FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FEUP

Crowd Simulation Applied to Emergency and Evacuation Situations

Fábio Homero Moreira e Aguiar

Master in Informatics and Computing Engineering

Supervisor: Rosaldo Rossetti (Ph.D)

Second Supervisor: Eugénio Oliveira (Ph.D)

28^{th} June, 2010

Crowd Simulation Applied to Emergency and Evacuation Situations

Fábio Homero Moreira e Aguiar

Master in Informatics and Computing Engineering

Approved in oral examination by the committee:

Chair: António Augusto de Sousa (Ph.D)
External Examiner: Elisabete Arsénio (Ph.D)
Supervisor: Rosaldo Rossetti (Ph.D)

31st July, 2010

Abstract

Efficiently managing and organizing crowds in emergency situations has become a more important area of study in the last years and plays an important role in the design of a building or area. The use of simulation techniques to predict these situations is becoming more common as it aims to predict and prevent major accidents. Unfortunately, studies that focus on the safety of a project take place after the project is designed. Performing these studies at this phase increase the project's cost and its duration, so it becomes important to study a solution that allows designers to verify the safety of the building as well the efficiency of its evacuation routes.

There are several simulation techniques from which we can highlight different approaches based on particle systems and heterogeneous multi-agent systems. There is also a range of existing simulators with different purposes. Considering a BDI agent architecture, there are also several approaches when it comes to the definition of the agents and their beliefs desires and intentions.

The problem on which this dissertation focus is the implementation of a module on a previously developed simulator that allows designers to create and configure agent types in order to create more realistic simulations, therefore improving their outcome.

The simulator used as a starting point for the implementation of this module is the ModP simulator. This simulator has all the needed features to implement the desired module and uses Qt for interface and the OpenSteer library for movement and steering behaviour.

The implemented agent editor module allows the creation of agent types trough a simple syntax that specifies the agents parameters, such as maximum speed and vision range, and its evacuation plan. This plan consists in a group of pre-implemented commands that specify which action should be taken by the agent and in which order. While modeling the simulation it is possible to select the types of agents that will be present in it and their distribution in percentage.

A batch of tests were made in order to show the realism improvement brought by the implementation of this module. It is possible to observe how changing parameters and agent types interferes with the crowd behaviour, speed and evacuation time.

In conclusion, the implemented module allows an easier and more realistic specification of the agents in the simulation, resulting in a more realistic simulation. Some interesting features were not implemented but would increase even more the realism of the simulator and better aid designers to assess the project's safety.

Resumo

Gerir e organizar multidões em situações de emergência e evacuação de forma eficiente tem-se vindo a tornar uma importante área de estudo nos últimos anos e representa um papel importante no desenho de um espaço. O uso de técnicas de simulação para prever estas situações está-se a tornar mais comum visto que se pretende prever e prevenir acidentes. Infelizmente, estudos que se focam na segurança de um projecto são efectuados depois da fase de desenho, o que pode aumentar os custos do projecto e a sua duração. Assim sendo, demonstra-se de grande importância estudar uma solução que permita aos desenhadores verificar a segurança de um espaço assim como a eficácia das suas rotas de evacuação.

Existem várias técnicas de simulação, das quais se podem destacar aproximações com base em sistemas de partículas e sistemas multi agente heterogéneos. Existem também vários simuladores com diferentes finalidades. Em relação a agentes BDI, existem também algumas aproximações no que diz respeito à definição dos agentes e as suas crenças, desejos e intenções.

O problema no qual esta dissertação se foca é a implementação de um módulo num simulador previamente desenvolvido que permita aos desenhadores criar e configurar tipos de agentes com o objectivo de criar simulações mais realistas e, consequentemente, melhorar os resultados obtidos das mesmas.

O simulador usado como ponto de partida para a implementação deste módulo é o ModP Simulator. Este simulador possui todas as funcionalidades necessárias para implementar o módulo desejado. Usa QT para a interface e a biblioteca OpenSteer para cinemática.

O módulo de edição de agentes implementado permite a criação de tipos de agentes através de uma sintáse que especifica os parâmetros do agente, como a sua velocidade máxima e campo de visão, e o seu plano de evacuação. Este plano é um conjunto de comandos pre-implementados que especificam que acção deve ser executada pelo agente e em que ordem. Durante a modelação da simulação, é possível seleccionar os tipos de agentes que estarão presentes nela e a sua distribuição em percentagem.

Foi efectuado um conjunto de testes com o objectivo de demonstrar o melhoramento do realismo obtido através da implementação deste módulo. É possível observar a interferência causada pela variação de parâmetros ou percentagem de agentes no comportamento da multidão, velocidade e tempo de evacuação.

Em conclusão, o módulo implementado permite uma especificação rápida e realista de agentes na simulação, tendo como resultado uma simulação mais realista. Algumas funcionalidades e outros módulos não foram implementados mas estes poderiam aumentar ainda mais o realismo do simulador, ajudando ainda mais os desenhadores a confirmar a segurança do projecto.

Acknowledgements

For all the support given troughout my thesis, I'd like to thank my girlfriend, all my family, specially my parents, and friends (you know who you are, don't whine about how you didn't get your name here, no one did) and everyone who somehow helped me getting this project done.

Without them, this project would have been a lot harder and probabily not possible at all.

I would like to thank my supervisor, Professor Rosaldo Rossetti for the guidance given during this dissertation.

Last, but not least, thank you coffee!

Fábio Aguiar

Contents

List of Figures

LIST OF FIGURES

Abbreviations

API	Application Programming Interface
BDI	Belief-Desire-Intention
CAD	Computer-Aided Design
CROSSES	CROwd Simulation System for Emergency Situations
DXF	Drawing Exchange Format
LIACC	Artificial Intelligence and Computer Science Laboratory

Chapter 1

Introduction

Efficiently managing and organizing crowds in emergency situations, let these be caused by man or nature, has become more important since 9/11[SVLS06]. Considering that human behavior can frequently be unpredictable[DFG09], it is of vital importance to consider these situations while designing a public area in order to minimize possible accidents as well as improve the response to and recovery from these accidents, avoiding further and more serious consequences[DFG09].

The ability to predict and control human behaviour when in emergency and evacuation situations is considered to be an important area of investigations not only in computing, mostly related to simulation and artificial intelligence, but also in such areas as psychology and sociology[LZC+08]. One of the most studied effects emerging from crowd behaviour occurs when a group of individuals behave as one unit, tending to consider individual actions less important and behave as the crowd itself[BMOB03][HFMV02].

1.1 Context and Motivation

As referred before on this document, ensuring the safety of an area or building represents a very important role in the design. Nowadays, studies that focus on the safety of an area or building and its evacuation routes take place after the project is designed. Making these studies in this phase can bring some problems such as increased costs relative to design updates to the project. For instance, removing a column that obstructs pedestrian flow or widen an emergency exit that is too narrow[HFMV02] for the expected flow are changes that can be very costly to make and, in some cases, impossible.

Furthermore, each crowd has its own characteristics and demographic configuration[HS04]. Therefore it is of great importance to study a solution that allows an easy configuration of the crowd to be simulated, generating results that would be closer to reality.

The successful use of the application would not only ensure the safety of the building and its evacuation plan but also allow this certification to happen while the project is being developed. Furthermore it will decrease the cost of eventual changes to the project as well as make it more time effective than verifying its safety after it has been designed.

1.2 Expected Contributions

The problems deriving from focusing on the safety of a project when it comes to pedestrian behavior only after the area or build has been designed bring to the matter the need to predict pedestrian behavior in emergency situations while the project is being developed, therefore allowing the designers to better suit the area or building to the expected users.

That being, the aim of this project is to create an application that allows designers to simulate pedestrian movement and behavior during emergency and evacuation situations while the building or area is being developed. This will allow them to quickly observe a prediction of the pedestrian flow throughout the building or area and easily find choke points, narrow routes and obstructed paths. Detecting these problems during the design phase of the project will allow faster and simpler changes and consequently, improve cost and time efficiency.

