BENOIT LAROCHELLE

Multi-Agent Geo-Simulation of Crowds and Control Forces in Conflict Situations

Models, Application, and Analysis

Mémoire présenté
à la Faculté des études supérieures de l'Université Laval
dans le cadre du programme de maîtrise en informatique
pour l'obtention du grade de Maître ès sciences (M.Sc.)

FACULTÉ DES SCIENCES ET DE GÉNIE
UNIVERSITÉ LAVAL
QUÉBEC

2009

Abstract

Few models and simulations that describe crowd behaviour in conflict situations involving control forces and non-lethal weapons (NLW) exist. This thesis presents models for crowd agents, control forces, and NLWs in crowd control situations. Groups as well as their interactions and collective actions are explicitly modelled, which pushes further currently existing crowd simulation approaches. Agents are characterized by appreciation of aggressiveness profiles and they can change their behaviours in relation with the Social Identity theory. A software application was developed and the models were calibrated with realistic scenarios. It demonstrated the technical feasibility of such complex social models for crowds of hundreds of agents, as well generating data to assess the efficiency of intervention techniques.

Résumé

Peu de modèles et de simulations qui décrivent les comportements de foule en situations de conflit impliquant des forces de l'ordre et des armes non-létales (NLW) existent. Ce mémoire présente des modèles d'agents de la foule et des forces de l'ordre ainsi que des NLWs dans des situations de conflit. Des groupes ainsi que leurs interactions et actions collectives sont explicitement modélisés, ce qui repousse les approches de simulation de foule existantes. Les agents sont caractérisés par des profils d'appréciation de l'agressivité et ils peuvent changer leurs comportements en relation avec la Théorie de l'identité sociale. Un logiciel a été développé et les modèles ont été calibrés avec des scénarios réalistes. Il a démontré la faisabilité technique de modèles sociaux aussi complexes pour des foules de centaines d'agents, en plus de générer des données pour évaluer l'efficacité des techniques d'intervention.

Acknowledgements

This thesis is the culmination of ten years of post-secondary studies. It is with great pleasure and pride that I have written this work, and I must acknowledge the constant support of those around me.

First of all, I would like to thank my research director, Bernard Moulin, for offering me some of the greatest opportunities that a student could receive. Bernard has welcomed me to his team, given me interesting and challenging work, and trusted me with key tasks and milestones in the project. Bernard has given me great freedom in my research and his wisdom has provided me with an environment in which I could abundantly flourish. I highly appreciated that Bernard was very meticulous and this trait made working with him a great pleasure.

I would like to thank the DRDC team for providing me with an environment in which I felt highly valued. I am immeasurably thankful to Luminita Stemate for always being so patient, understanding, and open-minded about research ideas. Also, I would like to express gratitude to Guillaume Gagné, who tremendously helped me with testing, calibrating, and experimenting with the system. He also was very patient when working with early versions of the software.

I also must thank Ghina Besbes, friend and colleague, for her warm-hearted support over the past two years. She has always been present for me, offering her support through all steps of my personal and professional life since I have known her. Ghina also provided me with endless motivation in school, especially for the writing of this thesis.

Finally, I would like to thank all of my friends, for encouraging me, helping me, guiding me, — and keeping me sane — during the past two years. I could never mention every person who has had a positive impact on me, but I am mostly grateful to: Caroline, Denis, Jean-François, Jean-Philippe, Jean-Sébastien, Marianne, Sahil, Tania, and Véronique.

Without all of your support and encouragement, this thesis might have ended here.

To Becky,
who was the first one to believe
that I could get this far

Contents

List of Tables

List of Figures

List of Abbreviations

Abbreviation	**Signification**
2D	Two-Dimensional
3D	Three-Dimensional
ASI	Adopted Social Identity
BDL	Behaviour Description Language
CF	Control Forces
DRDC	Defence Research & Development Canada
FSI	Fundamental Social Identity
GIS	Geographic Information System
GRIC	Groupe de recherche en informatique cognitive
GUI	Graphical User Interface
MABS	Multi-Agent-Based Simulation
MAGS	Multi-Agent Geo-Simulation
PLAMAGS	Programming LAnguage for Multi-Agent Geo-Simulation
SD	System Dynamics
SI	Social Identity
STG	Spatio-Temporal Group
VGE	Virtual Graphic Environment

Chapter 1

Introduction

Military forces are faced with an increasing number of crowd control situations and they are turning to computer simulation to help them improve their intervention techniques. Such is this thesis' research context, which is detailed in Section 1.1. Next, the motivation for the thesis is discussed in Section 1.2, followed by detailed objectives in Section 1.3. Afterwards, the research methodology is presented in Section 1.4. Finally, the organization of the thesis is presented in Section 1.5.

1.1 Research Context

This thesis was written following a 3-year research project with Defence Research and Development Canada (DRDC) Valcartier and the Cognitive Informatics Laboratory (Groupe de recherche en informatique cognitive — GRIC) at Université Laval. DRDC is an agency of the Canadian Department of National Defence that provides the scientific and technological expertise needed by the Canadian Forces. DRDC has an annual budget of $300 million and employs over 1600 people in seven research centres located across Canada, including the one in Valcartier (near Québec City). To fulfill its broad scientific program, DRDC regularly collaborates with industry and academia, such as in the project presented here. The GRIC is a research group led by Dr. Bernard Moulin and composed of post-doctoral fellows, doctoral- and master's-level students, and research professionals. This group actively collaborates with industry to integrate its developments into practical solutions. More information about the two project's partners is available at www.drdc-rddc.gc.ca and www.ift.ulaval.ca/~moulin. A description of the project (Crowd-MAGS), copied directly from (Stemate, 2006), is presented below.

> Models and simulations that describe crowd behaviour in conflict situations involving control forces (military or police) do not exist at the present time in a form that could be used to assess the impact of Non-Lethal Weapons (NLW) on crowd dynamics and the resolution of conflict. The aim of this project is to

> provide such a capability, with a view of reaching two main objectives. First, to make available a platform than can be used to assess the effectiveness of various types of NLW in situations of crowd control. Second, to improve warfighter's training by providing the most recent and reliable data on human behaviour in urban conflict situations. Several facts motivate the undertaking of this project. Among them, the expected increase in the number of crowd control situations in which the control forces (CF) troops may be involved in the future, due to the fact that future operations are expected to occur more often in urban environments, involving a mix of military and civilian personnel (in the context of the "three block war"). In addition, there is a widely recognized difficulty in evaluating the effectiveness of NLW. Therefore, there is a need for effective modelling and simulation tools to address this issue.

The simulation paradigm that was selected by the GRIC team, which includes the author of this thesis, is multi-agent geo-simulation (MAGS). Multi-agent systems are typically decentralized in terms of data and processing. Autonomous entities, called agents, are the basic components in such systems and interact to varying degrees with one another to solve a problem that could not be easily dealt with by traditional methods. Multi-agent simulation is a sub-field of multi-agent systems that aims at simulating events that are inherently composed of multiple entities, such as studying the flow of packets on a network or exploring the spread of a virus. In multi-agent geo-simulation, such as crowd simulation, the simulated events happen in virtual environments that are spatially referenced using real coordinates like geographic information systems (GIS) (Moulin et al., 2003). Crowd simulation deals primarily with agents that represent humans, commonly called virtual human agents. Crowd simulations must reproduce human behaviour and most projects have placed the agents in constrained environments, such as airplanes for assessing evacuation efficiency and urban environments for evaluating crowd movements. More details are presented in Chapter 2.[1]

1.2 Motivation

The field of crowd simulation is fairly recent and although theoretical agent models are numerous, none of them provides an explicit modelling of emerging groups, which are created, modified and disbanded throughout crowd events (Moulin, 2009). In addition, no model exists for agents that are able to plausibly simulate perception and interpretation of behaviours of other agents and groups and to plausibly change their social identities. Without these models, any simulation would not be highly plausible because the social aspects observed in real crowds could not be simulated. In consequence, it was necessary to extend the available agent and group models to simulate more plausible crowd behaviours (Moulin, 2009).

