

FACULDADE DE ENGENHARIA DA FACULDADE DO PORTO



FEUP

Utilização de agentes autónomos na simulação pedonal em interfaces multi-modais

Edgar Ferreira Esteves

VERSÃO FINAL

Relatório de Dissertação
Mestrado Integrado em Engenharia Informática e Computação

Orientador: Prof. Rosaldo Rossetti (Prof. Auxiliar da FEUP)

Março de 2009

Utilização de agentes autónomos na simulação pedonal em interfaces multi-modais

Edgar Ferreira Esteves

Relatório de Dissertação
Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo Júri:

Presidente: Eugénio da Costa Oliveira, Prof. Catedrático da FEUP

Arguente: José Nuno Panelas Nunes Lau, Prof. Auxiliar da Universidade de Aveiro

Vogal: Rosaldo José Fernandes Rossetti, Prof. Auxiliar da FEUP

20 de Março de 2009

Resumo

Um dos factores mais penalizantes no que se refere aos transportes colectivos é o tempo de espera para além, evidentemente, da noção generalizada de custo representada pelo tempo total de viagem. As interfaces de transbordo entre diferentes modos de transportes devem permitir uma rápida e cómoda transferência modal, permitindo que o uso do transporte colectivo se possa tornar cada vez mais atractivo e captar passageiros ao transporte individual privado. Contudo, este ponto não tem sido devidamente contabilizado aquando da concepção deste tipo de infra-estruturas, existindo actualmente poucos sistemas de apoio à decisão que tratam este tipo de problema. Com este projecto pretende-se dar os primeiros passos para o desenvolvimento de uma aplicação para modelação pedonal de estações multi-modais, que permita otimizar os transbordos tendo em atenção factores como o espaço, os modos de transporte e fluxos de tráfego pedonal, assim como a influência de todos esses parâmetros no processo de tomada de decisão dos indivíduos que utilizam a estação. Sendo evidentes as dificuldades em desenvolver uma aplicação completa e operacional em tão curto espaço de tempo, propõe-se a construir um protótipo que modele o problema referido a partir da utilização do conceito de agentes inteligentes, fazendo uso de características como autonomia, adaptação, interacção social e emergência.

Abstract

One of the most penalizing factors in what concerns public transport is the waiting time besides, of course, the generalised perception of cost represented by the total travel time. The public transport hubs connecting several modes should favour quick and convenient mode interchange, allowing transport to become more and more attractive and captivating travellers from private transport. However, such an issue has not been properly accounted for in the design of those infrastructures, and only very few systems currently available support decision-making to tackle this type of problem. In this project, one intends to give first steps towards the development of an application for modelling pedestrians in multi-modal stations, allowing the optimisation of transshipments whilst accounting for factors such as space, modes of transport and pedestrian traffic flows, as well as the influence of all these parameters upon the decision-making process of individuals using the station. As the hardness of developing a complete and operational application in such a very short period of time is evident, this work suggests the implementation of a prototype that models the problem mentioned above, relying on concepts such as intelligent agents and profiting from characteristics such as autonomy, adaptation, social interaction and emergency.

Agradecimentos

Gostaria de expressar a minha enorme gratidão a todos aqueles que contribuíram, de alguma forma, para a realização deste trabalho.

Em primeiro lugar, gostaria de agradecer ao meu orientador, o professor Rosaldo Rossetti, pelo apoio, colaboração e orientação que sempre me reservou.

Gostaria de agradecer também à TRENMO Eng. Ltd., ao Professor Álvaro Costa, do Departamento de Engenharia Civil e às Engenheiras Oana Santos e Vera Ferreira.

À Faculdade de Engenharia da Universidade do Porto, agradeço a disponibilização dos recursos necessários à execução deste trabalho.

À Agência de Inovação, agradeço o financiamento inicial deste trabalho, no âmbito do Projecto ModP – Modelação Pedonal, da Rede de Competência em Mobilidade.

“Agradeço a Deus, pai (todo poderoso), à mãe Maria e a ti, irmão, por todas as coisas possíveis e impossíveis. E a vós, amigos, que facilitastes este percurso sinuoso.

“Já devias ter isso feito! Já estás atrasado... esquece o design!” – sermão do Sr. Paulinho aos peixes.

“Explica-me como fizeste isso! O que querias fazer? Isso é...” – rogou Pedro aos Céus.

“Eu ajudo-te!” – implorou Daniel do alto de toda a sua disponibilidade.

Por minha culpa, minha tão grande culpa, peço à santa Catarina o meu perdão pelas noites mal passadas e pelos tempos de solidão.

Na última ceia e quando já passava da meia-noite, Late levantou-se e disse: “Tomai e bebei todos, esta é a forma como tenho para vos agradecer.” E sem mais nada para dizer, sentou-se.”

Conteúdos

1	Introdução	1
1.1	Objectivos.....	2
1.2	Organização da dissertação	2
2	Simulação Pedonal.....	5
2.1	Caracterização dos modelos de simulação	6
2.2	Agentes e Simulação	8
2.3	Plataformas de Simulação	8
3	Especificação da Solução.....	13
3.1	Softwares de Informação e de Manipulação.....	14
3.2	ModP – Modelação Pedonal.....	15
3.2.1	Domínio Real.....	16
3.2.2	Domínio Virtual.....	16
3.3	Modelo.....	17
3.3.1	Estrutura básica do cenário.....	18
3.3.2	Categorias dos componentes e parametrização	19
3.3.3	Regras de utilização dos componentes	20
3.4	Interface Gráfica de Edição	21
3.4.1	Esquematização da IGE.....	21
3.4.2	Mecanismo de Interacção com o Utilizador	23
3.5	Motor Controlador da Simulação	25
3.5.1	Arquitectura Lógica.....	26
3.5.2	Mecanismo de Interacção	27
3.5.3	Visualização e Análise de Dados.....	29
3.6	Agentes Autónomos	30
3.6.1	Aplicação de Forças e Comportamentos Direcctionais	32
3.6.2	Construção da Decisão	34
3.7	Síntese	36
4	Desenvolvimento do Protótipo.....	37
4.1	Tecnologias Utilizadas	37
4.2	Descrição das Interfaces Gráficas.....	38
4.2.1	Interface Gráfica de Edição	38
4.2.2	Interfaces Gráficas de Visualização 3D e Estatísticas.....	40

Introdução

4.3	Arquitectura Lógica	40
4.3.1	Módulo Editor do Cenário	42
4.3.2	Módulo Controlador da Simulação	46
4.3.3	Módulo Plug-in	49
4.3.4	Módulo Agentes Autónomos	52
4.4	Ciclo de Execução da simulação	57
4.5	Visualizador 3D e Analisador de Dados	58
4.6	Síntese	60
5	Resultados	61
5.1	Cenários, Parametrizações e Simulação	61
5.2	Dados obtidos através da simulação	63
5.3	Síntese	67
6	Conclusões	69
6.1	Desenvolvimentos Futuros	70
6.2	Perspectivas Futuras	71

Lista de Figuras

Figura 2.1 – Hierarquia dos estudos pedonais (baseado em [Tek02], Figura 2-1).....	6
Figura 3.2 – Modelo de interacção Utilizador-Software.....	14
Figura 3.3 – Diferenciação entre dois tipos de visualização de fluxo de peões.	15
Figura 3.4 – Relação entre os agentes especializados e os vários domínios.	15
Figura 3.5 – Arquitectura de alto nível do ModP.....	16
Figura 3.6 – Interacção entre os módulos IGE, MCS e Modelo.	17
Figura 3.7 – Ilustração dos componentes das interfaces multi-modais.	18
Figura 3.8 – Diagrama de classes de domínio do cenário.	19
Figura 3.9 – Representação esquemática da disposição IGE.	21
Figura 3.10 – Mecanismo de desenho de linhas através do compositor do cenário.....	24
Figura 3.11 – Mecanismo de agregação de componentes no cenário.	24
Figura 3.12 – Edição das propriedades de um controlador de fluxo (Entrada/Saída).	25
Figura 3.13 – Exemplo de edição de regras dos componentes.....	25
Figura 3.14 – Arquitectura do motor controlador da simulação.	26
Figura 3.15 – Sequência de interacção entre GDM, GS e AA.....	28
Figura 3.16 – Exemplo de um ciclo de execução de simulação.....	29
Figura 3.17 – Arquitectura dos módulos Visualizador 3D e Analisador.	30
Figura 3.18 – Arquitectura do módulo Agente Autónomo.	31
Figura 3.19 – Representação dos espaços Mundo Virtual, Local e Inércia.	32
Figura 3.20 – Ilustração de alguns comportamentos de direcção.....	33
Figura 3.21 – Ciclo de execução do plano (baseado em [Wei06], figura 4.14).	35
Figura 4.1 – Apresentação da plataforma ModP.....	38
Figura 4.2 – Interface Gráfica de Edição.	39
Figura 4.3 – Interfaces Gráficas de Visualização 3D e Estatísticas.	40
Figura 4.4 – Esquema representativo da arquitectura lógica.	41
Figura 4.5 – Arquitectura em camadas do módulo Editor do Cenário.....	42
Figura 4.6 – Esquema representativo do comportamento de não replicação de pontos.....	43
Figura 4.7 – Esquema representativo do comportamento de sugestão de zona.	44
Figura 4.8 – Características dos objectos criados pelo módulo editor cenário.	45
Figura 4.9 – Arquitectura em camadas do módulo Controlador da Simulação.	46
Figura 4.10 – Estrutura hierárquica de dados criada pelo Gestor de Dados.	47
Figura 4.11 – Estrutura de dados auxiliares criadas pelo Gestor de Dados.	48
Figura 4.12 – Mecanismo de gestão de <i>Plug-ins</i>	48
Figura 4.13 – Arquitectura em camadas do módulo <i>Plug-in</i>	50
Figura 4.14 – Particionamento de espaço e cálculo de vizinhos.....	51
Figura 4.15 – Camadas lógicas do módulo Agentes Autónomos.	52
Figura 4.16 – Propriedades do objecto móvel e mecanismo de aplicação de forças.....	53
Figura 4.17 – Comportamento direccionado “mover para alcançar alvo”.....	54

Introdução

Figura 4.18 – Comportamento direccional “mover evitar colisões”.	54
Figura 4.19 – Comportamento direccional “mover para evitar paredes”.	55
Figura 4.20 – Relação entre o comportamento “mover para evitar paredes” e a pilha de desejos.	56
Figura 4.21 – Ciclo de execução da simulação.	57
Figura 4.22 – Camadas dos módulos Visualizador 3D e Analisador de Dados.	59
Figura 5.1 – Exemplo da variedade de cenários editados.	62
Figura 5.2 – Afectação directa do valor do raio na simulação.	62
Figura 5.3 – Afectação da projecção da posição futura na resolução de cantos.	63
Figura 5.4 – Cenário de simulação.	64
Figura 5.5 – Estatísticas gerais do cenário de simulação com baixo fluxo de pões.	64
Figura 5.6 – Visualização 3D e mapa de densidade do cenário de baixo fluxo.	65
Figura 5.7 – Estatísticas gerais do cenário de simulação com médio fluxo de pões.	65
Figura 5.8 – Visualização 3D e mapa de densidade do cenário de médio fluxo.	66
Figura 5.9 – Panorama geral da simulação de elevado fluxo.	66

Lista de Tabelas

Tabela 2.1 – Sumário da análise de aplicações de simulação pedonal.....	8
Tabela 2.2 – Análise das plataformas OpenSteer e PedSim.	12
Tabela 3.3 – Resumo dos comportamentos direccionais.	32
Tabela 4.1 – Tabela de relação “Nome”→Tipo.	51
Tabela 4.2 – Tabela descritiva dos comportamentos direccionais.	53

Abreviaturas e Símbolos

ModP	Software de Modelação Pedonal
SINF	Software de Informação
SMAN	Software de manipulação
DV	Domínio Virtual
DR	Domínio Real
MR	Modelo Real
MV	Modelo Virtual
AE	Agente Especializado
DM	Delimitadores do Movimento
E/S	Componentes de Entrada e Saída
IGE	Interface Gráfica de Edição
MCS	Motor Controlador da Simulação
AA	Agente Autónomo
OQVVEOQVT	O Que Você Vê É O Que Você Tem
MCF	Menu Contextual de Ferramentas
VCC	Visualizador da Composição do Cenário
PI	Ponto de Interesse
CPI	Conjunto de Pontos de Interesse
CF	Controladores de Fluxo
CCF	Conjunto de Controladores de Fluxo
BRAF	Barra de Acesso Rápido a Ferramentas
MP	Menu de Projecto
ECP	Editor Contextual de Propriedades
CC	Compositor do Cenário
XML	<i>Extreme Modelling Language</i>
UML	<i>Universal Modelling Language</i>
BDI	<i>Beliefs Desires and Intentions</i>
API	<i>Application Programming Language</i>
DLL	<i>Dynamic-link library</i>

Capítulo 1

Introdução

A principal motivação deste trabalho parte da crescente necessidade de otimizar o conforto e a mobilidade das pessoas no que se refere aos transportes colectivos. Dado o panorama actual, a predição da afectação do planeamento urbanístico no conforto e mobilidade (planeamento pedonal) é, possivelmente, uma das tarefas mais importantes, dada a rotina stressante e evidenciada, no dia-a-dia contemporâneo.

Os factores mais penalizantes, no que se refere aos transportes colectivos, são os tempos de espera e não tanto o tempo de viagem no global. As interfaces de transbordo entre diferentes modos de transportes devem permitir uma rápida e cómoda transferência modal permitindo que o uso do transporte colectivo se possa tornar cada vez mais atractivo e tentando captar passageiros ao transporte individual. Contudo, este ponto não tem sido devidamente contabilizado aquando da concepção deste tipo de infra-estruturas, existindo actualmente poucos sistemas de apoio à decisão que trate este tipo de problema.

No que diz respeito a situações com alta densidade de peões, o volume de tráfego de peões tem especial interesse. Nestes casos, não basta apenas analisar o fluxo de peões como também se torna necessário analisar as conexões existentes nestas infra-estruturas. Neste âmbito, surgem questões como qual a conexão a utilizar para aceder ao próximo terminal ou piso e o quão rápida é esta transição, assim como de que forma os horários devem ser adaptados às diferentes situações.

Com este projecto pretende-se dar os primeiros passos para o desenvolvimento de uma aplicação para modelação pedonal de estações multi-modais que permita otimizar os transbordos tendo em atenção o espaço, os modos de transporte e fluxos de tráfego pedonal, assim como a influência de todos esses parâmetros no processo de tomada de decisão do indivíduo que utiliza a estação.

Perspectivando os vários passos que pressupõem a especificação e implementação de uma aplicação deste tipo, definiu-se um objectivo primário: especificar uma ferramenta de simulação que sirva de apoio à tomada de decisão, no planeamento de urbanístico em interfaces multi-modais. Pretende-se uma abordagem baseada em agentes, fazendo uso de características como autonomia, adaptação, interacção social e emergência.

1.1 Objectivos

O principal objectivo deste trabalho é especificar uma aplicação de simulação baseada em agentes autónomos para a avaliação das interacções sociais entre peões. Deve ser dado especial ênfase aos aspectos parametrizáveis do comportamento individual dos peões, assim como à interface de utilização da ferramenta enquanto ambiente de modelação e simulação. As características a serem suportadas pela aplicação incluem:

- Modelos de agentes para representação de peões em interfaces multi-modais, com as suas respectivas características sensoriais, de acção e tomada de decisão;
- Editor de populações de (agentes) utilizadores;
- Editor e visualizador de layout das estações e zonas de transbordo;
- Editor das origens e destinos, assim com rotas e opções de circulação;
- Recursos para avaliação de alguns modelos de comportamento e tomada de decisão;
- Recursos para análise, calibração e validação de modelos.

Como objectivo mais funcional, apresenta-se a implementação de um protótipo funcional que disponha de alguns dos conceitos conceptualizados e assuma as principais características. Essas características incluem:

- Conter um certo grau de modularidade, que permita posterior reutilização para outro tipo de simulação, como simulação em condições de emergência;
- Estar provido de uma estrutura de dados flexível;
- Possuir um editor gráfico que permita a edição e parametrização do cenário de simulação;
- Implementar a dinâmica do ambiente de simulação e simulador do modelo de interacção;
- Incorporar uma arquitectura de agentes e o seu processo de decisão;
- Definição e modelação de um cenário de estudo;
- Testes e avaliação de resultados.

1.2 Organização da dissertação

Este relatório está estruturado da seguinte forma:

- Capítulo 2 – examina o domínio de estudo de simulação pedonal e introduz os principais conceitos. Apresenta também algumas das ferramentas de simulação pedonal existentes, bem como as características superficiais mais relevantes e similares de cada uma delas.
- Capítulo 3 – analisa o panorama geral da área de foco deste tipo de software, como as características mais relevantes que deverão ser tidas em conta na implementação e especificação de uma aplicação de informação e manipulação. Descreve o domínio de referência e aplicabilidade da aplicação, sugerindo abstrações de alguns conceitos. Especifica uma arquitectura de alto nível, tanto do editor como do simulador e dos agentes, descrevendo as funcionalidades principais.

Introdução

- Capítulo 4 – apresenta o protótipo desenvolvido de acordo com a especificação, detalhando a arquitectura implementada. Exemplifica algumas das principais características e demonstra, em detalhe, alguns dos principais algoritmos. Também descreve as estruturas de dados e transformação dos mesmos durante a passagem entre módulos.
- Capítulo 5 – apresenta os resultados mais relevantes quer de parametrizações internas aos agentes e a sua afectação do comportamento, como da construção de diversos cenários e diversas parametrizações.
- Capítulo 6 – apresenta as conclusões, sugere directivas para o desenvolvimento futuro e apresenta perspectivas futuras.

Capítulo 2

Simulação Pedonal

Os estudos de modelação pedonal começaram a ganhar relevância desde algumas décadas atrás. O conhecimento principal nos sistemas de tráfego pedonal tem origem em estudos de observação empíricos, que podem ser consultados com um maior detalhe em [DHe01] e [DHe02]. De acordo com [Tek02], os estudos pedonais podem ser divididos em colecções de dados de peões e análise pedonal.

Enquanto a colecção de dados de peões foca a observação das características dos mesmos, a análise pedonal está preocupada com a interpretação dos dados recolhidos, tanto para entender as situações observadas, como para planeamento de novas instalações (como por exemplo no planeamento urbanístico).

Estes estudos têm como objectivo base prever os comportamentos pedonais de pessoas ou multidões, em situações normais ou em condições de emergência. Estes têm especial relevância no movimento de pessoas dentro de edifícios públicos, como estações de comboios, centros comerciais e aeroportos, entre outros.

Com a intenção de modelar e simular os comportamentos de multidões, são muito utilizados dois níveis distintos de análise pedonal:

- **Nível Macroscópico** – examina as características médias do fluxo de peões;
- **Nível Microscópico** - analisa o movimento e o comportamento a nível individual.

A utilização da simulação é fundamental no planeamento e na correcta correlação entre os parâmetros do domínio. No entanto, a maioria das análises são realizadas a um nível individual (independentes umas das outras), na tentativa de reduzir o número de variáveis observadas, de modo a simplificar o processo de procura das suas correlações. Isto levanta a questão de como os diferentes pontos de vista de análise do domínio, poderão ser agregados no mesmo modelo e ambiente de simulação, permitindo uma perspectiva de análise mais alargada [Bar91][Gra04][Ros99]. Este não é um problema recente. A ferramenta genérica para uma teoria de transporte identifica dois conceitos emprestados de Economia, que engloba todos os aspectos relacionados com a procura e a dinâmica da oferta dentro da ferramenta, incluindo selecção multi-modal e planeamento de actividades [MGM00]. Com alguma discussão, os modelos realistas são os primeiros instrumentos que permitem a integração de diferentes perspectivas de análises em qualquer domínio de aplicação. Porém, a modelação não é uma tarefa fácil e a abstracção é uma necessidade frequente de forma a tornar exequível a aplicação

prática do modelo. A metáfora “agentes autónomos” tem vindo a ser cada vez mais utilizada, como a abstracção de eleição para a construção do modelo adoptado, mantendo as importantes características comportamentais e cognitivas, que devem ser preservadas pelas entidades do sistema. Os avanços da Engenharia, nos sistemas multi-agente, têm fomentado a ideia do comportamento global do sistema, que emerge da interacção de entidades modeladas a nível microscópico.

Este capítulo descreve as possíveis caracterizações do modelo e respectivos tipos. Introduce, também, de que forma o conceito de agente pode ser caracterizado no âmbito de aplicações de simulação. Por último, apresenta algumas das ferramentas de simulação pedonal existentes e descreve sucintamente as suas principais características.

2.1 Caracterização dos modelos de simulação

Um modelo de simulação descreve, geralmente, um sistema dinâmico. Um modelo de simulação pode ser caracterizado como sendo contínuo ou discreto. As simulações discretas representam a realidade, calculando as acções ou actividades produzidas pelas suas entidades, que alteram o estado do ambiente após um intervalo de tempo fixo, ou depois de um evento.

As simulações contínuas representam a realidade, calculando o resultado de uma equação que representa o estado do ambiente em qualquer dado ponto da simulação [Cha99].

A simulação pode ser caracterizada como sendo Estocástica ou Determinística. Os modelos estocásticos possuem um conjunto de processos que, por sua vez, contêm funções probabilísticas. Assim, o estado de simulação actual não determina o seu próximo estado, significando que duas simulações com exactamente os mesmos parâmetros de entrada e a mesma calibração irão produzir resultados diferentes. Este factor torna este tipo de modelos bastante imprevisíveis. Por outro lado, os modelos de simulação determinísticos são bastante previsíveis. Isto porque não possuem variáveis aleatórias, significando que o estado da simulação determina o estado seguinte. Podem ser utilizados para calibração de modelos ou para quando existir a necessidade de reprodução dos resultados esperados [Cha99].

A Figura 2.1 ilustra esquematicamente as diferentes divisões que existem nos estudos pedonais.

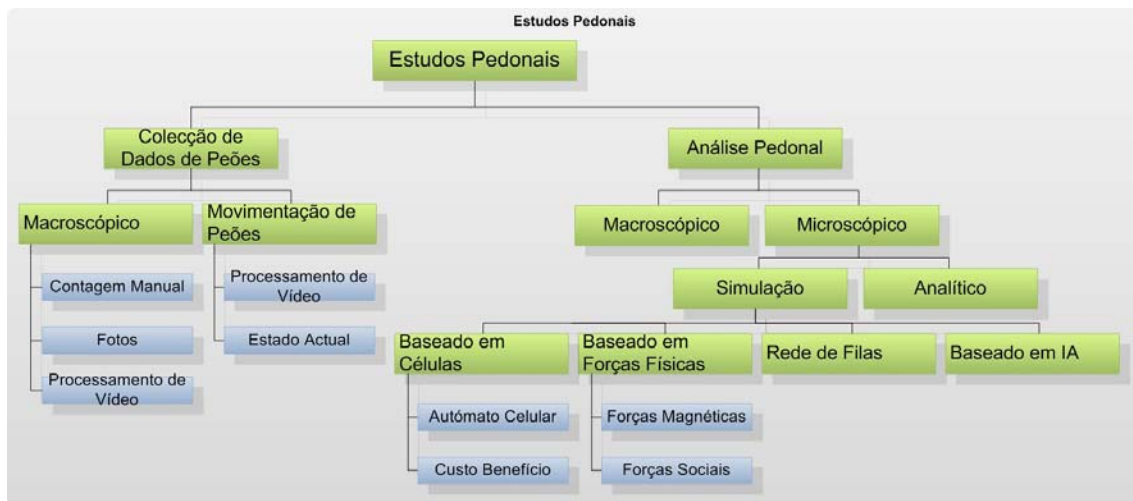


Figura 2.1 – Hierarquia dos estudos pedonais (baseado em [Tek02], Figura 2-1).

Nos estudos pedonais estão patentes dois níveis de análise distintos: o nível macroscópico e o nível microscópico [Fer082][Kes02][HK100]. O primeiro compara o fluxo pedonal com os comportamentos globais de fluxo de gases e líquidos, examinando a velocidade e densidade média. Os microscópicos, por sua vez, focam o seu estudo na velocidade e interacção individuais, comparando o fluxo de multidões com a dinâmica de fluidos e cinética de gases.

Para motivos desta dissertação, serão apenas focados os modelos microscópicos. Tal como ilustrado na Figura 2.1, estes podem ser categorizados em quatro tipos distintos:

- **Modelos Baseados em Células** – Neste modelo, os peões são simulados como uma partícula numa célula. O ambiente é representado como uma grelha de células (ex: 0.5mX0.5m por célula) e o movimento dos peões é modelado “saltando” de célula em célula. A célula seguinte é calculada de acordo com uma pontuação [Wei06][Tek02][Tek00]. Uma pontuação, à qual lhe são atribuídos valores negativos a partir da proximidade de peões (similar a forças repulsivas) entre células vizinhas e a própria e valores positivos, baseados na distância da célula ao alvo pretendido (expressando forças de atracção). A desvantagem deste modelo é de apenas permitir posicionamentos discretos e estáticos dos peões. No entanto, reduz significativamente o tempo de processamento para actualização de posições, quando comparado com modelos que permitem posicionamentos contínuos no espaço.
- **Modelos Baseados em Forças Físicas** – Em contraste com o modelo anterior, estes podem prever o posicionamento e localização dos peões com precisão [Qui03]. Existem duas aproximações a este modelo: Modelo de Forças Magnéticas e o Modelo de Forças Sociais. A primeira aproximação desenvolvida por Prof. Okazaki e Matsushita, aplica modelos magnéticos e equações de movimento a campos magnéticos, para a locomoção de peões. Ambos os peões e obstáculos possuem um pólo positivo e um pólo negativo, que são atribuídos de acordo com os objectivos dos mesmos. O movimento pedonal é afectado por duas forças, permitindo que cada indivíduo se mova para atingir o seu objectivo, evitando colisões com obstáculos ou outros peões existentes no meio, como explicado por Teknomo [Tek00]. A aproximação de Forças Sociais combina os princípios do Modelo de Custo/Benefício Celular e o Modelo de Forças Magnéticas, aplicando a cada peão forças sociais (forças repulsivas, atribuídas a outros peões e obstáculos; e forças de atracção, atribuídas ao objectivo), que combinadas actuam como motivação e criação de desejos, direcções e de acelerações. A cada instância de tempo, os valores podem ser alterados dinamicamente, com a preocupação de que os valores da velocidade máxima e mínima nunca se cruzem. Esta aproximação assume que todos os peões possuem um objectivo ou destino desejado.
- **Modelos Baseados em Redes de Filas** – Estes modelos são muito utilizados na simulação de multidões em situações de evacuação [DHe02], como por exemplo, em edifícios. É similar a um espaço discreto, pois cada espaço é designado por nó, sendo as portas entre os espaços, designados por conectores. Cada peão parte de um nó ou fila de um conector e atinge um outro nó. O objectivo dos peões, é atingir o ponto de saída da forma mais rápida e segura possível. A escolha do próximo nó é baseada no peso calculado pela densidade de peões, que se encontra no nó vizinho. Se um peão não conseguir aceder ao nó pretendido, espera ou escolhe outro nó para se dirigir.
- **Modelos Baseados em Inteligência Artificial** - Estes modelos complementam os restantes, trazendo novas funcionalidades à simulação pedonal. A contribuição principal dada aos outros modelos, é a melhoria de pesquisa no espaço e tomadas de decisões por parte dos peões. Isso é garantido com algoritmos como A*, D*, Dijkstra, entre outros algoritmos de pesquisa. Uma aproximação muito utilizada aos modelos baseados em inteligência artificial, é o modelo baseado em agentes. Neste modelo, a unidade básica de actividade é o agente [Sch99]. Geralmente, é utilizado um grande número de agentes (que representam uma possível multidão), permitindo modelar as interacções existentes entre eles mesmos e entre eles e o mundo. O objectivo deste modelo é construir comportamentos humanos mais complexos, uma vez que uma simulação realista requer um conjunto de parâmetros, como por exemplo parâmetros psicológicos, fisiológicos e sociais [Sar08].

A vantagem do modelo microscópico, quando comparado com o seu homólogo macroscópico, depende da sua capacidade de fornecer resultados com muito maior confiança e precisão, em casos em que a interacção dos peões é crucial nos resultados de simulação. Um dos casos, em que estas interacções são importantes, é a análise do fluxo pedonal em áreas com elevada densidade de peões. Outra vantagem, está relacionada com a capacidade de visualizar e

analisar resultados de forma atractiva e realista. O maior problema do modelo microscópico é a sua complexidade na calibração e validação de processos [Bur04].

2.2 Agentes e Simulação

O conceito baseado em agentes é relativamente novo. A sua aplicabilidade é de grande interesse para a comunidade científica e para a indústria de software. Não existe um consenso nem uma definição rígida para o termo agente, mas Wooldridge define-o como:

“Um agente é um computador capaz de acções autónomas flexíveis, situado num ambiente dinâmico, aberto e imprevisível” [Woo99]

As capacidades cognitivas dos agentes podem incluir a percepção, raciocínio, compreensão, aprendizagem, processamento de objectivos e processamento de conhecimento direccionado aos objectivos. Em [Ore00], os autores descrevem que os agentes e a simulação podem ser combinados em três aproximações básicas:

- **Simulação de Agente** – A simulação de agente pode ser definida como a simulação de entidades inteligentes, artificiais ou humanas que podem ser representadas por agentes, sendo este, o modo natural de simulação e modelação deste tipo de domínios [Ore00].
- **Simulação Baseada em Agentes** – A simulação baseada em agentes é definida quando os agentes são utilizados para gerar modelos comportamentais numa simulação. Estes tipos de sistemas, são similares a sistemas especializados e a sistemas de simulação qualitativos, mas oferecem possibilidades adicionais à simulação numérica [Ore00].
- **Simulação Suportada por Agentes** - A simulação suportada por agentes, pode ser definida como o uso de agentes no apoio a operações de simulação, como operações de suporte/apoio ao utilizador e sistema, assim como de outras actividades relacionadas com o software de simulação [Ore00].

2.3 Plataformas de Simulação

Tabela 2.1 – Sumário da análise de aplicações de simulação pedonal.

Software	Modelo de Simulação	Simulação em Tempo Real	Condições Emergência	Visualização 3D	Edição do Cenário
STEPS	Baseado em agentes	sim	sim	sim	sim
VISSIM	Modelo de Forças Sociais	sim	sim	sim	sim
SimWalk	***	sim	sim	sim	sim
Legion	***	sim	sim	sim	sim
Micro-PedSim	Baseado em agentes	sim	sim	sim	sim
AnyLogic	Sistemas dinâmicos Eventos Discretos Baseado em agentes.	sim	sim	sim	sim
SimPed	Baseado em agentes	sim	***	***	sim
UAF	Baseado em agentes	sim	sim	sim	sim

Este grupo de aplicações foi alvo apenas de um estudo superficial, de forma a serem apurados os requisitos funcionais mais relevantes para este tipo de aplicações. Neste ponto de vista, podem retirar-se algumas conclusões importantes e que ajudam a distinguir alguns aspectos óbvios das aplicações mais usadas no tipo de estudos em causa. A construção e edição

de um cenário de simulação (através de uma interface gráfica) são um ponto forte de todas as aplicações e que se traduz numa grande eficácia e dinâmica de simulação. A simulação 3D em tempo de execução enriquece todo o ambiente de simulação, possibilitando uma melhor e mais profunda visualização de diversos pontos-chave do cenário. Focando os tipos de simulação disponibilizados pelas diversas plataformas (situações normais ou de emergência), pode ser deduzido que a integração dos dois favorece um maior grau de análise à simulação.

STEPS

STEPS [Mac08] é um dos simuladores mais utilizados mundialmente para a modelação de estações de metro e sistemas de carris subterrâneos. Esta aplicação comercial utiliza uma aproximação baseada em agentes que prevê os movimentos dos vários indivíduos num espaço tridimensional. Esta aproximação utiliza alguns princípios da teoria de autómatos celulares que se encontra agora estabelecida na dinâmica da modelação de peões.

O STEPS foi utilizado na concepção de diversas infra-estruturas, entre as quais: London Heathrow Terminal 5: station and track transit system; Minneapolis Light Rail Transit, USA; Rotterdam Metro, The Netherlands; Delhi metro, India; KCRC Shatin-Central Link, Hong Kong; Luton Network Rail station, UK; e International Centre for Life, Newcastle, UK.

Algumas funcionalidades chave do STEPS: Micro simulação baseada em agentes; Condições normais e condições de emergência; Recolha extensiva dos vários percursos realizados; Eficiente em modelos complexos e de grande escala; Importação directa de modelos CAD 2D e 3D; Interactividade 3D (realidade virtual); Sistema de rotas em detrimento das matrizes origem-destino pesadas computacionalmente; Veículos móveis; Variedade de medições dos movimentos dos peões com representação gráfica; Exportação de dados analíticos no formato CSV (.csv) (para posterior importação em folhas de cálculo); Gravação de simulações (em formato AVI); Obtenção de imagens de sequências animadas (extensões JPG, BMP, PNG, TIFF).

VISSIM

O VISSIM [PTV08] é um software de código fechado que, recentemente, centrou as suas atenções na simulação de tráfego pedonal, acompanhando a crescente proeminência do impacto destes estudos, no planeamento urbanístico. Este, foca a sua atenção, não só na análise de dados resultantes da simulação pedonal, mas também na influência destes, sobre o tráfego veicular. O VISSIM é uma ferramenta de simulação microscópica em ambientes multi-modais que permite simular, realisticamente, as interações entre os peões e o fluxo de veículos. A análise de simulação pode incidir sobre ambos os tipos de tráfego (pedonal e veicular), ou apenas sobre um único tipo (independentemente do outro). O VISSIM é baseado no modelo de forças sociais e permite simulações em tempo real de situações de emergência, com o recurso a gráficos tridimensionais.

SIMWALK

O SimWalk[Sav06] é um simulador pedonal cujo objectivo é permitir a avaliação da segurança e do conforto dos peões, em ambientes onde estes predominam. A análise dos resultados gerados pelo simulador permite uma tomada de decisão mais assertiva, visando o conforto e segurança dos peões.