This project, specifically, aims towards the development of a tool that allows the creation of different agents and its parameters by the designers in order to create more realistic simulations. This is an important part of the simulation as every area or building should be used by different types of people and in different quantities. For example, a football stadium will have different types and quantities of agents when compared to a train station or a hospital. The general idea is that the tool to be created allows the creation of any number of agents and the configuration of their parameters, such as maximum speed, vision depth and emergency plan) as well as their expected quantities throughout the area of the simulation.

1.3 Methodological approach

The module to implement will be added to an existing simulator, ModP. The reasons why this simulator is used as base for the module are better explained in chapters 2 and 4. The application to be added to the existing simulator will allow an easy and quick creation of different agent types as well as their probability of existing at a given time in the area of the simulation. To do so, the user only has to add a text file containing the list of parameters that the agent requires and his emergency plan. These files are written in a simple syntax and the plan of each agent type uses a list of pre-implemented commands.

While editing the simulation in the user interface, when a drain source is placed in the simulation area, the agents option box will show all the available (previously created) agents and will allow the input of the percentage of each agent type that will exit that drainsource.

This way, it is viable to collect data while designing the project for a specific area or building and, through that demographic data, add the types of agents that are expected to use the area and in which percentages, making the simulation much more realistic.

1.4 Structure of this document

The following chapters will go into detail on the problem at hand. First of all, an overview of the state of the art will be presented, detailing some existing tools that aim to solve this problem or similar ones. Following that chapter, the problem at hands will be described and explained in detail.Next, a detailed description of the simulator to be used will be described, along with its main features. Afterwards, the implemented agent editor module will be explained. The main features and the used syntax that defines the agents will be described as well as the integrations with the Opensteer library and QT. In order to evaluate the implemented module, there will also be presented a chapter with tests and results evaluating the impact of the developed tool in the simulation. Finally, there will be a chapter on conclusions, analyzing the results and contribution of the implemented module as well as a description of further developments and future work that would improve the present simulator.

Chapter 2

State of the art

2.1 Introduction

Considering the amount of associations that take interest in these problems, for instance, government agencies and insurance companies, several approaches are being explored. The objective is to get a solution that is as close as possible to real scenarios of evacuation and emergency. Along with these approaches, different simulation technics are applied. In this section, the various approaches and technics will be described and evaluated, considering their application to the problem at hands.

2.2 Simulation Technics

Nowadays, there is a variety of different approaches and solution attempts directed at the problem of efficiently simulate pedestrian behaviour on emergency and evacuation situations. Throughout the various existing solutions, two groups stand out as the most explored. The first one is based on solutions that simulate the crowd as a particle system, i.e. considering the individuals as a homogeneous group in which everyone shares the same batch of behaviours. The second one is based on multi agent systems. This last one considers the heterogeneity of the agents, allowing them to have different behaviours and parameters. Both options will be evaluated in this section, considering their advantages and disadvantages.

2.2.1 Crowd as Particle Systems

The simulation of crowds as particle systems is an approach that has been carried out with successfully results. A few of these implementations can be studied in more detail in the following articles [BCN97].

When it comes to a solution based on particle systems we can consider as major advantage the fact that these solutions require less processing capacity. This arrives from the fact that, in this case, every agent shows the same behaviour and, in most situations, these behaviours are very simple[BMOB03].

Nevertheless, it is somewhat easy to notice the disadvantages of solutions based on particle systems. Considering that human behaviour varies from individual to individual and can, sometimes, be very hard to predict, it isn't accurate enough to simulate a crowd without considering the different behaviours between individuals. These behaviours can depend on a great amount of demographic characteristics, for instance, age, physical aptitude and even psychological state.

2.2.2 Simulation Based on Heterogeneous Agents

Considering the previous section, it is necessary to evaluate an approach based on heterogeneous multi agent systems. In this case, unlike the previous approach, the simulation will need a higher processing capacity due to the variety of agents and behaviours that are possible to simulate.

Despite this disadvantage, this approach proves to be closer to reality[BMOB03] because it allows a widest modelation of agent behaviours. That being, it is possible to model agents in a dynamic way, considering their previously referred characteristics. With this approach it is easier to apply the simulation to a wider range of situations, ranging from different countries, to different zones and cultures, allowing, through demographic studies a more accurate characterization of the population that will use the simulated area or building.

2.3 Existing Tools

Being this subject an important area of study when it comes to building safety, it is important to analyze existing tools and approaches to the problem of efficiently simulating

crowds in emergency and evacuation situations. In this section, some of them will be presented and analyzed in detail with the objective of choosing the best approach possible.

Lets start by analyzing the application developed by the Crowd Simulation Group of the School of Computing Sciences, UEA[Gro10]. This tool allows crowd simulation in open areas, in this case cities. The simulator aims for a more compelling visual reality[LRD07][RD06] and simulates pedestrian movement in normal situations, opposed to emergency or evacuation situations. In this case, the objective is to create a visually realistic environment considering not only the graphic detail but also the movement realism.

Another interesting simulator is CROSSES [BHH+]. This simulator was developed with the objective of aiding in the train and rehearse of crowds, in order to prepare them for emergency situations. Its target is not a random crowd but a known group of individuals. For instance, workers at an electrical station. Therefore its main goal is not to predict crowd behaviour, but to work as a tool to train the known users of a building so that when an emergency situations comes, the workers are better prepared for that situation, decreasing the time it takes to evacuate and, therefore, the risks of injuries.

This approach, like the one presented before, has very realistic graphics, using satellite images and pictures in order to better present the model in which the simulation will occur. Users can train by joining the crowd as an avatar, which correspond to their individual in the simulation and which they can control.

The simulator described by Tim Thirion and Suddha Kalyan Basu in Real-Time Crowd Simulation for Emergency Planning[TB] shows a simpler approach, in comparison with the previous ones, with less visual detail. It is stated that the agents have limited information when compared to a human being. For instance, the notion of *door* is not know by the agents. A door or a narrow passage is considered the same by the agent as for a human being they are different things.
On the other hand this simulator is not as efficient as one could expect when it comes to collision detecting. For every cycle of the simulation, collisions are calculated between every pair of agents. Nevertheless this simulator shows an interesting approach when it comes to the spacial perception by the agents.

2.4 BDI Agents and Configuration

Considering that the objective of this project is to implement a module that allows agent creation and configuration in the simulation environment, it is important to address the

different approaches taken so far to this same problem. Therefore, in this section, some of the present approaches will be presented in detail.

First of all, the reason for studying BDI agents as the main approach for the agent editor module is that over the past years, this agent systems have been focused in a great amount of studies[Rao96]. This agent systems are applied to situations in which the agents are fed continuous information in real-time and have to make decisions considering that information[Rao96]. The beliefs, desires and intentions are what makes the agent take a decision, considering the information it has collected at that point.

Many logical languages have been specified over the years. The most relevant to this project will be presented in the next paragraphs in order to better decide on the final syntax to be implemented in the agent editor module.

AgentSpeak(L) is a logical language describe in detail by Anand S. Rao[Rao96]. The implemented language can be interpreted as an abstraction of implemented BDI systems that use the three main attitudes (beliefs desires and intentions) as data structures instead of modal operators. The implemented language allows the configuration of BDI agents through a simple syntax. The plan consists of a triggering event and a context. Then a series of rules that dictate when and why the plan is triggered. Then the plan has a body that consists on goals or actions. Below is an example of the definition of a plan in AgentSpeak(L).

```
+location(waste,X):location(robot,X) &
  location(bin,Y)
  <- pick(waste);
   !location(robot,Y);
   drop(waste).
```

The first successful attempt AgentSpeak(L) language programs is described by Rodrigo Machado and Rafael H. Bordini in [MB02].

The most common used interpreter for this language is Jason which is a java based interpreter for an extended version of this language and is described in detail in [BWH07].

Another approach worth mentioning is Jam, which is described as BDI-theoretic agent architectures and can is explained in detail by Marcus J. Huber[Hub99]. One of the problems of using Jam, which is developed in Java or UMPRS, Jam's C++ version is that for

the user to define the agents configuration, i.e. beliefs, desires and intentions and respective plans, it is required to have some experience with either language (Java for Jam and C++ for UMPRS).