Another point that was noticed is that although existing simulation platforms provide several of the elements necessary for the Crowd-MAGS project's objectives, none of them

[1]N.B.: Since this thesis deals with crowd simulation, the term *agent* will be used exclusively in the sense of *virtual human agent*. Also, human pronouns (*he*, *him*, *his*) will be preferred to non-human pronouns (*it*, *its*) when referring to these agents. The masculine form will be used (*his* rather than *her*) to alleviate the text although the agents are gender-neutral.

provides all elements at once. For example, some solutions simulate crowds but not control forces; others simulate weapons but not their psychological and emotional impacts on crowd members.

Finally, it was observed that no or little output data is generated for analysis from current crowd simulation platforms. This drawback causes a serious problem for the Crowd-MAGS project because the efficiency of non-lethal weapons must be backed by solid quantitative and qualitative analysis and not simply the observation of simulation events by domain experts. Thus, this need motivated the development of a platform that would allow the collection and analysis of large and varied data about simulation events.

Chapter 2 provides a detailed overview of existing crowd simulation models and related systems. After that chapter, it will become clear to the reader that the main motivation of this thesis is the development of a simulation platform that integrates all of the necessary concepts so that an operational software can be developed to help control forces.

1.3 Objectives

The overall goal of the Crowd-MAGS project was the development of a simulation application that could assess the efficiency of non-lethal weapons in crowd control situations. This goal was to be achieved using different methodologies (system dynamics and multi-agent geo-simulation), but only a subset of the project's objectives was selected to be discussed in this thesis.

First of all, some existing simulation tools allow modelling groups of agents but they are limited since these groups must be predefined and cannot evolve (accept new members, change vocation, disband, etc.). Thus, the first objective was the elaboration of a model for explicitly modelling emerging groups. In hand with this objective was the necessity of an agent model that would interact plausibly with these emerging groups.

Next, these models needed to be implemented in a software platform that would allow simulating them with control forces, crowds, and non-lethal weapons. No such platform was available so the second objective was to extend an existing one or to develop a new one that would include all necessary concepts.

In addition, an information model was necessary to collect data during simulations so that the efficiency of intervention strategies and non-lethal weapons could be assessed. Thus, the third objective was the creation of a model that would not only fill this need but also be generic enough to generate data for needs that were not yet clearly identified, such as integrating the platform with another simulation system.

Finally, the simulation platform had to be calibrated so that it could be readily used by end-users without technical knowledge, such as military personnel. The platform also needed a scenario specification tool so that these users could use the system without constant interaction with the platform developers. Thus, the fourth and final objective was the task of producing an operational, well calibrated, easy-to-use application following the completion of the first three objectives.

1.4 Research Methodology

The methodology used in the Crowd-MAGS project is described below and presented in Figure 1.1.

The first step was a literature review that focused on 1) agent models and 2) simulation platforms. From the information gathered, the strong and weak points of every model and platform were identified and a subset of candidate platforms was identified. The PLAMAGS platform was selected because it offered the most advanced behaviour engine for the project's needs.

Next, it was necessary to decide what types of events and actions were to be simulated. A broad search for crowd control videos was undertaken and a corpus of frequent situations, actions, and objects observed in the videos was built up. All videos were documented using fiches to be used in the validation phase. Only crowd control events in urban environments were to be simulated at first, and only a subset of the observed actions were to be implemented. This choice guided the following development of the models and the platform.

With a clear scope on the events to simulate, users' needs with respect to the agents and the emerging groups were evaluated. Then, the main components of the models (perception, memory, formation, etc.) were created (Moulin, 2009; Paris et al., 2008). Both the agent model and the group model were developed in parallel to be as cohesive as possible.

The following step was the technical development of the software tool, called Crowd-MAGS, that uses the PLAMAGS platform as its underlying simulation engine. First, extensions for the physical simulation of non-lethal weapons, such as tear gas and plastic bullets, were built. Next, extensions for the social simulation of human agents were developed, for example the ability for agents to change behaviours as simulated events progress. In addition, the agent and group models were integrated and other required components were created for Crowd-MAGS to be functional.

Once the platform was ready, a list of scenarios to be simulated was elaborated. Since many of these scenarios involved a high level of complexity, basic scenarios were first created to test all models and components individually.

Next, social identities (SI) were identified to represent different types of crowd participants. Behaviours were designed for each social identity. Each behaviour was then implemented in Crowd-MAGS and tested individually with one of the basic scenarios. Afterwards, control forces and their behaviours were created, as well as components related to crowd control, non-lethal weapons and the management of health and harm.

Next, a reflection of what could be the future needs for simulation data analysis was carried out because the users' needs were still ill-defined. A generic information model was devised to start collecting simulation data and prepare for future extensions. This model was tested by generating data for a system dynamics (SD) system that modelled similar crowd control situations.

Finally, the last step was composed of calibration and experimentation tasks. First, a methodology was established, in which incrementally complex scenarios are run while adjusting system parameters. Next, a model for specifying scenarios was created and used promptly. The user interface of the system was improved so that end users could more easily use the system (to help them during calibration and during usage of the system for their own needs). Once the system was stable, scenarios were created and run according to the calibration methodology resulting in an operational tool. Experimentation with more specific and more complex scenarios was performed so that the system could eventually be validated against the videos collected earlier in the project.

1.5 Thesis Organization

The rest of this thesis is organized as follows.

Chapter 2 aims to introduce the reader to the key concepts of the crowd simulation domain. The chapter also presents a literature review of crowd simulation models and systems.

Chapter 3 shows an overview of the Crowd-MAGS models and software in preparation for the detailed descriptions of the following two chapters.

Chapter 4 discusses the various models that were developed, placing the emphasis on the agent model, the group model, and the information model. In addition, generic agents behaviours are presented.

Chapter 5 presents the system that was developed, including the implementations of the models, mechanisms, and user interface.

Chapter 6 contains details about the calibration procedure that were undertaken and the results of various experimentations with the system.

Chapter 7 concludes this thesis by summarizing the theoretical and the practical results as well as by discussing future works.

Finally, three appendices are included to provide more technical details about the system. Appendix A presents extra information that was not crucial in the main chapters. Next, Appendix B shows the exhaustive behaviour specifications for all types of agents. Finally, Appendix C presents performance tests for individual components as well as for typical simulation scenarios.

