No Jadex designam-se de *Plans* (planos) e são sequências de ações que um agente pode tomar para atingir os seus desejos. Os planos podem também incluir sub-planos.

Para além disso, o Jadex aglomera agentes, serviços e componentes utilizando *Service Component Architecture* (SCA) e possui uma arquitetura de aplicação seguindo os princípios das arquiteturas orientadas a serviços (SOA). Salienta-se que para a implementação deste trabalho a utilização de Jadex tem vantagens, uma vez que permite de forma direta:

- Simular agentes com *Beliefs*, *Goals* e *Plans*;
- Simular vários agentes em simultâneo num mundo;

- Simular espaços;
- Possiblitar intercomunicação entre os vários componentes.

4.2 Estrutura da Aplicação

A estrutura do modelo anteriormente apresentado, foi implementada segundo o diagrama UML representado na *Figura 2*.

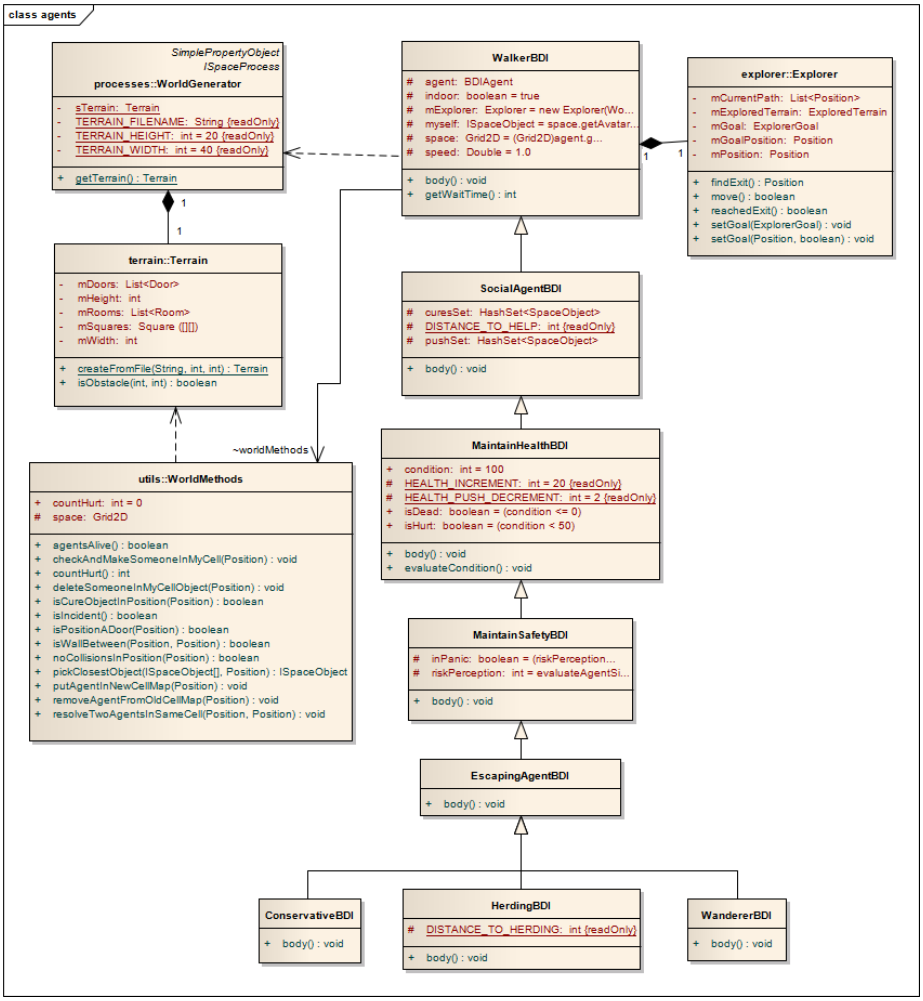

Figura 2: Diagrama de classes do módulo dos agentes. Todos os agentes derivam do agente *WalkerBDI*. O espaço é criado a partir de um ficheiro, no *WorldGenerator*. Para interagir com o espaço, os agentes utilizam o *WorldMethods*. Cada agente contém uma instância de um *Explorer*, que é responsável pela exploração do espaço e pelo cálculo do percurso que o agente segue.