O SimWalk é uma aplicação de código fechado, focado na simulação dos ambientes mais críticos, onde existem elevadas densidades de peões, como por exemplo, áreas públicas, estádios de futebol, estações de comboios e aeroportos. O SimWalk permite, também, o cálculo do risco humano em situações de emergência.

Este simulador, integra uma base de dados incluindo todos os dados empíricos importantes dos últimos 20 anos de pesquisa internacional sobre peões, como por exemplo, velocidade de locomoção e diâmetro médio do corpo nos diferentes países. A base de dados, suporta e simplifica estudos de simulação e calibragem, com dados comprovados, acerca dos peões.

Aplicações do SimWalk comuns: Testes de segurança nos pontos de entrada/saída em estações de comboio, aeroportos e outros edifícios complexos; Engenharia de tráfego; Concepção e validação de espaços públicos; Integração de cenários pedonais na simulação de tráfego veicular.

SimWalk oferece uma vasta gama de ferramentas de análise para avaliação dos resultados de simulação: Mapa de densidade (congestionamentos); Velocidade de movimento (média e por peão); Tempos de movimento (média e por peão); Tempos de atraso (médio e por peão); Contagens e taxas de fluxo; Contagem de saídas; Utilização de espaço; Pistas de peões; Níveis de serviço (pisos); Eficiência e tempo em filas de espera.

Os criadores deste software assumem como principais benefícios na utilização do SimWalk: Simulação e análise abrangente dos problemas dos peões, no que diz respeito à sua segurança, conforto e fluidez de locomoção; Análises de congestionamento, possíveis efeitos de funil e elevadas densidades em edifícios complexos; Estudos de praticabilidade em espaços urbanísticos ou concepção de edifícios para fomentar a fluidez de locomoção e redução de custos; Flexibilidade de aplicação em domínios de simulação distintos (tráfego, planeamento urbano, evacuação); Integração de desenhos CAD; Visualização de resultados a 2D e a 3D.

LEGION

O Legion [Leg08] é uma aplicação de código fechado que permite a simulação em diversos ambientes. As simulações utilizando o Legion são eficazes para: Planear novas estações; Remodelar projectos; Boas análises temporais e planeamento de calendários / horários; Integração operacional com acontecimentos desportivos e festivos; Evacuação e segurança; Design e o posicionamento dos comerciantes.

O Legion foi utilizado para simulações em vários locais / infra-estruturas, destacando-se as seguintes: London Underground; New York's 34th Street Station; Jogos Olímpicos de Atenas, Sidney e Pequim; Jogos Olímpicos de Londres (em 2012); Luton Airport (London)

No contexto de eventos, o Legion é eficaz para: Projectar novos estádios; Renovação de projectos; Testar e assegurar a entrada e saída dos peões dentro de um espaço temporal estabelecido; Treino de operações; Integração com a rede de transportes.

No sector dos aeroportos, as simulações Legion são eficazes para: Projectar novos aeroportos; Expansões de capacidades de um aeroporto já existente; Testes de impactos de novas medidas de segurança; Projectar e testar medidas de segurança e evacuações; Projectar as zonas comerciais; Identificar os espaços com baixa e elevada utilização.

MICRO-PEDSIM

O Micro-PedSim [Tek] é uma ferramenta de pesquisa para simulações de tráfego de peões, baseada num sistema de multi-agentes. O Micro-PedSim fornece o poder de visualizar e simular microscopicamente o comportamento dos agentes. Pode-se gerar agentes manualmente, através de um ficheiro de texto ou automaticamente. O sistema permite especificar também os parâmetros envolvidos na simulação e realizar experiências sobre eles. A simulação pode ser calibrada no espaço e tempo. A visualização dos dados obtidos pelo programa, podem ser mostrados numa simulação em tempo real ou numa visualização bidimensional, para melhor análise de dados. As suas principais características são: Software livre; Diferentes tipos de peões; Geração automática de peões; Detecção automática de colisões; Sistema anti-colisões; Ultrapassagem automática de peões mais lentos; Simulação em tempo real; Ajuste de parâmetros através de uma interface gráfica; Calibração de tempo e espaço; Geração manual e automática de peões; Traçado do caminho efectuado; Tráfego unidireccional e bidireccional; Performance do sistema; Fácil de instalar; Todas as análises podem ser guardadas num ficheiro de texto para posterior análise em folhas de cálculo e/ou MatLab.

ANYLOGIC

O AnyLogic [Tec08] é utilizado mundialmente por vários utilizadores, centenas de organizações (comerciais e governamentais) e centenas de universidades. Com o AnyLogic, pode-se simular a população inteira de uma grande cidade, preservando as propriedades individuais de cada peão. Esta ferramenta inclui uma grande variedade de análises de dados, entre os quais, diversos tipos de gráficos e histogramas. Estas análises de dados, permitem visualizar dinamicamente a eficiência de processos em tempos diferentes de simulação. Quando o modelo de simulação estiver completo, pode-se definir várias experiências e testes, como a simulação, execuções comparativas, variações de parâmetros e optimizações.

O AnyLogic é uma ferramenta que suporta todas as metodologias de simulação existentes: Sistemas dinâmicos; Eventos Discretos; Baseado em agentes.

Esta ferramenta permite, assim, visualizar a complexidade e heterogeneidade do sistema, com o nível de detalhe desejado. A interface gráfica do AnyLogic, ferramentas e livrarias, permitem modelar rapidamente diversas áreas, sobre as quais vão incidir vários estudos.

O editor de animação suporta uma grande variedade de formas gráficas e controlos. É permitida a importação de imagens e ficheiros CAD como elementos e fundos. A animação no AnyLogic é escalável e hierárquica. Pode-se obter vistas gerais de processos, bem como, informações detalhadas sobre esses mesmos processos.

SIMPED

O SimPed [Win02] permite a visualização e quantificação dos efeitos da escolha de determinado plano de uma infra-estrutura sobre o padrão dos fluxos de passageiros. Este é baseado numa arquitectura distribuída, na qual, os modos para entrada, simulação, animação, arquivo e análise estão conectados.

Uma das características do modelo é a sua capacidade de modelação dos processos de transporte público, especialmente a sua influência no comportamento dos peões (como transferências de um autocarro para um comboio e peões a comprarem bilhetes). Assim, os peões tomam decisões acerca das suas rotas (mais curtas) e movem-se na infra-estrutura.

Actualmente, a criação de horários envolve várias aproximações ao tempo que os passageiros realmente necessitam para se deslocarem entre diferentes plataformas. Diferentes tipos de passageiros necessitam de tempos diferentes para se deslocarem. Ao utilizar-se tempos de transferência bastante aproximados do real, remove-se tempos de espera excessivos do horário e reduz-se a probabilidade dos passageiros perderem a sua ligação até ao destino, aumentando assim, o conforto dos passageiros e a fiabilidade do horário.

O modelo utilizado analisa os impactos: na concepção de estações; na concepção de horários e sua relação com as estruturas utilizadas; na concepção de grandes espaços para peões; nas operações de controlo de fluxo; Analisar performance da plataforma em termos de efeito funil.

No entanto, esta ferramenta ainda não possui um modelo comportamental microscópico, mas utiliza o modelo macroscópico nas relações existentes entre a densidade nas várias áreas pedestres e entre a velocidade dos peões. Deixa os peões decidirem qual caminho escolher na infra-estrutura e decidirem as suas actividades. Efeitos de funil em certos espaços, antes de escadas rolantes, ou perto de entradas e saídas, podem ser encontrados durante a fase de concepção, evitando assim gastos adicionais.

O SimPed é utilizado para avaliar: o esquema de (partes de) estações de existentes; efeitos das extensões e/ou adaptações de estações existentes; esquemas alternativos para estações acabadas de desenvolver; alternativas na alocação de diferentes plataformas; slternativas e mudanças em diferentes horários; esquemas alternativos para áreas grandes para peões sem conexão a transporte público.

Durante a simulação, o modelo realiza as seguintes actividades repetitivamente: determina a origem e o destino dos peões presentes; atribuir rotas na rede de estações; calcular tempos de locomoção baseada em modelos comportamentais; executa avaliações à dinâmica e actualiza a situação.

UAF

O UAF (Urban Analytics Framework) [Qua08] desempenha um papel fundamental na engenharia de tráfego e no planeamento urbano. Combinando o modelo de tráfego micro simulado com o modelo baseado em agentes, permite, ao UAF, reproduzir uma vasta gama de cenários. O UAF pode ser utilizado como modelo de veículos e interacções entre agentes, ou pode ser utilizado para modelar estas componentes separadamente, sem as respectivas dependências.

O principal objectivo do UAF é a identificação dos problemas de segurança existentes num vasto leque de cenários. Nesta aplicação, os peões podem ignorar o sinal para não atravessar passadeiras e podem atravessar as vias no meio do trânsito, desde que se sintam seguros para tal. Existem, também, diferentes tipos de condutores (mais agressivos, mais pacíficos). Estas técnicas desenvolvidas, permitem obter um retorno fiável de problemas de segurança.

Todas as ferramentas fornecidas pelo UAF operam em tempo real, não necessitando assim de tempo adicional para um processamento posterior dos dados obtidos. No UAF, todas as métricas chave para avaliação, estão disponíveis quando se necessita delas.

As áreas chave de aplicação do UAF são: concepção de espaços urbanos e avaliação de tráfego operacional; níveis de serviço e qualidade do ambiente; terminais/Estações de transportes públicos e sua interacção com a rede de estradas; planeamento de eventos especiais; planeamento de evacuações; análise de tempos de evacuação em espaços urbanos; análise da segurança dos agentes; e análise de veículos e agentes.

As suas principais capacidades gráficas assumem: fundos a partir de fotografias aéreas; mapeamento de terreno tridimensional; realismo nas estradas (cores e estilos de marcas rodoviárias); sombras realistas em todo o cenário; modelos tridimensionais estáticos e dinâmicos; animação dos agentes; e efeitos meteorológicos – Sol, nevoeiro, nuvens, neve e chuva.

A ferramenta para visualizar colisões permite aos utilizadores ver onde se desenrolam as colisões entre peões, entre veículos e entre ambos. Assim, é possível utilizar as ferramentas espaciais métricas para identificar problemas de densidade excessiva.

As ferramentas espaciais métricas são as ferramentas mais poderosas disponíveis no UAF. Registam o percurso realizado por cada agente enquanto se move pelo cenário, registando também como o espaço pode ditar a qualidade do serviço. É utilizada para criar mapas coloridos complexos para examinar a velocidade, densidades, atrasos e utilização de espaços.

Agentes métricos incluem: densidade, velocidade e atraso dos agentes; probabilidade de mapas de conflitos; utilização do espaço; tempo de viagem; e regras para agentes (não atravessar no sinal vermelho).

Num ponto de vista mais aprofundado, foi realizada uma análise a duas aplicações de código aberto, que possibilitam simulação microscópica.

Tabela 2.2 – Análise das plataformas OpenSteer e PedSim.

Software	Modularidade	Linguagem Programação	Performance	Sistema Plug-ins	Motor Gráfico 3D
PedSim	razoável	C++	razoável	não	***
OpenSteer	razoável	C++	razoável	sim	sim

Com o intuito de diferenciar estas plataformas, foi realizada uma análise dos parâmetros ilustrados na Tabela 2.2. É de notar dois parâmetros com avaliação distinta. O facto de o OpenSteer incorporar um motor de visualização 3D e de o PedSim necessitar de recorrer a um output para um visualizador 3D externo, nomeadamente o Ped3D, aufere uma possível vantagem OpenSteer. Mas o parâmetro mais importante, e que pode fazer incidir numa eventual escolha no OpenSteer, é o facto de este disponibilizar um sistema de plug-in que facilita a integração de vários tipos de simulação: em condições normais, ou de emergência por exemplo.

Contudo, o OpenSteer [Ope04] não é propriamente um simulador pedonal, mas mais uma biblioteca de cinética e que incorpora um *plug-in* de demonstração de fluxo de pedestres. Este facto revela-se importante na criação de comportamentos heterogéneos, singulares e distintos [Rey00]. Uma abordagem utilizando as funcionalidades mais significativas da arquitectura do OpenSteer pode revelar-se importante.

Capítulo 3

Especificação da Solução

Este capítulo expõe uma especificação de alto nível de um software de modelação de peões em interfaces multi-modais, denominado “ModP – Modelação Pedonal”. Este baseia-se em agentes autónomos e segue a abordagem de simulação microscópica.

Nas duas primeiras secções, 3.1 e 3.2, são introduzidos o conceito ModP, as filosofias de software em que se baseia (software de informação e manipulação) e é estipulado o grau de detalhe e de complexidade que este pretende alcançar. Pode-se também encontrar uma especificação dos módulos que o compõem, bem como de que forma estes se correlacionam. Na segunda secção, é possível observar os domínios de referência (Domínio Real e Domínio Virtual) e a interacção pretendida entre eles. É também discutido o propósito da utilização do modelo microscópico e de agentes autónomos assim como o tipo de análises que a solução proposta deverá proporcionar (comportamento ao nível do indivíduo e/ou multidão).

A secção seguinte (secção 3.3) especifica o modelo de simulação e o tipo de abstracções que se podem efectuar quer ao nível da composição (generalização e agrupamento dos componentes), quer ao nível das regras de utilização dos componentes (só podem ter uma regra – probabilidade de utilização de outro), simplificando o processo de construção e facilitando a tomada de decisão dos agentes autónomos.

A secção 3.4 expõe uma especificação de alto nível do módulo “Interface Gráfica de Edição” (IGE) realçando aspectos fulcrais na concepção e especificação de interfaces deste género. Os seus pontos-chave focam-se na usabilidade e na restrição da interacção ao essencial, eliminando a necessidade de formação ou leitura pormenorizada de um manual de utilizador para um uso complexo e eficiente. Tal é possível com a implementação de ferramentas genéricas, representação dos objectos virtuais através de formas simples e com o auxílio dos mais recentes padrões de desenho (*design patterns*) de interfaces.

A explicação do processo de simulação está descrita nas secções 3.5 e 3.6 onde é esclarecida a interacção síncrona entre o motor de simulação e os agentes. Na secção 3.5, pode-se analisar a arquitectura lógica do motor de simulação e interpretar todo o processo, desde o início de simulação (auxiliado pela estrutura básica do cenário e pela parametrização dos vários componentes) até ao recolher de dados concordantes com as análises que se pretendem efectuar. Na secção 3.6, estão descritos a arquitectura lógica para os agentes autónomos e o tipo de características específicas que devem possuir, como a autonomia de decisão. Esta autonomia permite aos agentes a aquisição de certas capacidades de improvisação em relação às suas

acções. No fundo, pretende-se que se comportem como actores humanos num cenário de improvisação em que a sua composição e regras são conhecidas.

Por último, a secção 3.7 fornece uma síntese sobre este capítulo, reforçando o paradigma da modelação através de processos simplificados e a importância de integração com modelos de agendamento, capazes de executar um plano baseado no conhecimento sobre o mundo virtual, como o modelo BDI.

3.1 Softwares de Informação e de Manipulação

Victor Bret refere, no seu aclamado artigo “Magic Ink” [Vic06], que todo o software pode ser categorizado, quanto ao seu objectivo, em três grandes categorias: software de informação, de manipulação e de comunicação. Para efeitos desta dissertação, só irão ser focados os dois primeiros.



Figura 3.2 – Modelo de interacção Utilizador-Software.

O **Software de Informação (SINF)** é todo o software que possibilita ao utilizador a construção/adaptação de modelos mentais, como ilustrado na Figura 3.2.a). Este deve estimular a aprendizagem, isto é, encorajá-lo a perguntar e a responder a questões; fazer comparações e retirar conclusões. Se se imaginar um especialista de tráfego pedonal a realizar um estudo sobre a locomoção de peões numa estação de comboios, podemos antever algumas das suas questões: “Qual a zona por onde passam mais peões?”; “Qual a taxa de afluência aos multibancos?”; “Será que o multibanco está bem colocado?”. Um bom SINF deverá fornecer análises automatizadas e de fácil compreensão. Deverá, também, permitir o acesso rápido à informação pretendida e não prolongar a decodificação dos dados apresentados. Por exemplo, se o mesmo especialista pretender analisar o fluxo e a sua distribuição numa determinada área de um edifício (mapa de densidade), a melhor forma de disponibilizar essa informação será através de um gráfico e não através de números. O contrário aconteceria no caso de se pretender observar o número de pessoas que passam por uma determinada zona. A Figura 3.3 apresenta a diferença de visualização nessas condições.

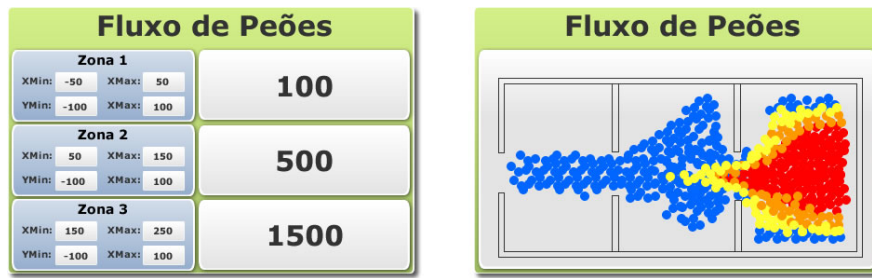


Figura 3.3 – Diferenciação entre dois tipos de visualização de fluxo de peões.

O **Software de Manipulação (SMAN)** deverá permitir a construção e a manipulação de modelos externos ao utilizador, como demonstra a Figura 3.2.b). A construção/manipulação desses modelos poderá ser entendida como a alocação de objectos virtuais (representativos dos objectos reais num determinado espaço físico) e a edição das suas propriedades ou parâmetros. Os SMAN podem, por isso, ser considerados ferramentas virtuais. Num espaço físico, cada operação é realizada com o auxílio de uma ferramenta especializada. Porém, o principal desafio deste tipo de software é desenhar uma “mega-ferramenta” que incorpore claramente todas as operações e que use apenas os gestos “clique” e “arrastar”. Um bom SMAN também deverá proporcionar uma boa visualização, permitindo ao manipulador ver os resultados da sua acção e promovendo a sua actividade criativa.

3.2 ModP – Modelação Pedonal

O conceito ModP, ferramenta de simulação pedonal baseada em agentes, assenta nos pressupostos apresentados na secção 3.1. Este apoia a tomada de decisão aquando do planeamento e concepção de um projecto para infra-estruturas que sirvam de interfaces de transbordo entre diferentes tipos de transportes.

Durante a interacção dos utilizadores, agentes humanos/virtuais especializados (conhecedores da dinâmica deste tipo de infra-estruturas), com o ModP deve ser estabelecida uma relação de confiança que lhes transmita certezas na adaptação do seu modelo interno e/ou externo. A figura seguinte, Figura 3.4, exemplifica não só as interacções dos diversos agentes com o ModP e destes com o ambiente externo, como os domínios em que se inserem.



Figura 3.4 – Relação entre os agentes especializados e os vários domínios.

Como é possível observar na Figura 3.4, existem unicamente dois domínios: **Domínio Virtual** e **Domínio Real**. A interligação entre estes é feita através dos agentes especializados, que apoiando-se na informação obtida do domínio virtual permite-lhes a intervenção na estrutura, composição e/ou regras do modelo real e vice-versa. O objectivo desta interligação é permitir a replicação de modelos reais em modelos virtuais para que de uma forma fácil, segura e barata seja possível realizar alterações de parâmetros nesses modelos. Através dessas alterações, é pretendido encontrar aquelas que têm um impacto positivo na optimização da locomoção dos peões, facilitando assim o acesso a meios de transporte e pontos de interesse existentes na infra-estrutura. Após estudado o impacto de determinadas alterações ao modelo do

domínio virtual e concluindo-se que estas trazem um impacto positivo, poder-se-á proceder à sua aplicação no modelo real de uma forma segura e eficiente.

3.2.1 Domínio Real

O domínio real (DR) é composto por modelos reais (MR's) e agentes especializados (AE's). Os MR's são formados por componentes reais que se enquadram numa determinada zona/local ou num determinado edifício. No caso específico desta dissertação, o MR base é que serviu de suporte para toda a especificação do ModP está relacionado com interfaces multi-modais, mais especificamente com estações de comboios que tenham ligação com interfaces de metro, autocarros e táxis.

São precisamente este tipo de infra-estruturas o objecto de estudo dos AE's e que servem de suporte aos seus modelos. Esses agentes, humanos ou virtuais, individualistas ou colaborativos, realizam o seu estudo na tentativa de prever a influência do planeamento urbano no conforto e mobilidade dos peões, tanto no interior dos edifícios públicos como em áreas abertas.

A análise de dados, recolhidos unicamente sobre cenários reais semelhantes, pode revelar-se insuficiente na medida em que existem inúmeras variáveis que podem afectar a amostra: externas (mês, dia da semana, hora do dia, eventos sociais do dia e condições meteorológicas) e internas (condição financeira, disponibilidade do veículo pessoal e tempo de viagem previsto). A implementação ou alteração do modelo real pode, por isso mesmo, revelar-se ineficaz e dispendiosa podendo trazer descrédito aos AE's. Uma análise eficaz sobre o modelo pré (se for o caso) e pós planeamento torna-se crucial. A melhor maneira de o fazer será através do recurso a um domínio virtual capaz de representar o MR e que permita análises e testes de novas parametrizações e configurações sobre o mesmo. Será possível, assim, não só obter um planeamento rápido, seguro e fidedigno, mas também reunir avaliações e contabilizações sobre as melhorias provocadas pelas alterações. A sua implementação torna-se indubitável.

3.2.2 Domínio Virtual

O domínio virtual (DV), designadamente o ModP, centra-se, sobretudo, nas abordagens de simulação pedonal microscópica [Est09] destinadas a estudar as interacções entre as pessoas que utilizam um ambiente comum, com diferentes finalidades e mobilidade sobre diferentes restrições. Contudo, uma abordagem que possibilite a compreensão do comportamento de multidões não está, de todo, posta de parte. Esta última abordagem pode elevar o ModP a um outro nível de melhoria de concepção espacial (encaminhamento de multidões para zonas específicas – ex: em situações de emergência). A Figura 3.5 ilustra a arquitectura de alto nível do ModP.

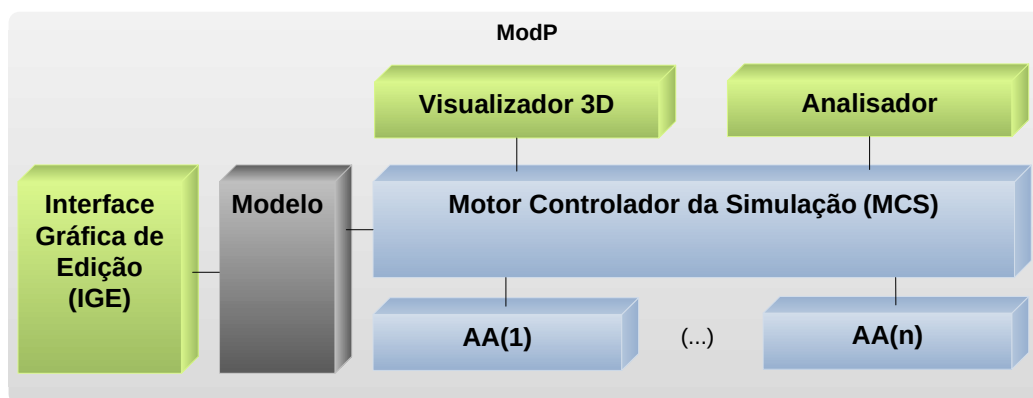


Figura 3.5 – Arquitectura de alto nível do ModP.

Analisando a Figura 3.5, observa-se que a plataforma ModP está estruturada em seis módulos que agregam funções específicas de acordo com as necessidades referidas até este momento.

O módulo **Interface Gráfica de Edição (IGE)** enquadra-se nos conceitos de SINP e SMAN e representa o ambiente de interacção com o utilizador. Este deverá permitir a manipulação total do modelo, desde a composição até à parametrização, e oferecer uma visualização concordante e não ambígua sobre as operações efectuadas. Os grandes desafios para a especificação e implementação deste módulo serão reduzir a interacção ao essencial e tornar a edição/manipulação OQVVEOQVT – O Que Você Vê É O Que Você Tem.

Modelo é um módulo essencial e deverá servir para enquadrar o modelo virtual ao seu homólogo MR. É sobre ele que se executará a simulação de acordo a parametrização e durante um determinado intervalo de tempo.

Toda a sincronização, coordenação e imposição de regras deverão estar ao cargo do módulo **Motor Controlador da Simulação (MCS)** com base na análise feita ao modelo. A simulação disponibilizada por este módulo deverá decorrer sobre um ambiente microscópico, facultando a obtenção de dados tanto a nível do indivíduo como da multidão. O MCS deverá também integrar dois módulos, **Visualizador 3D** e **Analizador**, de forma a permitir uma melhor observação espacial e uma rápida e eficaz análise sobre os dados recolhidos, respectivamente. O desafio mais relevante do MCS será a eficácia de execução.

A Figura 3.6 resume a relação e interacção entre os três módulos, primeiramente relatados.

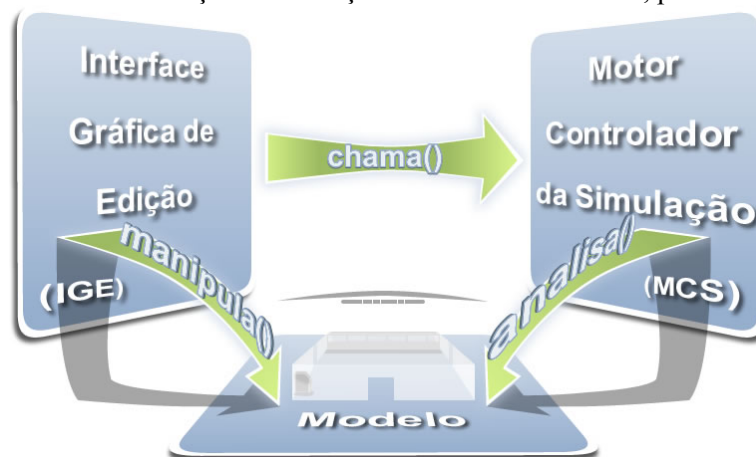


Figura 3.6 – Interacção entre os módulos IGE, MCS e Modelo.

Por último, temos o módulo **Agente Autónomo (AA)**. Os AA's são as entidades inteligentes que utilizam o MCS para coabitarem no modelo e interagirem entre eles e o modelo, segundo as suas próprias finalidades e mobilidade. Estas deverão ser integradas na simulação pelo MCS de acordo com as parametrizações quer de entrada quer de fluxo. O desafio que se impõe a este módulo está relacionado tanto com o realismo de execução de tarefas a nível individual, como o de locomoção individual e colectivo.

3.3 Modelo

Esta secção assume uma importância relevante na composição, estruturação e conteúdo das subsequentes, pelo que uma leitura cuidada facilita a compreensão das restantes.

Dadas as características de aplicabilidade e de usabilidade do domínio virtual, a especificação do modelo torna-se uma pedra crucial para o desenvolvimento de toda a plataforma ModP. Este deve comportar uma abstracção e flexibilidade tal, que permita a simplicidade e redução da interacção na construção do mesmo.

Em primeiro lugar, é importante perceber quais os componentes que constituem as interfaces multi-modais e que características é que eles podem adquirir.

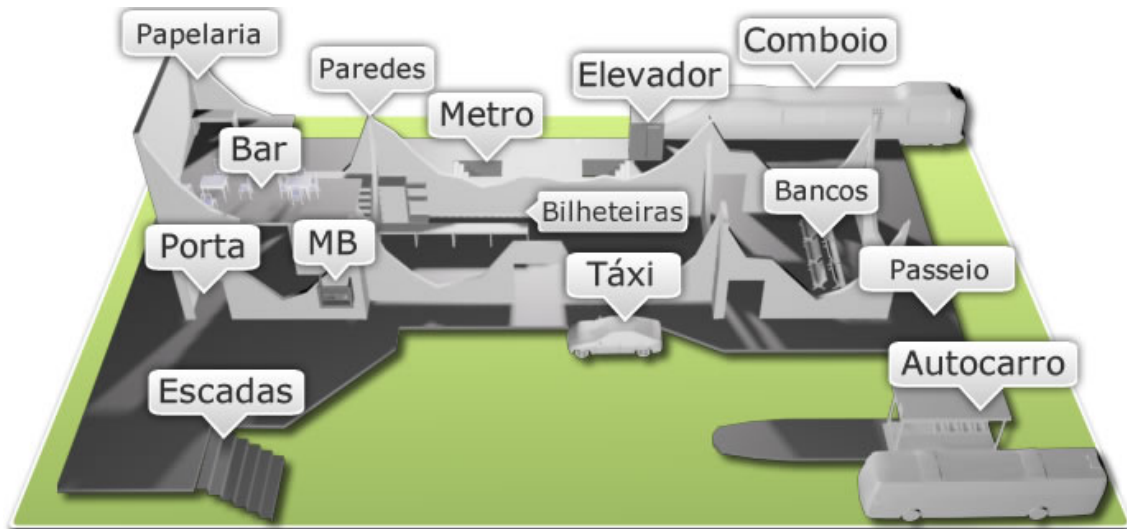


Figura 3.7 – Ilustração dos componentes das interfaces multi-modais.

Como é visível da Figura 3.7, existem inúmeros componentes que podem fazer parte de uma interface deste tipo e todos eles afectam o comportamento, quer de um indivíduo, como de uma multidão. Porém, essa afectação não é análoga a todos. Contudo, existem certos componentes que possuem características, intrínsecas à sua função, tão semelhantes, que nos leva, intuitivamente, a agrupá-las em categorias. Por exemplo, se olharmos para o conjunto constituído por metro, táxi, autocarro e comboio, automaticamente catalogamo-los de “meios de transporte”. Isto é verdade para a função que desempenham. Mas se os avaliarmos numa vertente qualidade/preço, intuitivamente colocamo-los em categorias diferentes. Isto influencia a decisão de maneira diferente. O objectivo é abstrair todos os componentes e catalogá-los de forma a simplificar o processo de modelação e manter a consistência da simulação.

No modelo de referência do ModP (interfaces multi-modais) existe um cenário composto pelos componentes representados na Figura 3.7, que se encontram dispostos numa estrutura básica. Estes são categorizáveis e parametrizáveis, podendo alguns deles ostentar regras de utilização.

3.3.1 Estrutura básica do cenário

Na Figura 3.7 podem ser também observadas estruturas simples, que podem restringir rigidamente a locomoção e o comportamento dos peões. O caso mais óbvio é a presença de paredes, mas existem também os passeios e os obstáculos. É com base neste tipo de estruturas que é feita a decomposição do cenário.

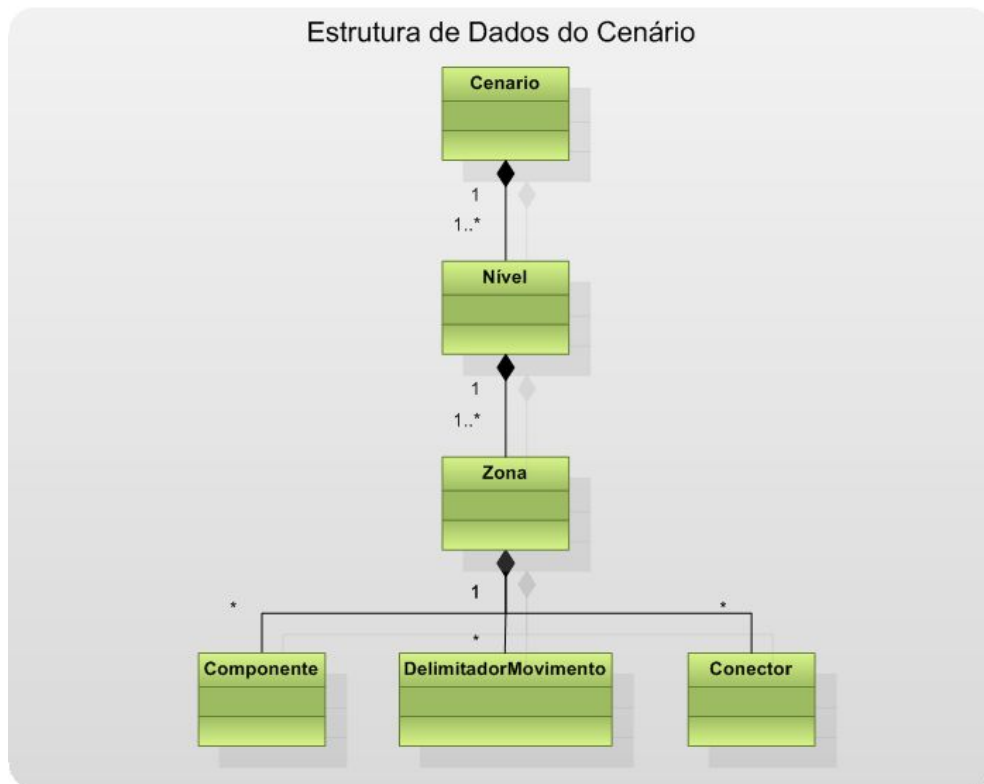


Figura 3.8 – Diagrama de classes de domínio do cenário.

Tal como ilustrado na Figura 3.8, a estrutura básica do cenário é simples e prevê a integração de todos os objectos que o podem compor. Descriminando essa estrutura, temos:

- **Delimitadores de Movimento (DM)** – Os DM's são estruturas físicas e rígidas que limitam ou delimitam a forma que os peões navegam pelos passeios. Nesta categoria estão integrados as paredes e os passeios, respectivamente. A paredes impedem o peão de passar de um lado para o outro enquanto os passeios guiam o peão de forma a tentar que este só passe de um lado para o outro no local certo.
- **Componentes/Obstáculos** – São quase todos os objectos que compõe o cenário e que estão etiquetados na Figura 3.7, à excepção dos DM's. Os obstáculos podem ser entendidos como barreiras físicas impostas pela estrutura intrínseca de cada componente que obriga os peões a contorná-los.
- **Zonas** – São áreas delimitadas por DM's. Estas zonas podem ser rotuladas abstractamente (ex: zona de restauração) e podem ter na sua composição qualquer componente.
- **Conectores** – Os conectores servem para ligar as zonas entre si (portas, elevadores e escadas).
- **Níveis** – Estes são representativos dos pisos e na sua composição integram uma ou mais zonas.