2.5 Summary

Considering the importance given to the problem of efficiently simulating crowd in emergency and evacuation situations, there has been a fair amount of different approaches to this problem. As explained in section **??**, when it comes to simulation technics, simulating a crowd as a heterogeneous multi agent system is the most realistic approach. Furthermore, considering all the different tools and approaches related to the definition of BDI agents, the one to be implemented in this project will be created specifically for this purpose and can be described as a simples version of AgentSpeak(L) that better fits the ModP Simulator. The next chapter will describe the problem at hands in detail and explain its importance.

Chapter 3

Problem Statement

3.1 Introduction

In this chapter, the focused problem of this project will be described in detail, considering all the studied technics and approaches in the previous chapter 2 as well as the tools previously detailed.

3.2 Problem Statement

As described in the introduction of this document, new approaches to design buildings considering their safety have been an important area of research over the last years. First of all, its is important to define what a crowd is. Although there are several definitions for what characterizes a crowd, the broad meaning of the term can be defined by a series of features that define a crowd and its individuals[CCR09]:

- A considerable amount of people;
- These people come together in a specific area, i.e. the same physical space;
- These people come together for a determined period of time, i.e. not momentarily;
- Individuals in a crowd share objectives or have common interests;
- Individuals in a crowd show similar behaviours when acting in group;
- Individuals in a crowd interact with each other;

A crowd can also be characterized through a number of parameters, for instance its size (number of individuals), density, period of time, level of objective and interest sharing and level of adjustment to the environment[CCR09]. There are other features that, even being considered secondary when defining a crowd [CCR09], are also important to the scope of the problem of efficiently simulating the crowd. This features are demographic data of the crowd, some individuals ability or inability to move quickly and even the weather, depending on the area, building or event.

Considering all this, it has become of great importance to allow the designers to define the crowd to simulate easily and with a fair amount of detail. The problem is then to create a module that allows designers to create types of agents and define their characteristics in accordance to the demographic data about the predicted users of the building that is being designed. It is also necessary to allow the configuration of those agents in the simulation, i.e. how many agents of each type should exist or on which percentage of a determined type of agents should spawn on a specific location. This module will be added to the ModP simulator, which will be described in detail in the next chapter 4.

3.3 Proposed Methodology

Considering all the approaches and technics presented in chapter 2 and the aim of the project, the most important decision to make is whether to simulate the crowd as a particle system or a heterogeneous multi agent system. Considering that the objective of the project is to allow designers to better and easier describe the crowd to be simulated, the only possible approach is heterogeneous multi agent systems. In this case, the ModP Simulator[Est09] developed by Edgar Esteves in LIACC will be the simulator used. In chapter 4, this simulator will be described in detail, along with the features that make it the best option as a base to implement the agent editor module.

When it comes to the syntax or tool that will implement and allow the configuration of the agents, JAM and UMPRS can not be implemented because this architectures require that the user has some experience with either Java or C++, which should not be required for the designers. Finally, the AgentSpeak(L) language has been considered. But because of its complexity when compared with the agents architecture of the modp simulator, the chosen approach is to create a simpler syntax that considers the needed agent parameters and plan. This syntax will be based on the structure of the AgentSpeak(L) language but simplified, in order to minimize the learning curve assossiated with the language.

The Agent Editor Module will be integrated with the Opensteer library in order to better translate the implemented commands into movement methods that can be applied

to the simulation engine and, therefore, allow the agent to move accordingly to the designated plan and simulation status.

3.4 System Requirements

The agent editor module should be fully integrated with the existing ModP Simulator, more specifically with the Graphic Editing Interface in order to allow an easy and dynamic agent distribution configuration throughout the simulation through a list of editable agent type percentages in the interface. The language that will allow the creation, definition and edition of agent types should be siple and easy to learn and master in order to allow designers to quickly learn the syntax and easily define the needed agents.

Finally, this module should maintain the simulator architecture, integration with the OpenSteer library and the Qt framework, connecting the agent types to the interface and the interpreted commands from the agent type files to the OpenSteer library in order to connect the syntax for agent definition and the simulation engine.

3.5 System Architecture

With the addition of the Agent Editor Module, the architecture of the ModP Simulator will have small changes which can be visualized in th following figure.

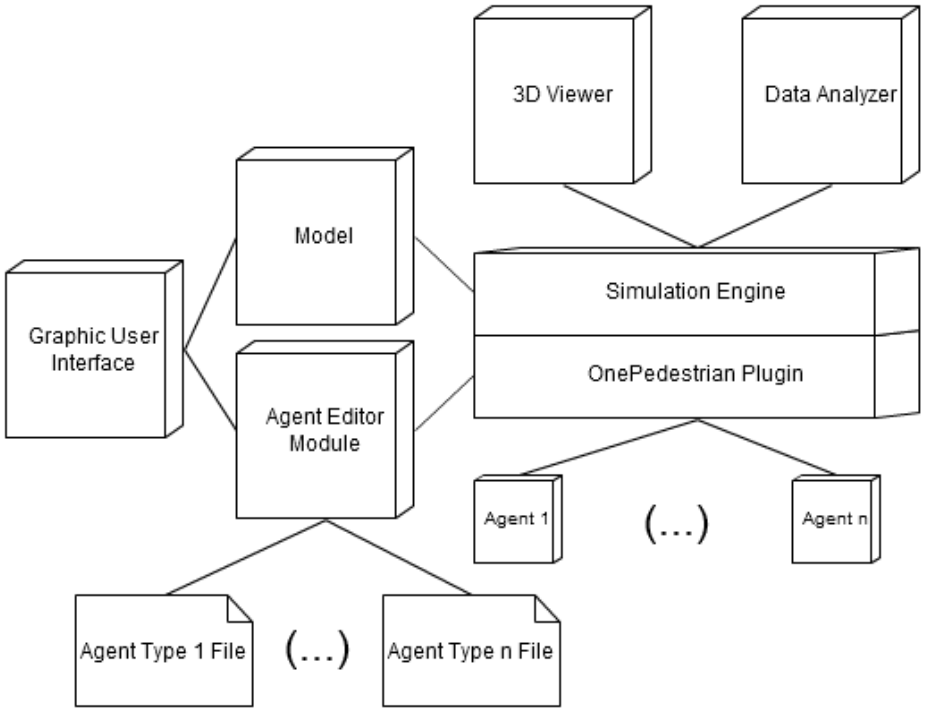

Figure 3.1: ModP Architecture with the Agent Editor Module

The Agent Editor Module will receive as input information the files describing all the previously defined agent types and the agents distribution in percentage from the Graphic Editing Interface. This information will allow, through integration with the OpenSteer library, the modelation of the crown in the OnePedestrianPlugin, which is the part of the Simulation Engine that coordinates and updates the agents in the simulation.

It is also important to explain that in order to read and apply the data from the agent type files, an interpreter will be implemented in the Agent editor module, in order to read the data from the files and apply that data to the OnePedestrianPlugin. The architecture of the agents will also change in order to accomodate the needed data structures that will allow the agent to follow its designed plan.

3.6 Summary

The problem on which this document will focus is the creation of a tool that allows designers to create and define types of agents in order to better characterize a crowd for simulation. This module will be added to the existent ModP simulator and will be described in detail on the following chapters.

It is of great importance that the Agent Editor Module does not alter the core architecture of the simulator. It should then be integrated with the existing modules and use part of the existing modules and structures to maintain the desired architecture. It is also important so specify that the language to be used in order to create and define agent types should provide a simple to learn syntax without compromising a wide range of agent options and definitions. The outcome of a practical solution for this problem will allow designers to simulate more realistic crowds while the project is being designed, resulting in faster and less complex changes in that same project.

Chapter 4

ModP Simulator

In this section an overview of the ModP Simulator will be presented, as well as its main features and modules. This is the simulator into which the Agent Editor Module will be integrated.

4.1 Introduction

The ModP simulator has been developed by Edgar Esteves[Est09] in 2009 in LIACC. This simulator was developed as a multi agent system which is the approach that was considered to be the best option in chapter 2. The objective is to, over time, complete this simulator with more modules and features in order to accurately simulate and predict crowd behaviour during emergency and evacuation situations.

4.2 Basic Features of ModP

The existing simulator was developed in C++ with the Qt framework[qt] for interface, OpenSteer[ope] library for steering behaviours and OpenGl[Gro] for the 3D viewer of the simulation. The simulator has four main modules: a graphic editing interface, a 3D viewer, a simulation engine and a data analyzer. These modules will be described in detail in the following section.