Para além disso, é de salientar ainda que foram utilizados alguns recursos, por forma a tornar o código mais reutilizável e fácil de testar:

- *Abstract Factory Design Pattern*, para poder implementar vários agentes (sem alteração do XML);
- Leitura do mapa de um ficheiro;
- Utilização de processos no Jadex, para gerar agentes, incidentes e estatísticas;
- Output directo das estatísticas em ficheiro *CSV*.

4.3 Detalhes Relevantes da Implementação – Algoritmos Utilizados

4.3.1 Representação do Terreno

Relativamente à representação do terreno, os agentes apenas conhecem a divisão em que se encontram. Assim, se os agentes do tipo verificarem que não existe uma porta de saída na divisão em que se encontram, então, dirigem-se para a porta mais próxima (de uma divisão que ainda não exploraram). Depois de explorar uma nova divisão, o processo repete-se, até encontrarem a porta de saída. Deste modo, foi necessário representar as divisões do mapa (*Rooms*), e as várias portas (*Doors*) que fazem a ligação entre as várias divisões (ver *Figura 3*).

Quando a aplicação é iniciada, o terreno é construído a partir de um ficheiro de texto:

- *X* representa uma parede;
- *D* representa uma porta, que liga duas divisões;
- *E* representa a saída;
- *A* representa um agente do tipo *Active*;
- *H* representa um agente do tipo *Herding*;
- *C* representa um agente do tipo *Conservative*.

Para representar as divisões, foi usado o algoritmo *Flood Fill* [6], de modo a associar uma divisão a um conjunto de quadrículas.

4.3.2 Encontrar o Caminho Mais Curto num Mundo Inexplorado

De modo a simular uma deslocação mais realista dos agentes pelo mapa, foi implementado o algoritmo $A*$ aplicado a um espaço 2D [7] [8] [9]. A heurística escolhida foi a *distância Manhattan*:

$$h(node, goal) = \mid node_x - goal_x \mid + \mid node_y - goal_y \mid \quad (1)$$

Nesta implementação, as paredes e as divisões ainda não exploradas são consideradas obstáculos. À medida que os agentes vão explorando novas divisões, as quadrículas correspondentes deixam de representar obstáculos. Deste modo, um caminho não é válido se passar por uma divisão que ainda não foi explorada pelo agente.