3.3.2 Categorias dos componentes e parametrização

No ponto de vista de simulação, existem algumas abstracções interessantes sobre os componentes, que são levadas em conta e que pretendem simplificar todo o processo de manipulação e simulação do modelo. Estas abstracções são relativas à natureza dos componentes e ao significado que têm para a simulação e para os agentes (peões) que participam na simulação. Em termos gerais existem duas categorias principais:

- **Pontos de Interesse (PI's)** – Os PI's generalizam os componentes que, por norma, ajudam ou são essenciais para cumprir o objectivo final. A distinção entre ser essencial e ajudar resulta do motivo de utilização. Por exemplo, utilizam-se as bilheteiras, porque para utilizar o comboio é preciso um bilhete, mas utiliza-se a papelaria porque uma revista ajuda a passar o tempo. Dentro desta super categoria temos multibancos, bilheteiras, bar, restaurante, papelaria e bancos. As suas parametrizações são muito genéricas e são efectuadas sobre a sua localização (posição e orientação), geometria (comprimento, largura e altura) e identificador próprio (nome do componente quando instanciado). Os dois primeiros contêm o parâmetro “tempo médio de espera” e “estado de utilização”.
- **Controladores de Fluxo (CF's)** – Os controladores de fluxo são a generalização de todos os componentes que proporcionam a entrada e saída de peões nas zonas correspondentes. Quanto às parametrizações genéricas, estas correspondem às dos PI's (localização, geometria e nome). Devido a uma distinção marcante entre alguns destes componentes, os CF's são naturalmente divididos em duas subcategorias:
 - **Entrada/Saída (E/S)** – Os E/S's são componentes responsáveis pela inserção de peões na simulação e pela sua retirada. Assumindo, assim, o papel de ponto de partida e ponto final. Nesta categoria encontramos comboio, táxi, autocarro e metro. O facto de possuírem a característica de ponto final torna-os especialmente importantes. O utilizador do ModP poderá parametrizar um escoamento de fluxo (ex: dos agentes que advêm do metro, 80% vão para o comboio e 20% para o autocarro), para um determinado E/S, obrigando os agentes a possuírem como objectivo final um desses pontos e a sair da simulação quando o atingirem. Quando se processa um escoamento de fluxo, é necessário garantir que este é atribuído a todos os agentes (ex: a soma das percentagens tem de ser 100%). Quanto à entrada de peões, esta também é parametrizada. Os E/S possuem uma característica própria de periodicidade, podendo-se assumir como contínuos, periódicos ou únicos. Ao especificar-se um destes tipos, deve-se atribuir, respectivamente, o fluxo médio por período de tempo ou o número de pessoas (ex: de 1 a 4 ou 30) por disponibilização.
 - **Conectores** – Possibilitam a passagem entre zonas de forma controlada. Os conectores seguem a abordagem de simulação por fila, o que permite restringir o número de pessoas que os podem utilizar simultaneamente, sendo esse um dos parâmetros. Os outros parâmetros estão relacionados com as zonas que conectam. Como só põem conectar duas zonas, possuem dois parâmetros semelhantes, repartidos pelas duas. A esta categoria pertencem as escadas (rolantes ou não), portas, elevadores e passadeiras (rolantes ou não).

3.3.3 Regras de utilização dos componentes

A incorporação de regras em determinados componentes pretende incrementar o realismo da simulação e adequa-la ainda mais à realidade. A necessidade da sua incorporação advém do uso normal deste tipo de interfaces, onde predominam regras como a necessidade de dinheiro para aquisição de um bilhete que permite usufruir do comboio. O grau de complexidade destas regras pode atingir um valor pesado na performance da simulação (necessário um verificador de itens para cada componente), um acréscimo exagerado de variáveis (dinheiro, cansaço, sede, motivação, paciência, etc.) nos agentes e uma base de conhecimento alargada (“o bilhete para comboio tal adquire-se na bilheteira tal”, “o dinheiro pode ser levantado no multibanco”, “para beber pode-se ir ao bar ou restaurante”, etc.).

A abstracção proposta para o modelo parte do princípio que cada componente só pode possuir uma regra de utilização e que essa regra é a probabilidade utilização de outro componente (que também pode ter regras). Essas regras podem ser estipuladas para um

componente instanciado e identificado ou para a sua representação genérica. Exemplificando, para um componente E/S, ao qual se pode dar o nome “comboio inter-regional”, atribui-se a regra PI (com o nome “bilheteira inter-regional”) com probabilidade 80%. Isto significa que, sempre que um agente mostrar a intenção de utilizar o “comboio inter-regional”, este tem 80% de hipóteses de ter que se dirigir à “bilheteira inter-regional”. Por sua vez, se se atribuir à “bilheteira inter-regional” a regra PI genérico (representativo do multibanco) com probabilidade 34%, significa que tem 34% de hipóteses de ser obrigado a ir a qualquer multibanco instanciado.

Para maior heterogeneidade de comportamento, é acrescentado a cada componente a afectação sobre a variável “energia” intrínseca a cada agente (mais detalhes na secção 3.6).

3.4 Interface Gráfica de Edição

Como já foi referenciado na secção 3.2.2, a interface gráfica de edição (IGE) representa o ambiente de interacção com o utilizador. Esta acarreta tanto características de SINP, como de SMAN potenciando a usabilidade e a transparência de utilização.

A IGE, ponto de partida para a utilização do ModP, não só fornece um conjunto de ferramentas e funcionalidades capaz de construir/manipular o modelo, como uma organização contextualizada das mesmas, tornando a sua utilização intuitiva.

Esta secção esquematiza a disposição da interface, as funcionalidades de cada zona e apresenta o mecanismo de interacção com o utilizador.

3.4.1 Esquematização da IGE

Num desenvolvimento deste tipo de interfaces, é importante um desenho adequado às necessidades do utilizador final e ao seu grau de conhecimento (adquirido no uso de ferramentas semelhantes). Mas desafiante, será desenhá-la a pensar num utilizador com pouco ou nenhum conhecimento, que não necessite de constante apoio da documentação e que lhe desperte o interesse na área.

Segue-se um esquema que representa uma visão, baseada nesses paradigmas, da disposição da interface principal.

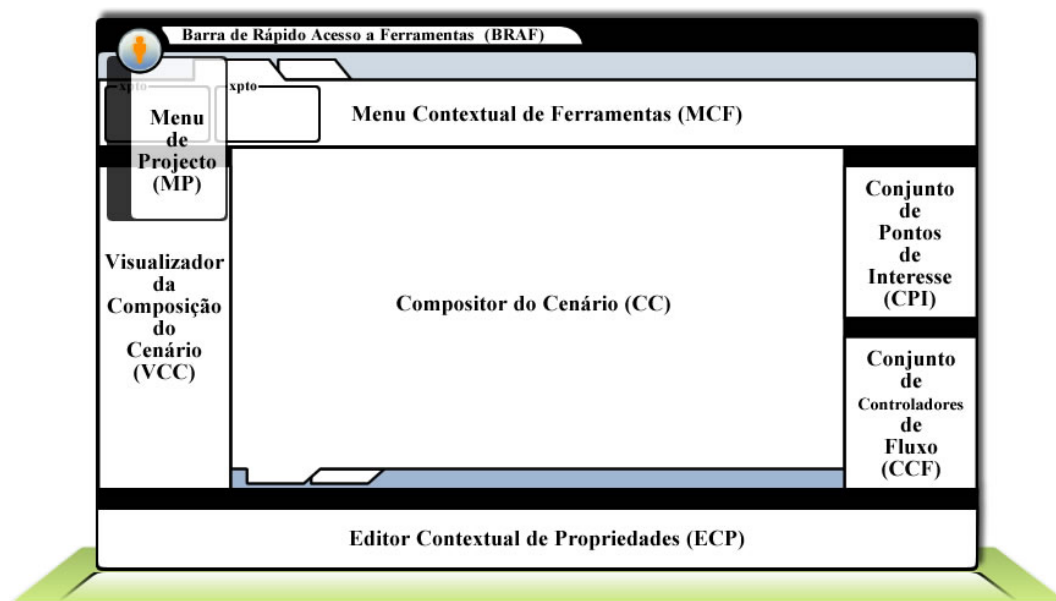


Figura 3.9 – Representação esquemática da disposição IGE.

Através da Figura 3.9, pode-se observar que os vários componentes gráficos da interface encontram-se dispostos em cinco zonas distintas: Norte, Sul, Este, Oeste e Central.

Na zona central, com maior disponibilização de espaço, encontra-se isolado, devido à sua proeminência, um componente:

- **Compositor de Cenário (CC)** – Este integra, no seu funcionamento mais básico, a capacidade de construir um cenário de forma esquemática e do nada. Como o principal objectivo é seguir o conceito OQVVEOQVT, a esquematização deverá ser representativa/semelhante (ex: linhas duplas para as paredes; linhas simples para os passeios; ícones dos PI's; e áreas dos CF's) e à escala do cenário do modelo real. O CC enquadra-se num ambiente de desenvolvimento vectorial, facilitando a edição/modificação das linhas (ex: deslocar os pontos extremos para a posição pretendida) e o reposicionamento quer dos PI's como dos CF's. De antemão, podemos presumir que a construção de objectos compostos por várias linhas agrupadas (a representar zonas do modelo) torna-se um pouco confusa ou impossível sem o auxílio de uma ferramenta ou um modo de agir diferentes. Mas, o que se pretende ao simplificar o desenho de linhas é simplificar também o desenho de zonas. Daí a necessidade de o CC ser capaz de antever alguns comportamentos de utilização, a partir de agora designado de comportamento de alto nível (ver Figura 3.10). Este comportamento de alto nível pressupõe duas previsões cruciais: prever a não replicação de um ponto (extremo de um segmento de recta) quando este se sobrepõe a outro de um segmento diferente; e prever, após um encadeamento de segmentos, que a junção a um dos pontos primordiais desse encadeamento poderá sugerir a criação de uma zona. Todos os objectos no interior da zona pertencem-lhe e os posteriormente inseridos também. Outra característica interessante do CC é a adopção de painéis dispostos por abas (*tabs*). Estes servem para diferenciar os diversos níveis do cenário. A colocação do selector de abas, na parte inferior do componente, serve para desambiguar o utilizador em relação à zona norte, que, por sua vez, também possui painéis.

A gestão e a organização de operações sobre o CC, gestão do projecto e de controlo de simulação, estão encarregues à zona Norte. Nessa zona encontramos três componentes:

- **Menu Contextual de Ferramentas (MCF)** – Este menu possui o padrão *Ribbon* (Fita) que facilita a navegação e acesso[Mic07] às ferramentas. A fita é constituída por abas exibindo as ferramentas que são mais relevantes para uma determinada área de tarefa. A atribuição de nomes e ferramentas às abas deve ser feita de modo contextual e de acordo quer com a especificidade da tarefa que pretendemos dar às ferramentas, como com a área a que estas pertencem. O MCF permite dois tipos distintos de influência¹: sobre o compositor do cenário (CC) e sobre a simulação (duas abas). Em relação à primeira, permite mudar o estado do tipo de operação que se executará sobre o CC. Nomeadamente, permutar entre ferramentas de selecção (de vértice e de objectos compostos ou não compostos) e ferramentas vectoriais (desenho de limitadores de movimento e paredes) que se encontram agrupadas dentro de uma caixa de grupo com o respectivo nome. Quanto à segunda, possibilita a execução de operações de controlo de simulação (iniciar, parar, disponibilizar estatísticas), que se encontram na mesma caixa de grupo e a edição das regras de utilização dos diversos componentes (outro grupo). É importante que todas as ferramentas sejam accionadas através de um botão com um ícone condizente à sua função.
- **Barra de Rápido Acesso a Ferramentas (BRAf)** – Esta barra destina-se a alocar ferramentas/operações de uso constante, como “guardar”, “desfazer” e “refazer”. Contudo, o utilizador tem permissão para colocar/retirar na BRAf qualquer outra ferramenta.

¹ O tipo de influências e ferramentas que lhes estão associadas não são tomados como definitivos ou únicos, apesar de serem cruciais e abstractos, o suficiente, para simplificar todo o processo.

- **Menu de Projecto (MP)** – Este menu é acedido através do botão ModP (canto superior esquerdo) e disponibiliza uma série de operações para gestão do projecto. Entre “Criar Novo”, “Guardar”, “Guardar Como”, “Abrir”, “Imprimir” e “Sair”, destacam-se as operações de “Importar” e “Exportar”. Estas permitem a adaptação a outros softwares de edição de cenários/plantas (ex: AutoCAD®), incentivando à adopção do ModP.

Nas restantes zonas encontra-se um conjunto de componentes gráficos distintos, mas com uma característica em comum: janela de alocação (permitem ser destacadas ou alocadas nas zonas especificadas). Esta característica oferece, ao utilizador, a capacidade de adaptar/optimizar o ambiente de criação às suas necessidades e torna-se particularmente eficaz quando existem muitas vistas. Enumerando estes componentes, temos:

- **Conjunto de Pontos de Interesse (CPI)** – Este componente incorpora uma lista de ícones representativos dos PI's que podem compor o cenário (multibanco, bilheteira, snackbar, papelaria e banco/cadeira). Em adição a estes PI's, é acrescentado um PI indiferenciado (com o ícone “?”) e que pode assumir todas as características dos outros. Existem três razões para os PI's não serem todos indiferenciados: não confundir o utilizador não técnico; pode ser desnecessário; e individualizar a representação 3D. O modo de interacção entre o CPI e o CC é “pegar” e “largar”. Depois de serem largados no CC, estes passam a ser objectos representados pelo ícone.
- **Conjunto de Controladores de Fluxo (CCF)** – O CCF incorpora também uma lista de ícones, mas, desta vez, representativos dos CF's (comboios, metros, táxis, autocarros, portas, escadas, escadas rolantes, tapetes, tapetes rolantes e elevadores). O CF também possui um ícone indiferenciado e que pode assumir todas as características de um componente de Entrada/Saída. A grande diferença entre o CPI e o CCF está na acção de largar no CC. Isto deve-se ao facto de alguns CF's possuírem características restritivas (ex: as portas só podem ser largadas em paredes).
- **Editor Contextual de Propriedades (ECP)** – Este componente permite a edição das propriedades dos objectos gráficos do cenário, desde a sua posição e geometria até à sua parametrização. O ECP assume uma importância essencial na precisão e geometria dos mesmos, pelo facto de permitir afinar esses valores de forma numérica. A sua contextualização depende da selecção ou não de objectos. Por exemplo, se um objecto for seleccionado, apresenta o formulário de edição de propriedades intrínsecas ao mesmo. Senão, apresenta o formulário de edição de propriedades do nível. A sua contextualização/dinamismo surge, uma vez mais, da necessidade de optimização de espaço.
- **Visualizador da Composição do Cenário (VCC)** – Fornece uma representação hierárquica da composição do modelo. Para o utilizador é apenas um componente de informação que ajuda a interpretar o cenário e a validar o mesmo. Mas para o ModP, o VCC é a interface de leitura do modelo, quer para análise do motor controlador da simulação, como para guardar/abrir o projecto. A sua transcrição para um modelo no formato XML é facilitada pela estrutura em árvore. O XML pode ser usado para atribuir uma semântica ao ambiente, podendo ser usada na cognição dos agentes autónomos.

3.4.2 Mecanismo de Interacção com o Utilizador

Neste momento, podemos subentender que a interface gráfica de edição (IGE) oferece simplicidade de utilização e facilidade de construção do modelo. Contudo, a melhor maneira de consolidar essa presunção será através da apresentação de casos de utilização concretos.

A estrutura base do modelo é constituída por paredes e passeios (delimitadores de movimento). O seu desenho é efectuado através da união de segmentos de recta. Para realizar

Especificação da Solução

essa operação, o utilizador deve seleccionar, no menu contextual de ferramentas (MCF), a ferramenta correspondente à tarefa desejada e, seguidamente, realizar operações de “clique” e “arrasto” sobre o compositor do cenário (CC). A Figura 3.10 demonstra este tipo de operações.

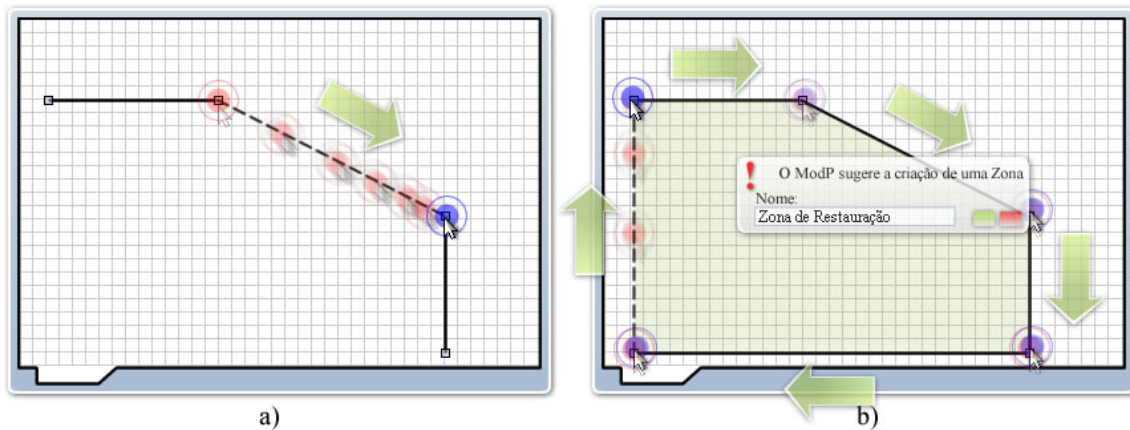


Figura 3.10 – Mecanismo de desenho de linhas através do compositor do cenário.

Depois de efectuar um primeiro “clique”, o primeiro ponto do segmento de recta fica marcado. Só quando esse evento for libertado é que se processa o segundo ponto do segmento. Ao se proceder ao desenho de um outro segmento é necessária a comprovação de que não existe replicação desnecessária de pontos. Como demonstrado na Figura 3.10.a), se os pontos extremos de um segmento coincidirem com algum ponto extremo de outro segmento, estes deverão ser o mesmo (ficando os segmentos ligados por esse ponto). Num encadeamento sincronizado deste tipo de operações, Figura 3.10.b), se e só se um ponto coincidir com o ponto de desencadeou o processo, deve ser sugerido, ao utilizador, a criação de uma zona rotulada.

Uma outra tarefa de composição do cenário é colocar pontos de interesse (PI's) e controladores de fluxo (CF's), recorrendo à lista de ícones da área Este (ver Figura 3.9). Esta acção é despoletada após a sequência “pegar” (o ícone da lista) e “largar” (o ícone no compositor de cenário), tal como demonstra a Figura 3.11.

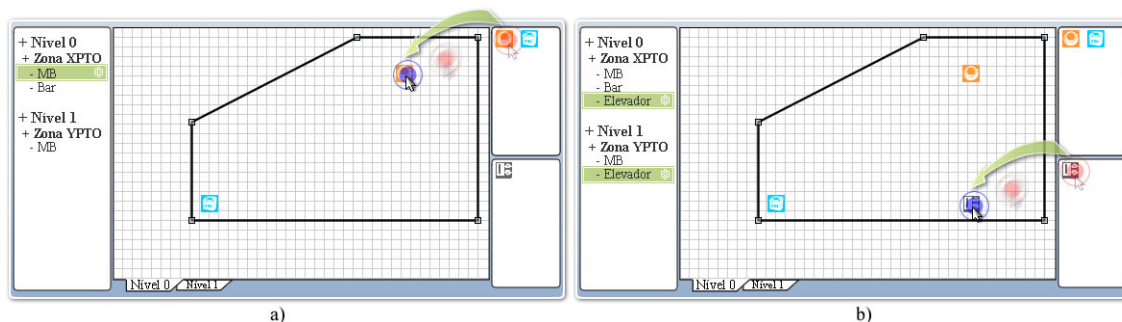


Figura 3.11 – Mecanismo de agregação de componentes no cenário.

Depois de colocados no compositor de cenário (CC), estes ícones/objectos ficam imediatamente alocados à zona e piso correspondentes (Figura 3.11.a)). No entanto, existe uma pequena diferença quando se agrega um CF de conexão entre níveis. Este fica alocado, tanto na zona do piso em que foi inserido, como no(s) nível(eis) superior(es), na zona correspondente à sua posição. Quando um CF é inserido e não existe mais do que um nível, o IGE deve sugerir a criação de um novo nível.

Resumindo, a construção do cenário resulta da execução destas acções de forma simples, rápida e eficaz. No entanto, a construção do modelo não se limita à construção do cenário, apesar de chegar para realizar uma simulação, já que os seus componentes possuem parametrizações predefinidas. O utilizador poderá querer parametrizar alguns parâmetros de um determinado componente do cenário. Para isso, basta seleccioná-lo e utilizar o editor contextual de propriedades (ECP), como demonstra a Figura 3.12.

Especificação da Solução

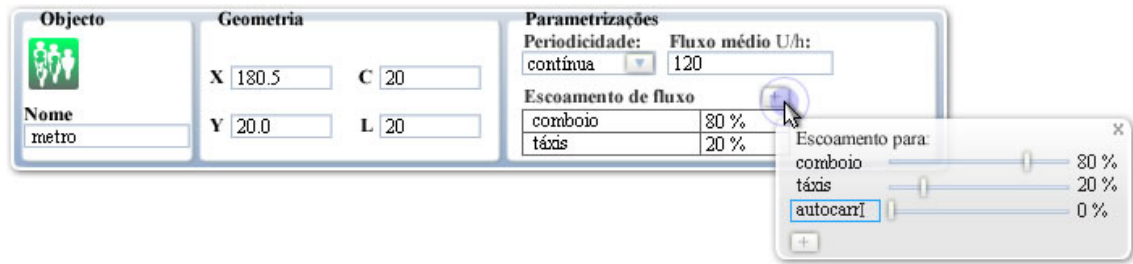


Figura 3.12 – Edição das propriedades de um controlador de fluxo (Entrada/Saída).

Os parâmetros expostos no ECP são concordantes com as parametrizações permitidas pelo componente. A Figura 3.12 exemplifica os campos de parametrização de um CF de entrada e saída de peões. Os padrões de desenho são, também, condizentes com o tipo de dados (ex: caixas de texto com filtro para números reais para secção de geometria). Para um CF deste tipo e no caso de o utilizador pretender adicionar parametrizações para escoamento de fluxo, este é feito através de barras deslizantes garantindo a restrição dos cem por cento. Algumas destas parametrizações repercutirão sobre o comportamento dos agentes autónomos, como é o caso do nome do objecto e do escoamento de fluxo. Este assunto será esclarecido nas subsecções seguintes.

A construção do modelo completa-se com a incorporação de regras dos componentes pretendidos. A sua inserção simplifica-se com um processo de “largar” e “arrastar”, demonstrado na figura seguinte.



Figura 3.13 – Exemplo de edição de regras dos componentes.

Como pode ser observado na Figura 3.13, a edição de regras pressupõe a existência de dois painéis (um para os ícones instanciados no cenário e outra para os ícones genéricos). A inclusão das regras processa-se com o largar dos ícones na zona inferior da janela, seguindo-se um largar de outro ícone para o seu interior. Este último passo pode não ser necessário, uma vez que o utilizador poderá querer apenas parametrizar a influência na energia e não atribuir nenhuma regra distinta.

3.5 Motor Controlador da Simulação

Na subsecção 3.2.2 foi introduzido o Motor Controlador da Simulação (MCS) como a essência de toda a simulação. O MCS é não só capaz de desempenhar a simulação microscópica de peões em condições normais, como possui uma modularidade tal, que permite a sua reutilização para outros tipos de simulação (simulação de evacuação de edifícios). Ao longo

desta secção será especificada a arquitectura lógica deste módulo e exemplificado o seu mecanismo de interacção.

3.5.1 Arquitectura Lógica

A arquitectura lógica do motor controlador da simulação (MCS) e todos os seus componentes estão ilustrados num diagrama UML, destacado na Figura 3.14.

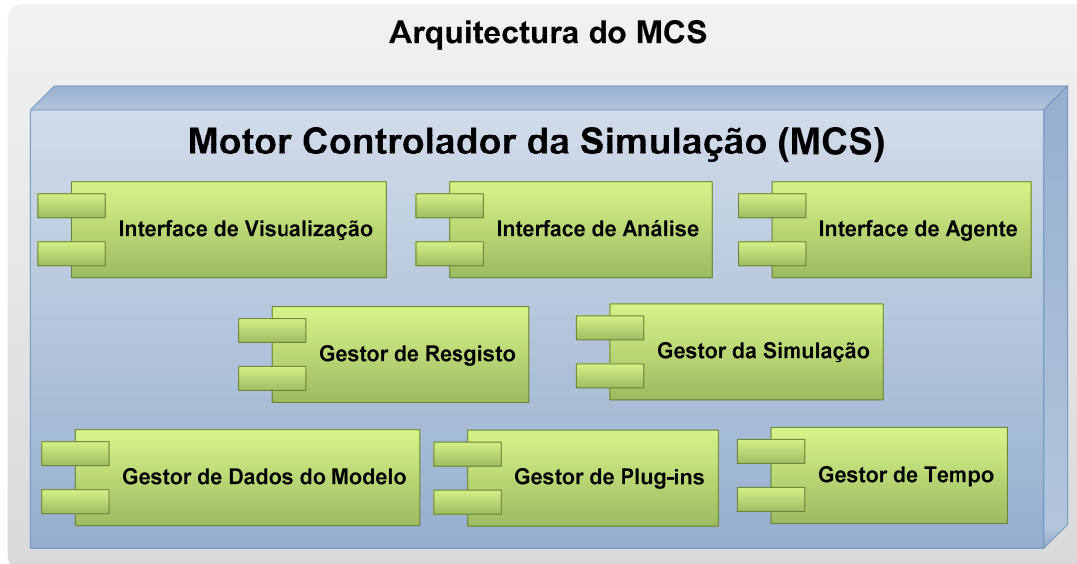


Figura 3.14 – Arquitectura do motor controlador da simulação.

Como se pode notar no diagrama, o MCS é constituído por sete componentes principais:

- **Gestor de Tempo** – Este componente é o provedor da habilidade e agilidade de gestão e coordenação do tempo. O gestor de tempo lida de tal maneira com a dinâmica do tempo, que o possibilita atender, de forma eficaz, a todos os eventos que irão não só afectar o estado dos vários componentes, como o despejo dos agentes. Qualquer ajuste à unidade de tempo de simulação (incremento ou decremento) é gerido por este componente.
- **Gestor de Dados do Modelo (GDM)** – A leitura de dados do modelo é essencial para o arranque da simulação, estando este componente encarregue de a realizar. Essa leitura deve ser feita de forma cuidada e estruturada, facilitando a pesquisa e acesso a dados. Durante todo o processo de leitura deve ser tido em conta a estrutura básica do cenário bem como a hierarquia providenciada pelo visualizador da composição do cenário. Este gestor deve fornecer mecanismos eficientes de pesquisa (“Quem está na zona tal?”) e possibilitar uma pesquisa semântica (“Em que posição se encontra o componente E/S comboio”), otimizando a interacção entre e com os agentes, respectivamente. A pesquisa eficiente pode ser alcançada com recurso à estrutura básica do cenário de níveis e zonas e/ou através de consultas espaciais (*space queries*, *locality queries* ou *neighbourhood queries*) [Rey00]. O GDM também está responsável por informar o gestor de registo sempre que existir a necessidade de inserção de agentes no cenário, devendo-o prover dos dados necessários para o efeito.
- **Gestor de Plug-ins** – É um componente que eleva a modularidade do MCS. Este permite a coexistência de Plug-ins de simulação com características diferentes (regras de simulação, como poder atravessar paredes, e agentes diferentes) mas que utilizam o mesmo modelo. A função principal é chamar o Plug-in pretendido e fornecê-lo com o modelo.

- **Gestor de Registo** – Este incorpora a capacidade de introduzir na simulação os agentes de acordo com as parametrizações. Essa incorporação deve ser efectuada na zona correspondente e na área oferecida pelo componente de E/S, tentando evitar, ao máximo, a sobreposição dos agentes. No instante da sua inserção, estes deverão ser providos das regras de utilização dos componentes e, se for o caso, do seu objectivo final. Durante a simulação, o gestor de registo está encarregue do registo dos agentes nas várias zonas (à medida que eles vão passando de umas para as outras) e de apagar o seu registo nas zonas que foram deixadas por eles. O problema de conflito na fronteira de zonas concorrentes é atenuado pela capacidade dos conectores processarem o tráfego baseado em filas.
- **Gestor da Simulação (GS)** – O gestor de simulação é o componente principal do MCS. Este é responsável por toda a sincronização e coordenação da simulação (durante a sua execução) e o seu modo de operação está intrinsecamente ligado ao Plug-in seleccionado. A sincronização é realizada de acordo com a unidade de tempo virtual (*time-step*) e esta obriga todos os agentes a realizarem uma actualização das suas acções, no decorrer da mesma unidade. Depois de agregadas as acções (a agregação pressupõe o respeitar das regras internas do Plug-in) são calculadas e actualizadas as afectações nos diversos componentes que as sofreram. No final da unidade de tempo, todo o estado do cenário é actualizado (ex: ocupação dos componentes e actualização das possíveis novas zonas de localização dos agentes) e processa-se uma nova unidade. A relação entre este componente e os agentes é simbiótica. O GS precisa dos agentes para processar o seu ciclo de execução obtendo registos resultantes do comportamento destes. Por seu lado, os agentes necessitam do GS para coexistirem e obterem informações sobre o estado do cenário e sobre os outros agentes (precisamente da zona em que se encontram), de forma a efectuarem as suas acções.
- **Interface de Análise** – Esta interface pretende organizar todos os dados recolhidos pelo gestor de simulação numa determinada unidade de tempo e encaminhá-los logo de seguida para o módulo Analisador.
- **Interface de Visualização** – Uma das melhores maneiras de visualizar uma simulação sobre um modelo virtual baseado num modelo real é apresentá-la de forma semelhante. Esta interface tem a função de comunicar com o módulo Visualizador 3D despoletando-lhe os métodos de desenho requeridos e fornecendo-lhe os dados fundamentais para a representação.
- **Interface de Agente** – Esta interface permite o transporte de informação/dados estruturados com o agente, facultando uma comunicação não ambígua.

3.5.2 Mecanismo de Interacção

Depois de inicializada a simulação existe a necessidade não só de incorporar, no cenário, AA's capazes de interagir com o mesmo, como também forçá-los a uma interacção. Este último processo realiza-se através de uma sequência sincronizada entre o gestor de dados do modelo (GDM), o gestor da simulação (GS) e o agente autónomo (AA), como demonstrado no diagrama de sequência ilustrado na Figura 3.15.

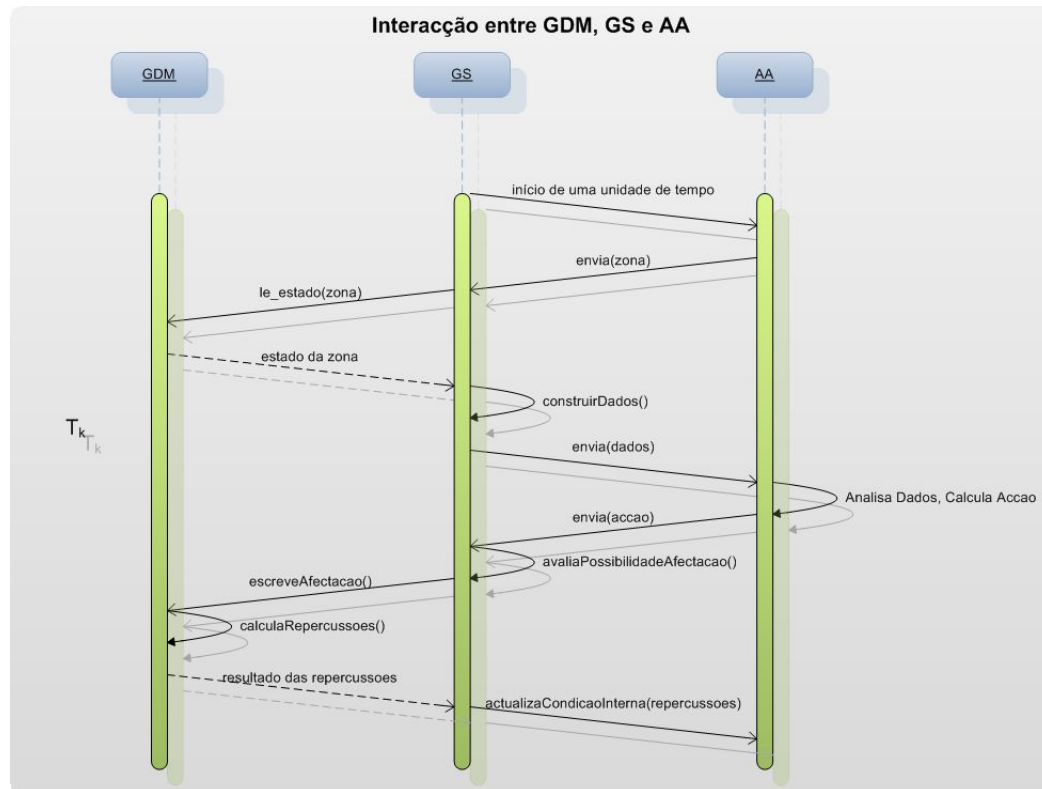


Figura 3.15 – Sequência de interação entre GDM, GS e AA.