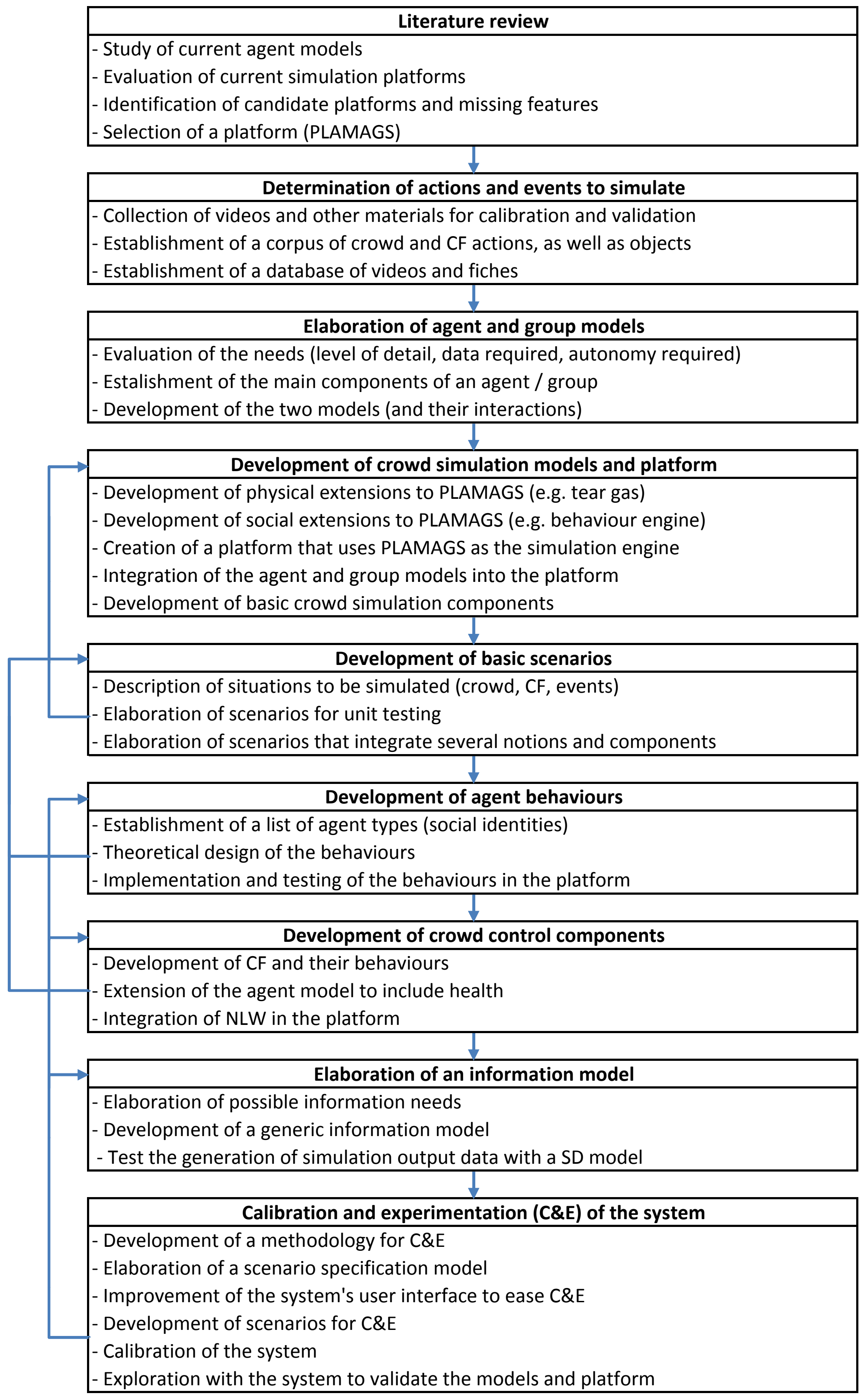

Literature review
- Study of current agent models
- Evaluation of current simulation platforms
- Identification of candidate platforms and missing features
- Selection of a platform (PLAMAGS)
Determination of actions and events to simulate
- Collection of videos and other materials for calibration and validation
- Establishment of a corpus of crowd and CF actions, as well as objects
- Establishment of a database of videos and fiches
Elaboration of agent and group models
- Evaluation of the needs (level of detail, data required, autonomy required)
- Estalishment of the main components of an agent / group
- Development of the two models (and their interactions)
Development of crowd simulation models and platform
- Development of physical extensions to PLAMAGS (e.g. tear gas)
- Development of social extensions to PLAMAGS (e.g. behaviour engine)
- Creation of a platform that uses PLAMAGS as the simulation engine
- Integration of the agent and group models into the platform
- Development of basic crowd simulation components
Development of basic scenarios
- Description of situations to be simulated (crowd, CF, events)
- Elaboration of scenarios for unit testing
- Elaboration of scenarios that integrate several notions and components
Development of agent behaviours
- Establishment of a list of agent types (social identities)
- Theoretical design of the behaviours
- Implementation and testing of the behaviours in the platform
Development of crowd control components
- Development of CF and their behaviours
- Extension of the agent model to include health
- Integration of NLW in the platform
Elaboration of an information model
- Elaboration of possible information needs
- Development of a generic information model
- Test the generation of simulation output data with a SD model
Calibration and experimentation (C&E) of the system
- Development of a methodology for C&E
- Elaboration of a scenario specification model
- Improvement of the system's user interface to ease C&E
- Development of scenarios for C&E
- Calibration of the system
- Exploration with the system to validate the models and platform

Figure 1.1: Crowd-MAGS' Research Methodology

Chapter 2

Literature Review: Crowd Simulation Models and Platforms

Crowd simulation is a relatively recent field of research and this thesis would be incomplete without an in-depth introduction to the field. First, Section 2.1 presents a simple but broad introduction to computer simulation in general. Section 2.2 is more advanced and presents the computer simulation's sub-fields that are related to crowd simulation. Next, Section 2.3 introduces the most complete and interesting agent models developed so far. Section 2.4 is similar but discusses models of groups and crowds. Next, Section 2.5 presents models of control forces. Afterwards, Section 2.6 discusses various other models that are related to crowd simulation and that could be useful for the Crowd-MAGS project. Then, Section 2.7 explores the existing crowd simulation platforms that could be used directly or extended. Finally, Section 2.8 sums up the literature review by discussing the interesting elements of the models as well as the elements that are missing or require further research.

The effort of Moulin's team must be acknowledged for (Moulin et al., 2007). In fact, Sections 2.3 through 2.8 are heavily based on Moulin's literature review.

2.1 Introduction to Computer Simulation

Computer simulation is an umbrella term that encloses several sub-fields, including crowd simulation. This section provides an overview of computer simulation and provides useful references but does not describe simulation in detail. The advanced reader may skip this section and start directly with Section 2.2.

A short but rich definition of computer simulation is "the technique of imitating, on digital computer, the behavior of some situation or system (economic, mechanical, etc.) by means of analogous models, situation, or apparatus, either to gain information more conveniently or to train personnel." (Ali, 2006) This interpretation is broken down in the following paragraphs in order to be fully appreciated.

Motivation

"Simulations are often portrayed as solutions to out-of-reach problems" (Lenhard et al., 2006), but why are so many problems out-of-reach? (Ali, 2006) states a number of reasons that explain why computer simulations are often necessary. These reasons, which are reproduced below, are based on a 30-year old paper from Brian R. Gaines but they are still completely relevant today.