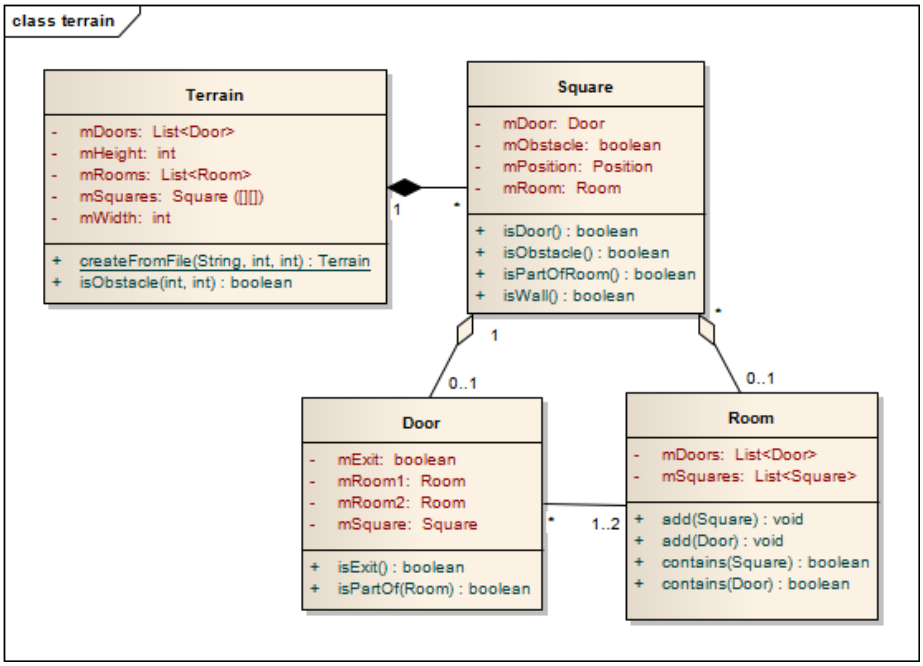

Figura 3: Diagrama de classes do módulo da representação do terreno. O *Terrain* é constituído por um conjunto de *Squares*. Um *Square* pode conter uma *Door* ou *Wall*, ou, ainda, fazer parte de um *Room*. Um *Square* também pode representar um *Obstacle*. Uma *Door* pode fazer a ligação entre dois *Rooms* ou ser a porta de saída (*exit*).

Este algoritmo é utilizado para calcular o caminho mais curto desde a posição do agente até uma posição no mundo. Por exemplo, o algoritmo é utilizado para descobrir o caminho mais curto até uma determinada porta, ou agente *ferido*.

4.3.3 Encontrar a Porta Inexplorada Mais Próxima

Como foi explicado até aqui, um dos objetivos dos agentes é descobrir a saída. No entanto esta pode não estar diretamente ao seu alcance, tendo o agente de atravessar várias divisões até a descobrir. No início, o agente apenas tem conhecimento das portas da divisão em que se encontra, mas se a divisão em que este está contiver uma saída, então o agente dirige-se imediatamente para aí. Se tal não se verificar, então o agente vai de encontro à porta mais próxima que ainda não explorou.

De modo a descobrir qual a porta inexplorada mais próxima, foi aplicado o algoritmo A* a um grafo, em que os nós correspondem às posições das portas que estão ao alcance do agente, isto é, todas as portas de todas as divisões já exploradas. Assim, o algoritmo termina quando encontra o nó com menor custo, correspondente a uma posição de uma porta que ainda não foi explorada pelo agente.

4.3.4 Encontrar a Saída

De modo a encontrar a saída, foram utilizados os dois algoritmos anteriormente descritos em 4.3.2 e 4.3.3. O *Algoritmo 2* descreve a forma como os agentes calculam a próxima posição, quando o seu objetivo é encontrar a saída. A *Figura 4* apresenta o diagrama de classes, mostrando a interação entre os vários componentes.

Algoritmo 2 Pseudocódigo do algoritmo para encontrar a saída.

```
1:  function moveExplorer
2:      if explorer.goal = ExplorerGoal.FindExit then
3:          exitSquare ← exploredTerrain.findNearestExit()
4:          exitPosition ← exitSquare.getPosition()
5:          if ¬ exitSquare.isObstacle() then
6:              destination ← exitPosition
7:          else
8:              // Perform A* on graph of doors to find the nearest unexplored door:
9:              destination ← DoorFinder.run(exploredTerrain, exitPosition)
10:         // Perform A* on the terrain to find the shortest path to the destination:
11:         currentPath ← PathFinder.run(exploredTerrain, currentPosition,
    destination)
12:         nextPosition ← currentPath.getFirst()
```

4.3.5 Deteção de Paredes entre Agentes

Para simular de forma mais realista a visão que o agente tem do mundo, foi necessário implementar uma função que testa se existe uma parede entre dois objectos. Esta função foi útil para isolar agentes que não estejam na mesma divisão, bem como isolar incidentes de agentes.

O algoritmo seguido consistiu em encontrar a linha que une dois pontos e todas a posições da *grid* que pertencem a essa linha. Para este efeito utilizou-se o algoritmo de *Bresenham*.