Durante uma determinada unidade de tempo discreto T_k , decorre um ciclo de execução, cujo seu início é alertado ao agente autónomo (AA). Este AA é, por sua vez, responsável por divulgar a zona onde se encontra, com o intuito de obter informação sobre o estado da mesma (ex: localização e ocupação dos componentes e localização dos outros AA's). O recolher dessa informação está encarregue ao gestor de dados (GDM), quando requisitado. Após a requisição da informação ao GDM, o gestor de simulação (GS) processa-a de forma concordante com a estrutura de transporte aceite pelo AA (ex: lista de AA's na periferia, identificação e localização dos pontos de interesse e conectores). O ciclo encontra-se neste momento estagnado no agente que analisa a informação recolhida para tomar uma decisão e transcrevê-la numa acção. Essa decisão é baseada num processo interno e que será explicado na secção seguinte. Depois de transcrita a acção, esta é transmitida ao GS que avalia a possibilidade dessa acção ser executada, com base nas regras de simulação. Se tal for possível, é efectuada a recolha de dados de simulação e invocado o gestor de dados para actualizar o estado da simulação com base na afectação enviada pelo gestor de simulação. A partir desse momento, o gestor de simulação é informado sobre as repercussões da acção do agente, devendo, de imediato, pedir a actualização do seu estado interno. Essas repercussões baseiam-se no gasto/ganho de energia pela acção tomada e pela utilização de componentes que têm afectação nesse campo e também pela alteração posicional (podendo ocorrer uma mudança de zona). Terminado este ciclo, inicia-se um novo ciclo T_{k+1} em tudo semelhante ao anterior.

No decorrer de um ciclo de execução, existem alguns casos particulares. O cenário exposto na Figura 3.16, representativo de um ciclo de execução possível, pretende ilustrar alguma dessas particularidades.

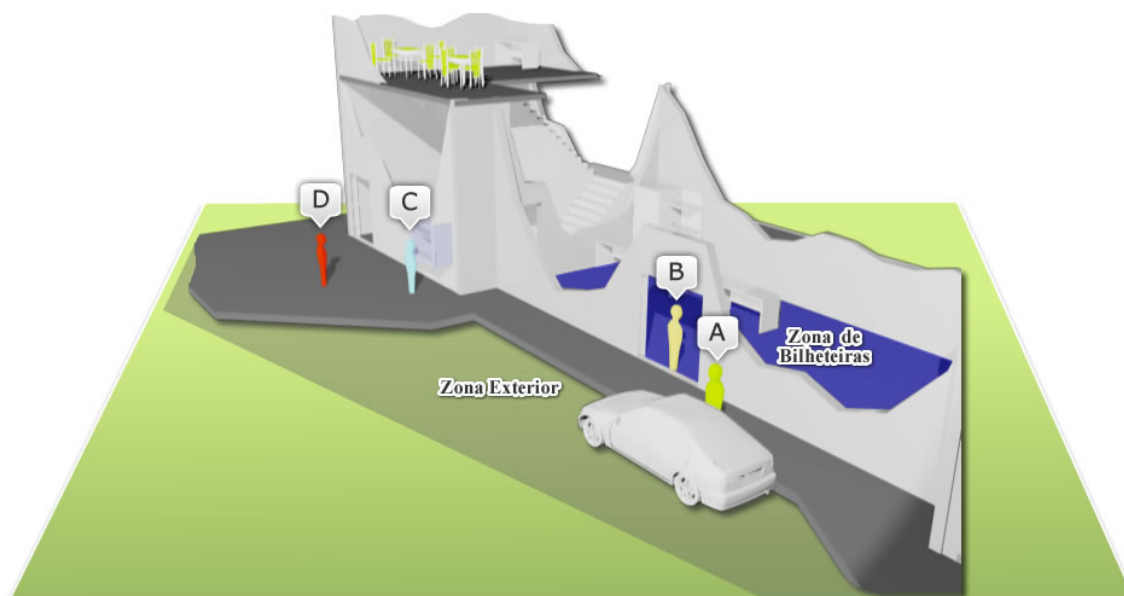


Figura 3.16 – Exemplo de um ciclo de execução de simulação.

No cenário apresentado na Figura 3.16, pode-se observar a existência de quatro agentes (**A**, **B**, **C** e **D**), duas zonas etiquetadas (“Zona Exterior” e “Zona de Bilheteiras”), um componente de Entrada/Saída, representado por um táxi e um componente Ponto de Interesse do tipo Multibanco.

Nos casos dos agentes **C** e **D**, assumindo que a sua intenção é usar o multibanco, desenrola-se o que se pode chamar de acção condicionada. Esta acontece quando, após ser permitida a acção (por parte do gestor de simulação) de utilização do multibanco (pretendida pelo agente **C**), o estado de utilização do multibanco passa a “em utilização”. Nesse momento, será negado o pedido de utilização do multibanco ao agente **D**, obrigando a optar por outra acção (ex: esperar). Na mesma situação, o agente **D** pode nem optar pela acção “utilizar multibanco”, dado que no início do ciclo ele recebe o estado da zona e consequentemente o estado do multibanco.

Numa situação completamente diferente, encontra-se o agente **B**. Este finalizou a utilização de um conector (neste caso, uma porta), efectuando uma mudança de zona (da “Zona Exterior” para a “Zona de Bilheteiras”). No decorrer destas situações, o gestor de registo deve ser notificado para processar a transferência de zona, possibilitando a informação de zona concordante com a sua localização.

O agente **A** demonstra a inserção dos agentes através de um componente de E/S do tipificado “táxi”. Este acontecimento advém da parametrização deste tipo de componentes. Neste caso específico, está parametrizado como periódico (com uma ocorrência de por exemplo 5 minutos) e com um intervalo discreto de 1 a 4 agentes por ocorrência. No instante da sua inserção, o agente **A**, recebe informação sobre o cenário (composição básica do cenário), estado da zona (“Zona Exterior”) e é forçado a tomar uma acção.

Em todas as situações existe um decremento da energia de cada agente, sendo esse valor de decremento semelhante para os agentes **A**, **B** e **D** e distinto para o agente **C**, caso exista um valor de incremento estipulado no componente.

3.5.3 Visualização e Análise de Dados

A visualização e análise de dados de simulação são procedimentos essenciais no que diz respeito à usabilidade do ModP. Estes estão encarregues a dois módulos distintos, cuja arquitectura lógica está ilustrada na Figura 3.17.a) e Figura 3.17.b), respectivamente.

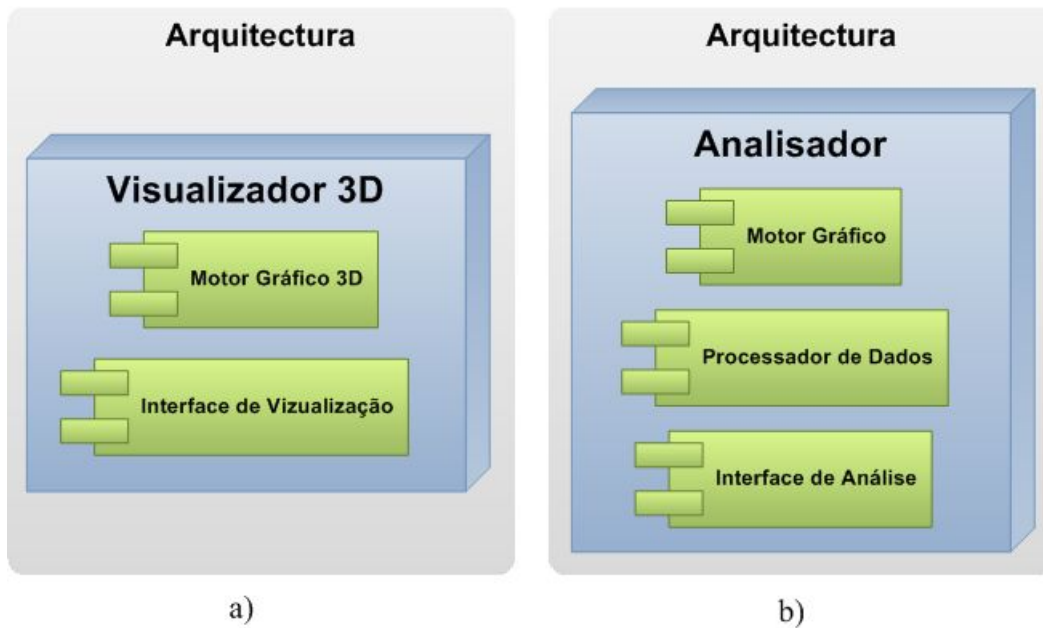


Figura 3.17 – Arquitectura dos módulos Visualizador 3D e Analisador.

O **Analisador** efectua o tratamento dos dados recebidos pelo motor controlador da simulação (MCS), disponibilizando-os de forma coerente e intuitiva de acordo com a necessidade do utilizador. A sua função não se restringe a mostrar os dados numa forma gráfica, mas pressupõe um conjunto de tratamentos dos mesmos, aumentando o seu grau de detalhe e informação. São considerados cruciais, para efeitos de resultados de simulação, estatísticas de utilização de componentes, bem como de peões (tempo e trajectória do percurso realizado) e de multidões (mapa de densidade)

Quanto ao **Visualizador 3D**, pretende-se que este represente o estado da simulação em cada unidade de tempo virtual e que esboce o desenho do cenário e agentes com formas semelhantes às do modelo real. Essa semelhança deve ser tanto a nível físico como a nível dimensional, provendo o utilizador de uma visualização em tempo de execução, tridimensional, com um aspecto concordante com as características reais e completamente navegável.

3.6 Agentes Autónomos

Durante este capítulo, têm sido referenciados os agentes, algumas das suas características e alguns dos seus comportamentos. Contudo, essa referência não é suficiente para interpretar todo o conceito por detrás da sua especificação. Esta secção destina-se a colmatar essa lacuna e apresentar uma especificação de alto nível dos Agentes Autónomos (AA).

Em toda a bibliografia consultada, encontram-se várias definições de agentes autónomos. Mas a que melhor se enquadra com as características e especificações da plataforma ModP e com o objectivo desta dissertação é a seguinte:

“Um agente autónomo é um sistema situado dentro e pertencente a um ambiente, que sente o ambiente e age sobre ele, ao longo do tempo, em perseguição da sua própria agenda de tal modo que afecta o que ele vai sentir no futuro”. [Buc05], Capítulo 3

A partir deste momento é usado o termo “agente autónomo” para referenciar um agente que possui um certo grau de movimento autónomo. Por exemplo, se um agente estiver em vias de se esbarrar contra uma parede, ele possui a habilidade de responder à situação e ajustar o seu movimento de forma a contorná-la. Do ponto de vista do movimento, este pode ser decomposto em três níveis [Buc05][Rey99]:

- **Seleccção da acção** – responsável pela escolha de objectivos e por decidir o plano a seguir.

- **Direcção** – responsável por calcular trajetórias capazes de alcançar os objectivos e planos traçados pela camada anterior.
- **Locomoção** – Responsável por activar os mecanismos necessários que permitam a uma entidade deslocar-se a fim de executar uma determinada acção.

Para uma melhor interpretação da diferença entre a direcção e a locomoção, retenha-se que cada entidade/agente tem a capacidade de controlar um corpo/veículo. Agora imagine-se que a uma entidade, que estava a controlar um cavalo, é-lhe pedido que passe a controlar uma moto. O seu plano, as suas acções e comportamentos direccionais continuam os mesmos. A única coisa que muda são os mecanismos usados para executar o movimento. No caso de simulação de peões, existe uma entidade que controla um corpo. Esse corpo possui mecanismos de locomoção, que são as pernas.

Para efeitos desta dissertação, são tidos em consideração, apenas os dois primeiros níveis e é assumido que os agentes autónomos são incorporados (estão embutidos num corpo virtual ou objecto móvel que lhes permite mover, sem preocupação de activação dos mecanismos). A locomoção é realizada através da aplicação de forças de atracção e repulsão (abordagem de forças sociais) e cabe ao agente decidir que pontos é que o atraem (decisão de alto nível baseada nas regras parametrizadas e na optimização dos campos internos estipulados) e que pontos é que o repulsam (comportamento de baixo nível, como desviar paredes e outros peões). A especificação da arquitectura lógica dos Agentes Autónomos (AA's) é baseada nestes dois níveis e encontra-se representada no diagrama destacado na Figura 3.18.

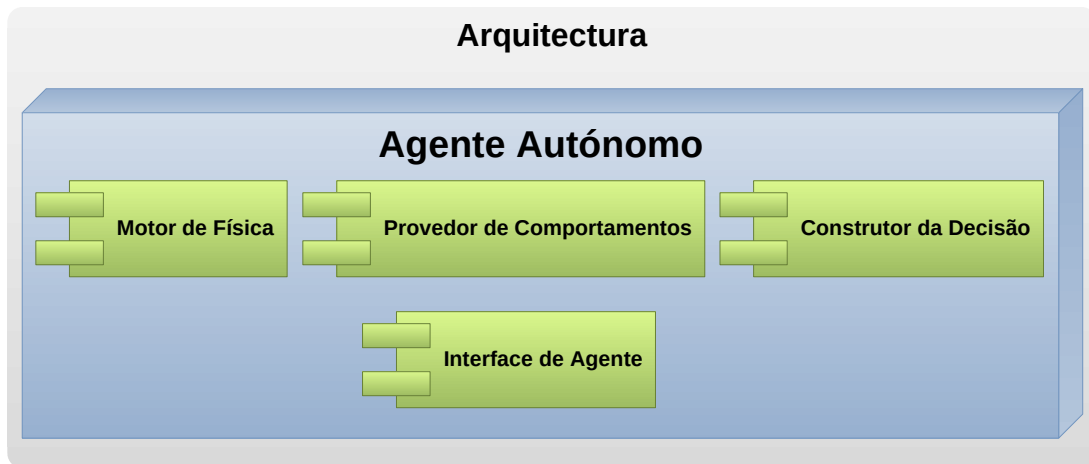


Figura 3.18 – Arquitectura do módulo Agente Autónomo.

Tal como é possível interpretar através da Figura 3.18, o módulo AA é composto por quatro componentes essenciais:

- **Motor de Física** – Este componente realiza todas as operações relacionadas com o cálculo vectorial, promovendo a aplicação de forças sobre um corpo e a sua consequente deslocação pelo espaço tridimensional.
- **Provedor de Comportamentos** – Possui uma biblioteca de comportamentos direccionais, que podem ser usados isoladamente ou serem combinados de forma a se realizar comportamentos de locomoção mais complexos. Dentro desses comportamentos, existem alguns fundamentais, como evitar paredes, evitar peões/obstáculos e mover para alcançar o objectivo.
- **Construtor da Decisão** – Responsável pela definição de um plano e escolha de objectivos. Este pode ser entendido como aquele que habilita ao agente um comportamento de alto nível, como cumprir regras, escolher o melhor caminho, avaliar as condições do cenário e propor alternativas. Deve preocupar-se também com a manutenção ou optimização de bem-estar interno ao agente, que neste caso específico será o campo energia.

- **Interface de Agente** – Esta interface permite a comunicação estruturada e/ou a troca de dados com o motor controlador da simulação.

3.6.1 Aplicação de Forças e Comportamentos Direccionais

Como já foi referido, os agentes autónomos estão incorporados num objecto móvel e são capazes de o controlar. Qualquer objecto móvel é definido pela posição do seu ponto de massa, velocidade, que pode ser alterada pela aplicação de forças, força máxima (a força máxima que pode ser aplicado a um objecto móvel baseado nas suas próprias capacidades motoras), uma velocidade máxima e uma orientação.

A posição do ponto de massa e a orientação de um objecto definem o que se pode chamar “espaço local” [Dun02] e adicionam ao objecto uma referência cartesiana (Figura 3.19.a)) em relação ao mundo virtual e a capacidade de aplicação de forças orientadas a esse espaço.

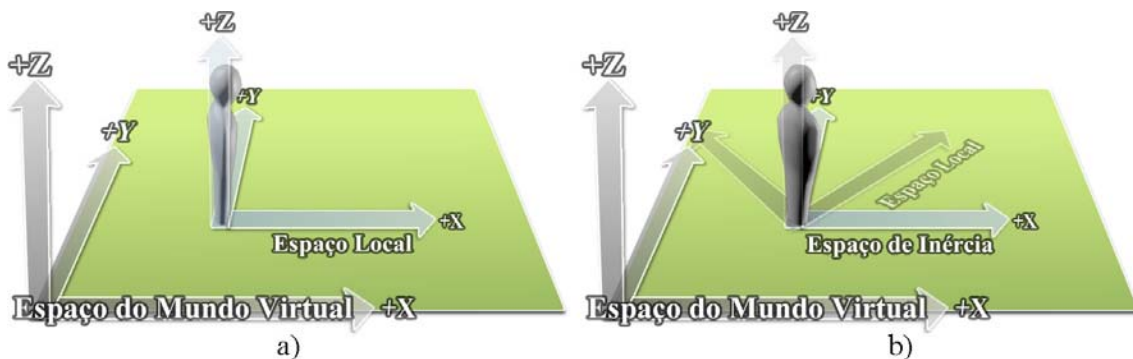


Figura 3.19 – Representação dos espaços Mundo Virtual, Local e Inércia.

É possível observar, na Figura 3.19.b), a existência de um espaço de inércia. Este espaço assume duas características distintas: permite transformar um ponto entre o “Espaço Local” e o “Espaço de Inércia” somente através de uma rotação e permite transformar um ponto entre o “Espaço de Inércia” e o “Espaço Mundo Virtual” através de uma translação.

Ao agregar estes dois tipos de transformação, capacita-se o objecto de ser deslocado e direccionado. Um agente autónomo tem a habilidade de direccionar o objecto móvel através da aplicação de forças de direcção comportamental ao ponto de massa. Dependendo da orientação e do valor da força resultante, é produzida uma aceleração que, por sua vez, é adicionada à velocidade antiga do objecto. Este procedimento produz uma nova velocidade que é truncada com o valor da velocidade máxima estipulada para o objecto. Baseado no novo valor do vector velocidade, são actualizados os valores posição e orientação do objecto.

Reynolds e Buckland sugerem em [Rey99] e [Buc05], respectivamente, um conjunto de comportamentos básicos que, se combinados, podem gerar movimentos bastante complexos. Os comportamentos sugeridos (compilados e implementados, em código aberto, numa biblioteca chamada OpenSteer podem ser encontrados em [Ope04]) estão sumarizados na tabela seguinte.

Tabela 3.3 – Resumo dos comportamentos direccionais.

Comportamentos Direccionais			
Seek	Flee	Pursuit	Evasion
Offset Pursuit	Arrival	Obstacle Avoidance	Wander
Path Following	Flow Field Following	Unaligned Collision Avoidance	Separation
Flocking	Cohesion	Alignment	Leader Following

Especificação da Solução

Nem todos os comportamentos são considerados cruciais, pelo que serão apenas exemplificados os mais importantes. Segue-se uma figura ilustrativa dos comportamentos “mover para alcançar o alvo”, “mover para evitar parede” e “mover para evitar uma potencial colisão”, respectivamente Figura 3.20.a), Figura 3.20.b) e Figura 3.20.c).

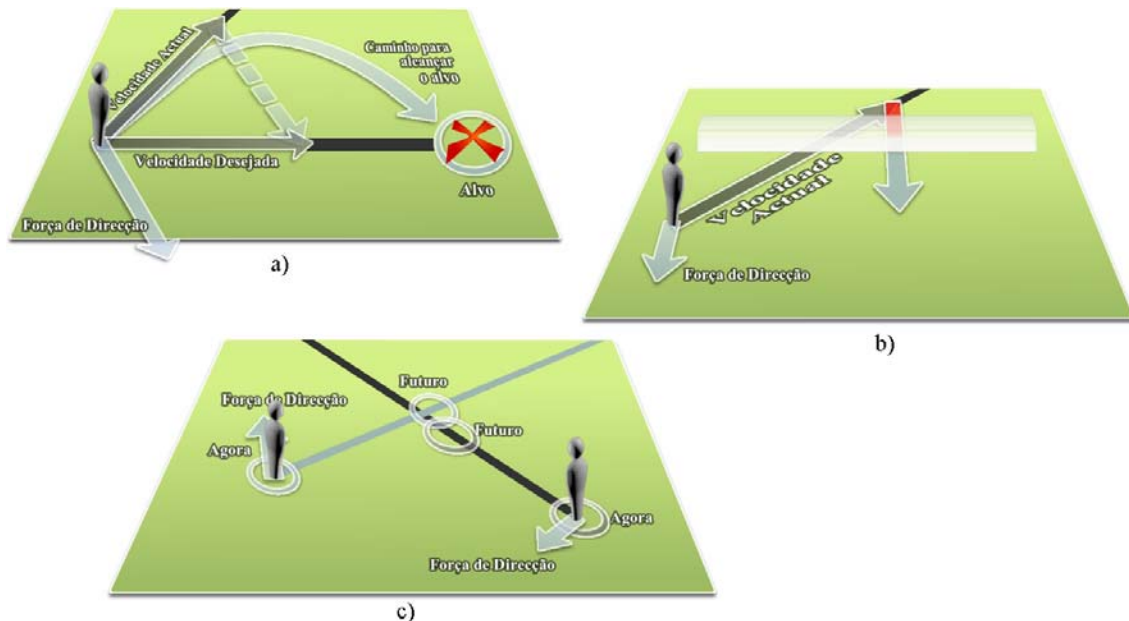


Figura 3.20 – Ilustração de alguns comportamentos de direcção.

O primeiro comportamento, **“mover para alcançar o alvo”** (baseado no comportamento *seek*), actua de forma a orientar o objecto para uma determinada posição no mundo virtual. Este comportamento ajusta o objecto de modo a que a sua velocidade é radialmente ajustada para o alvo. A velocidade desejada é um vector na direcção da personagem para o alvo, cujo comprimento é no máximo igual ao valor da velocidade máxima estipulada. A força de direcção é a diferença entre a velocidade desejada e a velocidade actual. Se o agente continuar a realizar este comportamento, irá ultrapassar o alvo à velocidade máxima e, seguidamente, volta a realizar a mesma operação entrando em ciclo, até por fim parar. Isto pode ser evitado através de uma decisão de alto nível (parar quando está a uma distância suficiente para “utilizar” o alvo). Pode-se também efectuar um abrandamento aquando da aproximação ao alvo. Para tal, pode adaptar-se este movimento (comportamento *arrival*) introduzindo um raio de abrandamento. Uma vez dentro deste raio, a velocidade desejada é diminuída (ex: linearmente) até zero.

Os outros dois comportamentos auxiliam os agentes quer a se desviarem uns dos outros, quer a evitarem as colisões com os delimitadores de movimento. Quanto ao **“mover para evitar parede”** (semelhante ao *path following*), este rege-se pela projecção do vector velocidade actual e teste de colisão com uma parede (se existir). Se alguma colisão for detectada, calcula-se uma força de direcção baseada na magnitude de penetração e com direcção igual à normal da parede.

Por último, temos o comportamento **“mover para evitar uma potencial colisão”** (baseado no comportamento *Unaligned Collision Avoidance*). Este comportamento envolve a previsão de potenciais colisões e alteração da direcção e velocidade para evitá-las. Para se obter uma força de direcção efectua-se uma pesquisa a todos os outros objectos móveis e, seguidamente, calcula-se uma previsão de quando e onde poderá ocorrer uma colisão. É considerada, uma colisão eminente, a que se encontrar mais próximo e cuja distância é considerada suficientemente pequena (a medição dessa distância está relacionada com a soma centro a centro e dos raios). No caso de tal acontecer, são verificadas algumas condições da direcção das velocidades, calculando-se a força resultante, dependendo se estas são paralelas (no mesmo sentido), anti-paralelas (em sentidos opostos) ou perpendiculares. Por exemplo, no caso da Figura 3.20.c) as velocidades são consideradas perpendiculares. O objecto móvel que tiver menor valor de velocidade sofre uma força de desaceleração na direcção da parte de trás do outro. Por sua vez, o outro objecto direcciona-se acelerando para fugir da frente deste. No caso

de serem consideradas paralelas ou anti-paralelas, os objectos são desviados da posição do outro ou da potencial posição, respectivamente.

Os dois últimos comportamentos apresentados, “**mover para evitar parede**” e “**mover para evitar uma potencial colisão**” possibilitam, também, evitar obstáculos inerentes aos componentes do cenário, sendo eles representados por circunferências ou por um grupo de segmentos de recta.

3.6.2 Construção da Decisão

A construção da decisão assume bastante relevância na heterogeneidade do movimento. Esta deve garantir a interacção do agente com o cenário de forma coerente, assemelhando-se ao comportamento humano em interfaces multi-modais. Isto pressupõe o respeitar das regras inerentes a este tipo de interfaces e realizar uma coreografia sincronizada de forma a efectuar a sequência correcta para cumprir o objectivo final.

Uma das abordagens mais populares e que providencia ao agente uma certa habilidade mental é o paradigma BDI (*Beliefs Desires and Intentions*)[Ros02][Ron05][Kin96]:

- **Beliefs (crenças)** – representam o estado informacional do agente. Por outras palavras, representam as suas crenças sobre o mundo e sobre o estado interno, em relação a si mesmo e aos outros agentes (ex: Existe um comboio no cenário; Para usar um comboio é preciso um bilhete).
- **Desires (desejos)** – expressam o estado motivacional do agente. Constituem os objectivos que o agente gostaria de realizar (ex: Desejo apanhar o comboio).
- **Intentions (Intenções)** – representam o estado deliberativo do agente. São o resultado do processo de decisão, ou seja, acção que o agente decidiu tomar (ex: Andar até ao comboio e entrar nele).

A especificação de um modelo BDI, para simular o comportamento humano, pode ser bastante complexa (elevado número de variáveis), o que pode aumentar o realismo do comportamento. Contudo, para efeitos desta dissertação e de exemplificação, a abordagem sugerida, apesar de ser similar, é simplista. A principal simplificação encontra-se ao nível das crenças e desejos, simplificando o processo de edição dos parâmetros, bem como, o processo de decisão dos agentes.

Como já foi referido anteriormente, na subsecção 3.3.3, as regras de utilização dos componentes são baseadas na probabilidade de utilização de outro componente e a sua utilização afecta o campo energia. Esta abstracção rege-se pelo facto de um conjunto de regras/crenças traduzir-se numa probabilidade de utilização de um determinado componente. Para uma melhor compreensão, imagine-se uma pessoa que quer apanhar o comboio. Ela pode possuir ou não um bilhete adequado. Se não possuir o bilhete, é provável que tenha que ir à bilheteira. Para adquirir o bilhete, ela tem que possuir dinheiro suficiente para a compra. Se tal não acontecer, é provável que tenha que ir ao multibanco. No decorrer do percurso (plano), a pessoa referida pode encontra-se cansada, esfomeada ou aborrecida, sendo provável que tenha de se deslocar ao bar ou a um quiosque e pode não ter dinheiro para adquirir o que pretende. Como se pode observar, existe um elevado número de crenças e variáveis que precisam de ser introduzidas. Todas elas aumentam ou diminuem a probabilidade da pessoa adquirir um desejo. Na abordagem sugerida, os bens pessoais/palpáveis (dinheiro e bilhetes) são eliminados e são apenas adicionadas as probabilidades de ser ou não incutido um desejo (que é adicionado à pilha, construindo o plano). Por exemplo, para a mesma pessoa que quer apanhar o mesmo comboio é-lhe atribuída uma probabilidade de ter que utilizar a bilheteira. Se essa probabilidade se confirmar é-lhe incutido o desejo de querer ir à bilheteira, sendo que esta possui uma probabilidade de ter que utilizar o multibanco.

Quanto às outras crenças como cansaço, aborrecimento e paciência, estas são abstraídas em estado de espírito e a variável que o discrimina é “energia”. Se o valor da energia for considerado baixo, é incutido ao agente o desejo de utilizar um componente que dê energia. A

escolha é efectuada com base na energia que é preciso gastar (distância a que se encontra o componente) e na energia que o componente dá. Quando se processa a criação do agente, é necessário transmitir-lhe a composição do cenário. Essa composição serve de referência sobre a localização dos diversos componentes (zonas em que se encontram) através da qual o agente se guia para estimar a distância.

O agente pode iniciar o seu ciclo de execução com um plano (pilha de desejos) pré-definido (se for especificado o escoamento – subsecção 3.3.2) que vai sendo ajustado à medida que o estado interno se altera, ou sem nenhum plano e que é gerado dinamicamente (pela alteração do estado interno).

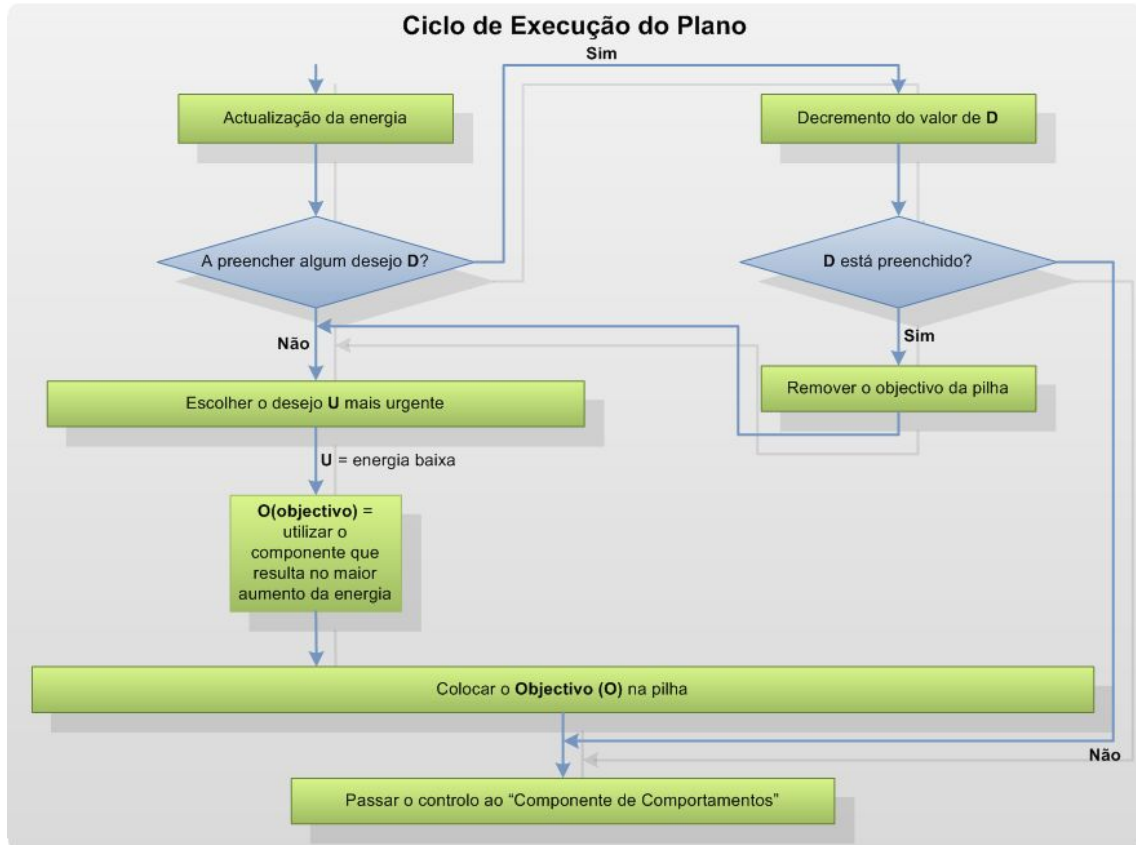


Figura 3.21 – Ciclo de execução do plano (baseado em [Wei06], figura 4.14).

Como se pode observar no diagrama ilustrado na Figura 3.21, cada ciclo de execução do plano inicia-se com a actualização do campo de energia (este é incrementado ou decrementado dependendo da acção que foi tomada). Seguidamente, o agente verifica a satisfação de preenchimento de um desejo, continuando a executá-lo ou passando para o desejo seguinte consoante o valor de satisfação. No decorrer de um ciclo é também verificado o campo de energia. Se este assumir um valor considerado baixo, o agente adiciona à pilha o desejo de utilizar um componente que lhe aufera o maior incremento da energia (distância que está e energia que pode ser adquirida). O ciclo termina com a passagem do controlo ao componente de comportamentos de forma a deliberar o desejo com a aplicação do comportamento “mover para alcançar alvo”.

Quando a regra de utilização de um componente dita a probabilidade de utilização de um tipo de componentes, o agente calcula a instância do componente (rotulado) que pretende usar com base no provável gasto de energia.

O campo energia pode ser determinante numa eventual simulação em situações de emergência. Bastando, para tal, introduzir os agentes na simulação com um valor de energia baixo e atribuir aos componentes (representativos dos “pontos de encontro”) uma capacidade de aumento de energia elevada. Levando os agentes à procura constante deste tipo de componentes, simulando, assim, uma evacuação por saídas de emergência.

3.7 Síntese

Este capítulo apresentou a especificação de uma ferramenta de simulação microscópica pedonal, baseada em agentes, em interfaces multi-modais, apelidada de ModP. Estão também descritas a arquitectura global, como as dos diversos módulos que a compõem.

Cada um dos diversos módulos pressupõe um conjunto de abstracções e/ou características particulares, promovendo a usabilidade, redução da interacção e um certo grau de modularidade que permite a criação de vários cenários de simulação de forma rápida e simples. Desde a edição do modelo de simulação, através de uma interface gráfica com os mais recentes padrões de desenho, de uso fácil e intuitivo, até à sua simulação, através de um motor de simulação, que usa a estrutura básica do modelo simplificada e totalmente abrangente. O motor de simulação possui a possibilidade de introdução de Plug-ins de simulação, alargando os tipos de simulação e permitindo futuros desenvolvimentos de forma simples e com total aproveitamento de toda a estrutura construída.

A especificação dos agentes baseia-se em comportamentos de baixo nível (abordagem de “forças sociais”) e de alto nível (abordagem BDI). Apesar de existirem variadas arquitecturas que podem capacitar os agentes de certas habilidades mentais, foi sugerida uma arquitectura simplificada baseada do paradigma BDI (abordagem bastante popular). Esta aproveita a abstracção realizada sobre as regras de utilização dos componentes e apenas o aumento ou diminuição do valor da energia, para criação a actualização do seu plano.

O campo energia é não só importante para a improvisação (comportamento heterogéneo), como para a simulação em situações de emergência.

Capítulo 4

Desenvolvimento do Protótipo

No último capítulo, foi especificada uma arquitectura de alto nível para a plataforma de simulação pedonal ModP. O protótipo desenvolvido foi baseado nessas especificações, tendo como principal objectivo demonstrar a flexibilidade da arquitectura especificada e abordar os principais paradigmas e directrizes intrínsecos ao desenvolvimento de uma plataforma deste tipo. Este capítulo descreve detalhadamente os pormenores mais importantes da implementação do protótipo.