- *The physical system is not available:* Often, computer simulations are used to determine whether or not a projected system should ever be built; so obviously, experimentation is out of the question. This is common practice for engineering systems (e.g., an electrical circuit) with well established and widely applicable meta-knowledge. It is rather risky to rely on such a decision in the case of systems from soft science (the so-called ill-defined systems) since the meta-knowledge available for these types of systems is usually not validated.
- *The experiment may be dangerous*: Often, simulations are performed in order to find out whether the real experiment might 'blow-up', placing the experimenter and/or the equipment under danger of injury/damage or death/destruction (for example, an atomic reactor, or an aircraft flown by an inexperienced person for training purposes).
- *The cost of experimentation is too high*: Often, simulations are used when real experiments are too expensive. The necessary measurement tools may not be available, or are too expensive to buy. It is possible that the system is used all the time, and taking it 'off-line' would involve unacceptable cost.
- *The time constants of the system are not compatible with those of the experimenter*: Often, simulations are performed because the real experiment executes so quickly that it can hardly be observed (for example, an explosion), or because the real experiment executes so slowly that the experimenter is long dead before the experiment is completed. Simulations allow us to speed up or slow down experiments at will.
- *Control variables, and/or system parameters may be inaccessible*: Often, computer simulations are performed because they allow us to access all input (variables), whereas, in the real system, some inputs may not be accessible for manipulation.

(Sulistio et al., 2004) adds that "simulation tools support the creation of repeatable and controllable environments for feasibility study and performance evaluation. These simulation environments facilitate researchers, educators and students to conduct effective research, teaching and learning with ease."

Objectives

The above definition mentions that the objectives of computer simulation are to "gain information more conveniently or to train personnel". These two objectives are the most common, with examples such as (Daiger, 2005), (Aggarwal et al., 2006), (Bürki-Cohen et al., 2003) and (Vincenzi and Wise, 2008). Another goal of simulation is forecasting, which is illustrated in (Franses, 1998) and (Chevillona and Hendry, 2005) for economics and (JAMSEC, 2009), (Desgagné et al., 2006) and (Lin et al., 2004) for weather.

Lately, computer simulations have also been used by the entertainment industry to produce realistic movements and behaviours of animated characters and crowds in movies and games. These commercial products are all considered to be simulators: the 'MASSIVE' simulation and visualization system for crowds, (`http://www.massivesoftware.com/whatismassive/`), the 'Gran Turismo' car racing simulator (`http://www.gran-turismo.com/`), the 'SimCity' city management simulator (`http://simcitysocieties.ea.com/index.php`, (Adams, 1998)), and the 'The Sims' family simulator (`http://thesims.ea.com/`).

Concepts

According to the definition above, the means used to achieve the computer simulation's goals are "analogous models, situation, or apparatus". The "analogous models" refer to the theoretical models that are implemented in the system. For a simulation of an electrical circuit, the models would represent how energy interacts with batteries, resistors, switches, etc. For an economical simulation, the models would explain how a population would react to given monetary policies. In a social simulation, the models would try to reproduce human reasoning and behaviours. The "situations" are usually represented by scenarios, or scripts, that describe what conditions or events the modellers try to reproduce. For example, flight pilot trainers might want to reproduce a situation of high stress due to engine failure. Thus, they could create various scenarios where the engines fail in different manners and conditions. The "apparatus" commonly refers to the software application that takes the models and scenarios as inputs and performs the simulation, whether is consists of driving a physical system, showing animations on a computer screen or generating output data.

Techniques

The above definition mentions that computer simulation is a "technique of imitating"; however, other techniques to achieve the same goals exist. For example, instead of imitating it, one could forecast the behaviour of a system through any means, such empirical data or mathematical models. Another technique would be duplicating the system and observing it, instead of imitating it. Computer simulation is most useful when these other techniques are not feasible (Shi and Brooks, 2007).

Several techniques of simulation have emerged in the past half-century as more and more advanced techniques are required to simulate the more complex domains. (Klügl, 2004) argues that two main types of simulation exist: macro-simulation simulates a complete system perspective while micro-simulation simulates smaller entities with distinct states and behaviours. In fact, other properties could be used to classify simulations, such as those in (Sulistio et al., 2004). For example, a simulation may or may not include the concept of time (dynamic vs.

static). The modelled entities may be allowed to take any value/state from a finite set (discrete) or within any range (continuous). Also, the simulation could be deterministic, where no random event or uncontrollable element is possible, or stochastic, where different simulation runs may yield different results. Based on these criteria and several others, different simulation techniques can now be identified; Table 2.1 lists a few of the common ones.

Table 2.1: Some Techniques of Computer Simulation

Technique	**Description**
Petri Networks	Graphical and mathematical modelling language to simulate the dynamics of systems
Queuing Networks	A system represented by a network of queues (service centers and customers) that is evaluated analytically
System Dynamics	Internal feedback loops and and stocks and flows help in understanding the behaviour of dynamic systems from a macro point of view
Particle Systems	System of small particles and entities that apply forces onto the particles to simulate physical phenomena
Cellular Automata	Discrete model that consists of a regular grid of cells, each with a neighborhood and a finite number of states
Multi-Agent	Multiple agents represent the simulated reality with specific behaviours

Only a few of the computer simulation's sub-fields were presented here. However, these papers (Ali, 2006; Maier and Größler, 2000; Davidsson, 2000) provide a more detailed summary of the various techniques.

Limitations

Although the field of computer simulation has shown tremendous advances in the 1990's and 2000's, limitations still exist. One of the obvious inconveniences of simulation is that any time and money invested in the simulation is not invested in the real system. For example, spending 100 000 $ on the simulation of an urban intersection means that this money can no longer be invested in the road infrastructure. This situation may be acceptable because the simulation may bring significant savings and may reduce the possibility of problems with the system (e.g. traffic jams in this case). A balancing act must be performed to distribute funds between the system and its simulation.

Another difficulty is the complexity of creating the models. In some cases, the models are mathematical formulas representing physical laws such as gravity. These can be simple, but the users' needs may warrant a higher level of realism, for example removing the assumption of constant gravity around the globe. Such needs make modelling more complex. Other types of models, such as the ones used to simulate social systems, can be much more complex. For example, representing how a human brain works on a computer is currently infeasible and only selected parts are simulated in each application, such as the memory or the path planning reasoning. (Sulistio et al., 2004) even claims that "the design and development costs for complex simulation models are sometimes comparable to the costs of building the actual systems. In addition, not all researchers are experts in simulation, thus they have problems creating simulation models successfully."

Next, performance considerations can be a limitation to simulation platforms. For example, multi-agent systems with complex agents slow down as soon as the number of agents grows, up to a point where the system is not practical anymore. This drawback forces users to design scenarios with lower complexity, such as reducing the number of agents.

Finally, the processes of validation, verification, and calibration can make simulation appear like a less interesting choice (Klügl, 2004). Indeed, simulations are sometimes used because real experimentation is not possible. In many cases, simulations must be based on empirical data and/or predictions from domain experts. Unfortunately, such data are often unavailable (e.g. if real experimentation is impossible), leaving the simulations uncalibrated and/or unvalidated.

It is important to keep in mind that models are abstractions of reality, which leave out certain details for various reasons. Thus, it is generally infeasible to reproduce *all* eventualities that could happen in a system. This drawback is illustrated in a humorous manner in the movie Demolition Man[1] that takes place in 2032. When Los Angeles' most dangerous and ingenious criminal escapes from prison, the police department underreacts because their computer simulation has analyzed *all* possible scenarios involving the fugitive. The movie shows that unexpected thinking from the criminal leads to scenarios that are far from anything that the simulation envisaged.