4.3.6 Personalidades e Velocidade

Para se implementar de forma mais realista o comportamento de evacuação, foi tido em conta que a perceção do risco depende de um conjunto de variáveis, que se designaram de *personState*. Este é constituído por um conjunto de variáveis:

- Género;
- Impulsividade;
- Confiança nas autoridades;
- Ansiedade traço e estado;
- Medo traço e estado;
- Raiva traço e estado;

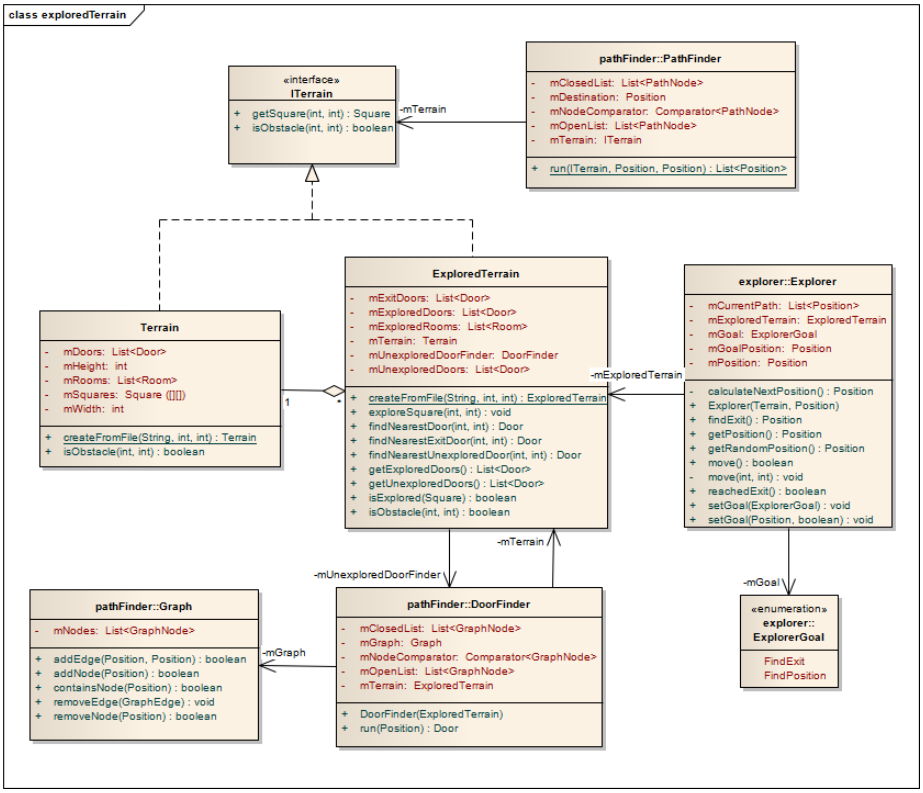

Figura 4: Diagrama de classes do módulo de exploração. Cada agente contém uma instância de um *Explorer*. Cada *Explorer* contém um *ExploredTerrain*, que, por sua vez, contém uma referência para um único *Terrain* partilhado por todos os agentes. O PathFinder encontra o caminho mais curto num determinado *ITerrain*. Se o *ITerrain* for um *Terrain*, então apenas as *Walls* são consideradas *Obstacles*. Se o *ITerrain* for um *ExploredTerrain*, todos os *Rooms* não explorados correspondem a *Obstacles* adicionais. O DoorFinder encontra a *Door* inexplorada mais próxima de uma dada posição.

- Locus de controlo;
- Paciência.

Para trazer alguma variabilidade aos agentes, uma vez que cada pessoa é única, esse estado foi calculadas de forma aleatória. Contudo, escolheu-se atribuir um valor para cada um dos campos gerando um aleatório proveniente de uma distribuição normal (exceto género), com média 0 e desvio padrão 1.