As primeiras duas secções (4.1 e 4.2 deste capítulo) pretendem dar a conhecer as interfaces gráficas implementadas, a sua composição e o tipo de usabilidade e interacção pretendidos. São também referidas, as tecnologias utilizadas e de que forma é que a adopção de uma plataforma de desenvolvimento de software (Qt) e o recurso a uma biblioteca de desenho 3D (OpenGL®) podem ser úteis para a implementação destas.

A secção 4.3 descreve a arquitectura lógica do protótipo desenvolvido, detalhando os aspectos mais relevantes dos módulos implementados. Esta secção realça a arquitectura modular da plataforma, como um meio para atingir a sua expansibilidade e clarifica a interligação entre os módulos. É possível também interpretar o modo como são construídos os dados e como é elaborada a sua estruturação, desde os processos simples do módulo de edição gráfica até ao seu uso semântico por parte dos agentes. A parte final desta secção está destinada à explicação detalhada de micro comportamentos baseados em forças sociais e do modelo de decisão sobre a atracção aos vários pontos, com recurso a uma pilha de desejos.

A secção 4.4 apresenta o ciclo de execução da simulação, detalhando os processos ocorrentes nos diferentes estados intermédios de cada ciclo de execução.

Por último, na secção 4.5, será apresentado uma síntese sobre este capítulo realçando os aspectos mais característicos de cada módulo.

4.1 Tecnologias Utilizadas

O protótipo da ferramenta ModP foi desenvolvido em C++ utilizando a versão 4 de código aberto da plataforma de desenvolvimento Qt[Nok08]. Este é um sistema multi-plataforma que providencia uma API intuitiva, uma biblioteca abastada de classes C++, um conjunto de ferramentas para desenvolvimento avançado de interfaces gráficas do utilizador (GUI) e permite

o desenvolvimento de aplicações de desktop que funcionem em diferentes sistemas operativos sem a necessidade de reescrever o código fonte. A principal motivação na escolha desta plataforma recai sobre dois aspectos importantes:

- **Signals and Slots** – é uma linguagem de construção de código que permite uma fácil e rápida implementação do padrão de desenho de software *Observer* [Nok08][Eal05] (um objecto mantém uma lista de objectos dependentes e notifica-os automaticamente se houver alguma mudança de estado). O conceito consiste na possibilidade de um objecto controlo (*widget*) poder enviar sinais contendo informação sobre eventos (por exemplo, a ferramenta de edição de paredes foi seleccionada, o compositor de cenário muda o estado para desenho de segmentos de recta do tipo paredes) e estes poderem ser recebidos por outros objectos controlo através de funções *slot*. O sistema *Signal/Slot* enquadra-se bem com a forma como são concebidas as GUI.
- **Motor Gráfico 2D e 3D** – oferece um excelente suporte para desenho de gráficos 2D e 3D. As classes internas de gráficos 2D suportam imagens bitmaps e gráficos vectoriais, enquadrando-se com a construção do cenário e desenho dos gráficos estatísticos. Por sua vez, as classes internas de gráficos 3D permitem o acesso às funcionalidades da biblioteca OpenGL® [Gol09], facilitando a integração de gráficos 3D, o que sustenta a ideia do visualizador de simulação 3D.

A arquitectura do Motor Controlador da Simulação com um sistema de integração de *Plug-ins*, bem como, a implementação dos comportamentos direccionais, são baseados na biblioteca de código aberto OpenSteer [Ope04]. Esta biblioteca disponibiliza um conjunto de componentes e ferramentas para ajudar a construir comportamentos direccionais de personagens autónomas em jogos e outras formas de simulações multi-agente. Além da biblioteca, o OpenSteer fornece uma aplicação chamada OpenSteerDemo baseada numa arquitectura de *Plug-ins*: uma plataforma vazia em que podem ser adicionados incrementalmente *Plug-ins*.

4.2 Descrição das Interfaces Gráficas

O desenvolvimento do protótipo deu origem a um conjunto de interfaces gráficas (Figura 4.1) que assumem papéis e características completamente distintos.

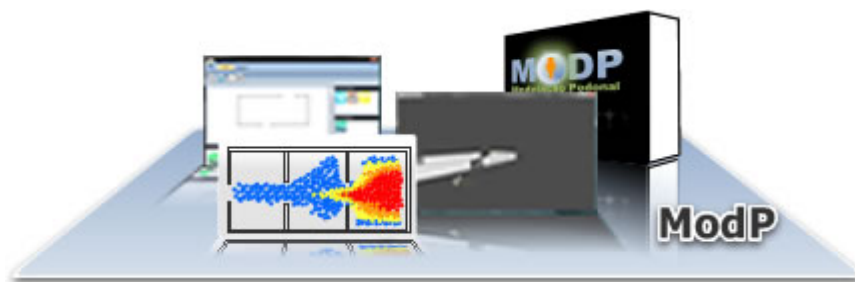


Figura 4.1 – Apresentação da plataforma ModP.

Esta distinção vem de acordo com as características que o ModP pretende atingir, como referenciado na secção 3.2 e com o paradigma do que um bom software de informação e manipulação deve disponibilizar (secção 3.1). Durante esta secção, serão evidenciadas essas interfaces e será feita uma analogia com a especificação correspondente.

4.2.1 Interface Gráfica de Edição

Como especificado na secção 3.4, a Interface Gráfica de Edição (IGE) pressupõe uma disposição e funcionamento simples, usáveis e intuitivos seguindo alguns dos mais recentes

padrões de desenho/concepção de interfaces gráficas do utilizador. A Figura 4.2 representa a IGE desenvolvida e baseada nesses princípios.

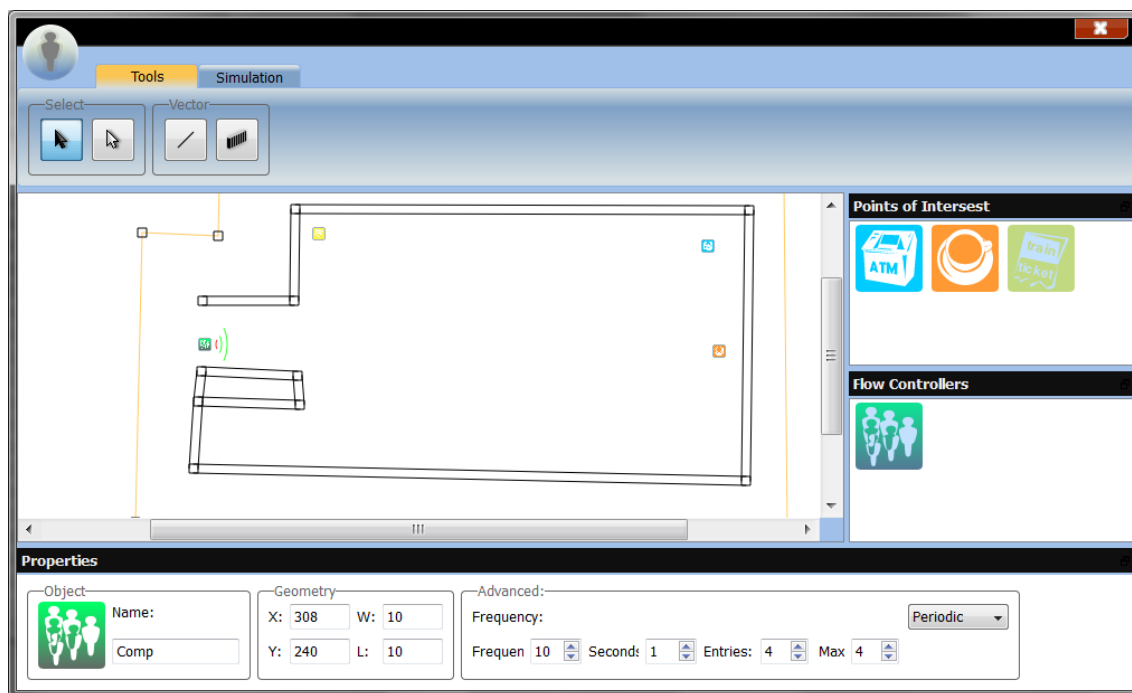


Figura 4.2 – Interface Gráfica de Edição.

Na Figura 4.2, estão destacadas quatro zonas distintas localizadas e compostas por um conjunto de funcionalidades e/ou componentes de forma análoga à especificação:

- Zona 1 (ModP Menu)** - Esta zona corresponde ao Menu Contextual de Ferramentas. Este organiza essas ferramentas em abas e agrega-as pelo grupo de funcionalidades genéricas (padrão *Ribbon*). Como é possível observar na Figura 4.2, os ponteiros de selecção de objectos ou vértices (preto e branco, respectivamente) encontram-se num grupo designado “Select”, enquanto as ferramentas para desenhar linhas ou paredes encontram-se no grupo “Vector” e ambos os grupos na mesma aba. Sempre que uma destas ferramentas é seleccionada é mudado o estado do compositor do cenário. A segunda aba é relativa ao controlo da simulação e agrega funções como “play”, “stop” e “display statistics”. O canto superior desta zona destina-se ao Menu de Projecto e está representado pelo botão redondo com o símbolo do ModP.
- Zona 2 (ModP Component Dock Widgets)** – É representativa das janelas de alocação que possuem a lista de componentes de composição de cenário, representados por ícones, que podem ser agarrados e largados no compositor do cenário. Na parte superior, encontra-se o conjunto de pontos de interesse, mais especificamente, multibanco, bar e bilheteira. Já na zona inferior está o conjunto de controladores de fluxo (componente de Entrada/Saída).
- Zona 3 (Properties)** – Esta zona, alocada na parte inferior da janela, permite a edição das propriedades do objecto seleccionado. Reage de forma contextual e disponibiliza um formulário de edição concordante com as propriedades dos objectos que podem ser parametrizadas. Nesse formulário, os campos de edição possuem uma filtragem do tipo de dados ou formato de acordo com as características da propriedade. Por exemplo, não pode ser inserido texto nas propriedades de geometria do objecto.
- Zona 4 (Edition Widget)** – É a zona que acarreta maior relevância e assume um papel fundamental para a construção do cenário, ocupando, por isso, a zona central e uma área maior. Esta corresponde ao compositor do cenário e permite a construção deste através de processos simples, como “clicar, arrastar e largar” e “pegar e largar”. O compositor do cenário segue o padrão “O Que Você Vê É O Que Você Tem” (linhas

duplas para paredes, simples para delimitadores do movimento e ícones representativos dos componentes compositores do cenário), comporta o comportamento de alto nível de junção de segmentos de recta e o de sugestão de criação de zona, cujo funcionamento será explicado mais à frente. O utilizador pode também realizar operações de aproximação e afastamento sobre esta zona.

4.2.2 Interfaces Gráficas de Visualização 3D e Estatísticas

Um dos principais focos da especificação do ModP é relativo à abordagem dos softwares de informação, prende-se no detalhe e qualidade de informação não ambígua que se deve obter através da simulação. As interfaces implementadas, que possuem características de visualização da informação, pretendem ser exemplificativas dos modelos de informação que podem ser gerados. Estes passam pela visualização 3D da simulação, estatísticas de locomoção dos peões e do uso do tipo de componentes compositores de cenário. A Figura 4.3 apresenta essas interfaces.

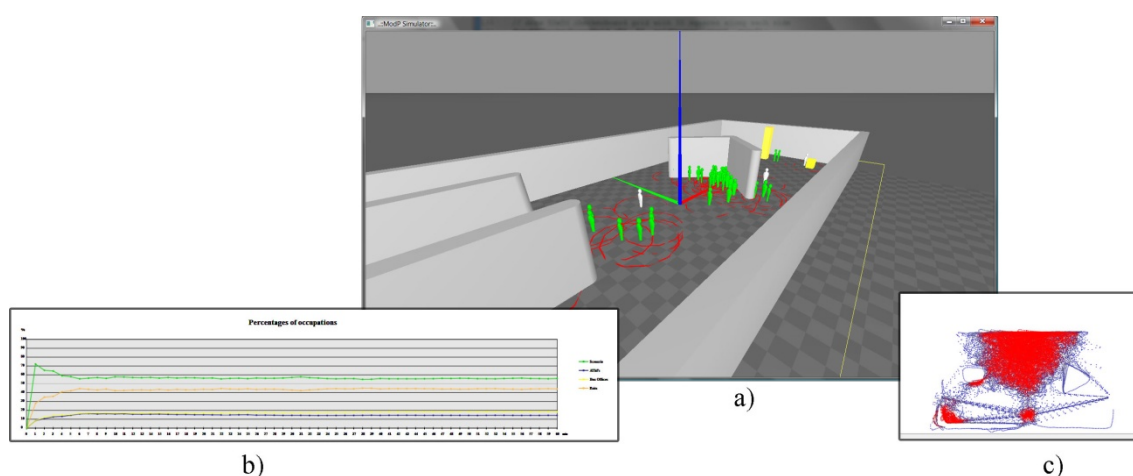


Figura 4.3 – Interfaces Gráficas de Visualização 3D e Estatísticas.

A Figura 4.3.a) exhibe o detalhe da interface de visualização 3D. Esta caracteriza os objectos num espaço tridimensional e assemelha a sua forma à gerada pelo compositor do cenário, permitindo uma comparação mais adequada. A representação dos peões é efectuada através de um corpo semelhante ao esboço do contorno do corpo humano.

As figuras, Figura 4.3.b) e Figura 4.3.c), representam, respectivamente, um gráfico de taxas de utilização dos tipos de componente e um mapa de densidade. Este último representa as zonas por onde passaram os peões, demarcando a vermelho as mais sobrecarregadas.

4.3 Arquitectura Lógica

Tendo em conta as necessidades de performance de execução e das boas práticas de desenvolvimento de software, o protótipo desenvolvido foi dividido em quatro módulos lógicos principais, tal como ilustrado na Figura 4.4. O objectivo desta divisão baseou-se também em proporcionar a capacidade de extensibilidade ao protótipo de forma a ser possível o seu reaproveitamento para a implementação de outro tipo de simulações (através do desenvolvimento de diferentes *Plug-ins*).

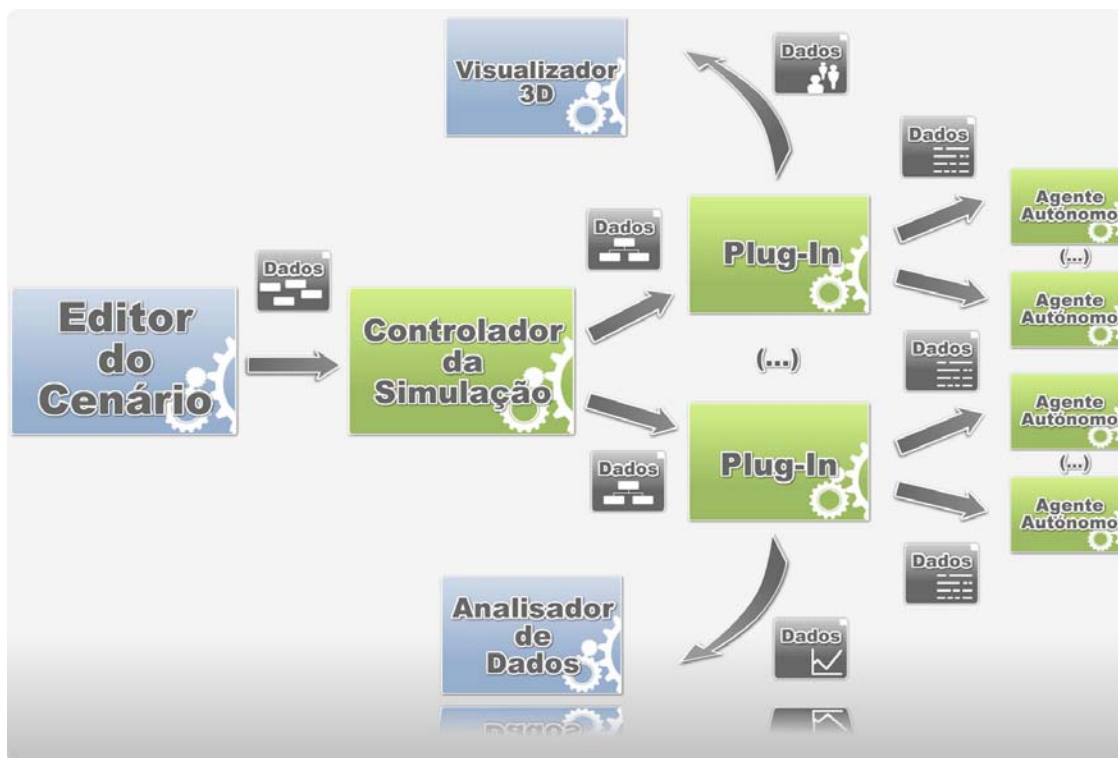


Figura 4.4 – Esquema representativo da arquitectura lógica.

Para efeitos deste protótipo os módulos ilustrados não foram separados fisicamente. Contudo a sua arquitectura/implementação foi desenvolvida para permitir uma eventual separação de uma forma fácil, sendo apenas necessária a especificação e desenvolvimento de um protocolo de comunicação bem como as necessárias alterações às interfaces de comunicação entre os módulos. No diagrama ilustrado na Figura 4.4, é possível observar a comutação de dados entre os diferentes módulos lógicos.

O processo de construção inicia-se no **Módulo Editor do Cenário** através da interface gráfica do utilizador fornecida, permitindo a construção de um cenário de simulação bem como a edição de algumas das suas propriedades, constituindo assim o modelo de simulação.

O **Módulo Controlador da Simulação** executa uma estruturação de dados oriundos do Módulo Editor do Cenário e processa-os de uma forma abstracta (processamento não semântico). Fornece também métodos que permitem a consulta e alteração dos dados, que serão posteriormente processados (semanticamente) pelos diferentes *Plug-ins*. Este módulo é o responsável por toda a gestão dos diferentes *Plug-ins* e pela respectiva sincronização, de acordo com o seu relógio interno.

O **Módulo Plug-in** está capacitado de uma análise semântica permitindo-o recorrer de forma adequada às estruturas de dados alojadas no módulo controlador da simulação, satisfazendo assim, os pedidos dos agentes. Cada *Plug-in* possui um ciclo de execução interno, despoletado a cada unidade de tempo (*time-step*), materializador do estado evolutivo da simulação. No decorrer desses ciclos, poderá existir uma eventual necessidade de criação, inserção e registo de agentes, satisfazível através deste módulo.

O **Módulo Agentes Autónomos** é responsável por caracterizar os actores interactivos do cenário. Este é baseado numa arquitectura de agentes autónomos. Estes são capazes de perceber o ambiente/cenário em que são inseridos e executar um conjunto de acções sobre o mesmo. Essas acções são tomadas de acordo com a informação consultada (através do *Plug-in*) sobre o estado do cenário e atendem às regras estipuladas. Estes agentes possuem autonomia de locomoção e um certo grau de habilidade de decisão.

As próximas subsecções descrevem estes módulos com maior detalhe, assim como as ligações entre eles. É de notar, ao longo desta secção, a presença de camadas de interfaces de interligação entre módulos. O facto de aparecerem expressas em dois módulos distintos e com o mesmo nome, não significa que estas são distintas e comuniquem. De facto, estas interfaces são

o mesmo objecto (até porque os módulos não se encontram fisicamente separados) e apenas se encontram expressas em módulos distintos, para uma maior facilidade de compreensão.

4.3.1 Módulo Editor do Cenário

Este módulo oferece ao utilizador, tal como referido anteriormente, um conjunto específico de funcionalidades que lhe permite, de forma intuitiva, a edição e parametrização do cenário. Essas funcionalidades estão distribuídas por quatro camadas específicas representadas na Figura 4.5.



Figura 4.5 – Arquitectura em camadas do módulo Editor do Cenário.

A camada **Gestor de Eventos** trata toda a informação proveniente das acções realizadas pelo utilizador. Essas acções podem ser categorizadas pelo tipo de operações a que se destinam ou pelo tipo de mecanismos que accionam:

- **Mudança de Estado** – este mecanismo é accionado através da selecção de ferramentas que se encontram no ModP Menu na aba *Tools* (subsecção 4.2.1). Dependendo do tipo da ferramenta que é pressionado, o estado interno do módulo de edição é alterado. Por exemplo, se for seleccionada a ferramenta de selecção de vértices, o estado interno é alterado e o utilizador, a partir deste momento, só pode seleccionar/arrastar vértices de segmentos de recta. Já se for seleccionada a ferramenta de edição de paredes, o utilizador pode executar operações de desenho vectorial, através de “clicar, arrastar e largar”. A permutação dos estados internos de módulo de edição de cenário, através deste tipo de mecanismos, não impossibilita a inserção de outros objectos através do método “pegar e largar”.
- **Inserção de Objectos** – esta acção é despoletada através dos conjuntos de sequências “pegar e largar” e/ou “clicar, arrastar e largar”. Quando alguma sequência deste tipo é executada, este mecanismo informa quer o Motor Gráfico 2D, quer a Interface de Inserção de Dados de forma a serem cumpridas as intenções do utilizador (inserção de objectos de “Entrada/Saída”, “Conectores”, “Pontos de Interesse” e “Delimitadores de Movimento”).
- **Alteração das Propriedades dos Objectos** – este conjunto de acções destina-se a alterar/parametrizar os objectos inseridos no compositor do cenário (*Edition Widget*, subsecção 4.2.1). Dependendo do tipo das propriedades dos objectos, podem existir dois métodos de alteração das suas propriedades. Um dos métodos ocorre através do uso do ponteiro do rato, depois de se efectuar a “mudança de estado” para modo de selecção, bastando arrastar o objecto (alterando-lhe a posição). O outro é fruto da utilização do editor contextual de propriedades (*Properties*, subsecção 4.2.1), permitindo alterar as propriedades que facultam a sua parametrização. Essa permissão é intrínseca ao objecto e é este que constrói o formulário de edição de propriedades no momento da sua selecção.

- **Controlos da Simulação** – conjunto de acções que permitem a selecção de um determinado *Plug-in* de simulação através de uma lista providenciada pela camada interface de controlo da simulação (no protótipo desenvolvido esta lista é unitária uma vez que só existe um *Plug-in*, pelo que esta acção é desvalorizada) e o controlo de início e fim de execução da simulação através do *Plug-in* seleccionado. Estas acções são despoletadas pelas ferramentas na aba “*Simulation*” no grupo “*Control*” (subsecção 4.2.1). Quando alguma destas ferramentas é pressionada, é enviado um sinal à interface de controlo da simulação com a informação do respectivo evento.
- **Disponibilização de Estatísticas** – esta acção permite a disponibilização da interface gráfica do módulo de análise de dados, quando pressionada a ferramenta correspondente na aba “*Simulation*” (subsecção 4.2.1).

A inserção de Delimitadores de Movimento pressupõe dois comportamentos de alto nível: “não replicação de pontos coincidentes” e “sugestão de criação de zona”. O primeiro comportamento, exemplificado na Figura 4.6, pretende automatizar o processo de criação de segmentos de recta vectoriais ligados, assumindo que essa é a vontade do utilizador (ex: quando coloca um vértice de um segmento em sobreposição ao vértice de outro).

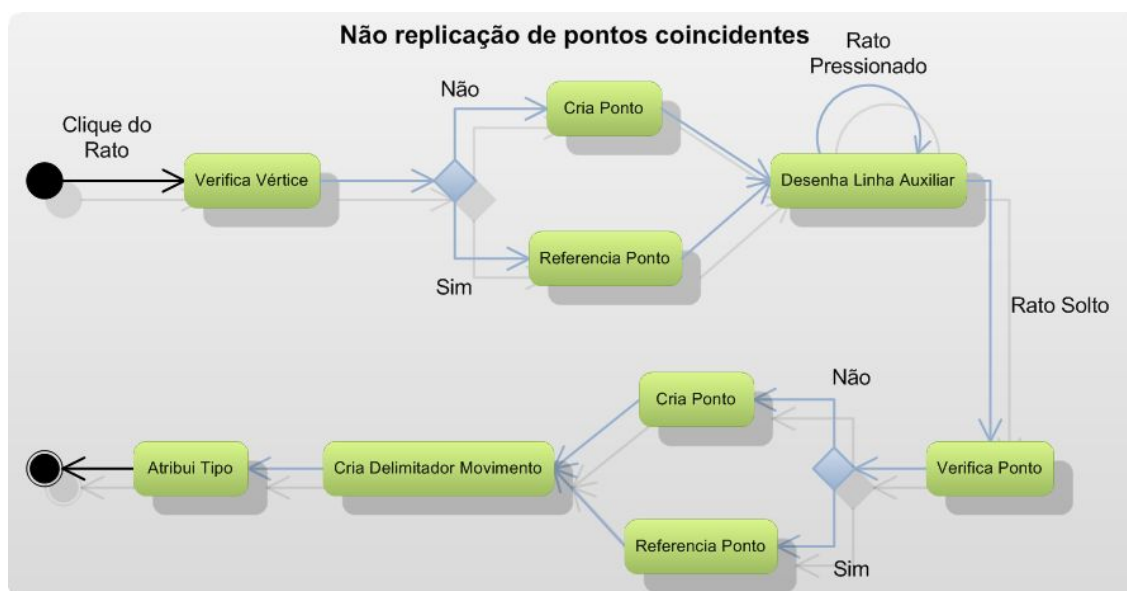


Figura 4.6 – Esquema representativo do comportamento de não replicação de pontos.

Este processo pressupõe, em primeiro lugar a “mudança de estado” para o modo de construção de passeios ou paredes e sendo despoletado aquando um “clique” no compositor de cenário (zona central da janela, *Edition Widget*, subsecção 4.2.1). Nesse instante, é testada a sobreposição do vértice que se pretende inserir sobre algum outro. Se tal não acontecer é criado esse vértice, caso contrário, o vértice que se pretende inserir é substituído pelo vértice existente naquela posição. Em qualquer um dos casos, o processo continua e enquanto o botão do rato não for largado, é gerada uma linha de auxílio permitindo elucidar o utilizador sobre a linha que poderá gerar. No instante em que ocorre o “largar” do botão do rato, é testado o mesmo tipo de sobreposição, tal como para a inserção do primeiro vértice e é efectuado o mesmo processamento. No final, é criado um segmento de recta com os vértices resultantes e definido o tipo consoante a ferramenta estipulada.

O segundo comportamento baseia-se no ciclo de execução do comportamento anterior, mas assume um objectivo mais alargado: sugerir a criação de uma zona. Este tem como principal desafio simplificar o processo de criação de zonas, através de uma suposição. Quando um utilizador executa um ciclo de criação de segmentos de recta ligados e passa pelo vértice de início do ciclo, criando uma estrutura fechada (delineada por esses segmentos) provavelmente quer criar uma zona. A Figura 4.7 representa o ciclo que leva a essa suposição.

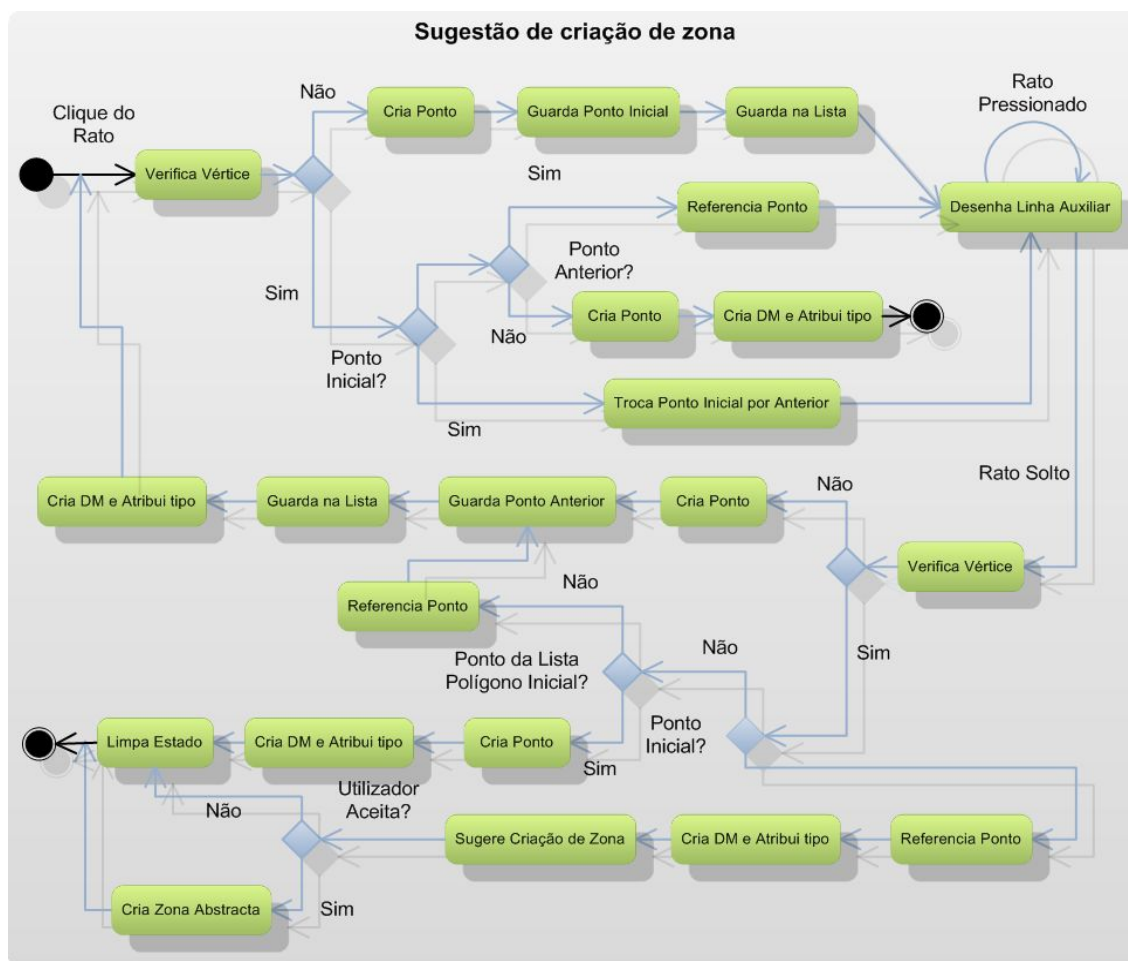


Figura 4.7 – Esquema representativo do comportamento de sugestão de zona.

O comportamento “sugestão de criação de zona” inicia o seu ciclo tal como o primeiro comportamento. Após um “clique” é testada a sobreposição. Se for negativo, é criado o primeiro vértice na área gráfica específica, é anotado como vértice de início de ciclo e guardado numa lista (vértices do polígono, caso a zona seja criada). As grandes diferenças, com o comportamento anterior, começam neste momento. Quando o teste de sobreposição dá positivo, é verificado se esse vértice é o de início de ciclo. Isto acontece, porque se o processo se encontrar num ponto avançado do ciclo, não quer dizer que o utilizador abandonou o processo, mas pode apenas ter trocado de sentido. Caso o vértice seja o de início é efectuada a troca de referência com o último vértice criado (o vértice de início passa a ser o último e vice-versa), promovendo a troca de sentido. Se este vértice não for o de início, é verificado se é o último criado condicionando o prosseguimento o processo. Caso o processo continue, é desenhada a já referida linha auxiliar até se processar o “largar” do botão do rato. Nesse momento e se não houver sobreposição é criado vértice final do segmento, que por sua vez é anotado como o último vértice criado e guardado na lista de vértices. Mal se gere a criação do delimitador de movimento com o tipo correcto o ciclo retorna ao estado inicial, ficando à espera de um outro “clique”. No caso de a adição do segundo vértice significar uma sobreposição, é verificado se é o vértice inicial. Em caso negativo, é verificado se é algum dos pontos que estão na lista de um possível polígono (evitando a criação da zona só com dois segmentos) e a negação permite continuar o processo, gerando-se o vértice (passa a ser o último criado), o delimitador de movimento e o ciclo retorna ao início. Finalmente, e quando o teste de vértice de início é ultrapassado, gera-se o delimitador e é sugerido a criação de uma zona. Caso o utilizar confirme a sugestão a zona é criada.

A camada intermédia, **Motor Gráfico 2D** é constituída por um conjunto de métodos que permitem o desenho de bitmaps e gráficos vectoriais, facilitando a disponibilização em modo gráfico de algumas das acções tomadas pelo utilizador. Nomeadamente, o desenho dos diversos

componentes do cenário (com aspecto concordante com o tipo e/ou ícone representativo do tipo de objecto) bem como a sua disposição.

A camada **Interface de Inserção de Dados** é responsável pela interligação com o Módulo Controlador de Simulação. Esta permite que os dados/objectos gerados, no decorrer dos vários eventos realizados pelo utilizador (no intuito de construir o modelo de simulação), sejam processados e inseridos no módulo Controlador da Simulação. A escrita desses dados não recai, unicamente, sobre a inserção de novos objectos, mas deve também proceder à escrita de novas parametrizações (caso sejam efectuadas). A vantagem desta constante escrita reflecte-se na permissão de inserção e manipulação de parâmetros no decorrer da simulação.

A inserção e alteração das propriedades dos objectos do cenário pressupõe o respeitar das suas características e dos seus tipos de dados. A Figura 4.8, que se segue, demonstra os objectos que podem ser criados, bem como os parâmetros que possuem.



Figura 4.8 – Características dos objectos criados pelo módulo editor cenário.