History

Like many scientific fields, computer simulation originates from a World War II military project (Lenhard et al., 2006). The Manhattan project started around 1939 and used Monte-Carlo methods to simulate physical systems involving nuclear weapons.[2] Several years later, in 1955, the first civil computer simulation was the Fermi-Pasta-Ulam experiment. Originating from the Los Alamos National Laboratory like the Manhattan project, this experiment was conducted by the three scientists Fermi, Pasta, and Ulam (who also worked on the Manhattan project) and its goal was to better the understanding of the thermalization process of a solid, which is a physical process related to energy equilibrium.

Nowadays, some form of computer simulation is used in virtually every domain, such as aerospace (Bürki-Cohen et al., 2003; Vincenzi and Wise, 2008), military (Manojlovich et al., 2003), public safety (Santos and Aguirre, 2004; Murakami et al., 2002), medicine (Daiger, 2005; Aggarwal et al., 2006), economics (Franses, 1998; Chevillona and Hendry, 2005), earth

[1]Demolition Man came out in 1993 and starred Sylvester Stallone, Wesley Snipes, and Sandra Bullock. `http://www.imdb.com/title/tt0106697/`

[2]Interesting fact: (Masco, 2006) talks about "one of the major achievements of post-Cold War weapons science: the completion of the first three-dimensional computer simulation of a thermonuclear detonation in 2001. [...] The simulation ran for 122.5 days [...] it would take a state-of-the-art home computer 750 years to complete the calculation". This project is a good example that the military sector has not only given birth the field but is also pushing its limits further and further.

sciences (Desgagné et al., 2006; Lin et al., 2004), and many more. “In fact, virtually all areas where humans pursue understanding can and will benefit from the dynamic and malleable representations that simulation renders.” (Vincenzi and Wise, 2008)

Methodology

Due to wide variety of simulation projects, it is difficult to establish a ‘standard’ methodology that is commonly followed. Certainly, some important steps are an integral part of many projects: requirements elicitation (in particular determining the situations or scenarios to be simulated), elaboration of the models, implementation and execution of a prototype, analysis of the results, calibration of the parameters, and verification and validation. Moreover, the steps involved in a project are often performed in several iterations. In other words, once initial results have been analyzed, users’ needs can be refined, models can be improved, other simulations can be run, and results can be analyzed further. (Klügl, 2004; Min et al., 2007; Sargent, 2007)

An effort has been made in Figure 2.1 to present a ‘common’ methodology, although it is not claimed to be the ‘standard’ methodology in simulation projects.

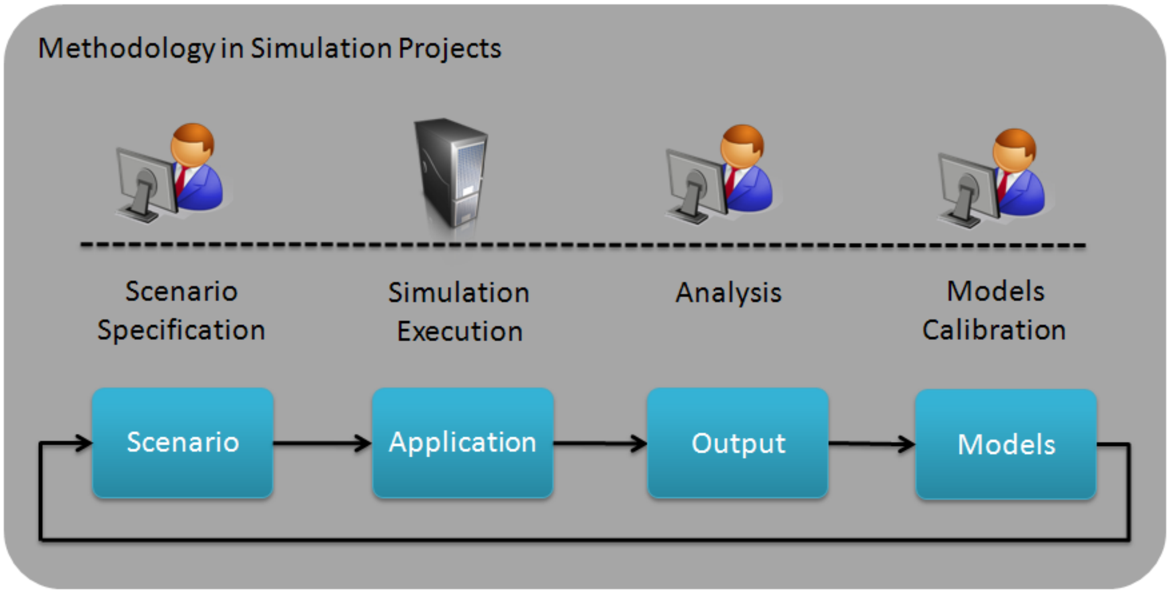


Figure 2.1: Methodology in Simulation Projects

The first phase consists of specifying the scenarios, which define what events will happen during the simulation. Next, the scenarios are run on a simulation application. Depending on the modelling paradigm and the application, it may be possible to watch the simulation as it runs. In the third phase, analysis, the simulation output data is usually much more important than the visualization aspect. Macro-level simulations tend to give aggregated information while micro-simulations provide much more detailed information, which can be more difficult to analyze. Finally, the models are calibrated after the analysis is performed.

Once the calibration seems acceptable, the cycle starts over by running the scenarios again, which could have been modified.

Complete simulators provide tools for all phases, but the minimal core is only the execution part. Scenarios can be edited by hand or using a scenario editor. Output data can be analyzed directly in the application, in a third-party analysis tool, or by hand. The models can be calibrated manually or directly in the application.

2.2 Origins of Crowd Simulation

This section describes a series of domains from which the recent field of crowd simulation derives. Figure 2.2 shows a graphical summary of how these domains are related.

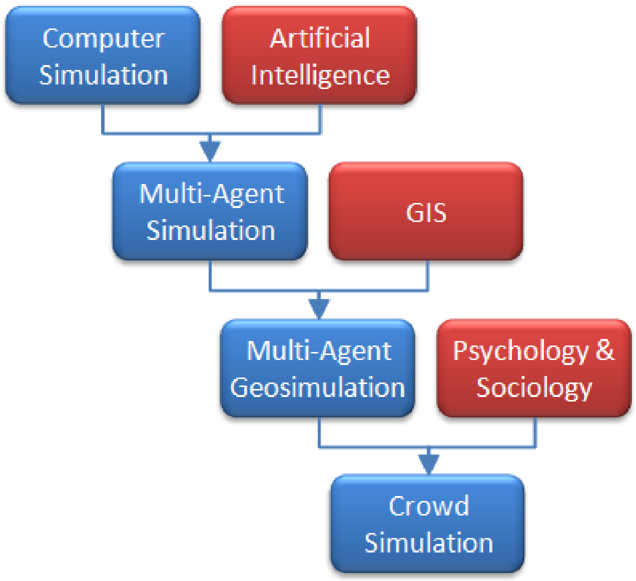


Figure 2.2: Crowd Simulation Origins

2.2.1 Multi-Agent Simulation

Multi-agent simulation, or multi-agent-based simulation (MABS), is one of the techniques of computer simulation that were introduced in Section 2.1. As its name indicates, this technique is merely a computer simulation that uses multiple agents as its architectural framework. Before going in more details, the concept of agent[3] and multi-agent systems in general should be introduced.