Estas características vão interferir no cálculo da perceção de risco, dado que ao valor de risco atribuído, mediante as condições do meio, lhe vão ser somados os valores em cada uma destas variáveis. Esses valores são adicionados com sinal de acordo com o referido no capítulo 3. Por exemplo, se a mais ansiedade corresponde maior percepção de risco, então ao valor que se obteria para a perceção de risco será adicionado o valor de cada agente em cada variável.

Posteriormente o valor da velocidade dos agentes é calculado em função da perceção do risco e da condição física. A velocidade de um agente é dada pelo tempo em milissegundos para percorrer uma célula e é equivalente ao tempo em que a *thread* do agente faz *sleep*. O cálculo da velocidade é dado pela fórmula:

$$speed = \frac{condition}{100} + \frac{4 \cdot valueForRiskPerception}{100} \tag{2}$$

Assim sendo, a velocidade de cada agente vai variar entre 0.5 e 5 se o agente estiver em boa condição física. Caso o agente esteja ferido, ficará parado.

4.3.7 Tipos de Incidente

Tal como foi proposto no enunciado, foram utilizados vários tipos de incidente. Para se gerar estes objetos é utilizado um processo de Jadex. Neste é guardado uma variável, que marca a taxa com que estes são gerados. Os seus diferentes comportamentos apresentam-se de seguida:

- *Fogo* consiste em gerar aleatoriamente uma posição no terreno com uma célula de fogo, esta vais progredindo gradualmente célula a célula, sempre numa direção aleatória, podendo obstruir paredes e portas.
- *Cheia* consiste em fazer um algoritmo de flooding que se expande iteração a iteração. Note-se que, ao contrário do fogo, a água nunca ocupa as paredes.
- *Terrorismo* consiste em gerar elementos aleatoriamente, em qualquer posição do mapa, inclusive portas, mas nunca numa parede.

5 Experiências e Resultados

5.1 Objectivos e Questões de investigação

Com o objectivo de testar a capacidade do programa desenvolvido funcionar como ferramenta de simulação, foram realizados várias experiências. As questões de investigação em análise foram as seguintes:

- O tipo de incidente influencia os resultados obtidos para o tempo de evacuação, número de agentes mortos e que consegue escapar?
- A presença de agentes Activos interfere no tempo de evacuação?
- A presença de agentes *Herding* interfere no tempo de evacuação?
- A presença de agentes Conservativos interfere no tempo de evacuação?
- A presença de agentes Activos interfere no número de agentes mortos ou que conseguem escapar com vida?
- A presença de agentes *Herding* interfere no número de agentes mortos ou que conseguem escapar com vida?
- A presença de agentes Conservativos interfere no número de agentes mortos ou que conseguem escapar com vida?

5.2 Metodologia

Para avaliar estas questões foram realizados 26 ensaios em que foram manipuladas individualmente cada uma das seguintes variáveis:

- O número de agentes Activos, os restantes factores mantidos constantes;
- O número de agentes *Herding*, os restantes factores mantidos constantes;
- O número de agentes Conservativos, os restantes factores mantidos constantes;
- O número de portas e os restantes factores mantidos constantes.

É de salientar que relativamente aos incidentes estes permaneceram com tipo e comportamento aleatórios. Os resultados foram imprimidos para um ficheiro *CSV*, com as respectivas variáveis nas colunas:

- Tipo de incidente;
- Número inicial de Activos;
- Número inicial de *Herding*;
- Número inicial de Conservativos;
- Número inicial de portas total;
- Número total de agentes a escapar;
- Número total de agentes mortos;
- Tempo total de evacuação em segundos.

Posteriormente, recorrendo-se ao *software* IBM SPSS, analisaram-se algumas métricas que passam a ser apresentadas no próximo subcapítulo.