4.3 Interacting Modules

All the modules present in the ModP simulator play an important role in the final result of the simulation. That being, they need to be correctly connected to each other in order to work as intended. The structure of the simulator can be better understood on the following image.

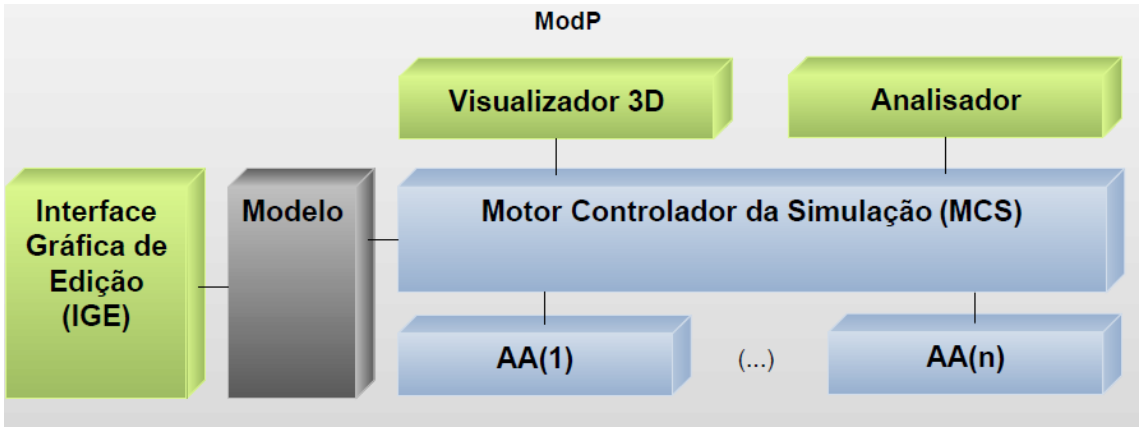

Figure 4.1: ModP Simulator Architecture. Obtained from [Est09]

4.3.1 Graphic Editing Interface

This module allows the user to edit the model of the area or building in which the simulation will occur. It is possible to add and edit walls and doors as well as drain sources. The drain sources are the points in the model from which the agents will emerge. They are fully configurable when it comes to the generated flow of agents, allowing an accurate control of the number of agents present in the simulation as well as their flow throughout the model.

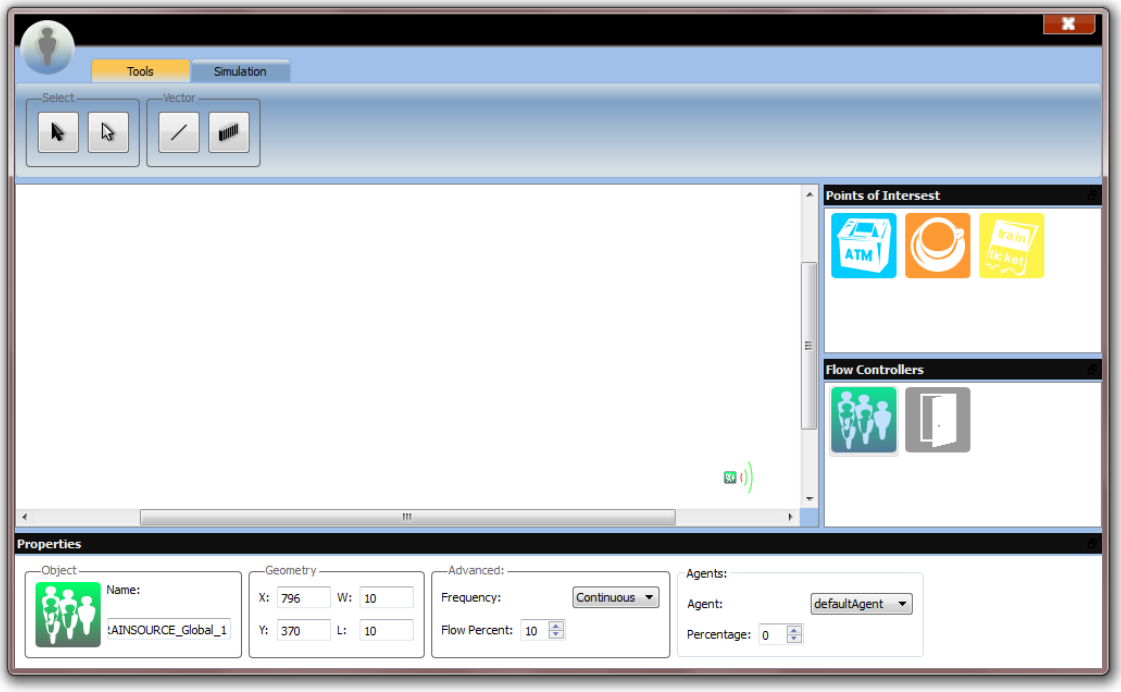

Figure 4.2: Graphic editing interface

4.3.2 3D Viewer

This module is responsible for the visual feedback of the simulation. All the elements inserted into the model in the editor will appear in the viewer and the agents flow will be shown in accordance to the previously configured drain sources in the graphic editing interface. While the simulation occurs in the viewer, it is possible to go back to the editor and add new entities to the model or edit the already existing ones. For instance, add a wall or a new drain source or increase or decrease the flow of an already existing drain source.

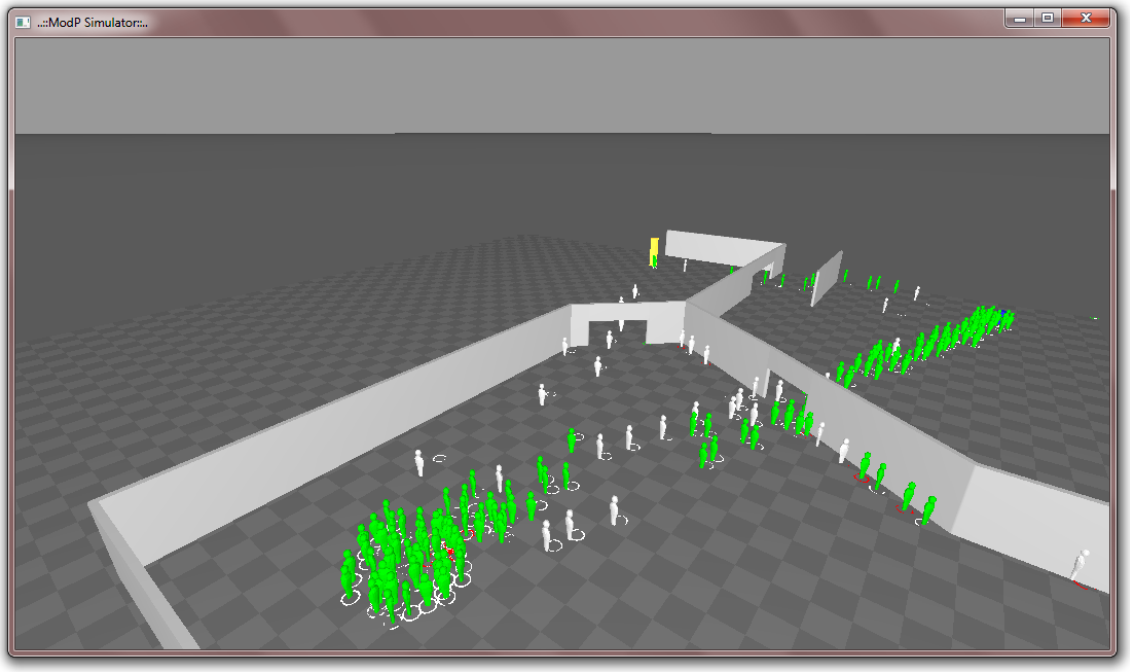

Figure 4.3: 3D Viewer

4.3.3 Simulation Engine

The simulation engine is responsible for all the agent-model and agent-agent interactions throughout the simulation. This module updates the position and parameters of the agents according to their state in the simulation and considering all the entities present in the model using attraction and repulsion forces.

This module also reads the drain source information from the graphic editing interface and, according to the configuration of each drain source, launches the agents into the simulation.