Na Figura 4.8, estão representados quatro objectos que resultam de algumas das abstracções do modelo de referência do ModP (secção 3.3) e da especialização do componente de cenário de Entrada/Saída. A adição deste tipo de componente resulta das suas características singulares e da importância que estes adquirem para a simulação. Quase todos os objectos possuem um conjunto de propriedades genéricas, permitindo na sua identificação (id e tipo), editar a sua geometria (posição, tamanho e orientação) e localizá-lo através da zona (de modo abstracto, já que a zona não possui estrutura física neste módulo, sendo apenas um conceito contextual). Contudo existem outras propriedades que os distinguem:

- **Ponto de Interesse** – este objecto pode representar, tal como descrito na subsecção 3.3.2, o conjunto de componentes compositores do cenário que podem auxiliar o agente a atingir o objectivo final (interferência estipulada pelas regras). A sua inserção, no cenário, realiza-se através de eventos de “pegar e largar” para o interior de uma zona de um determinado nível. Quanto às suas propriedades individuais, estes possuem “Tempo Médio de Utilização” obrigando o agente a esperar um determinado tempo, para satisfazer o seu desejo. A propriedade “Fila de Espera” permite os agentes registarem-se de forma a marcarem a ordem de utilização e a variável “Ocupação Máxima” permite estipular o número máximo de agentes que podem utilizar o ponto de interesse em simultâneo.
- **Entrada/Saída** – estes objectos são responsáveis tanto por parametrizar o local e a taxa de entrada de agentes, como por servir de objectivo final (sendo os que satisfizerem esse desejo, serão eliminados). Também permitem a parametrização do período de tempo em que pode ser efectuada a entrada de agentes, bem como a intermitência desse período. Dependendo da periodicidade (“contínuo”,

“periódico” e “único”) podem ser estipuladas a taxa de entrada (percentagem/unidade de tempo), intervalo de unidades (5-10 pessoas) por disponibilização e intervalo de unidades, respectivamente. No caso do valor da periodicidade ser “periódico” ou “único”, o seu estado interno alterna entre “disponível” e “indisponível”.

- **Conector** – permite a passagem dos agentes de uma zona para outra, efectuando esse processamento através de uma fila de espera. Este objecto possui um número máximo de ocupação em simultâneo (isto é importante para o caso dos elevadores) e dois extremos ligados, cada um deles, a duas zonas distintas (permitindo ao agente distinguir a entrada e saída do componente).
- **Delimitador de Movimento** – este tipo de objectos assume características distintas dos outros objectos. Os delimitadores de movimento possuem uma estrutura vectorial, um tipo (paredes ou passeios) e dois vértices que podem ser movimentados de forma a se ajustar a orientação e comprimento. Para além do comprimento, estes também possuem as propriedades de altura e espessura.

Por último temos a camada **Interface de Controlo da Simulação** que permite requisitar, ao módulo controlador de simulação, informações sobre os *Plug-ins* existentes e a execução de operações de arranque e término sobre *Plug-in* seleccionado.

4.3.2 Módulo Controlador da Simulação

O Módulo Controlador de Simulação é o módulo base de toda a aplicação ModP, no que diz respeito à simulação. Este assume características que permitem um certo grau de modularidade e uma consulta estruturada aos dados do modelo de simulação. O seu funcionamento é auxiliado por conjunto de mecanismos capazes de sincronizar uma determinada simulação, bem como, possibilitar o acesso a dados de um modo abstracto. As camadas responsáveis por estes mecanismos estão ilustradas na Figura 4.9 e as suas características principais serão descritas nesta subsecção.



Figura 4.9 – Arquitectura em camadas do módulo Controlador da Simulação.

A camada **Interface de Inserção de Dados** providência a recepção de dados provenientes do módulo de editor de cenário. Uma vez que os dados recebidos não se encontram estruturados, esta é responsável por accionar os mecanismos necessários para que a estruturação, dos mesmos, seja realizada de acordo com o tipo de componente de cenário e no contexto da sua localização.

Os mecanismos de estruturação de dados encontram-se na camada **Gestor de Dados**. Esta é responsável não só por estruturar os dados, mas também por criar estruturas auxiliares que facilitem e automatizem a consulta dos mesmos. A primeira estrutura de dados gerada está

representada na Figura 4.10 e vem de encontro com a especificação da estrutura do cenário (Figura 3.8) e das categorias dos componentes (subsecção 3.3.2).

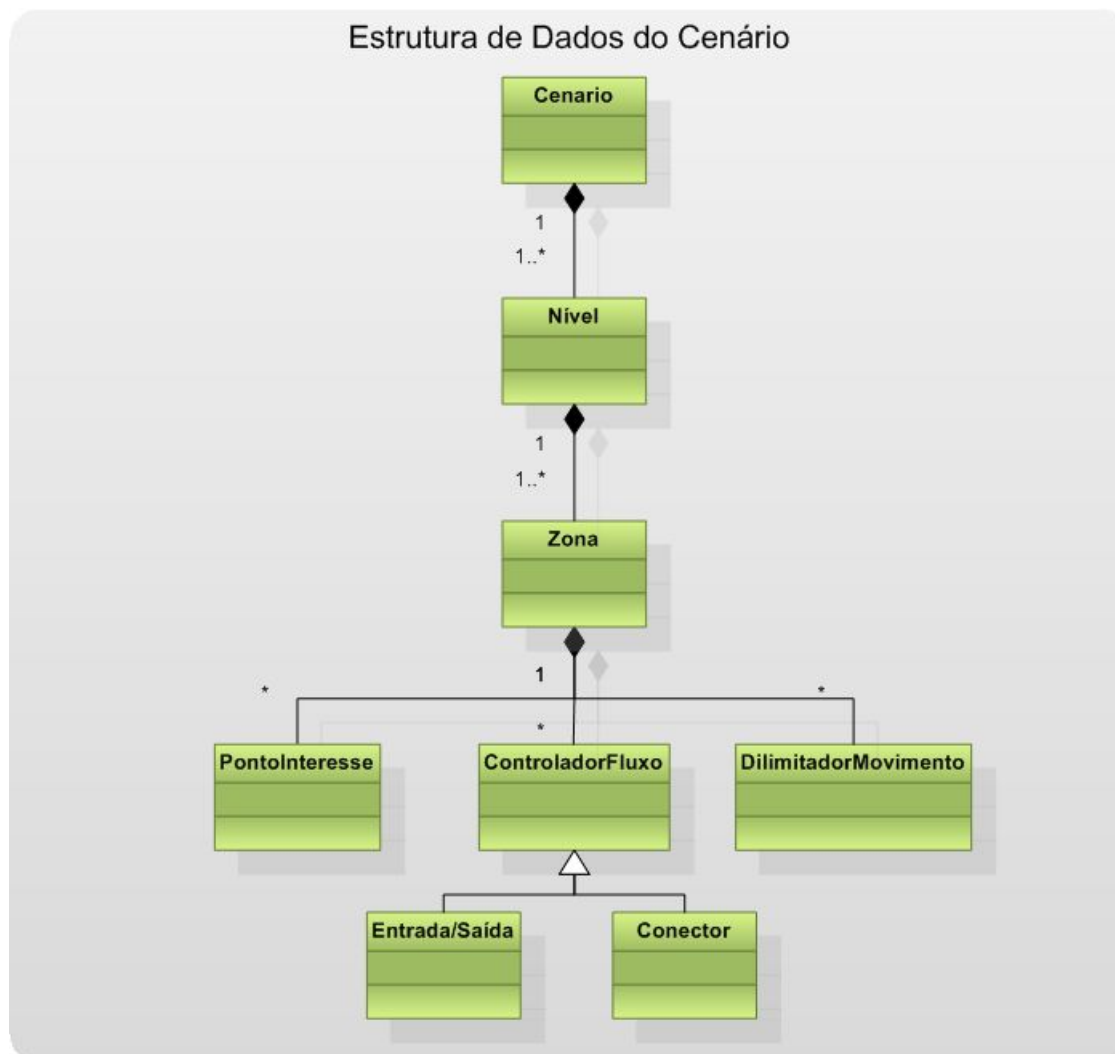


Figura 4.10 – Estrutura hierárquica de dados criada pelo Gestor de Dados.

Como se pode observar na figura o cenário é composto por níveis representativos dos pisos (no caso do protótipo desenvolvido, o cenário é somente composto por um nível) que podem possuir uma ou mais zonas. Essas zonas (identificadas por um id) são compostas por delimitadores de movimento (os delimitadores inseridos na fronteira da uma zona podem fazer parte de outra), por controladores de fluxo (especificamente Entrada/Saída e Conector) e também por pontos de interesse. Estes últimos são guardados de forma genérica/abstracta e não semântica. Quer isto dizer que o módulo de simulação assume a existência de vários tipos, mas não os consegue identificar pelo tipo de objectivos a que se destinam. Por exemplo, sabe que existe um ponto de interesse do tipo 3 mas não sabe se é “multibanco” ou “bilheteira” nem para que serve. Essa diferenciação é realizada pelo módulo Plug-in, dependendo do tipo de simulação que pretende realizar e de acordo com as intenções dos agentes, que este insere na simulação.

Como falado anteriormente, a gestão/manipulação de dados pressupõe a criação de estruturas auxiliares. Estas estruturas são baseadas no tipo dos objectos recebidos e nas instâncias dos mesmos. Para uma melhor interpretação destas estruturas, será exemplificado um processo de construção destas estruturas, com base nos objectos/componentes de cenário inseridos no protótipo. Imagine-se o conjunto de componentes de cenário tipificados como 1 (“multibanco”), 2 (“bilheteira”), 3 (“bar”) e 4 (“entrada/saída”) e com, respectivamente 3, 5, 1 e 3 instâncias. Tenha-se em atenção que as definições semânticas servem para uma melhor compreensão e não são tomadas em conta por este módulo. A Figura 4.11 representa as estruturas de dados geradas com base neste exemplo.

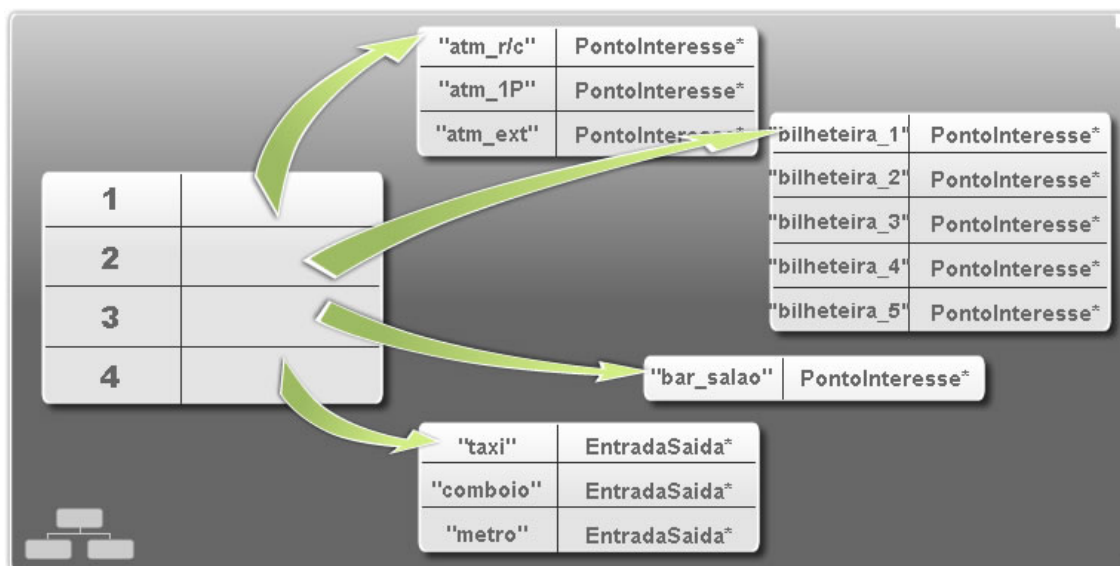


Figura 4.11 – Estrutura de dados auxiliares criadas pelo Gestor de Dados.

Quando o gestor de dados recebe um pedido de inserção de um objecto, pela interface de dados, é efectuada uma leitura da informação da zona transportada no objecto, do seu tipo e do seu identificador (id). Com base no tipo, é gerada uma linha na tabela de referência (*hashtable*), com chave igual ao tipo e valor igual à referência para uma tabela de objectos. Estas tabelas de objectos guardam a referência dos objectos de um determinado tipo (chave igual ao id do objecto e valor igual à referência do objecto – apontador *Objecto**). No caso de o tipo do objecto já existir na tabela, o gestor acede à tabela referenciada nessa linha, inserindo-lhe a instância do objecto. Estas estruturas permitem o acesso a dados por tipo de objecto e/ou pelo identificador do objecto instanciado.

A camada **Gestor e Sincronizador da Simulação** é responsável pela gestão e sincronização dos módulos *Plug-ins* existentes. A existência deste tipo de módulos é um factor importante para a expansibilidade da aplicação ModP. O protótipo desenvolvido possui uma arquitectura de *Plug-ins*, mas não permite a adição dinâmica desses (esse dinamismo está remetido para trabalho futuro), sendo necessária a recompilação do código de forma a estes serem registados. Contudo, se estes já existirem e no momento da compilação, são registados de forma incremental numa lista de *Plug-ins* disponíveis. Esta camada permite o acesso a essa lista e o forçar dos mecanismos arranque, término e sincronismo desses *Plug-ins*. A Figura 4.12 ilustra o processo de arranque de uma simulação.



Figura 4.12 – Mecanismo de gestão de *Plug-ins*.

Para que uma simulação comece, é necessária a selecção de um *Plug-in* e um pedido de execução oriundo da interface de controlo da simulação. Quando é requisitado o início de simulação de um determinado *Plug-in*, o gestor e sincronizador procura-o na lista e destaca-o/referencia-o como “seleccionado”. De seguida dá início ao processo de simulação, activando

o relógio interno (através do gestor de tempo) e forçando o *Plug-in* seleccionado a executar o seu ciclo de simulação através da interface de *Plug-in*. Arrancada a simulação, esta camada fica responsável pela sua sincronização.

A sincronização é realizada de forma cíclica e com recurso a unidades de tempo de simulação (*time-step*). Durante cada ciclo, o *Plug-in* é forçado a actualizar o seu estado de simulação (com base no valor da unidade de tempo) e este ciclo só termina quando terminada a actualização. No final de cada ciclo de actualização, o gestor de tempo é avisado para terminar a unidade de tempo actual e dar início à próxima. Sempre que a interface de controlo da simulação executa um pedido de término, o Gestor e Sincronizador da Simulação força o *Plug-in* seleccionado a terminar todo o seu processo. O relógio interno é desligado.

A camada, **Gestor de Tempo de Simulação**, é responsável por gerar as unidades de tempo de simulação, que assumem um valor fixo (ex: 0.1 segundos). Esta possui o controlo do relógio interno da simulação e providência, ao gestor e sincronizador da simulação, o tempo total de simulação e o valor de cada unidade de tempo. Esta camada impede o início de uma unidade de tempo nova, enquanto a anterior não terminar.

A camada **Interface de *Plug-in*** é responsável por fazer a interligação com o módulo *Plug-in*. Esta possui dois grupos distintos de métodos: de controlo do *Plug-in* e de consulta às estruturas de dados geradas pelo gestor de dados. Os primeiros são invocados pelo Gestor e Sincronizador da Simulação de forma a iniciar, parar e actualizar a execução do ciclo de actualização do *Plug-in*. Já os métodos de consulta são solicitados pelo *Plug-in*. Estes transferem-lhe a capacidade de consulta aos dados, consoante a sua necessidade. Por exemplo, pode fazer consultas como “Quais os componentes do tipo 1 que se encontram no cenário?”, “Em que zona se encontra o objecto com o id igual a tal?”, “Quais os componentes da zona x” ou até mesmo “Quais as zonas vizinhas da zona x?”. A resposta à última pergunta é baseada nas ligações dos conectores existentes nessa zona.

A camada **Interface de Controlo da Simulação** é responsável por informar o início e fim de um determinado *Plug-in*, mas também providenciar a lista de nomes de *Plug-ins* disponíveis ao módulo de editor de cenário.

4.3.3 Módulo *Plug-in*

Este módulo é responsável por realizar a simulação propriamente dita e por recolher dados resultantes da mesma. Cada *Plug-in* assume as suas próprias regras, define, de forma semântica, os tipos de dados estruturados pelo módulo controlador de simulação e decide se os quer usar e de que forma. Por exemplo, pode definir o que é que são paredes e o valor da sua existência (poderem ser atravessadas ou não). Aliado a isto, possui autonomia de decisão em relação aos agentes que incorpora. Estas características permitem criar vários *Plug-ins* que realizem simulações variadas como simulação de automóveis e de bandos de pássaros, podendo aproveitar os mecanismos de sincronização do módulo controlador de simulação e funcionalidades dos módulos Visualizador 3D e Analisador de Dados.

No caso específico deste protótipo, foi desenvolvido um *Plug-in* de simulação pedonal em interfaces multi-modais. Este assume dois tipos de delimitadores de movimento, paredes e passeios, com estruturas rígidas (linhas que não podem ser atravessadas) e flexíveis (linhas de guia ao movimento, podendo ser atravessadas), respectivamente. Pressupõe dois tipos de controladores de fluxo, de Entrada/Saída como zonas parametrizadas de entrada de peões e conectores do tipo Porta como pontos de ligação de zonas. Por último, disponibiliza Pontos de Interesse com tipos (multibanco, bar e bilheteira) e parametrizações diferentes. A Figura 4.13 ilustra a arquitectura lógica, dividida em camadas, deste módulo, cujas características serão explicadas no decorrer desta subsecção.



Figura 4.13 – Arquitectura em camadas do módulo *Plug-in*.

A camada **Interface de Plug-In**, tal como foi explicado anteriormente, permite realizar uma interligação com o módulo controlador de simulação a fim de ser realizado o sincronismo de simulação e consulta aos dados armazenados (estrutura principal e auxiliares).

A camada **Gestor do Estado da Simulação** assume um papel fundamental para o funcionamento deste módulo, pois detém as funções padrão (*open()*, *update()* e *redraw()*) que todos os *Plug-ins* necessitam para serem executados pelo módulo controlador de simulação. Este gestor tem como principal objectivo actualizar o estado da simulação em cada unidade de tempo de simulação, desde o estado interno cenário até o estado externo (posição) dos vários agentes. Sempre que recebido um sinal (por parte da interface de *Plug-in*) de início de uma unidade de tempo, esta camada executa uma série acções cruciais para se realizar a simulação pretendida. Começa por consultar (através do gestor de dados do cenário) todos os controladores de fluxo do tipo Entrada/Saída de forma a perceber a sua disponibilidade (dependendo da periodicidade, tempo de disponibilização do objecto e tempo total decorrido) e a necessidade de inserção de agentes na simulação (dependendo da taxa e da área). Se houver necessidade de inserção de agentes, estes são criados (dentro da área do objecto de Entrada/Saída) e guardados numa lista (pelo gestor de registo). De seguida, percorre a lista de agentes. Se os agentes tiverem informado que atingiram o seu objectivo final, estes são eliminados. Caso contrário, estes são forçados a executarem o seu ciclo de execução, sendo validadas as acções tomadas e actualizado o estado do cenário de acordo com a aceitação dessas acções. Findadas as actualizações estas são redesenhadas no módulo de Visualização 3D, registadas no Módulo Analisador de Dados e o módulo Controlador de Simulação é avisado do término do ciclo.

A camada **Gestor de Registo de Agentes** é responsável por registar e apagar, a pedido do gestor do estado da simulação, os agentes numa base de dados. Esta camada, também é responsável por devolver aos agentes, quando requisitado, a lista de agentes nas redondezas/vizinhança. No protótipo desenvolvido, este método é baseado em consultas de vizinhança (*neighbourhood queries*)[Rey00][Buc05], como referido na subsecção 3.5.1. Este método baseia-se numa base de dados espacial que armazena os objectos, sendo cada um associado a um ponto tridimensional. Estes pontos servem como chave de pesquisa para o objecto associado, possibilitando a consulta de forma mais rápida que uma pesquisa a todos os pontos conhecidos. Podem ser feitas perguntas como: “Quais os objectos que estão dentro de um raio R na localização L?”.

A base de dados é construída a partir de um espaço tridimensional em forma de paralelepípedo (caixa global) e tentando abranger toda a área de simulação. Por sua vez, essa caixa é subdividida em caixas mais pequenas (sub-caixas) e todas das mesmas dimensões. Essas sub-caixas são representadas por um ponteiro a uma lista de objectos duplamente ligados. Em termos práticos, é criada uma base de dados com uma caixa de dimensões e subdivisões pré-definidas. Quando um agente é criado, este é alocado na sub-caixa que integra a sua posição e sempre que este se move é efectuada uma actualização da base de dados. Este procedimento verifica se o agente troca de sub-caixa e, em caso positivo, todos os agentes, das caixas em questão, são devidamente avisados (isto é possível através da ligação dupla). A Figura 4.14 exemplifica essa divisão espacial e o mecanismo de detecção de possíveis vizinhos que possam causar uma colisão.

Desenvolvimento do Protótipo

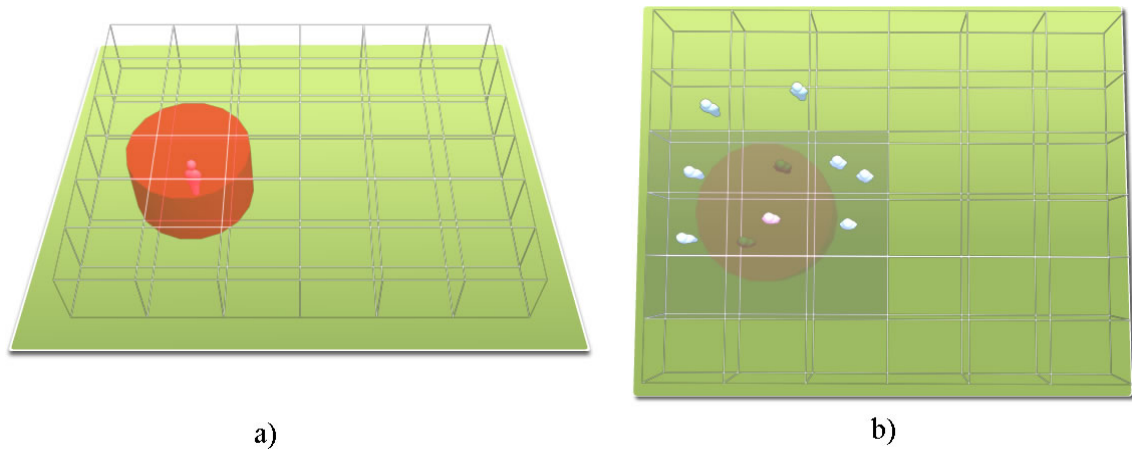


Figura 4.14 – Particionamento de espaço e cálculo de vizinhos.

Quando um agente, durante o seu processo de actualização, pretende consultar os seus vizinhos, requisita essa pesquisa à camada Gestor de Registo de Agentes, informando-a da sua localização (sub-caixa) e do raio em que pretende efectuar a pesquisa. Esta, por sua vez, verifica as sub-caixas que estão dentro do raio (total ou parcialmente) e retorna todos os agentes alocados nessas caixas (Figura 4.14.b)).

A camada **Gestor de Dados** providencia uma análise semântica aos objectos compositores do cenário. Essa análise é efectuada de acordo com a chave de pesquisa proveniente dos agentes, podendo ser processada através de uma tabela de relação (“nome”→tipo) e, forçosamente, através das estruturas de dados do controlador de simulação. A diferença de processamentos depende da chave introduzida. Por exemplo, quando o agente envia um pedido para obter informação relativa à cadeia de caracteres “ATM”, o gestor verifica na tabela de relação se existe alguma chave com essa palavra. Se existir, é porque se trata de um tipo de objectos conhecidos. Consequentemente, o valor retornado é a lista resultante da pesquisa por tipos de objectos à tabela de referência adequada (que se encontra no módulo controlador de simulação). O valor retornado pode, no entanto, ser uma lista vazia. Um outro tipo de processamento é efectuado quando, por exemplo, a cadeia de entrada é “bilheteira_internacional BOX_OFFICE”. Neste caso, o gestor de dados cria duas chaves de pesquisa: uma referente ao tipo de componente “BOX_OFFICE” e outra relativa à instância de um objecto específico “bilheteira_internacional”. O tratamento da primeira chave é, em tudo, semelhante ao anterior. Já a segunda é utilizada para identificar o objecto na lista resultante da primeira consulta, sendo o retorno uma referência ou um valor nulo. Estes processamentos possibilitam o respeitar das regras, seguindo as especificações referidas nas subsecções 3.3.3 e 3.6.2, na medida em que podem ser construídas para um determinado objecto ou para um tipo de objectos. No protótipo implementado, a tabela de referência, interna a este módulo, é fixa e assume os valores representados na Tabela 4.1.

Tabela 4.1 – Tabela de relação “Nome”→Tipo.

Nome	Tipo
“ATM”	1
“BOX_OFFICE”	2
“BAR”	3
“DRAIN_SOURCE”	4

Tal como ilustrado na Tabela 4.1, existem 4 tipos de componentes de cenário (“ATM”, “BOX_OFFICE”, “BAR” e “DRAIN_SOURCE”) que o Gestor de Dados pode tratar através da tabela. Os restantes (delimitadores de movimento: passeios e paredes; e portas) são acedidos através da estrutura de hierarquias do cenário (subsecção 4.3.2, Figura 4.10) e a sua requisição não depende do nome ou tipo de objecto, mas sim da zona pretendida.

A interligação deste módulo, com os agentes, é feita através da camada **Interface de Agente**. Esta processa os pedidos de acordo com as necessidades dos agentes e é responsável por lhes sinalizar o início de uma nova unidade de tempo, transmitindo o valor da mesma. Os agentes actualizam o seu estado com base nesse valor e, terminado o seu ciclo de execução, é obtida uma resposta do tipo acção (posteriormente enviada ao gestor do estado da simulação).

Este módulo possui mais duas camadas de interface, que servem de interligação com os módulos visualizador 3D e analisador de dados. A primeira, **Interface de Visualização**, habilita a chamada de um conjunto de métodos, permitindo o desenho da simulação (em cada unidade de tempo) num espaço tridimensional. A camada **Interface de Análise de Dados** permite enviar os dados recolhidos da simulação (posições dos agentes e número de utilização dos componentes) de forma a serem tratados e visualizados através de gráficos e estatísticas.

4.3.4 Módulo Agentes Autónomos

Um factor fundamental para a existência de interacção com o cenário de simulação é a inserção de actores/agentes. Estes devem ser capazes de se movimentar de forma autónoma, interagindo entre eles e tentando respeitar as regras e parametrizações inseridas. No protótipo desenvolvido, foi implementado um módulo interno baseado numa arquitectura de agentes, denominado **Agentes Autónomos**. Os agentes são situados, incorporados, reactivos e com autonomia de locomoção, como descrito em [Rey99][Buc05]. Estes possuem um sistema de direcção baseado em forças sociais e uma certa habilidade de construção de decisões. Essa construção é baseada nas regras estipuladas, cujo resultado é revertido na aplicação de forças direccionadas para o ponto escolhido, nomeadamente pontos de interesse e componentes de Entrada/Saída (objectivo final). O percurso realizado é não só o resultado dessa aplicação, mas também da aplicação de forças repulsivas, derivadas dos outros agentes e obstáculos. Esta subsecção detalha a arquitectura lógica utilizada neste módulo (ilustrada na Figura 4.15), bem como os aspectos mais relevantes da sua implementação.



Figura 4.15 – Camadas lógicas do módulo Agentes Autónomos.

Para além da camada **Interface de Agente**, que, como referido anteriormente, permite a interligação com o módulo *Plug-in* e o acesso a um conjunto de métodos de consulta, existem mais duas camadas lógicas essenciais para a movimentação e decisão do agente.

A camada **Motor de Comportamentos Direccionais** permite o acesso a um conjunto de comportamentos direccionais e de métodos de cálculo vectorial, que auxiliam a elaboração desses comportamentos. Estão incorporados, nesta camada, comportamentos distintos que os agentes podem utilizar (de forma isolada ou combinada), a fim de realizarem algumas trajectórias análogas ao comportamento humano. A Tabela 4.2 apresenta os comportamentos disponíveis.

Tabela 4.2 – Tabela descritiva dos comportamentos direccionais.

Tabela de Comportamentos Direccionais	
Comportamento	Breve Descrição
"mover para alcançar o alvo"	Direcciona o objecto móvel, de forma radial, para o alvo
"mover para evitar colisões"	Permite a separação dos objectos móveis, quando a distância de conforto é violada.
"mover para evitar paredes"	Força o objecto móvel a executar um movimento paralelo à linha que define a parede.
"mover para seguir passeios"	Direcciona o objecto móvel, levando-o a executar um movimento orientado com a linha de passeio.
"mover para vagar"	Permite a realização de trajectórias aleatórias.

Como pode ser observado na Tabela 4.2, existem cinco comportamentos distintos. A aplicação destes resulta, tal como referido na subsecção 3.6.1, numa força direccional que afecta o ponto de massa de um objecto e, consequentemente, a sua orientação. Para uma melhor percepção do funcionamento dos movimentos direccionais, é importante compreender o modo como são feitas as actualizações da posição e orientação. A Figura 4.16 ilustra, tanto as características básicas de um objecto móvel simples, como o procedimento de actualização das mesmas.

Objecto Movel: <table> <tr> <td>massa</td><td>escalar</td></tr> <tr> <td>posicao</td><td>Vec3D</td></tr> <tr> <td>raio</td><td>escalar</td></tr> <tr> <td>velocidade</td><td>Vec3D</td></tr> <tr> <td>forca_maxima</td><td>escalar</td></tr> <tr> <td>velocidade_maxima</td><td>escalar</td></tr> <tr> <td>orientacao</td><td>EspacoLocal</td></tr> </table>	massa	escalar	posicao	Vec3D	raio	escalar	velocidade	Vec3D	forca_maxima	escalar	velocidade_maxima	escalar	orientacao	EspacoLocal	<pre> applyMovingForce (force, elapsedTime) { Vec3D newAcceleration = (adjustedForce / mass()); velocity += newAcceleration * elapsedTime velocity = truncate(velocity, maxSpeed); setSpeed(velocity.length()); setPosition (position + (velocity * elapsedTime)); regenerateLocalSpace (velocity); } </pre>
massa	escalar														
posicao	Vec3D														
raio	escalar														
velocidade	Vec3D														
forca_maxima	escalar														
velocidade_maxima	escalar														
orientacao	EspacoLocal														

Figura 4.16 – Propriedades do objecto móvel e mecanismo de aplicação de forças.

Cada objecto móvel simples é definido pela massa, posição, velocidade, força máxima, velocidade máxima, raio e orientação (espaço local - sistema ortogonal, representando frente, lado, cima). Calculada a força direccional, é possível obter um valor de aceleração, utilizando a primeira lei de Newton e, com este novo valor, actualizar o vector da velocidade. A posição do objecto pode ser agora actualizada, incrementando, à posição antiga, o novo vector de velocidade truncado com a velocidade máxima. Como descrito anteriormente, cada objecto possui um sistema ortogonal local que necessita de ser actualizado a cada unidade de tempo de simulação. A posição do objecto deve estar sempre alinhada com a sua velocidade, pelo que o vector “frente” do objecto é actualizado, tornando-se igual ao vector velocidade normalizado. Por sua vez, o vector lado é calculado através do produto externo entre os vectores “novo frente” e “cima” (no caso deste protótipo, foi considerado que a rotação é apenas realizada sobre o vector “cima”, pelo que neste caso é constante). Utilizando o espaço local e o espaço de inércia referido na subsecção 3.6.1 é possível calcular o ângulo de rotação.

O comportamento direccional **“mover para alcançar o alvo”** baseia-se, sobretudo, na posição/ponto central do objecto alvo. Este retorna uma força que direcciona o objecto para esse ponto. A Figura 4.17 ilustra esse comportamento e o seu funcionamento.

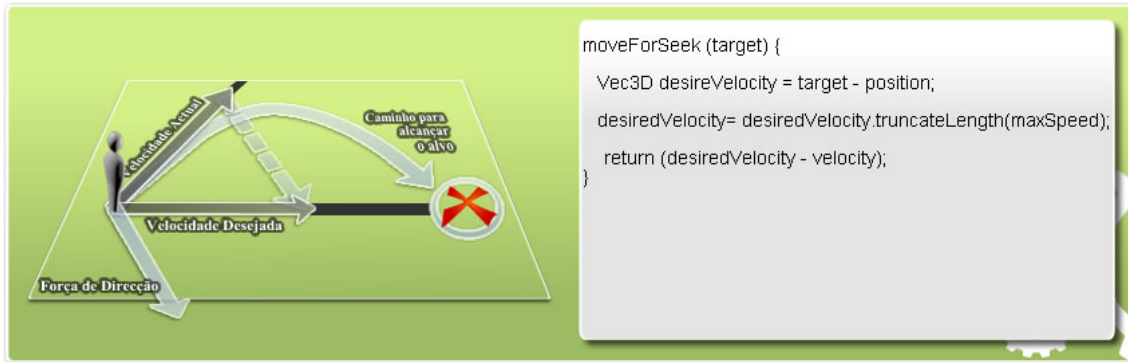


Figura 4.17 – Comportamento direccional “mover para alcançar alvo”.

Como é possível verificar, é um método bastante simples. Em primeiro lugar é calculada a velocidade desejada. Esta representa a velocidade necessária para o corpo atingir o alvo, que por vez é truncada com o valor da velocidade máxima possível. A força direccional retornada é a força necessária que, quando adicionada à velocidade actual do objecto, transmite a velocidade desejada. Para conseguir isso, basta subtrair à velocidade desejada o vector da velocidade actual. Estes procedimentos são suficientes para o objecto móvel alcançar o alvo pretendido, contudo provocam que o objecto o atinja à velocidade máxima. Caso seja pretendido um abrandamento da velocidade, no aproximar do alvo, é necessário definir um raio de abrandamento. Fora deste raio, a velocidade desejada é truncada pela velocidade máxima. Dentro do raio, a velocidade desejada é decrementada progressivamente (até zero), com base na distância ao alvo e consequente desaceleração.

O comportamento **“mover para evitar colisões”** baseia-se no “mover para evitar uma potencial colisão” apresentado na subsecção 3.6.1, porém possui um processo simplificado. Essa simplificação reflecte-se na redução dos cálculos efectuados, resultando num comportamento semelhante, somente no seu caso mais base (quando a potencial colisão mais próxima ocorre entre os raios dos objectos). A Figura 4.18 exemplifica este método.