5.3 Resultados e sua Análise

5.3.1 Tipo de Incidente

Para concluir se o tipo de incidente desenvolvido no programa influencia as diferentes variáveis em estudo, foram calculadas as médias obtidas nos ensaios, agrupando-se por tipo de incidente. Obtiveram-se os seguintes resultados:

	N	Média	Desvio Padrão
Fogo	11	31.73	9.645
Cheia	6	33.17	13.408
Terrorismo	9	36.22	8.541

Tabela 1: Médias para os ensaios com vários tipos de incidentes, na variável número de agentes a escapar.

	N	Média	Desvio Padrão
Fogo	11	2.36	3.828
Cheia	6	1.00	1.265
Terrorismo	9	6.44	4.746

Tabela 2: Médias para os ensaios com vários tipos de incidentes, na variável número de mortos.

	N	Média	Desvio Padrão
Fogo	11	39.89209	9.512567
Cheia	6	37.03250	13.522025
Terrorismo	9	38.45322	16.809776

Tabela 3: Médias para os ensaios com vários tipos de incidentes, na variável tempo de evacuação.

Para testar se as diferenças entre as médias obtidas tem relevância estatística foi usado o teste "One-way ANOVA", dado que os testes t são apenas adequados até um máximo de comparação entre duas médias e neste caso estamos na presença de três grupos de incidentes (fogo, cheia e terrorismo).

Os resultados obtidos indicaram que não se encontraram diferenças no número de agentes a escapar ($F = 0.484$, $p = 0.622$), nem no tempo de evacuação ($F = 0.093$, $p = 0.912$). Contudo, o número de mortos apresentou diferenças nos vários tipos de incidente, sendo o caso do terrorismo o mais mortífero ($F = 4.455$, $p < 0.05$).

5.3.2 Tipo de Agente e Número de Portas

Por sua vez, para verificar de que forma as restantes variáveis se relacionam, foi utilizada a correlação de Pearson, tendo-se obtido a matriz, que se apresenta de seguida.

		Número de agentes a escapar	Número de agentes mortos	Tempo de Evacuação
Número inicial de Activos	Pearson Correlation	0.44	0.479	NR
	p	<0.05	<0.05	
	N	26	26	
Número inicial de portas total	Pearson Correlation	NR	NR	-0.765
	p			<0.001
	N			26
Número de saídas para o exterior	Pearson Correlation	NR	NR	-0.676
	p			<0.001
	N			26

Tabela 4: Correlações significativas encontradas entre as várias medidas em estudo (os valores marcados com NR são não relevantes estatisticamente).

Estes resultados permitem concluir, que, de acordo com a implementação seguida, é quando existem mais agentes ativos que também existem mais agentes que conseguem escapar, contudo, também é neste caso que mais agentes morrem. Por sua vez, existe uma forte correlação entre o número de portas total e o tempo de evacuação, ou seja, quantas mais portas existem no edifício menor o tempo necessário aos agentes saírem. O mesmo se verifica para o número de portas que dão para o exterior.

6 Conclusões

Relativamente ao trabalho desenvolvido, o grupo cumpriu com os objetivos inicialmente propostos, tendo desenvolvido um módulo de comportamento de evacuação. Este foi implementado recorrendo-se à biblioteca *Jadex*. Para além disso, para testar se a ferramenta desenvolvida poderia ser utilizada para simulação deste cenário foram recolhidas algumas estatísticas. Chegou-se à conclusão que o número de mortos nas simulações vai depender do tipo de incidente e que é quando existem mais agentes Activos que mais agentes conseguem escapar e também é neste caso que mais agentes morrem, apontando que possivelmente estes agentes tanto contribuem para uma maior capacidade de fuga mas ao mesmo tempo implicando mais risco, até porque a estratégia que estes usam para conseguir fugir de situações de encravamento é procurar células aleatoriamente. Por sua vez, existe também uma forte correlação entre o número de portas total e o tempo de evacuação, o que vai de acordo ao esperado.