4.3.4 Data Analyzer

The data analyzer module manages the data received from the simulation engine and displays them in a intuitive way according to the users needs. The data analyzed by this module is focused on timing and trajectory of the pedestrians and the crowd itself, displayed in a density map.

4.4 Summary

The ModP simulator developed by Edgar Esteves in LIACC and described in detail in [Est09] proves to be a good option when it comes to the base simulator on which the agent editor module will be added. This simulator features a graphic editing interface, a 3D viewer, a simulation engine and a data analyzer. It uses OpenSteer[ope] as movement library and QT[qt] for interface. The ease of communication with the developer also favors the choice of this simulator. That being, the next chapter will focus on the implemented module, the agent editor module, and how it is integrated in the ModP simulator.

Chapter 5

The Agent Editor Module

This chapter will focus on the developed module that was added to the ModP simulator. The objective of this agent editor module is to allow the designers to simulate a broader range of crowds, creating the possibility to add new types of agents and configure their quantities across the simulation.

5.1 Introduction

One of the problems of realistically simulate a crowd is that the individuals that are part of the crowd at hand vary from country to country, zone, culture and the characteristics of the are itself as it can be a stadium, a shopping mall or a concert hall. Considering this problem, it is of great importance to add a dynamic agent editor to the simulator. This will allow a better characterization of the crowd considering the demographic information about the people that are expected to use the area or building.

This module allows the designers to create, configure and deploy an endless amount of agent types. Not only does it make the simulation more real, as human beings vary in behaviour and characteristics, as well as allows a precise configuration of the crowd just by inputing the percentages of each agent type that should exit each drain source.

5.2 Agent Parameters and Plan

In order to create an agent, the user must input the parameters that define that agent as well as its plan, i.e. what are the objectives of this agent.

The parameters that define an agent are its name, maximum steering force, maximum speed, comfort radius, mass and vision range. With all these parameters one can easily visualize what parameter values some types of agents will have. For example, an agent of the type "elderly" will have a low maximum speed and low vision range and, possibly a wider comfort radius. This will allow users to create whichever agents they find more suitable for the project at hand.

In addition to the parameters, each agent type created has a plan. This plan is defined as a series of commands that the agent will follow in order. The last command on the plan should always be the command exit, which symbolizes the point where the evacuation ends, i.e. the exit of the area or building where the agent is considered safe.

```
name:defaultAgent1
maxforce:300.4
maxspeed:1.5
radius:0.5
mass:1
visionrange:100
plan:find(10,-60);exit(10,-10)
```

In the snippet above it is possible to see all the agents parameters and plan. In the next subsection, the existing commands will be explained in detail.

5.2.1 Implemented Commands

The plan of each agent type works as a list of commands that should be followed in order by the agent, i.e. the agent can only start following command number two when he succeeds at command number one. In later developments, a timer should be added to each command in order to prevent impossible actions to impede the agent from successfully exit the building.

At the time there are only two commands that the user can add to the agent's plan. The main command is the exit command. This command tells the agent where the exit of the building or area is located. The syntax for the exit command is *exit(x coordinate, y coordinate)*. The other implemented command is the find command. This command can be used in a wide variety of situations. It can symbolize that the agent must pick up some documents at a determined point in the building before exiting or that it should push an alarm button somewhere in the area. The syntax for this command is *find(x coordinate, y*

coordinate).

Further commands that are relevant to the problem are the follow command to allow the creation of agents that behave without a concrete notion of the space around them or are unfamiliar with the building, therefore not knowing where the exit is. These agents could be children that need to follow their parents or even visitors in a museum that need to follow the guide. Another important command relevant to the simulation would be a simple wait command that instructed the agent to spend a predetermined amount of time in a specific spot.

```
plan:find(60,-60);find(10,-60);exit(10,-10)
```

The syntax to create a plan is very simple. Considering only the implemented commands, the plan should always end in an *exit* command and all commands should be separated by a semicolon, as shown in the snippet above.

5.3 Integration with Opensteer

When an agent is deployed in the simulation environment, it will already have all its parameters as well as its plan already prepared to be used. With the Opensteer library, all the information will be used in order to follow the plan. In this case, both the *exit* command and the *find* command, will use the *moveForSeek()* method. This method will receive the coordinates of the objective as well as the parameters of the agent such as maximum speed, comfort radius, etc. and will return a force to be applied by the agent in order to move in the direction of the objective. This force is then aggregated with the forces returned from the *moveToAvoidObstacles()* method in order to prevent agents from colliding with each other and with obstacles throughout the simulation.

The *moveForSeek()* method still needs improving when it comes to path finding as, for the moment, the agents are only aware of an obstacle when it enters its comfort radius, resulting in some unintelligent routes.

5.4 Integration with QT Interface

In order to allow the users to configure which agents should exit each drain source and in which quantities, a new tab was added to the drain source properties box in the graphic editor interface. This tab has a combo-box which displays all the available agents and an

input box that allows the user to enter the percentage of agents of a type that will exit that drain source.

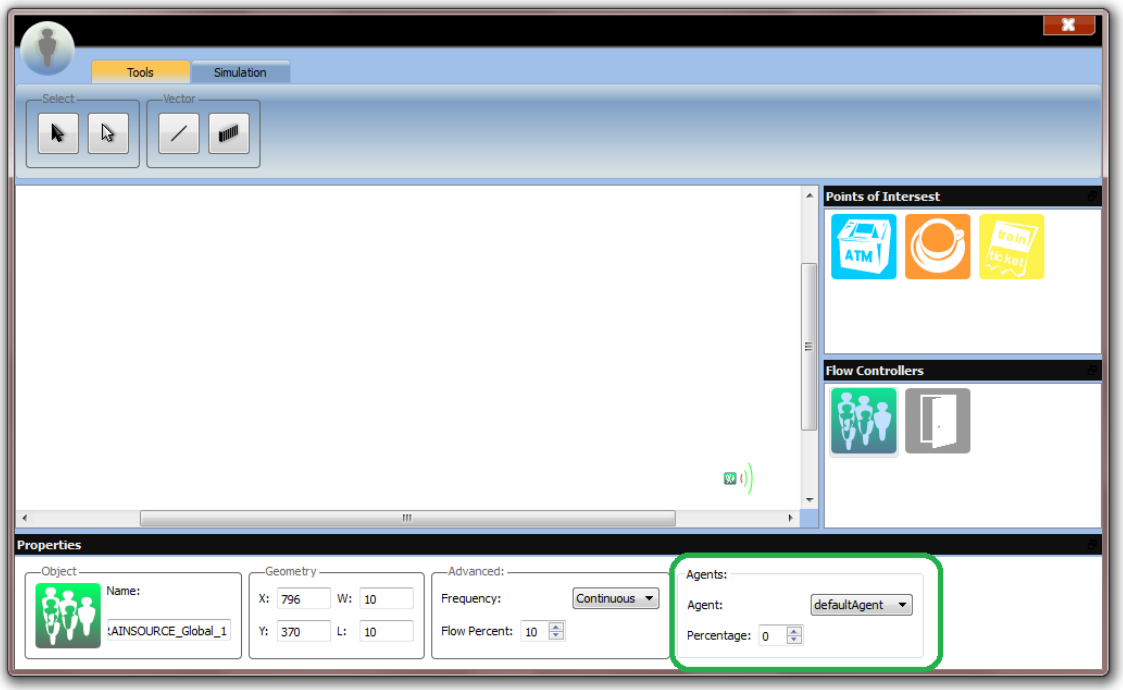

Figure 5.1: Agent and percentages tab

The changes made in the editor can be made while the simulation is running and will have real time results. Changing the percentage of a type of agent will update the drain source in the simulation.

5.5 Summary

As one can observe, the need to configure the agents that are present in the crowd to be simulated is of great importance. Therefore, allowing the building designers to easily create and configure different types of agents and their presence (in percentage) in the crowd is an important step towards a more realistic simulation.
Furthermore, with this tool, it is now possible to use collected demographic data to better model different crowds that are expected to use different environments and better predict their movement and behaviour.
Considering the implementation of this module, the next chapter will focus on the results obtained from its usage as well as the ease of usability of the syntax and configuration tab

on the simulator.