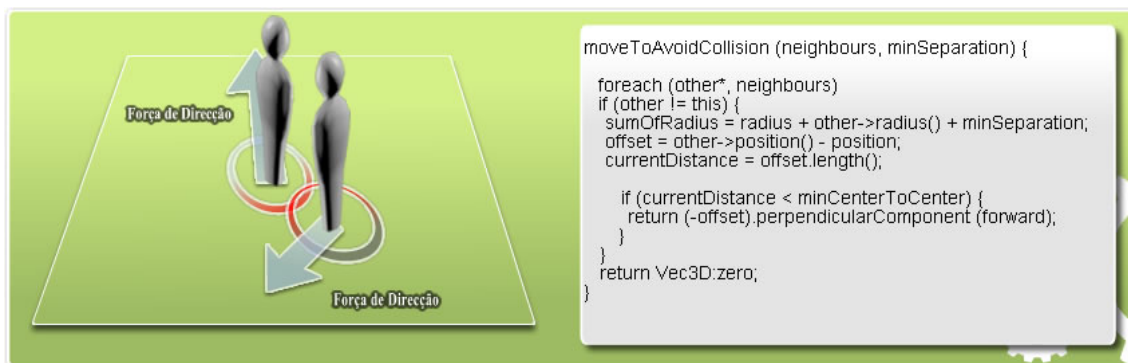


Figura 4.18 – Comportamento direccional “mover evitar colisões”.

O princípio base deste método está relacionado com a soma dos raios (e um possível acréscimo de distância de segurança). Em primeiro lugar é calculado um vector da posição objecto a um vizinho. Se o comprimento deste vector for menor que a soma dos raios (mais o possível acréscimo), é produzida uma força correspondente à componente perpendicular do inverso desse vector com o vector “frente”. A performance de execução deste método é, em muito, condicionada pelo cálculo dos vizinhos, pelo que o método utilizado (baseado em consultas de vizinhança implementada – “*neighbourhood queries*”) ajuda a otimizar este processo.

O comportamento **“mover para evitar paredes”**, como descrito na subsecção 3.6.1, possibilita um objecto móvel detectar a presença de uma parede (através da projecção do seu vector frente) e contorná-la através de forças direccionais. Estas são resultantes da amplitude penetração na parede e possuem uma direcção igual à normal da parede detectada. No protótipo

desenvolvido, este método foi implementado seguindo uma abordagem um pouco diferente, tal como demonstra a Figura 4.19.

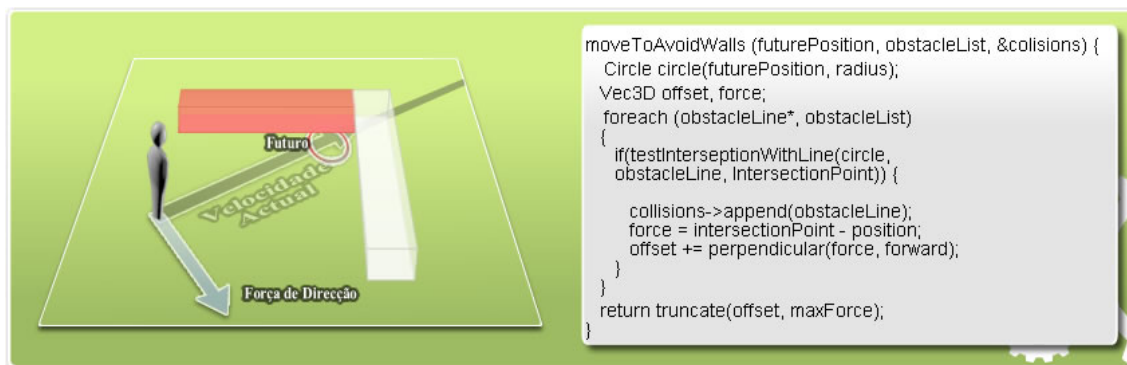


Figura 4.19 – Comportamento direccional “mover para evitar paredes”.

A principal diferença nesta abordagem é a projecção de uma circunferência com raio igual ao do objecto móvel, em vez de uma projecção do vector “frente” ou velocidade actual. Esta circunferência representa uma possível e futura área de ocupação. Neste caso, e se for detectada uma colisão entre um segmento de recta (parede) e a circunferência, é calculado um vector do ponto de intersecção ao posição do objecto. A força direccional produzida é a componente perpendicular desse vector com o vector “frente” truncada com a força máxima. A vantagem de se projectar uma circunferência, com o raio do objecto móvel, reside na possibilidade de detecção de colisões entre duas paredes em simultâneo. Isto pode significar que é um canto ou uma passagem demasiado estreita (por onde o objecto móvel não deve conseguir passar). A força de direcção resultante, deste tipo de situações, é calculada através do somatório das forças de repulsão de cada parede. Este método é simples de implementar, contudo não se torna suficiente para contornar um conjunto de paredes com disposição bastante sinuosa (contendo vários cantos). Este tipo de procedimentos (resolução de cantos) deve ocorrer numa camada de mais alto nível (método posteriormente explicado), porém, este método, assume-se como um importante apoio a essa resolução. Isto porque envia uma lista das rectas, em que se detectou colisão, à camada Provedor de Acções, junto da informação de possibilidade de ser um canto.

Na Tabela 4.2 pode-se observar a presença de dois comportamentos adicionais aos expostos na subsecção 3.6.1. Designadamente, **“mover para vaguear”** e **“mover para seguir passeios”**. Este último é uma simplificação do comportamento “mover para evitar paredes”, com as diferenças de não tratar de colisões de múltiplas linhas em simultâneo e a força resultante ser truncada com a velocidade máxima. Assim, é possível garantir um percurso guiado, mas não totalmente restrito, uma vez que os objectos móveis conseguem ultrapassar estas linhas. O comportamento **“mover para vaguear”** surge na possibilidade de acréscimo de mais uma alternativa em situações de espera em demasia ou falta de objectivos. Por exemplo, se um agente desejar dirigir-se a um componente de Entrada/Saída (parametrizado como periódico, com intermitência de 15 minutos e período de disponibilização de 5 minutos), é verificado o estado interno do objecto. Se estiver indisponível, o agente pode esperar (não aplicação de forças direccionais) ou então, e se faltar muito tempo para a sua disponibilização, pode tomar a acção “mover para vaguear”. A força resultante deste método é calculada através da geração aleatória dos valores “1” e “-1” multiplicados pelo vector “lado”.

Na maioria das vezes, ou quase sempre, o uso destes comportamentos é realizado de forma combinada. Neste caso, é importante ter em atenção que um objecto móvel possui uma força de direcção máxima, pelo que o somatório da combinação das forças deve ser truncado com esse valor. Esta questão só é relevante, obviamente, quando a combinação é feita através do somatório. Se se optar por uma combinação por prioridades, é importante definir quais os comportamentos mais prioritários, não existindo no entanto, nenhuma norma fixa para atribuição das mesmas. No protótipo implementado foi abordada uma combinação por prioridades com a seguinte ordem: “mover para evitar paredes”; “mover para evitar colisões” e “mover para seguir passeios”.

A camada **Provedor de Acções** habilita, ao agente, alguma capacidade de decisão sobre quais as acções a tomar. Basicamente, permite a escolha dos pontos a que se sente atraído, expressando assim, a sua motivação. Todo o comportamento operacional desta camada, gravita em torno de uma pilha, representativa do plano traçado pelo agente, que em termos gerais, é uma sequência de pontos (de atracção), pelos quais o agente terá de passar, a fim de atingir o seu objectivo final. Esta pilha, em analogia com a subsecção 3.6.2, pode ser representativa da pilha de desejos presente no paradigma BDI (alicerce da estrutura do mesmo). A Figura 4.20 ilustra um possível estado da pilha e uma situação de actualização da mesma.



Figura 4.20 – Relação entre o comportamento “mover para evitar paredes” e a pilha de desejos.

Analisando a composição da pilha (parte direita da Figura 4.20), pode supor-se que o agente expressa um desejo final de apanhar o “comboio_urbano” (este desejo é atribuído no momento da criação do agente e inserido de imediato na pilha). Para o realizar, terá que se deslocar a um multibanco e posteriormente à “bilheteira_urbana”. Esta sequência é fruto das regras estipuladas, baseadas na definição de regras por probabilidades (subsecção 3.6.2). Com base neste exemplo (que é um plano possível), é assumido que o agente está, neste momento, a preencher o desejo de utilização de um ponto de interesse do tipo “ATM”. Neste caso e porque se trata de um tipo de componentes, o agente recebe, por parte do módulo *Plug-in*, uma lista de apontadores de objectos (deste tipo). A escolha recai sobre o que se encontrar mais próximo. Até o desejo se encontrar preenchido (utilização do multibanco), é tomada a acção “mover para alcançar alvo”.

Durante o seu percurso para alcançar o ponto desejado, são aplicadas várias forças de acordo com a combinação, por prioridades, dos movimentos direccionais. A parte esquerda da Figura 4.20 ilustra a situação em que o agente se depara com uma situação de colisão iminente com um canto, pelo que será necessário resolvê-lo. Um dos princípios base, da abordagem usada para a sua resolução, é a descodificação de inclusão ou exclusão no plano formado pelos dois vectores dos segmentos de recta (representado na figura pela zona escurecida). É nesta altura que se torna relevante a informação da lista de linhas (retornadas pelo comportamento direccional) que estão a ser intersectadas pela circunferência projectada (representação das paredes). A descodificação inclusão/exclusão é realizada com auxílio a três vectores: dois vectores representativos das duas paredes em questão (do ponto mais próximo da projecção da posição futura ao mais distante), V1 e V2; e um vector do ponto PR (resultante da soma de V1 e V2) à posição actual V3. Caso o segmento de recta (representado por V3) colida com algum dos outros segmentos de recta, o agente está excluído do plano. Caso contrário, está incluído. Em qualquer um dos casos, a força aplicada é a resultante no método “mover para evitar parede”. A grande diferença é que, em caso de inclusão, o agente irá colocar na pilha de desejos, um dos pontos extremos (círculos a azul na Figura 4.20), com base na sua distância ao objectivo que está a cumprir. Resolvido o canto, resta resolver o problema do percurso cíclico que pode ser gerado (ex: localização do ponto de atracção por detrás de um conjunto de paredes dispostas em “U”). Este problema é resolvido com o apoio à lista de pontos que já resolveram cantos. Sempre que um agente insere um **Ponto de Resolução de Cantos** (“PRC X Y”) na pilha de desejos, guarda-o também numa lista de memória de pontos de resolução. Por algum motivo, o agente pode deparar-se com um ponto de resolução de um determinado canto, que se encontra na lista de memória. Neste caso, é assumido que a primeira escolha esteve errada, pelo que agora, tomará a opção contrária.

Sempre que um desejo da pilha é preenchido, o agente elimina-o e tenta preencher o desejo seguinte. No entanto, é preciso ter em atenção que um desejo “PRC X Y” pode nunca ficar preenchido. Isto pode acontecer se, no percurso de resolução de um canto, o agente encontrar outro canto (ponto de resolução é inatingível). Quando isto acontece, é necessário verificar se o desejo actual é um “PRC X Y”, substituindo-se o desejo, em vez de se adicionar um novo.

Quando o objectivo final é atingido, o agente informa o módulo Plug-in, que retira este agente da simulação.

4.4 Ciclo de Execução da simulação

Na secção anterior foram apresentados, todos os módulos que compõem o protótipo ModP. Apesar de terem sido explicados os seus funcionamentos e a sua interligação, pode não ter sido clarificado o ciclo de execução da simulação. Esta secção descreve todo o ciclo de execução desde o início de uma unidade de tempo, até ao seu término. A Figura 4.21 seguinte ilustra os estados decorrentes de todo o ciclo.

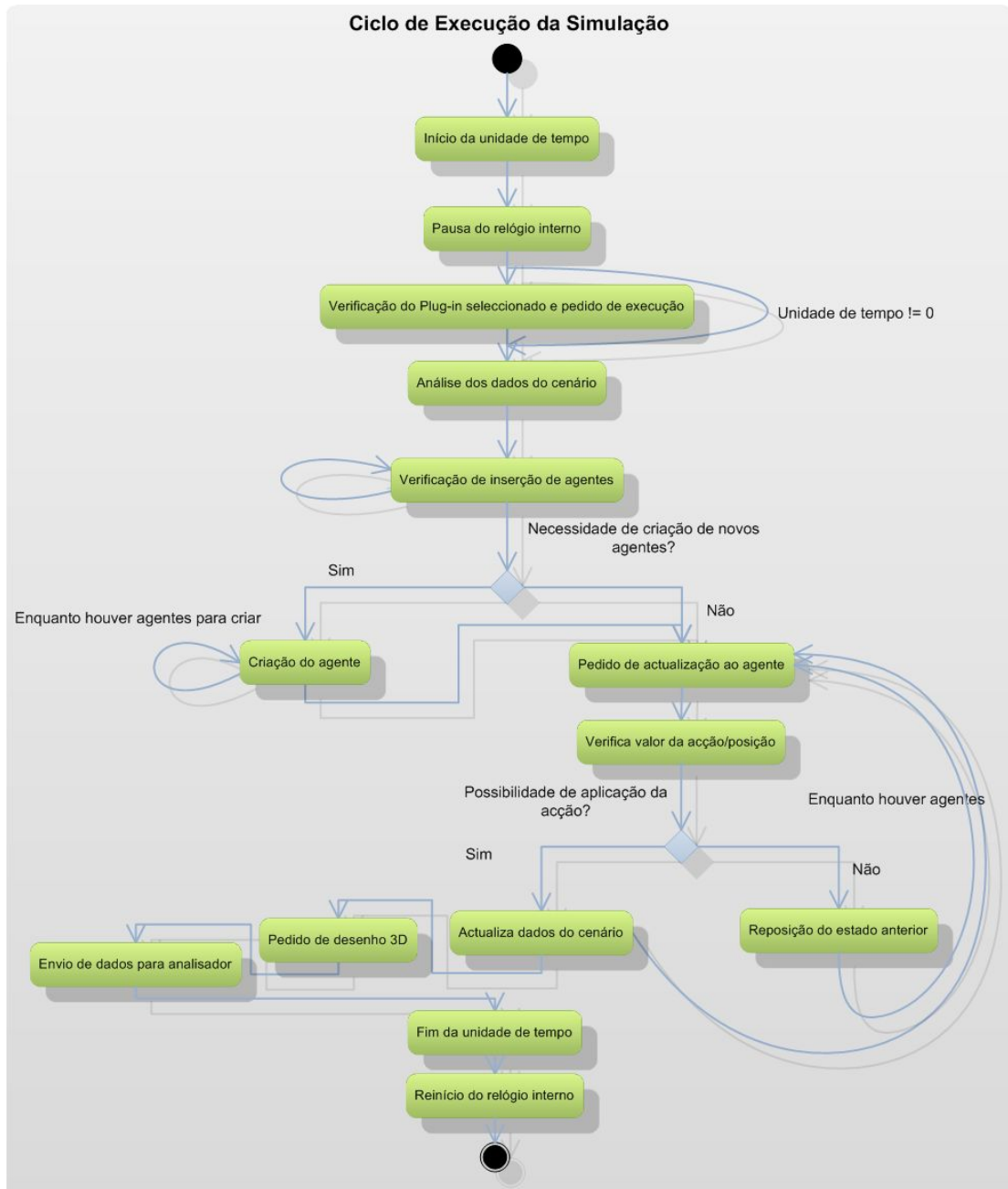


Figura 4.21 – Ciclo de execução da simulação.

Iniciada uma nova unidade de tempo, o relógio interno da simulação pára o seu funcionamento. Este procedimento garante o não início de uma nova unidade de tempo, antes do término desta. Num primeiro instante, se e só se coincidir com o arranque de toda uma simulação (unidade de tempo zero), é verificado se existe algum *Plug-in* que esteja seleccionado para se iniciar o seu arranque. Caso isto não se verifique, é realizado um pedido de execução ao primeiro *Plug-in* da lista. Uma vez ultrapassado este passo, todo o procedimento é igual e sistemático, para todas as unidades de tempo de simulação.

Em cada ciclo, o *Plug-in* seleccionado adquire/requisita os dados do cenário, de forma a validar os seus requisitos de funcionamento. No caso do *Plug-in* desenvolvido, esta análise foca-se, principalmente, no verificar da existência de Controladores de Fluxo de Entrada/Saída. Estes são a fonte de informação sobre a possibilidade de inserção de agentes na simulação. Essa inserção é baseada das propriedades internas de cada um dos objectos de Entrada/Saída. Se estes forem do tipo “contínuo” o processo de inserção é estocástico, pelo que a inserção depende do valor probabilístico inserido. Nos casos de serem “periódicos” ou “únicos”, são verificados os valores mínimos e máximos do número de agentes que devem ou podem ser inseridos, sendo o valor de inserção calculado aleatoriamente, dentro desse intervalo. Nestes dois últimos casos, verifica-se, se a inserção já foi efectuada ou se (no caso de serem “periódicos”) é permitida na unidade de tempo actual (uma vez que só fica disponível em intervalos de unidades de tempo específicas). Se algum desses valores se verificar, inicia-se o processo de inserção dos agentes. Mais uma vez, este depende das características internas desses objectos, mais propriamente da sua geometria. Quando um agente é criado, este é colocado aleatoriamente dentro da área delimitada pelo comprimento e largura do objecto. Estas características, das propriedades dos componentes de Entrada/Saída, permitem não só inserir agentes de um modo coordenado, como em locais específicos, com áreas específicas.

Verificadas as inserções, inicia-se a fase de actualização do estado do cenário. Em primeiro lugar, é requisitado a cada um dos agentes que actualize o seu estado interno e que tome uma acção, quer em relação à definição da sua nova posição, quer em relação à possibilidade de utilização de um ponto de interesse. Qualquer uma destas situações é alvo de avaliação, ou seja, é verificado se a posição pode ser ocupada e se a acção de utilizar um ponto de interesse pode ser tomada. No primeiro caso, é principalmente verificada a existência de alguma parede na posição que o agente quer ocupar. Se tal acontecer, este é recolocado na sua anterior posição. Quando à utilização dos pontos de interesse, a verificação reside na disponibilidade do componente, sendo que tal disponibilidade depende da sua parametrização. No caso de o ponto de interesse se encontrar ocupado, o agente pode requisitar a inscrição na fila de espera, cujo atendimento é sempre positivo. A libertação de um ponto de interesse é também verificada e sempre aceite (assumindo-se que o agente respeitou o tempo médio de utilização).

Findado o ciclo de actualização dos agentes e respectivas validações, efectua-se uma mudança de estado, no sentido de disponibilizar ao utilizador o resultado final de uma unidade de tempo. Este resultado decorre em dois processos distintos. Um primeiro, que visa a disponibilização em 3D de todo o cenário, bem como a localização e orientação de cada agente. Um outro, que disponibiliza, de um modo gráfico, um conjunto de estatísticas sobre a utilização dos diversos componentes e um mapa de densidade sobre a ocupação do cenário.

No final de todo este processo, dá-se por terminada a unidade de tempo e o relógio interno é reinicializado.

4.5 Visualizador 3D e Analisador de Dados

O visualizador 3D e o analisador de dados, são os dois módulos que permitem, ao utilizador, tirar algum partido de todo o processo de simulação, já que são as únicas fontes, de visualização gráfica de dados de simulação, inseridas no protótipo implementado. Na figura seguinte, estão ilustradas as suas arquitecturas lógicas em camadas.

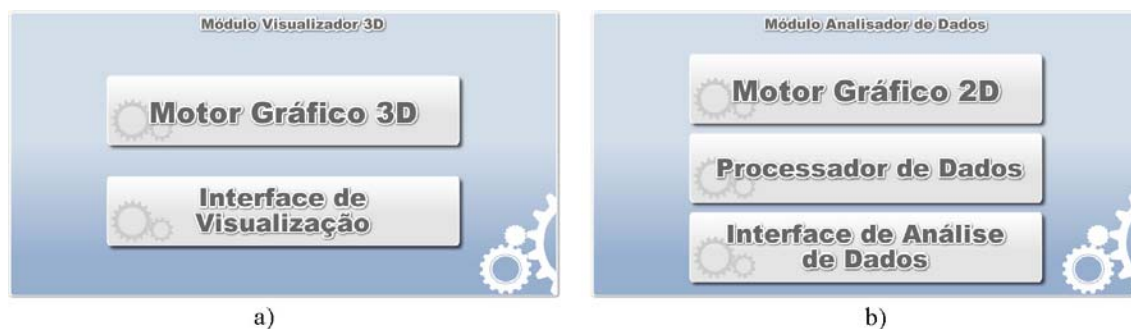


Figura 4.22 – Camadas dos módulos Visualizador 3D e Analisador de Dados.

O módulo **Visualizador 3D** faculta ao módulo *Plug-in*, o uso de um conjunto de métodos de desenho 3D baseados na tecnologia OpenGL[®]. Como se pode observar na Figura 4.22.a), este é constituído por duas camadas:

- **Interface de Visualização** – esta interface permite, como relatado anteriormente (secção 4.3.3), uma interligação com o módulo *Plug-in*, de modo a que este peça a execução de funções como desenhar paredes, desenhar linhas dos passeios, desenhar pontos de interesse, desenhar componentes de Entrada/Saída e desenhar peões.
- **Motor Gráfico 3D** – esta camada é na prática, um conjunto de métodos/funções que possibilitam o desenho de figuras geométricas simples de forma combinada. É de realçar a possibilidade de desenho de modelos complexos, através de um leitor de ficheiros do tipo STL (*StereoLitho* files). Este tipo de ficheiros, utilizados para prototipagem (podem ser gerados a partir do AutoCad[®]), pode acrescentar algum realismo ao cenário. Como possuem uma estrutura bastante simples (planos triangulares e respectivas normais) a sua leitura é facilitada. O grau de complexidade dos modelos, contidos nos ficheiros, pode afectar o tempo de arranque da simulação, contudo não afecta, em demasia o tempo de desenho. Isto porque, os objectos são alocados em memória, na inicialização do simulador, em listas do tipo *GLuint* e desenhados pela chamada de listas (*glCallList*)[Gol09], eliminando a necessidade de recalculo da iluminação/cor a cada ciclo de execução. Uma outra característica, desta camada, é a agregação das chamadas aos métodos da biblioteca disponibilizada pelo OpenGL[®], num contexto distinto. Isto facilita uma eventual mudança de tecnologia.

A interface gráfica, disponibilizada pelo módulo visualizador 3D, é completamente navegável e disponibiliza o desenho do cenário de forma concordante com o editor gráfico do cenário.

O módulo **Analisador de Dados** é responsável por tratar os dados oriundos do *Plug-in* e transformá-los em representações gráficas facilitando a leitura dos resultados obtidos no decorrer da simulação. Como é possível observar na Figura 4.22.b), este módulo é constituído por três camadas:

- **Interface de Análise de Dados** – é responsável por reencaminhar a informação vinda do *Plug-in* para a camada processador de dados de acordo com os tipos de análise pretendidos.
- **Processador de Dados** – é responsável por processar os dados recebidos, quer para criação de estatísticas de ocupação dos componentes, quer para criação do mapa de densidade. As estatísticas de ocupação dos componentes são tratadas em termos do número total de agentes que foram criados. Já o mapa de densidade é construído com base no posicionamento dos peões, em cada unidade de tempo. Sempre que inserido um ponto de posição, é verificada a colisão com os restantes pontos. Se o número de colisões ultrapassar determinados valores (previamente estipulados), a sua cor muda de azul para vermelho, passando pelo laranja.

- **Motor Gráfico 2D** – disponibiliza ao processador de dados a possibilidade de transpor os dados gerados, para o ecrã. Este gera duas janelas de visualização, para as estatísticas dos componentes e para o mapa de densidade.

A geração deste tipo de mapas vem de acordo com o especificado na subsecção 3.5.3, permitindo obter de uma forma rápida e eficaz o panorama geral do cenário de simulação.

4.6 Síntese

No decorrer deste capítulo foram apresentados os diversos módulos integrantes do protótipo para a plataforma ModP, bem como as suas interfaces gráficas e as tecnologias utilizadas na sua implementação (Qt e OpenGL®). A implementação de forma modular, facilita uma futura separação física dos diversos módulos e/ou uma possibilidade de substituição dos mesmos.

Cada um dos diversos módulos reúne um conjunto de características distintas que harmonizam quer o processo de utilização, por parte do utilizador final, quer o processo de agregação de novas funcionalidades, por parte do programador.

O módulo Editor de Cenário é caracterizado pelo recurso a operações simples (“clique, arrastar e largar” e “pegar e largar”), pelos comportamentos de alto nível (“não replicação de pontos/vértices coincidentes” e “sugestão de criação de zona”) e pela zona de edição contextual (com formulários gerados pelos próprios componentes de acordo com as suas características). Este possui uma de uma interface gráfica, que utiliza os mais recentes padrões de desenho de interfaces, possibilitando uma redução do número de operações necessárias, por parte do utilizador e promovendo uma rápida adaptação ao sistema.

O módulo Controlador da Simulação agrega um sistema de *Plug-ins* permitindo a sua expansibilidade a outro tipo de simulações microscópicas. A estruturação de dados, oriundos do editor gráfico, é realizada de um modo dinâmico e potenciada por estruturas auxiliares, permitindo aos *Plug-ins* efectuarem pesquisas de diversas ordens (por tipos de objectos, por identificadores dos objectos ou pela hierarquia do cenário).

São esses *Plug-ins* que definem as regras e tipos de simulação, bem como os agentes que são inseridos. O *Plug-in* de simulação pedonal em interfaces multi-modais, inserido no protótipo, serve-se de uma tabela de relação “nome do componente”→tipo, para realizar pesquisas nas estruturas do módulo controlador da simulação. Essas pesquisas pretendem dar resposta às consultas de ordem semântica requeridas pelos agentes.

Os agentes, por sua vez, baseiam-se numa pilha de desejos, com elementos de ordem semântica. Essa pilha representa o plano traçado para o agente, e permite-lhes a realização de um determinado percurso. As contrariedades no decorrer do percurso, são resolvidas com recurso a comportamentos direccionais de baixo nível (baseado em forças de atracção e repulsão – de fácil implementação), que podem apoiar o ajuste do plano, de forma a ser possível atingir o objectivo final.

Capítulo 5

Resultados

Nos dois últimos capítulos desta dissertação, foram descritos e especificados vários aspectos considerados importantes, no decorrer da implementação de uma ferramenta de simulação microscópica, baseada em agentes autónomos, para interfaces multi-modais. Esses aspectos evidenciam principalmente, a usabilidade da interface gráfica de edição, o realismo da simulação, a afectação da parametrização dos diversos parâmetros e o tipo de resultados obtidos. Ao longo deste capítulo estão descritos alguns dos resultados obtidos no desenrolar da implementação do protótipo.

A secção 5.1 demonstra o tipo de cenários que podem ser criados através da interface gráfica e sua afectação directa, aliada a um conjunto de parâmetros, nos resultados da simulação. Encontram-se também destacados, alguns pormenores dos comportamentos direccionais, que de uma forma ou de outra, são a justificação de tais resultados.

Na secção 5.2, estão descritos os resultados obtidos através da simulação, num mesmo cenário, com parametrizações distintas.

A última secção, 5.3, apresenta uma breve síntese sobre todo este capítulo.

5.1 Cenários, Parametrizações e Simulação

Nem todas as propriedades, quer dos objectos de cenário quer dos agentes, podem ser parametrizáveis através da interface gráfica. Contudo, é importante realçar, também, algumas dessas propriedades e a sua importância directa no realismo da simulação e no comportamento dos agentes.

A construção do cenário pressupõe o desenho de um conjunto de segmentos de recta (linhas duplas para paredes e simples para passeios), como a incorporação de outros e vários componentes (Entrada/Saída, Multibanco, Bar e Bilheteira). A Figura 5.1 ilustra um leque de cenários que podem ser criados e a sua representação no mundo virtual 3D.

Resultados

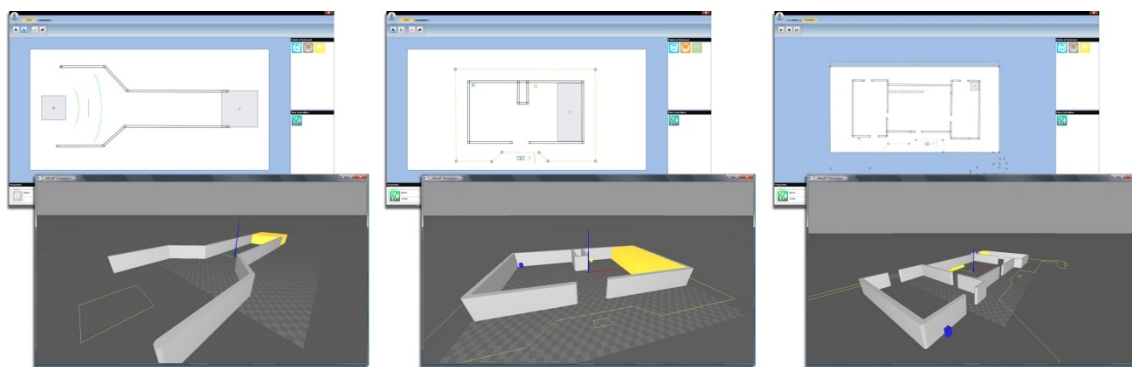


Figura 5.1 – Exemplo da variedade de cenários editados.

Como é possível observar na figura, existe uma variedade de cenários que pode ser criada através do editor gráfico, sendo fielmente representados num mundo virtual e tridimensional. O seu grau de detalhe depende, única e exclusivamente, da vontade do utilizador. Uma outra observação que pode ser realizada, é a não presença de peões. Como referido nas subsecções 3.3.2 e 4.3.1, os componentes Entrada/Saída podem ser parametrizáveis como “contínuos”, “periódicos” ou “únicos” e dependendo das suas propriedades de entrada de peões, esta pode ou não ser realizada. Uma não colocação destes componentes ou parametrização de entrada “zero” permite ao utilizador, após pedido de início de simulação, a visualização e navegação por todo o cenário de forma a confirmar a sua composição e disposição.

Quando se procede à parametrização dos componentes de Entrada/Saída, deve-se ter em atenção o tipo de objecto do mundo real que se pretende simular. Por exemplo, se pretendemos simular um ponto de táxis, podemos estipular as dimensões “280 x 160 cm”, periodicidade “periódico”, frequência “5 minutos” e entrada de peões “de 0 a 4 pessoas”. Já se pretendemos simular uma entrada de metro, podemos estipular uma área “500 x 500 cm”, periodicidade “periódico”, frequência “15 minutos” e entrada de peões “5 a 50 pessoas”. No caso da estação de metro, podia-se optar, por exemplo, por periodicidade “contínuo”, com taxa de entrada “10%”, assumindo que existe essa probabilidade a cada instante/unidade virtual de tempo.

Para além deste tipo de parametrizações, existem algumas variáveis que influenciam directamente a simulação. Estas prendem-se com as propriedades dos agentes, afectando as forças resultantes dos movimentos direccionais. A Figura 5.2 demonstra a influência directa do valor do raio, na distância que estes tentam garantir em relação aos outros (podendo ser designada por área de conforto).

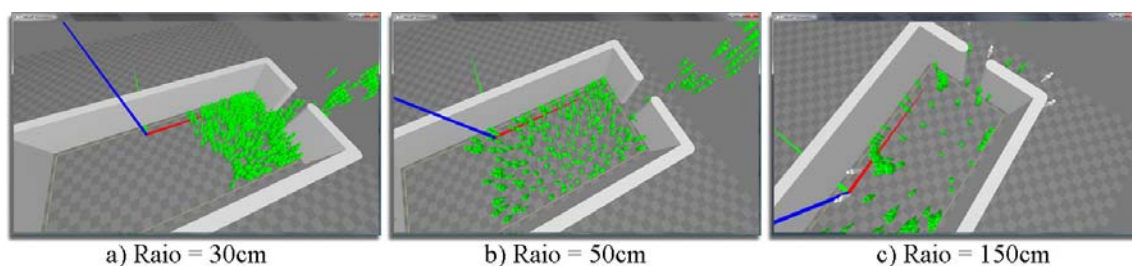


Figura 5.2 – Afectação directa do valor do raio na simulação.

Como pode ser observado na figura, a influência de parametrização do raio é bastante evidente. A decisão do valor correcto, a usar na simulação, depende do tipo de análise que se pretende realizar. Se a preocupação for otimizar o conforto dos peões, provavelmente, o valor mais correcto rondará os 50 cm, como demonstrado na Figura 5.2.b). Por seu lado, se o objectivo for otimizar o espaço, com vista a uma rápida evacuação, este valor pode ser reduzido, por exemplo para 30 cm (Figura 5.2.a)). Valores muito pequenos ou muito grandes podem presentear a simulação com características irrealistas, tal como é possível verificar na Figura 5.2.c). Nas simulações que deram origem às imagens, foi parametrizado um componente de Entrada/Saída com dimensões iguais às da sala (3000 x 1300 cm), periodicidade “único” e valores mínimos e máximos de entrada de 500 pessoas. Este tipo de parametrização é bastante

útil, quer para a realização deste tipo de testes, quer para uma eventual simulação de evacuação em condições de emergência.

Um outro factor importante para a simulação, tal como relatado na subsecção 4.3.4, é a previsão de colisões com as paredes e resolução de cantos. Um dos factores, que mais influencia o comportamento “mover para evitar paredes” e a resolução de cantos, é a distância à qual é projectada a circunferência, de raio igual ao do agente, representativa da posição futura. A Figura 5.3 ilustra duas situações distintas, onde a afectação da distância de projecção da posição futura é notória.

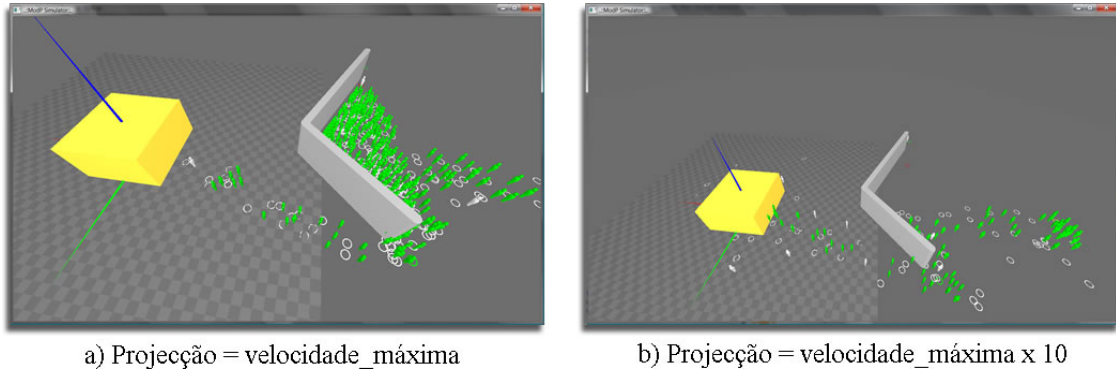


Figura 5.3 – Afectação da projecção da posição futura na resolução de cantos.