[3]In this sub-section only, the term 'agent' is general and does not mean 'virtual human agent' like in the rest of the thesis

What is an 'Agent'?

The concept of an agent was introduced in the early 1990's but it still does not hold a universal definition (Jennings et al., 1998; Shi and Brooks, 2007). However, one could argue that the following minimalist description is well accepted: an agent is an autonomous entity that, based on its goals and environment, makes decisions that could affect itself and its environment.

No clear definition exists because the types of agents that have been created in various domains are too varied. Some agents (*reactive*) only react to stimuli from their environments; others (*cognitive*) plan their actions. Some agents have full knowledge of the environment; some have only a partial view. Some agents always act in the same manner; others learn and can change their behaviours. Some agents work alone; others work with other agents — in collaboration or competition. (Russell and Norvig, 2003) presents a more detailed taxonomy as well as several examples.

The agents used in multi-agent simulations are more specific and usually posses some of the following properties:

- *Sensors and Actuators:* An agent possesses sensors to perceive stimuli from the environment (which includes other agents). An agent possesses actuators to interact with the environment (which includes again other agents). As a comparison, a human "has eyes, ears, and other organs for sensors and hands, legs, mouth, and other body parts for actuators" (Russell and Norvig, 2003). Figure 2.3, also from (Russell and Norvig, 2003), shows how an agent interacts with its environment.
- *Internal state:* An agent can be characterized by its internal state(s), such as READY / NOT READY, FRIEND / NEUTRAL / ENNEMY, or more complex 'emotional' states that represent emotions like TIRED, HUNGRY, MOTIVATED, HAPPY, ANGRY, VENGEFUL, etc.
- *Cognitive:* An agent possesses basic cognitive skills, such as memory, reasoning, and learning.
- *Autonomy:* An agent controls its internal states and actions based on what it has to accomplish. External interaction, from a human or another system, is thus not vital to the agent's operation.
- *Knowledge Representation:* An agent organizes his knowledge in structures that are used for reasoning and making decisions. This knowledge may include a mental map of the environment, a list of other agents known, some 'know-how' to accomplish tasks, and ways to acquire needed resources.
- *Intentions and Goals:* An agent has an overall task to perform. This task is divided into sub-parts that the agents plans to perform (intentions). Once the agent has decided to accomplish one intention and has planned how to achieve it, this intention becomes a goal.
- *Communication:* An agent is able to communicate and interact with other agents. Communication is performed via an agent communication language (ACL), such as

KQML and FIPA-ACL. The mode of interaction could be indifference, collaboration, competition, or others. (Ferber, 1995)

- *Rational:* An agent chooses the action that will best accomplish its goal, usually according to mathematical and logical formulas or more advanced methods like rules and behavioural graphs.

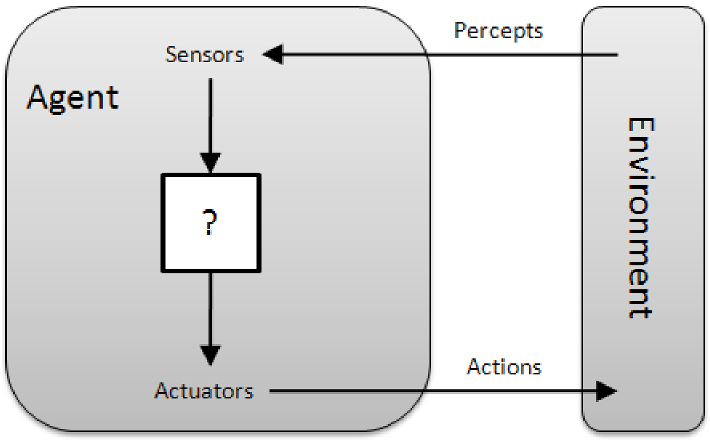


Figure 2.3: Agent's Interaction with its Environment

In addition, the environment in multi-agent simulations is typically a 2D or 3D virtual geometric space similar to what is seen in video games. Such an environment may contain constraints (walls, tunnels, buildings), objects (cars, trees), various resources, as well as the agents. These agents could all be identical (*homogeneous*) or be of different types (heterogeneous).

Multi-Agent Systems

An agent-based system is one in which the key abstraction is an agent. Multi-agent systems are simply systems that use more than one agent. More specifically, a multi-agent system can be defined as a collection of possibly heterogeneous computational entities (agents), each with its own problem solving capabilities, that interact together to achieve an overall goal that is beyond the individual capabilities of each agent (Ferber, 1995), (Jennings et al., 1998). Multi-agent simulation is one type of such systems, and it is discussed under the next heading.

Some of the key features of multi-agent systems are (Jennings et al., 1998):

- each agent has incomplete information, or capabilities for solving the problem, thus each agent has a limited viewpoint;
- there is no global system control;
- data is decentralized; and
- computation is asynchronous.

One of the most early and famous types of multi-agent systems is the implementation of the "blackboard" problem. A blackboard system is a common knowledge base, that is asynchronously updated by a knowledge sources and read by problem solving units. In the multi-agent solution, some agents are the knowledge sources and some are the problem solving units. However, these two sets of agents are not necessarily mutually exclusive. Others types of multi-agent systems include master-slave and benevolence assumptions (Conte et al., 1998). Although agents often work in collaboration, some systems include agents who are in relations of competition, negotiation, subordination, and others.

Multi-agent systems have led system developers to use new techniques, tools, paradigms, and philosophy when creating systems (Sahli, 2006). Although such systems provide great advantages, such as robustness, efficiency, and models that are close to reality, the field is still young, design methodologies are evolving and many questions remain open. (Jennings et al., 1998; Lynch and Rajendran, 2005). For example, multi-agent systems exhibit concurrency, emergent (unpredictable) structures and behaviour characteristics (Vassileva, 2001; Lynch and Rajendran, 2005). Nonetheless, multi-agent systems are "flexible and adaptive enough to solve different distributed problems such as scheduling, internet search, resources allocation, virtual environment for training and education, information retrieval, and military applications." (Ali, 2006)

Simulation Based on Multi-Agent Systems

As expected, multi-agent simulation, or multi-agent-based simulation (MABS), is a type of computer simulation that is based on the multi-agent paradigm. Compared to other approaches, such as discrete-event simulation, continuous-event simulation, and object-oriented simulation, MABS has a number of interesting properties that make it more attractive and useful. For example, MABS allows for modelling structures that preserve the simulated reality, which enables more researchers to use simulation and facilitates communication. MABS allows also for the design of highly specific behaviours for individual entities, highly customizable and dynamic scenarios, different levels of representation (individual, group, cluster), the analysis of emergent phenomena, and parallel computation. (Davidsson, 2000; Klügl, 2004; Sahli, 2006)

"MABS is nowadays used in a growing number of areas and domains, as a result of its ability to cope with very different models of 'individuals', ranging from simple entities (reactive agents), to more complex ones (cognitive agents)." (Ali, 2006) A short list of projects in various domains is presented in (Sahli, 2006).

Why Multi-Agent Simulation?

Since the field of multi-agent simulation is fairly recent, one might ask why this type of simulation is used over others, such as mathematical formulations, system dynamics (SD) or cellular automata.