É ainda de salientar que para um cenário de simulação em evacuação, sistemas multi-agente são uma alternativa realista uma vez que permitem modelar cada pessoa individualmente e em interação com o meio, o que é congruente com o que ocorre no mundo real. Contudo, esta abordagem tem a desvantagem de facilmente apresentar complexidades temporais e espaciais elevadas.

7 Recursos de Software e Linguagens Utilizadas

Para o desenvolvimento deste projeto foi utilizado o *IDE IntelliJ IDEA/Eclipse* em ambiente *Windows* com a linguagem *Java 7* e com recurso à biblioteca *Jadex*. Foi também utilizado o *software* de estatística "*IBM SPSS*". É ainda de salientar que foram realizados testes unitários para os algoritmos e foram criados também ficheiros teste para avaliar os comportamentos dos agentes.

Os três elementos do grupo distribuíram o trabalho equitativamente entre si.

Referências

[1] Simo Heliövaara et al. «Pedestrian Behavior in Evacuations–Simulation Models and Experiments». Em: (2014).

[2] Max T Kinateder et al. «Risk perception in fire evacuation behavior revisited: definitions, related concepts, and empirical evidence». Em: *Fire Science Reviews* 4.1 (2015), pp. 1–26.

[3] Erica D Kuligowski e Steve MV Gwynne. «The need for behavioral theory in evacuation modeling». Em: *Pedestrian and Evacuation Dynamics 2008*. Springer, 2010, pp. 721–732.

[4] João Lopes e Henrique Lopes Cardoso. *AIAD - Tutorial Jadex*. 2015. URL: `https://web.fe.up.pt/~eol/AIAD/jadex/doku.php?id=start` (acedido em 07/11/2015).

[5] Anand S Rao, Michael P Georgeff et al. «BDI agents: From theory to practice». Em: *ICMAS*. Vol. 95. 1995, pp. 312–319.

[6] *Flood fill - Wikipedia*. 2015. URL: `https://en.wikipedia.org/wiki/Flood_fill` (acedido em 09/12/2015).

[7] *Grid pathfinding optimizations*. 2015. URL: `http://www.redblobgames.com/pathfinding/grids/algorithms.html` (acedido em 09/12/2015).

[8] Daniel Imms. *A* pathfinding algorithm*. 2015. URL: `http://www.growingwiththeweb.com/2012/06/a-pathfinding-algorithm.html` (acedido em 09/12/2015).

[9] Ray Wenderlich. *Introduction to A* pathfinding*. 2015. URL: `http://www.raywenderlich.com/4946/introduction-to-a-pathfinding` (acedido em 09/12/2015).

Apêndice

A Manual do Utilizador

De modo a executar a aplicação, na interface do *Jadex* adiciona-se a pasta *bin*. De seguida, seleciona-se o ficheiro *bin/evacuation/Evacuation.application.xml* e inicia-se a aplicação.

Depois de todos os agentes terem morrido ou escapado, a aplicação termina e é adicionada uma nova linha ao ficheiro *EvacuationStatistics.csv*, no diretório base do projeto, com a informação estatística relativa à evacuação executada.

A aplicação cria o espaço baseando-se no ficheiro *resources/MainMap.txt*. A correspondência entre carateres e objetos criados no espaço é descrita na *Secção 4.3.1*.

Para testar a aplicação de uma forma mais eficaz, foram preparados alguns ficheiros com espaços previamente definidos, que estão contidos na pasta *resources*:

- *teste0 - saida e termina.txt*
- *teste1 - exploração.txt*
- *teste2 - herding.txt*
- *teste3 - herding com parede.txt*
- *teste4 - cure e hurt com parede.txt*
- *teste5 - velocidades.txt*
- *teste6 - estrategia activos.txt*
- *teste7 - estrategia conservativos.txt*

De modo a correr estes exemplos, deve-se copiar o conteúdo dos mesmos para o *MainMap.txt* e executar, de seguida, a aplicação *Jadex*.