Como é possível observar, quanto mais distante é a projecção da circunferência representativa da posição futura, mais cedo os agentes se apercebem das paredes e consequentemente, dos cantos. Essa projecção pode ser calculada com base na velocidade máxima de cada objecto móvel. Nos casos específicos, representados pelas Figura 5.3.a) e Figura 5.3.b), os valores das projecções variam entre 1.2 e 1.6 metros e 12 e 16 metros, respectivamente (depende do valor atribuído à velocidade máxima, que varia entre os primeiros valor em m/s). O valor exacto para essa projecção deve ser ponderado, já que, o que se pode ganhar com a antecipação da detecção, pode-se perder com o atingir do objectivo. Isto pode acontecer quando o objectivo está perto de um canto. A melhor opção a tomar, será a de realizar essa verificação antes de o agente tentar aplicar a força direccional.

5.2 Dados obtidos através da simulação

Como foi descrito anteriormente existem inúmeros factores que podem afectar uma simulação, pelo que a escolha da parametrização deve ser ponderada. Contudo e assumindo que esses valores possuem a parametrização correcta, existem outras condicionantes que afectam toda a simulação. Desde a disposição das paredes, pontos de interesse e componentes de Entrada/Saída, até aos valores internos de cada um.

Para demonstração dos resultados obtidos através da simulação, foram realizadas três simulações distintas de vinte minutos cada. A Figura 5.4 ilustra a composição do cenário de simulação.

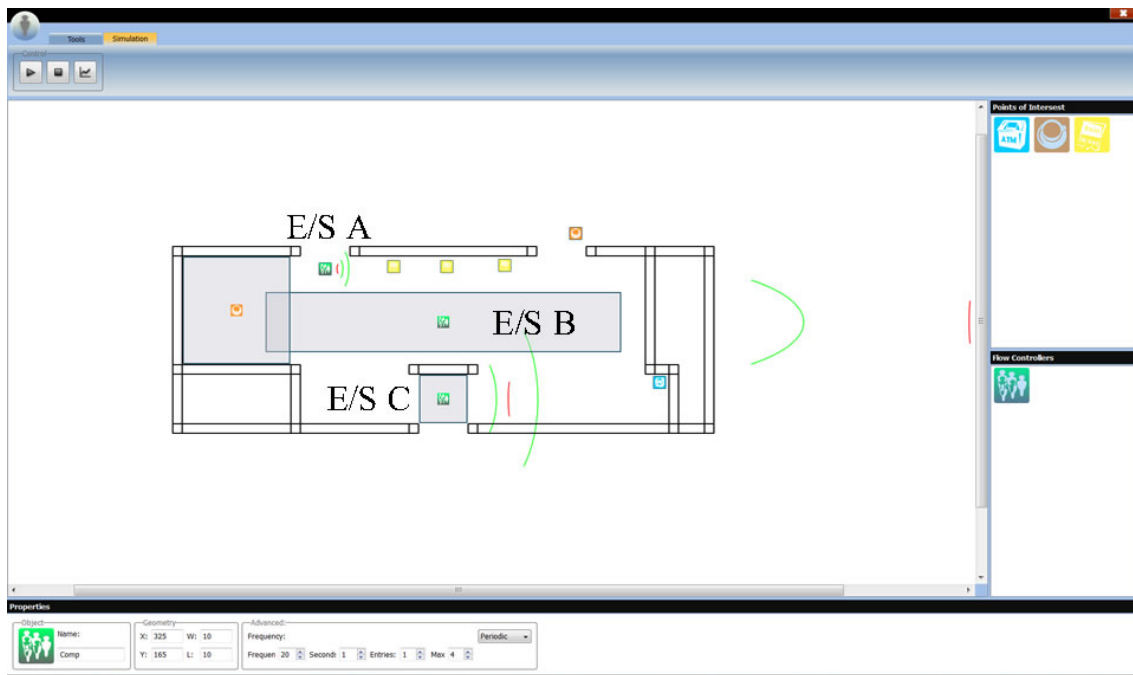


Figura 5.4 – Cenário de simulação.

A diferença, entre as simulações, baseia-se sobretudo na taxa de entrada de agentes. Uma primeira com baixo fluxo, seguida de uma de médio fluxo e, por fim, com um fluxo elevado. Quanto ao seu comportamento/percurso, estes possuem uma pilha inicial pré-definida e que lhes atribui um plano estocástico, baseado nas probabilidades de utilização de pontos de interesse (multibanco, 20%; bilheteiras, 50%; e bar, 30 % - por esta ordem). Sempre que os agentes chegam ao bar, são eliminados (é o seu objectivo final). Os pontos de interesse, representativos dos multibancos e bilheteiras, possuem uma taxa de utilização média de cem unidades de tempo virtual. Cada unidade de tempo de simulação corresponde a 0,1 segundos.

O cenário de simulação de baixo fluxo apresenta como taxas de entrada de agentes, a seguinte configuração:

- **E/S A** – “Periódico”, com frequência de 60 segundos, número mínimo de pessoas 1 e máximo 4;
- **E/S B** – “Único”, com número fixo de 10 pessoas;
- **E/S C** – “Contínuo”, com taxa de entrada de pessoas de 1%.

A Figura 5.5 representa o gráfico de estatísticas de utilização dos vários componentes, bem como a taxa de peões que se encontram no cenário em relação à população global gerada (linha verde). A azul está representada a taxa de utilização de multibancos, a amarelo, a taxa das bilheteiras e a laranja, a taxa dos bares.

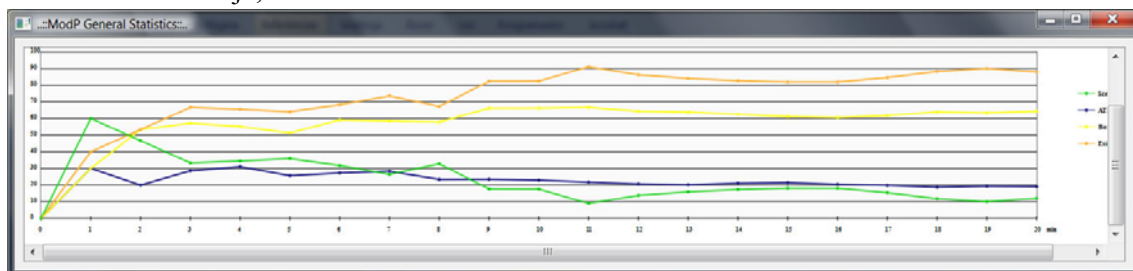


Figura 5.5 – Estatísticas gerais do cenário de simulação com baixo fluxo de pões.

Como é possível observar através do gráfico, ao fim de aproximadamente 10 minutos de simulação, os valores de utilização dos componentes aproximam-se dos valores probabilísticos do plano dos agentes. Isto deve-se à pouca densidade de povoamento do cenário e ao fácil acesso aos vários componentes. A fluidez do percurso pode ser avaliada através da linha verde

Resultados

(ao fim de 20 minutos, só se encontram no cenário aproximadamente 10% da população total gerada), mas também através do mapa de densidade e do Visualizador 3D, representados na Figura 5.6.

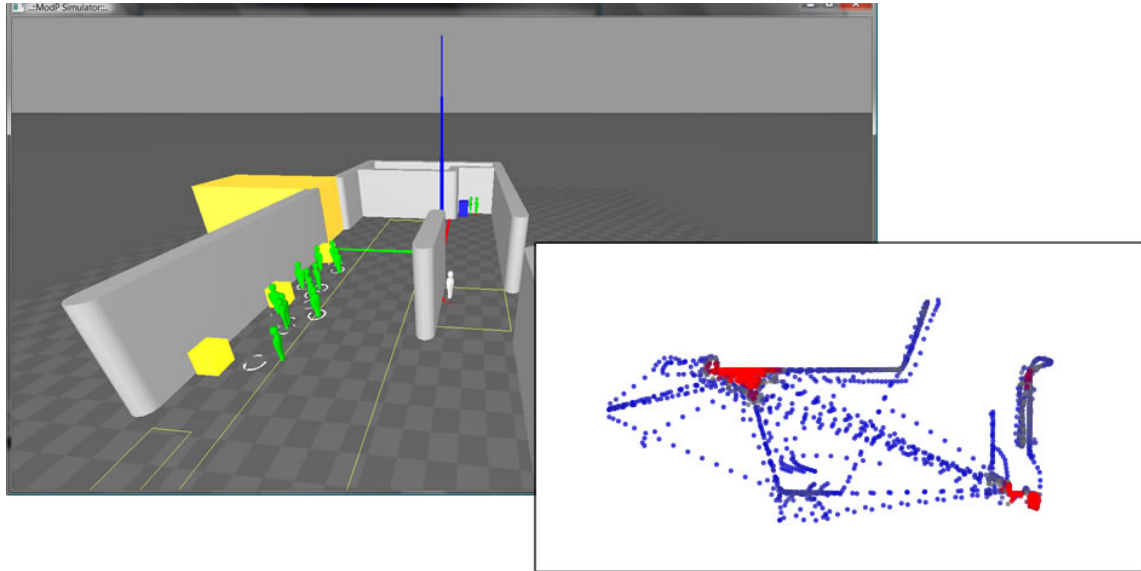


Figura 5.6 – Visualização 3D e mapa de densidade do cenário de baixo fluxo.

Como era previsto, a maior densidade populacional ocorre perto dos pontos de interesse, mais propriamente, multibanco e bilheteira, que possuem tempo médio de utilização e fila de espera.

Quanto ao cenário de simulação de médio fluxo, este apresenta como taxas de entrada de agentes, a seguinte configuração:

- **E/S A** – “Periódico”, com frequência de 20 segundos, número mínimo de pessoas 1 e máximo 4;
- **E/S B** – “Único”, com número fixo de 20 pessoas;
- **E/S C** – “Contínuo”, com taxa de entrada de pessoas de 2%.

A figura seguinte apresenta o gráfico de estatísticas obtido através dos 20 minutos de simulação deste cenário.

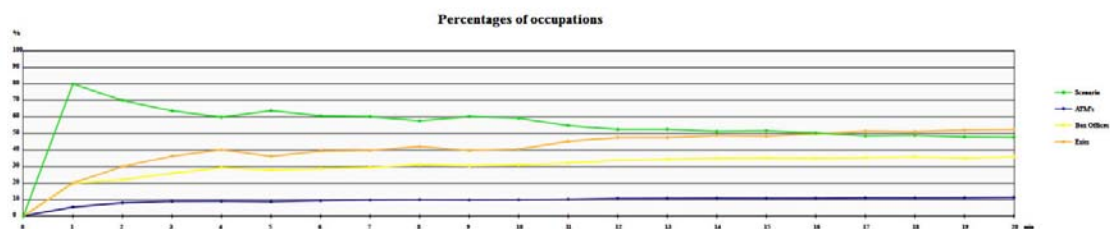


Figura 5.7 – Estatísticas gerais do cenário de simulação com médio fluxo de pões.

Nesta simulação pode-se observar que as taxas de utilização nunca chegam a se aproximar aos valores probabilísticos do plano dos agentes. É de esperar que a densidade populacional tenha aumentado, em relação ao cenário de baixo fluxo e que a densidade populacional tenha alastrado para áreas mais abrangentes do que só a proximidade dos pontos de interesse. A Figura 5.8 demonstra isso mesmo.

Resultados

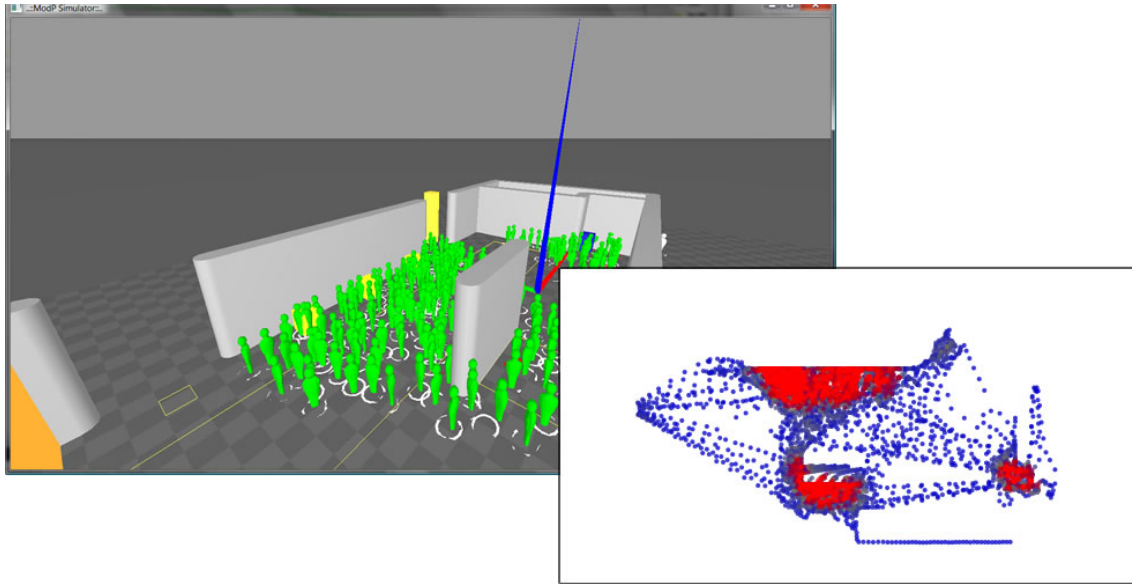


Figura 5.8 – Visualização 3D e mapa de densidade do cenário de médio fluxo.

O último cenário de simulação, de elevado fluxo, apresenta como taxas de entrada de agentes, a seguinte configuração:

- **E/S A** – “Periódico”, com frequência de 10 segundos, número mínimo de pessoas 1 e máximo 4;
- **E/S B** – “Único”, com número fixo de 100 pessoas;
- **E/S C** – “Contínuo”, com taxa de entrada de pessoas de 5%.

A figura seguinte apresenta o panorama geral da simulação de fluxo elevado.

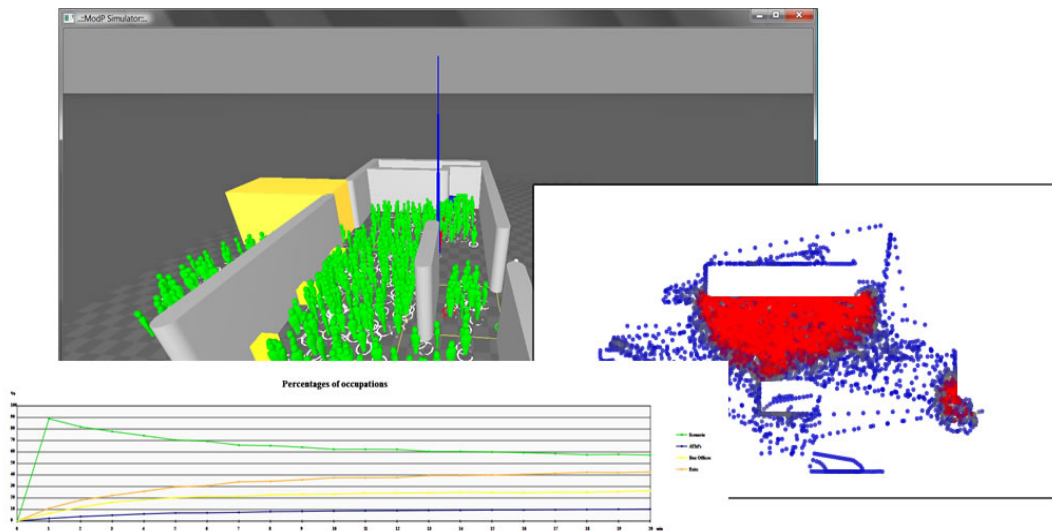


Figura 5.9 – Panorama geral da simulação de elevado fluxo.

Como deduzível, a situação piorou em relação ao cenário anterior. No mapa de densidade apresentado na Figura 5.9, pode-se observar algumas trajetórias descontextualizadas. Estes tipos de acontecimentos devem-se à elevada ocupação das filas de espera, uma vez que os agentes, a partir de um determinado valor de ocupação, tomam a opção de vaguear.

5.3 Síntese

Nesta secção foram apresentados os resultados mais relevantes de toda a implementação, destacando-se a afectação dos diversos parâmetros nos resultados da simulação.

O editor gráfico do cenário permite a criação de diversos ambientes e a parametrização de valores como geometria (comprimento e largura), nome identificador do objecto (id semântico) e propriedades de periodicidade dos componentes de Entrada/Saída.

Apesar de nem todos os parâmetros serem parametrizáveis através do editor gráfico, todos eles necessitam de uma calibração de forma a transportar realismo para a simulação. Desde os parâmetros intrínsecos aos agentes (como distância de projecção da posição futura e definição de raio de conforto) até às parametrizações dos componentes de cenário (Entrada/Saída e tempo médio de utilização dos restantes componentes).

A afectação desses parâmetros na simulação pode ser, tanto observada através do visualizador, como através do gráfico de estatísticas gerais (taxa de utilização de vários pontos de interesse e taxa de presença de peões em relação à população geral gerada) e do mapa de densidade.

Capítulo 6

Conclusões

As questões relativas à afectação do planeamento urbanístico na mobilidade e conforto dos peões são tão evidentes, que os utilizadores regulares já se aperceberam das condições precárias de algumas infra-estruturas de transbordo e do tempo de espera em demasia na interligação a outros tipos de transporte público. O recurso a ferramentas de simulação que possam prever essa afectação torna-se crucial. É também importante perceber como é que o conforto e mobilidade das pessoas afecta o planeamento.

O principal objectivo desta dissertação foi a conceptualização e especificação de uma arquitectura de alto nível da plataforma ModP: uma ferramenta de modelação pedonal baseada em agentes. A arquitectura sugerida foi desenhada de forma a garantir a usabilidade e expansibilidade, tendo como principal objectivo uma aplicação de uso fácil e intuitivo, permitindo uma rápida construção do modelo e uma fácil e útil parametrização. Esta possui algumas características particulares, tanto ao nível de abstracção de dados como de regras, que aplicadas aos agentes, podem facilitar a sua construção e garantir algum grau de heterogeneidade do comportamento dos peões. Um outro destaque da especificação recai sobre a arquitectura baseada em *Plug-ins*, de forma a possibilitar uma rápida integração de outros temas/tipos de simulação.

O protótipo desenvolvido (de forma modular) segue as principais características da especificação e promove tanto a usabilidade como a expansibilidade. O módulo de edição gráfica existente recorre aos mais recentes padrões de desenho de interfaces gráficas, mostrando-se intuitivo e capaz de gerar, de uma forma rápida, uma variedade de cenários. Este módulo possibilita também, a edição de um conjunto finito de parâmetros que afectam directamente a simulação. A arquitectura seguida e implementada, para os módulos de simulação, baseia-se num sistema de *Plug-ins* e encontra-se disponibilizado um *Plug-in* de simulação pedonal em interfaces multi-modais. O protótipo possui uma estruturação progressiva dos dados, reduzindo o seu processamento durante a sua construção (na passagem pelos diferentes módulos) e/ou consulta (particularmente, ao nível dos *Plug-ins* e agentes). Essa progressividade inicia-se com um conjunto de objectos simples, divulgados pelo editor e termina de forma semântica, de uso fácil e útil ao agente. O facto de os objectos serem

referenciados, durante todo o percurso, possibilita a edição de parâmetro e criação de objectos em tempo real de simulação.

O Plug-in de simulação pedonal e os agentes assumem toda a responsabilidade da simulação, no que diz respeito à interacção de acordo com as regras estipuladas e à realização toda a coreografia pretendida de forma acertada. A locomoção dos agentes é realizada através de comportamentos direccionais, baseados em forças sociais. Estes comportamentos reflectem as acções dos agentes resultantes da sua decisão. O processo de decisão gravita em torno de uma pilha de desejos, construída através de regras pré-estipuladas no instante em que os agentes são criados. Essa pilha, em conjunto com os movimentos comportamentais, permite ao agente solucionar uma série de problemas básicos do percurso, através da actualização do seu plano.

O protótipo da plataforma ModP desenvolvido está munido de um conjunto de módulos de visualização que permitem a observação e análise dos dados decorrentes da simulação. Essa visualização é realizada de um modo bidimensional (através de gráficos estatísticos e um mapa de densidade) e também de um modo tridimensional.

Em termos gerais, o objectivo principal deste trabalho foi concluído e a arquitectura especificada foi em grande parte implementada num protótipo funcional.

6.1 Desenvolvimentos Futuros

Foram dados os primeiros passos para o desenvolvimento de uma aplicação para modelação pedonal em estações multi-modais que permita otimizar os transbordos, tendo em atenção o espaço, os modos de transporte e fluxos de tráfego pedonal, assim como a influência de todos esses parâmetros no processo de tomada de decisão do indivíduo que utiliza a estação.

A primeira fase de desenvolvimento dos diversos módulos da plataforma ModP fica terminada por aqui. A modularidade conseguida permite, no entanto a adição de diferentes *Plug-ins* sem necessidade de reestruturação ou de profunda alteração da plataforma. Contudo ainda existem algumas considerações que precisam de ser tomadas. A primeira orienta-se à interface gráfica e à sua necessidade ou não de incorporar níveis/pisos. Essa necessidade dependerá somente do utilizador final, uma vez que este tipo estruturas pode ser simulado, simplesmente com o recurso a conectores, que podem transportar horizontalmente, os peões de uma área para outro. A segunda remete-se ao facto de o módulos *Plug-in* não serem inseridos dinamicamente, o que podia ser resolvido através de um sistema de *dll*'s.

Quanto aos agentes, estes podem incorporar ainda mais comportamentos direccionais, de forma a realizarem movimentos mais complexos e coreografados de forma a transportar mais realismo para a simulação. Um dos comportamentos que poderá suscitar maior curiosidade/interesse será, sem dúvida, “**mover para seguir um líder**” (*leader following*). A sua introdução poderá gerar dois tipos distintos de comportamentos em grupos: agrupamento em filas de espera para futura utilização dos vários pontos de interesse; e agrupamentos desalinhados permitindo a representação de grupos familiares ou grupos de amigos.

Seria também interessante possibilitar a atribuição de um conjunto de regras diferentes a grupos distintos de agentes, habilitando a introdução na simulação de agentes com características completamente distintas. Um outro factor importante, numa futura implementação de agentes, será incorporar um sistema de cálculo automatizado quer do raio, quer das forças máximas e velocidade máxima. Uma das maneiras de ajustar o raio, poderá passar pelo campo energia, ou seja, quanto menor o valor do campo energia, maior a necessidade de procura da mesma e menor a preocupação com a área de conforto.

No sentido de se realizar uma optimização do cálculo do melhor caminho a percorrer, poderia ser aproveitada a noção de zonas e conectores de forma a serem utilizados algoritmos de pesquisa como a A*. Num sentido prático, as zonas seriam os nós e os conectores seriam as ligações entre eles (obtendo-se um grafo dirigido). O cálculo do peso do caminho poderia ser realizado com base no congestionamento de cada zona e/ou do número de agentes registado em cada conector (assumindo que estes funcionam como filas).

Por último, a implementação de um editor de regras, como especificado na subsecção 3.4.2, poderia simplificar todo o processo de parametrização das regras de utilização dos vários componentes do cenário e da sua afectação do campo energia.

6.2 Perspectivas Futuras

A especificação de uma arquitectura baseada em *Plug-ins* envolve duas preocupações distintas:

- Em primeiro lugar, permitir uma fácil incorporação de diferentes tipos de simulação, na mesma plataforma de simulação, com recurso à mesma arquitectura de agentes e estrutura de dados do cenário. Neste caso, o campo “energia” (cujo valor é afectado de forma negativa/positiva pela utilização dos diversos componentes e essa afectação pode ser definida pelo utilizador) poderá assumir um papel chave no processo de decisão dos agentes que possam vir a ser inseridos/criados. Por exemplo, no caso de os agentes representarem peões e/ou automóveis o processo de decisão poderá focar a optimização desse campo, levando os agentes a utilizarem os possíveis bares (no caso de representarem peões) e as possíveis bombas de gasolina (no caso de representarem automóveis).
- Em segundo lugar, permitir uma possível separação física dos mesmos, o que deve melhorar a performance de simulação. Ao aliar-se esta separação à execução em simultâneo de vários *Plug-ins* (que executem quer o mesmo tipo de simulação em edifícios/interfaces diferentes, quer tipos diferentes de simulação, como tráfego veicular) e partilha de dados entre eles, pode-se obter uma simulação sobre cenários com maiores dimensões e composição mais complexa. A possível partilha de dados pode pressupor uma permutação de agentes entre *Plug-ins*, ou simplesmente informações sobre entradas e saídas de agentes, de forma a serem eliminados ou criados nos diferentes *Plug-ins*.

Uma simulação em larga escala e sobre uma diversidade de cenários, pode obrigar a uma necessidade importação de mapas e/ou outros tipos de formatos de dados quer respectivos a edifícios, quer respectivos a mapas citadinos (estradas, zonas, quarteirões e passeios).

Referências

- [Bar91] J. Barcelo, "Software environment for integrated RTI simulation systems," in *Advanced Telematics in Road Transport, Proceedings of the DRIVE Conference*, vol. II, Amsterdam, 1991, pp. 1095-1115.
- [Buc05] M. Buckland, *Programming Game AI by Example*. Texas, United States of America: Wordware Publishing, Inc., 2005.
- [Bur04] W. Burghout, "Hybrid microscopic-mesoscopic traffic simulation," in , Royal Institute of Technology Doctoral Dissertation, 2004.
- [Cha99] A. Champion, "Traffic generation with the SCAneR II simulator: toward a multi-agent architecture," in *Proceedings of the first Driving Simulation Conference*, 1999, pp. 311-324.
- [DHe01] D. Helbing, P. Molnar, I. J. Farkas, and K. Bolay, "Self organizing pedestrian movement," in *Environment and Planning B: Planning & Design*, 2001.
- [DHe02] D. Helbing, I. J. Farkas, P. Molnár, and T. Vicsek, "Simulation of Pedestrian Crowds in Normal and Evacuation Situations," in *Pedestrian and Evacuation Dynamics*, Berlin, 2002, pp. 21-58.
- [Dun02] F. Dunn and I. Parberry, *3D Math Primer for Graphics and Game Development*. 2320 Los Rios Boulevard Plano, Texas 75074: Wordware Publishing, Inc., 2002.
- [Eal05] A. Eales, "The Observer Pattern Revisited," in *18th Annual Conference of the National Advisory Committee on Computing Qualifications*, New Zealand, 2005, pp. 163-166.
- [Est09] E. F. Esteves, R. J. F. Rossetti, P. A. F. Ferreira, and E. C. Oliveira, "Conceptualization and Implementation of a Microscopic Pedestrian Simulation Platform," in *Technical Track on Advances in Computer Simulation, the 24th Annual ACM Symposium on Applied Computing, SAC.*, Waikiki Beach, Honolulu, Hawaii, USA, 2009 (accepted).
- [Fer082] P. A. F. Ferreira, "Specification and Implementation of an Artificial Transport System," Master Dissertation, Engineering Faculty of Porto University, Porto, 2008.
- [Gol09] G. S. Group. , 2009, Jan.. OpenGL - The Industry Standard for High Performance Graphics.. Available at <http://www.opengl.org/>
- [Gra04] P. C. Graziotin, B. Turkienicz, L. Sclovsky, and C. M. D. S. Freitas, "CityZoom - A Tool for the Visualization of the Impact of Urban Regulations," in *Proceedings of the 8th Iberoamerican Congress of Digital Graphics*, 2004, pp. 216-220.
- [HKl00] H. Klüpfel, T. Meyer-König, J. Wahle, and M. Schreckenberg, "Microscopic Simulation of Evacuation Processes on Passenger Ships," in *Theoretical and*

Referências

- Practical Issues on Cellular Automata*. London: Springer-Verlag, 2000.
- [Kes02] A. Keßel, H. Klüpfel, J. Wahle, and M. Schreckenberg, "Microscopic simulation of pedestrian crowd motion," in *Pedestrian and evacuation dynamics*, pp. 193-202, 2002.
- [Kin96] D. Kinny, M. Georgeff, and A. Rao, "A Methodology and Modelling Technique for Systems of BDI Agents," in *7 th European Worksgop on Modelling Autonomous Agents in a Multi-Agent World, LNAI* , 1996.
- [Leg08] L. I. Limited. , 2008, . Legion. Available at <http://www.legion.com/>
- [Mac08] M. MacDonald and Y. R. , 2008, . STEPS software | Mott McDonald. Available at <http://www.mottmac.com/skillsandservices/software/stepssoftware/>
- [MGM00] M. G. McNally, "The four step model," in *Handbook of Transport Modelling*, Oxford: Pergamon Press, 2000, pp. 35-52.
- [Mic07] M. Corp. (2007, Jan.) Microsoft Office Fluent User Interface. Fact Sheet.
- [Nok08] N. a. i. s.-i. Corporation. (2008) Qt Extended. Whitepaper.
- [Ope04] OpenSteer. , 2008, . OpenSteer. Available at <http://opensteer.sourceforge.net/index.html>
- [Ore00] T. I. Oren, S. K. Numrich, A. M. Uhrmacher, L. F. Wilson, and E. Gelenbe, "Agent-Directed Simulation - Challenges to meet defense and civilian requirements," in *Proceedings of the 2000 Winter Simulation Conference*, vol. II, 2000, pp. 1757-1762.
- [PTV08] P. AG. (2008) VISSIM PEDESTRIAN MODELING. Scope of Application.
- [Qua08] Q. P. Ltd. , 2008, . The Urban Analytics Framework. Available at http://www.paramics-online.com/product_pedestrian.php
- [Qui03] M. J. Quinn, R. A. Metoyer, and K. Hunter-Zaworski, "Parallel Implementation of the Social Forces Model," in *Proceedings of the Second International Conference in Pedestrian and Evacuation Dynamics*, London, 2003, pp. 63-74.
- [Rey00] C. W. Reynolds, "Interaction with Groups of Autonomous Characters," in *Game Developers Conference*, San Francisco, California, 2000, pp. 449-460.
- [Rey99] C. W. Reynolds, "Steering Behaviors For Autonomous Characters," in *The proceedings of the 1999 Game Developers Conference*, 1999.
- [Ron05] N. Ronald and L. Sterling, "Modelling pedestrian behaviour using the BDI architecture," in *IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'05)*, Compiegne, France, 2005, pp. 161-164.
- [Ros02] R. Rossetti, et al., "Using BDI agents to improve driver modelling in a commuter scenario," *Transportation Research, Part C (Emerging Technologies)*, vol. 10, pp. 373-398, 2002.
- [Ros99] R. J. F. Rossetti and S. Bampi, "A Software Environment to Integrate Urban Traffic Simulation Tasks," in *Journal of Geographic Information and Decision Analysis*, vol. 3, 1999, pp. 56-63.
- [Sar08] S. Sarmady, F. Haron, and A. Z. H. Talib, "Multi-Agent Simulation of Circular Pedestrian Movements Using Cellular Automata," in *Asia International Conference on Modelling and Simulation*, 2008, pp. 654-659.
- [Sav06] S. S. AG. (2006) The pedestrian simulation software. documentation.

Referências

- [Sch99] T. Schelhorn, D. O'Sullivan, M. Hakley, and M. Thurstain-Goodwin, "STREETS: An Agent-Based Pedestrian Model," in *Centre for Advanced Spatial Analysis (University College London)*, London, 1999.
- [Tec08] X. Technologies. , 2008, . AnyLogic – multi-method simulation software. Available at <http://www.xjtek.com/anylogic/>
- [Tek] K. Teknomo. Micro-PedSim. User Manual.
- [Tek00] K. Teknomo, Y. Takeyama, and H. Inamura, "Review on Microscopic Pedestrian Simulation Model," in *Proceedings Japan Society of Civil Engineering Conference*, Japan, 2000.
- [Tek02] K. Teknomo, "Microscopic Pedestrian Flow Characteristics: Development of an Image Processing Data Collection and Simulation Model," Ph.D. Dissertation, Department of Human Social Information Sciences, Graduate School of Information Sciences, Tohoku University, Japan, 2002.
- [Vic06] V. Bret. (2006, Mar.) Magic Ink, Information Software and the Graphical Interface.
- [Wei06] W. Shao, "Animating Autonomous Pedestrians," Ph.D. Dissertation, Courant Institute of Mathematical Sciences, New York University, New York, 2006.
- [Win02] W. Daamen, "SimPed: a pedestrian simulation tool for large pedestrian areas," in *Conference Proceedings EuroSIW*, London, UK, 2002.
- [Woo99] M. Wooldridge, "Intelligent Agents," in *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, 1999.

Índice Remissivo

A

AA, 17, 27, 28, 30, 31

AE, 16

API, 37

B

BDI, 14, 34, 36

BRAF, 22

C

CC, 22, 23, 24

CCF, 23

CF, 20, 22, 23, 24, 25

CPI, 23

D

dll, 70

DM, 19

DR, 16

DV, 16

E

E/S, 20, 21

ECP, 23, 24, 25

G

GDM, 26, 27, 28

GS, 27, 28

I

IGE, 13, 17, 21, 23, 24

M

MCF, 22, 24

MCS, 17, 23, 25, 26, 27, 30

ModP, 13, 15, 16, 17, 18, 21, 23, 30, 36, 37, 45, 46, 60

MP, 23

MR, 16, 17

O

OQVVEOQVT, 17, 22

P

PI, 20, 21, 22, 23, 24

S

SINF, 14, 17, 21

SMAN, 17, 21

U

UML, 26

V

VCC, 23

X

XML, 23