First, multi-agent simulation naturally supports micro-simulation and it is often used when macro-simulation is clearly inadequate. For example, when the assumptions of homogeneous space and population are not reasonable, or when the causes of emerging phenomena are unknown and must be discovered from simulating individual behaviour. (Klügl, 2004)

Another reason for the use of multi-agent systems is when the spatial aspect of agents and their behaviours is important. For example, simulating the power demand of millions of home does not require that homes move or interact with each other; however, simulating the migration of birds requires interaction and realistic movements. Thus, multi-agent systems are often selected over cellular automata (which are spatially fixed) and particle systems (which navigation is usually based on physical forces rather than on a rational thought process).

The next sub-section introduces a type multi-agent simulation that is specifically oriented towards applications where the spatial aspect is crucial.

2.2.2 Multi-Agent Geo-Simulation

Multi-agent geo-simulation (MAGS) is a type of multi-agent simulation that is coupled with a GIS. This sub-section first introduces GISs and then discusses multi-agent geo-simulation.

Geographic information systems are those that allow storing, viewing, editing, and analyzing geographic data. GIS data exist in two formats: raster and vector. The raster format subdivides semantic information (e.g. road, building, river) into regular square boxes that represent one specific semantic type. This approach's precision is determined by the accuracy of the spatial subdivision. A coarse subdivision is fast, takes little space, but it is imprecise; a finer subdivision is slower, takes more bytes, but it is more precise. The vector format locates semantic information exactly with unconstrained geometric shapes (i.e. not only squares). This approach can be much more precise and accurate because the real life shapes can be represented exactly instead of being approximated. However, the vector format is more complex to handle computationally.

Originating in the 1960's, they were used mostly on central computers, such as mainframes, with static information. During the 1990's, researchers started thinking about ways to tackle more dynamic information, such as civil evacuations and wild life disasters. GISs play an important role for the authorities because they can represent in a very precise manner the real world with various types of data arranged in layers, but they are often confined to the tasks of preparing for an eventual crisis. "During major disasters, the chaos and the rapidly changing situations have prevented the use of GIS in response [...] the coupling of GIS with communication and modelling tools are a necessity in using GIS for disaster response." (Herath, 2001)

The ability of GIS to process complex spatial information and the potential of multi-agent simulation to closely represent reality with micro-level models gave birth to a new simulation sub-field: multi-agent geo-simulation. In that sub-field, phenomena of interest, urban or on a larger scale, "are considered as the outcome of the collective dynamics of multiple animate and inanimate urban objects." (Benenson and Torrens, 2003) This paper adds that new methodologies for manipulating and interpreting spatial data have provided new information sources at fine resolutions, both spatial and temporal. In turn, these methodologies, along with the coupling to geo-simulation models, have created added-value for these data.

The added value comes from the ability to situate the simulation in the exact spatio-temporal conditions desired. In the past, researchers and analysts had to settle for "close-enough" initial conditions or had to infer from similar experiments with rules that could difficultly be validated. The advantage of geo-simulation is to provide more appropriate, customized, and accurate experiments, possibly leading to fewer experiments to obtain useful results. In addition, these results are more plausible, accurate, and precise than without the use of a GIS within the experiments.

Another added value of geo-simulation is the ability to analyze the output data in a richer manner. It is possible to analyze geo-simulation output with traditional methods such as statistics or manual inspection of data. However, "it is recognized that 80% of data have a spatial component" (Wrembel and Koncilia, 2006), and this number is probably higher is geo-simulation output data. "Having the possibilities to display data on maps, to compare maps of different phenomena or epochs, and to combine maps with tables and statistical charts allows one to get more insight into spatial datasets." (Wrembel and Koncilia, 2006) Methodologies such as spatial on-line analytical processing (SOLAP) are best suited to use at best the combination of simulation and GIS (Bédard et al., 2006).

2.2.3 Crowd Simulation

Although human beings possess several capacities that other beings do not enjoy, the ability to reason is what truly differentiates humans from other species. Indeed, humans are able to mentally plan and visualize actions that they want to execute, expliciting any relations between the actions themselves and between the actions and the environment (Paris, 2007). For this reason, social systems are too complex to be directly modelled; rather, their interest comes from observing the emerging properties of the system that were not present at the component level (Reeves and Reeves, 2009). Rather than exploring macro-level results about "average" individual elements, crowd simulation offers the possibility of directly interpreting individual behaviour (Benenson and Torrens, 2004).

Technically, crowd simulation[4] is a sub-field of multi-agent geo-simulation that aims at simulating crowds of tens, hundreds, or thousands of human individuals in urban environments. Because the number of agents is relatively small (compared to other types of simulation), the agent models must closely reflect reality in order to give plausible results. Thus, a lot of research in crowd simulation has been devoted to the elaboration of agent models. However, "one major omission of traditional simulators is that they do not allow for the personal differences in behavior" (Murakami et al., 2002). Indeed, more and more research is being oriented towards developing complex behaviour models rather than complex agent models.

Completely modelling human behaviour would include mechanisms such as control of the physical body (muscles), perception, memory, emotions, communication, planning goals,

[4]The terms 'urban simulation' and 'social simulation' also are commonly used

evolving (learning), etc. Unfortunately, not all of these mechanisms can be simulated at once with the same set of models. Some simulators emphasize emotions, others learning, and others memory. In fact, in addition the agent themselves, models of interaction between agents are becoming increasingly important. Whether agents interact with other agents or with their environment directly, crowds are based on numerous and varied interactions. These interactions can be modelled by the perception-decision-action loop representation (Lord and Levy, 1994), which has proven its suitability to model human behaviour. This loop explains that humans perceive their environment through perceptors (e.g. eyes, ears, hands), make a decision based on their knowledge of their environment, and react using actuators (e.g. mouth, arms, legs). The decision process of this loop, which corresponds to the five behavioural stages of a human being, has been detailed in (Newell, 1990) with the well known behavioural pyramid (Figure 4.22). This pyramid shows that the human interactions are complex and happen at several levels. Depending on the application domain, different simulation models emphasize interactions at different levels.

Crowd simulation has been used for crowd evacuation (Murakami et al., 2002; Pan, 2006), mall shoppers' behaviour (Ali, 2008), railway station design (Paris, 2007), traffic prediction (Champion et al., 2001), as well as other projects related to crowds in urban environments.

Because this thesis is about crowd control, a situation where leadership plays a crucial role (Murakami et al., 2002), the following sections will present the existing social models and discuss whether they provide sufficient support for groups, leadership, and the associated interactions to be plausible in situations of crowd control.

2.3 Agent Models

This section presents models of agents that are meant to be part of crowds. The focus in this section is on the agent's architecture, while Section 2.4 will focus on the crowd's 'architecture' or representation.

For brevity's sake, only the models that seem pertinent for crowd simulation, and most specifically crowd control simulation, will be presented. Other models, such as macro-simulation and fluid dynamics can be found in (Moulin et al., 2007).

2.3.1 Moulin's MAGS System

The authors of (Moulin et al., 2003) developed a generic software platform for the creation of multi-agent geo-simulations (MAGS) involving several thousands of agents interacting in virtual geographic environments and endowed with spatial cognitive capabilities. They applied the geo-simulation approach in the simulation of crowd behaviours in urban environments. The individual agents in MAGS are equipped with several knowledge-based capabilities such as perception, navigation, memorization, communication, and objective-based behaviour, which allow them to display an autonomous behaviour within a 2D/3D geographic virtual environment. An agent is characterized by a number of variables whose values describe the agent's state at any time. The static state does not change during the simulation, and is represented by a variable and its current value (e.g. gender, occupation, marital status). A dynamic state is one that can possibly change during the simulation (e.g. hunger, tiredness, stress). This dynamic state is represented by a variable, associated with a function that computes how its values change during the simulation. The variable is characterized by an initial value, a maximum value, an increase rate, a decrease rate, an upper threshold, and a lower threshold, which are all used by the function. Using these parameters, the system can simulate the evolution of the agents' dynamic states, as well as trigger the relevant behaviours. Behaviours are based on automata where each node represents an objective. Activation and completion rules can be assigned to each objective, as shown in Figure 2.4.