Chapter 6

Preliminary Results

6.1 Introduction

In this chapter, a batch of tests will be presented along with the results obtained from those tests. The objective of this tests is to assert the influence of each parameter of an agent has on the simulation and whats the outcome of a change in one of those parameters or in the percentage of a determined agent type.

For these tset, lets consider three types of agents, *child*, *adult* and *elderly* which have the configuration shown on the snippet below.

```
name:child
maxforce:250.0
maxspeed:1.0
radius:1.0
mass:0.5
visionrange:75
plan:exit(10,-10)

name:adult
maxforce:300.4
maxspeed:1.5
radius:0.5
mass:1
visionrange:100
plan:exit(10,-10)

name:elderly
```

```
maxforce:200.0
maxspeed:0.75
radius:1.0
mass:1
visionrange:60
plan:exit(10,-10)
```

As one can observe from the configurations displayed above, although the plan of each agent type is the same for the test purpose, they differ on a number of parameters. For instance, agents of the type *child* have half the mass of agents of the type *adult*, agents of the type *adult* have twice the maximum speed of agents of the type *elderly*. These values are not to scale, as they are only for test purposes. On a simulation of a building, demographic data would have to be collected in order to apply realistic values to these parameters.

6.2 Simulation Tests and Results

The aim of these tests, as explain before, is to better show how different agents with different configurations or even their percentages in the simulation can alter the outcome of the simulation. The charts in figures 6.2, 6.1 and 6.3 show the speed over time of each of the previously defined agent types when alone in the simulation, i.e. without any obstacles or any other agents, just following a straight line to their objective. The distance travelled by each agent is 1000 simulator units.

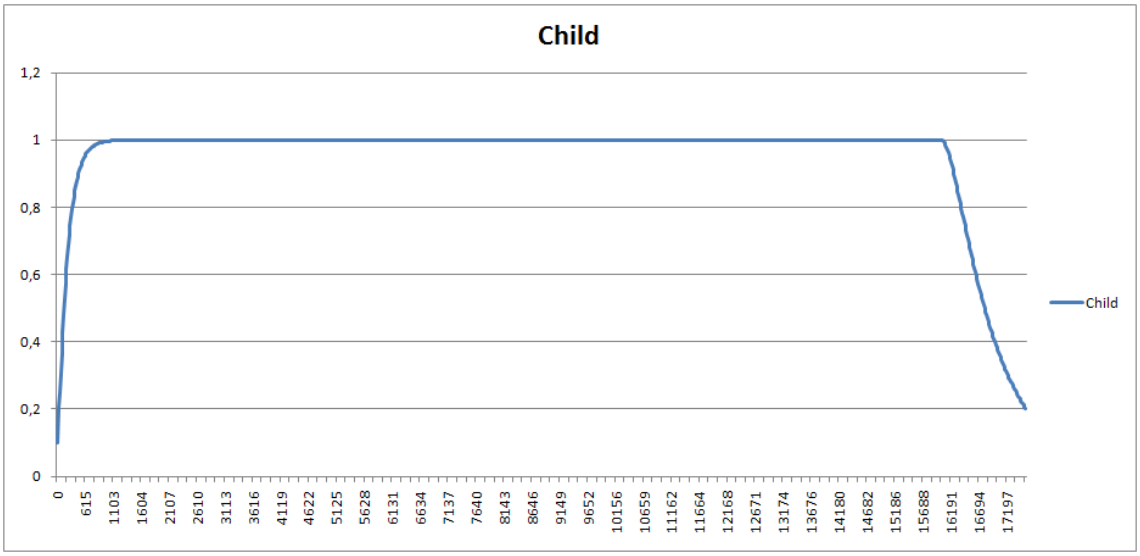

Figure 6.1: Child speed over time without obstacles

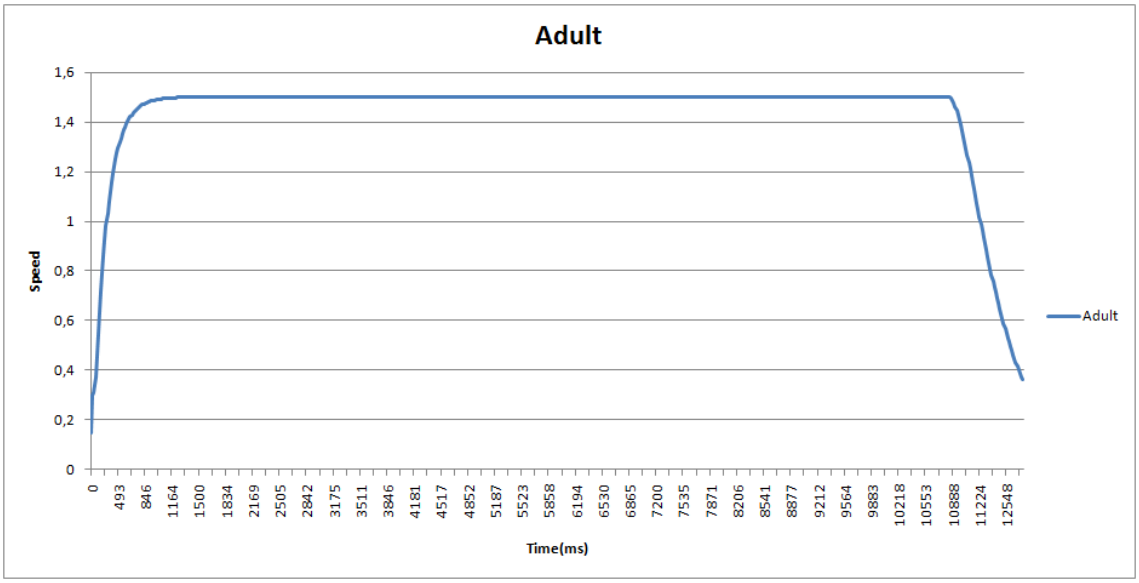

Figure 6.2: Adult speed over time without obstacles

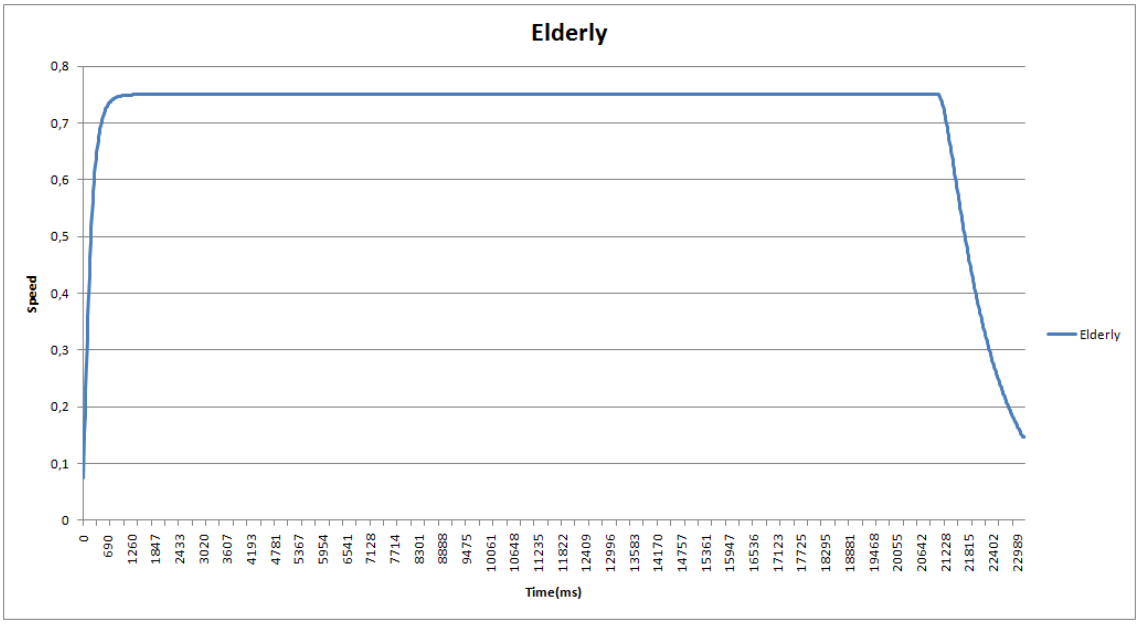

Figure 6.3: Elderly speed over time without obstacles

As one can observe, without any obstacle or neighbour agents impeding the movement of the agents, they accelerate until maximum speed is reached and then slow down to better reach their objective. This charts are presented as control data for the tests that will be presented later on this chapter.

The following test will feature 30 agents distributed amongst the three created types (33% for child, 33% for adult and 34% for elderly). These agents will spawn at the same point as the previous tests, 1000 simulation units away from their objective. The aim of this test is to evaluate the changes in speed of each agent type when in a more realistic environment with other agents and different types of agents. Th results arising from this test will allow a better understanding of the influence agents of different types have on each other.

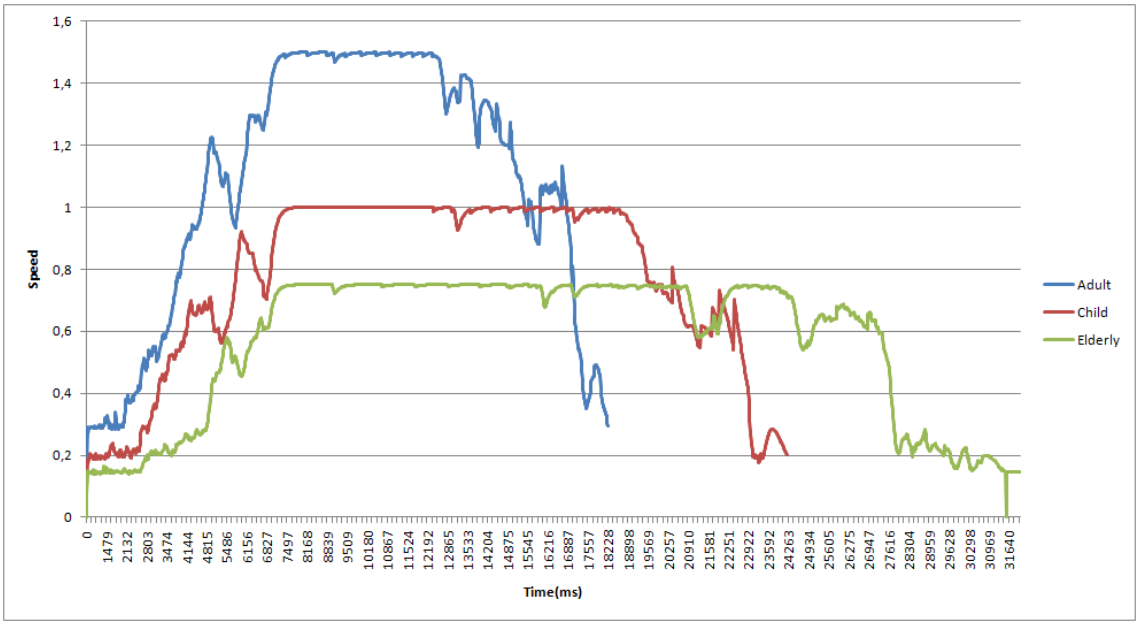

Figure 6.4: Speed over time of the three agent types (33%, 33% and 34%)

Evaluating the chart on figure 6.4 it is possible to observe the differences of speed between figures 6.2, 6.1 and 6.3 and the respective speed of each agent type when interacting with each other. Most times, faster, more agile agents (in this case, agents of the type adult) can some times be slowed down by other agent types. Another interesting information one can take from this chart, is when a number of agents come to close and start impede each other movement. Between 20300 milliseconds and 21600 milliseconds it is possible to observe one of these situations between agents of the types child and elderly.

This next test aims to show how a high amount of slow agents can induce slower movement on other agents. For this test, the same types of agents will be used, but their percentages in the simulation will be changed to 40% for child, 40% for elderly (which are slower agents) and 20% for adults.

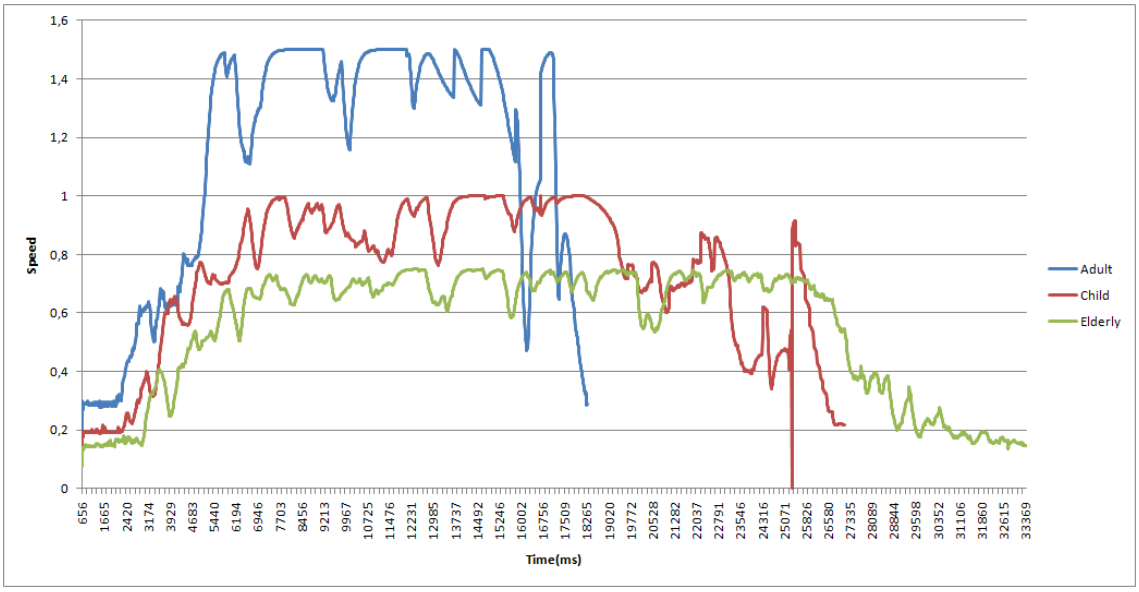

Figure 6.5: Speed over time of the three agent types (20%, 40% and 40%)

As one can see in figure 6.5, there is a big difference in the speed variation, when compared to the chart in figure 6.4. It is possible to observe more and deeper variations of speed. This happens because of the high amount of slow agents, impairing the movement of each other and other (faster) agents.

6.3 Summary

This chapter focused on testing and evaluating results from different scenarios where agents of different types interacted with each other. The results showed that, as expected, having different types of agents in a simulation will change its outcome. In this case, the two most important results to point are the overall changes in agent speed when there is a high amount of slower agents in the simulation as well as the changes in finishing time (in this case, evacuation time). In the next chapter, this results will be used, in addition to previous chapter in order to present conclusions and further developments to this tool.

Chapter 7

Conclusions

In this chapter, the final remarks for this document will be presented, taking into consideration the obtained results and the implemented agent editor module. There will also be two sections describing what further developments would improve the implemented module and the ModP simulator as well as future work on this area.

7.1 Final Remarks

Considering the problem stated at chapter 3 and the described implemented module in chapter 5 it is possible to analyze the results obtained from the previous chapter.

The importance of realistically simulating evacuation and emergency situations is the main focus of this project. Therefore, the aim of this project was to implement a module that would allow designers to create, edit and configure agent types that are relevant to the area that is intended to simulate, increasing its realism. The outcome of the addition of this module to the simulator has proved to be very positive, as shown on chapter 6, showing that the differences between parameters and plans of the different types of agents present in the simulation can alter significantly the speed and agility of each agent and, more importantly the evacuation time of a certain area.

With this tool it is in fact possible for designers to establish a more realistic simulation of an environment. With previously obtained demographic data, it is now possible to create and configure the crowd that will be present in the building or area resulting in better decisions while the project is being developed. This module can be considered one more

step towards a more accurate simulation but both the agent editor module and the simulator can still be improved. The next sections will focus on the possible improvements for the simulator and agent editor module.

While developing this project, a great amount of knowledge was gathered. The most important areas of study from which this knowledge was acquired were the familiarization with the ModP simulator and the way it handles agent interactions (attraction and repulsion forces), usage of QT interface and OpenSteer library and finally different approaches to define BDI agents.

7.2 Further Developments

Initially, this project included a greater amount of features to be added to the ModP simulator. These features proved to be to complex to implement in the time frame of this project, being that the reason why they were not referred nor implemented. From these features, the most important will be detailed in the following sections.

7.2.1 Agent Editor Module

When it comes to the agent editor module, in order to make it more dynamic and allow the creation of more types of agents, some commands should be added to the syntax. Although the *exit* and *find* can describe many real situations, they are not enough to describe all the possible behaviours the agents should be able to have. Some important commands to be added are, for example, the *follow* command to allow agents to follow some other agents. For instance, plans for child agents could include following adult agents and visitors at a museum could follow guides.

Another important improvement to the agent editor module would be to add a timer to each commands parameters in order to simulate the importance of that command. For example, instead of *find(x,y)*, the syntax would be *find(x,y,timer)*. This would allow the user to define the importance of each command, which can be interpreted as ïthe agent should only try to perform this action while it is safe to do so; when it becomes to dangerous to perform it (timer hits zero) the agent should proceed to the next action in the stack.:

7.2.2 DXF File Importer

The objective of this module was to allow user to automatically import CAD files into the simulation in order to prevent designing the building twice, once in the CAD software and a second time in the simulation editor. It possible to export projects from AutoCAD[Auta] to a readable file, DXF[Autb].Some DXF interpreter libraries were studied, such as Kabeja[kab] and have shown to be to complex to convert the DXF data to the model structure on the ModP simulator. Nevertheless, this is an important issue that should be addressed in future works on this simulator, as it is of vital importance from the designers point of view.

7.2.3 Data Gathering Module

This module would allow, through different tools, the gathering of data related to pedestrian movement and behaviour in order to refine the behaviour of agents in the simulator. This module was expected to gather information through two different approaches. The first one was to allow the introduction of avatars in the simulator. This avatars would be user controlled individuals that would respond to a persons input in the simulation. The objective of this approach was to assert whether the routes taken by the agents were compliant with the routes taken by the user controlled individual and with those results, refine the agents path finding algorithm. The other approach was to use ubiquitous computing with RFID to carry out real world evacuation simulations and collect the data from the RFID devices from each person in order to use that data to refine agents maximum speed and rotation force. The use of RFID to track people has already been successfully demonstrated in, for instance, hospitals to track patients [FL05]. This module would require a high amount time to implement, being that the reason why it was not studied in more detail.

7.3 Future Work

The added agent editor module increases the simulator's realism but there are many features to be added in order to use it as a reliable prediction tool. Two of the most important features to add are the ones described in the previous section, i.e. import DXF files and create a data gathering module. Other improvements would be related to the movement library, OpenSteer. The path finding algorithm should be improved in order to obtain a more reliable simulation. When it comes to the Agent Editor Module, future improvements considered were already referred in the previous section.

The approach taken in this project can also be applied not only in evacuation and emergency situations to other pedestrian movement situations. For example, predicting visitors flow in a museum or in a subway station. Summarizing, it can be applied to more normal situations as well, rather than evacuation and emergency situations.

References

[Auta] Autodesk. Autocad. http://usa.autodesk.com/adsk/servlet/pc/index?siteID=123112&id=13779270. Last accessed June 28th 2010.

[Autb] Autodesk. Dxf reference. http://usa.autodesk.com/adsk/servlet/item?siteID=123112&id=12272454&linkID=10809853. Last accessed June 28th 2010.

[BCN97] Eric Bouvier, Eyal Cohen, and Laurent Najman. From crowd simulation to airbag deployment: particle systems, a new paradigm of simulation. *Journal of Electronic Imaging*, 6(1):94–107, 1997.

[BHH+] Jens BLAUERT, Christian HEIPKE, Yves HENAFF, Elsabeth LIDDELL, Nadia MAGNENAT-THALMANN, Philippe NONIN, and Daniel THALMANN. Crosses : Crowd simulation system for emergency situations.

[BMOB03] Adriana Braun, Soraia R. Musse, Luiz P. L. de Oliveira, and Bardo E. J. Bodmann. Modeling individual behaviors in crowd simulation. In *CASA '03: Proceedings of the 16th International Conference on Computer Animation and Social Agents (CASA 2003)*, page 143, Washington, DC, USA, 2003. IEEE Computer Society.

[BWH07] Rafael H. Bordini, Michael Wooldridge, and Jomi Fred Hübner. *Programming Multi-Agent Systems in AgentSpeak using Jason (Wiley Series in Agent Technology)*. John Wiley & Sons, 2007.

[CCR09] Rose Challenger, Chris W. Clegg, and Mark A. Robinson. Understanding crowd behaviours: Supporting evidence, 2009.

[DFG09] Supervised by Dr Burkhard Wünsche Daniel Flower and Assoc-Prof Hans Werner Guesgen. Crowd simulation for emergency response planning. 2009.

[Est09] Edgar Ferreira Esteves. Utilização de agentes autónomos na simulação pedonal em interfaces multi-modais. Master's thesis, Faculdade de Engenharia da Universidade do Porto, 2009.

[FL05] Emory A. Fry and Leslie A. Lenert. Mascal: Rfid tracking of patients staff and equipment to enhance hospital response to mass casualty events. In *AMIA Annual Symposium Proceedings*, pages 261–265, 2005.

[Gro] G. S. Group. Opengl - the industry standard for high performance graphics. http://www.opengl.org/. Last accessed June 28th 2010.

[Gro10] Crowd Simulation Group. Corwd simulation group, 2010.

[HFMV02] D. Helbing, I. J. Farkas, P. Molnár, and T. Vicsek. Simulation of pedestrian crowds in normal and evacuation situations. In *Pedestrian and Evacuation Dynamics*, pages 21–58. 2002.

[HS04] K. J. Hoskin and M. Spearpoint. Crowd characteristics and egress at stadia. In T. J. Shields, editor, *Human Behaviour in Fire*, London, 2004. Intersience.

[Hub99] Marcus J. Huber. Jam: A bdi-theoretic mobile agent architecture. In *In Proceedings of the Third International Conference on Autonomous Agents (Agents'99*, pages 236–243. ACM Press, 1999.

[kab] Kabeja. http://kabeja.sourceforge.net/. Last accessed June 28th 2010.

[LRD07] R.G. Laycock, G.D.G. Ryder, and A.M. Day. Automatic generation, texturing and population of a reflective real-time urban environment. *Computers and Graphics*, 31(4):625 – 635, 2007.

[LZC+08] Linbo Luo, Suiping Zhou, Wentong Cai, Malcolm Yoke Hean Low, Feng Tian, Yongwei Wang, Xian Xiao, and Dan Chen. Agent-based human behavior modeling for crowd simulation. *Comput. Animat. Virtual Worlds*, 19(3-4):271–281, 2008.

[MB02] Rodrigo Machado and Rafael H. Bordini. Running agentspeak(l) agents on sim_agent. In *ATAL '01: Revised Papers from the 8th International Workshop on Intelligent Agents VIII*, pages 158–174, London, UK, 2002. Springer-Verlag.

[ope] Opensteer: Steering behaviors for autonomous characters. http://opensteer.sourceforge.net/. Last accessed June 28th 2010.

[qt] Qt - cross-platform application and ui framework. http://qt.nokia.com/. Last accessed June 28th 2010.

[Rao96] Anand S. Rao. Agentspeak(l): Bdi agents speak out in a logical computable language. pages 42–55. Springer-Verlag, 1996.

[RD06] G. Ryder and A. M. Day. High Quality Shadows for Real-Time Crowds . pages 37–41, undefined, 2006. Eurographics Association.

[SVLS06] Ameya Shendarkar, Karthik Vasudevan, Seungho Lee, and Young-Jun Son. Crowd simulation for emergency response using bdi agent based on virtual reality. In *WSC '06: Proceedings of the 38th conference on Winter simulation*, pages 545–553. Winter Simulation Conference, 2006.

[TB] Tim Thirion and Suddha Kalyan Basu. Real-time crowd simulation for emergency planning. http://www.cs.unc.edu/ skbasu/multirobot/PAPER/evacuation.pdf.