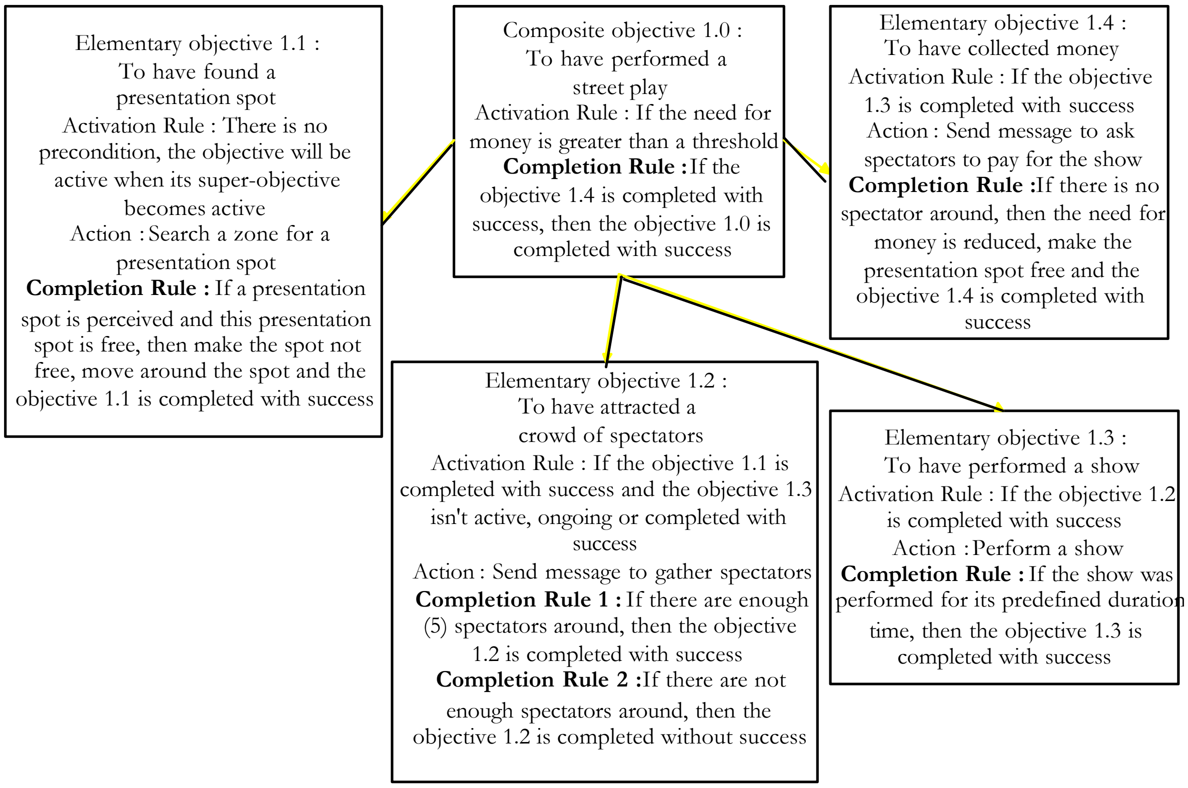

Elementary objective 1.1 :
To have found a presentation spot
Activation Rule : There is no precondition, the objective will be active when its super-objective becomes active
Action : Search a zone for a presentation spot
Completion Rule : If a presentation spot is perceived and this presentation spot is free, then make the spot not free, move around the spot and the objective 1.1 is completed with success
Composite objective 1.0 :
To have performed a street play
Activation Rule : If the need for money is greater than a threshold
Completion Rule : If the objective 1.4 is completed with success, then the objective 1.0 is completed with success
Elementary objective 1.4 :
To have collected money
Activation Rule : If the objective 1.3 is completed with success
Action : Send message to ask spectators to pay for the show
Completion Rule : If there is no spectator around, then the need for money is reduced, make the presentation spot free and the objective 1.4 is completed with success
Elementary objective 1.2 :
To have attracted a crowd of spectators
Activation Rule : If the objective 1.1 is completed with success and the objective 1.3 isn't active, ongoing or completed with success
Action : Send message to gather spectators
Completion Rule 1 : If there are enough (5) spectators around, then the objective 1.2 is completed with success
Completion Rule 2 : If there are not enough spectators around, then the objective 1.2 is completed without success
Elementary objective 1.3 :
To have performed a show
Activation Rule : If the objective 1.2 is completed with success
Action : Perform a show
Completion Rule : If the show was performed for its predefined duration time, then the objective 1.3 is completed with success

Figure 2.4: Example of MAGS objectives (Moulin et al., 2003)

2.3.2 Thalmann's Basic Agent Architecture

(Thalmann et al., 2000) presents an overview of a basic agent architecture for use in micro-level simulations. The authors focus on the need for agents to be equipped with sensors, a memory, perception abilities, and a behavioural engine that enable them to interact with their environment. Figure 2.5 is reproduced from the article and the following agent architecture is based on this figure.

Thalmann defines perception "as the awareness of the elements in the environment through physical sensation. This is achieved by equipping the agents with visual, tactile and auditory

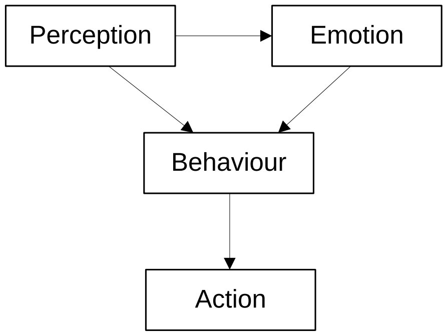


Figure 2.5: Thalmann's Agent Architecture (Thalmann et al., 2000)

sensors". He also suggests that agents be able to perceive three elements: 1) the presence of objects and actors, 2) actions of actors, and 3) actors performing actions on objects. Thalmann clearly mentions that if an actor's perception is restricted to the objects and other agents (their actions not being taken into account), then the behaviours will be limited because "only the presence and the characteristics of an object or an actor are implied in selecting a behaviour.

Next, Thalmann defines an emotion as "a person's reaction to a perception". For high realism, agents "must be capable of responding emotionally to their situation and acting physically within it." Three emotion types can be distinguished, based on three types of events that could cause these emotions: 1) potential events, 2) events affecting the fate of others, and 3) events affecting the well-being of the agent.

"Behaviour may be described in a hierarchical way. The behavioural model decomposes a behaviour into simpler behaviours which may themselves be decomposed further. Each level of this hierarchical decomposition contains one or more behaviours performed either sequentially, or concurrently." "An elementary behaviour is situated at the bottom of the hierarchical decomposition and encapsulates a specialized behaviour which directly controls one or more actions." "A high level behaviour reacts to sensorial input and uses special knowledge. A means of modelling behaviours is the use of an automaton. Each actor has an internal state which changes with each time step according to the currently active automaton and its sensorial input." A set of high-level behaviours is normally used to represent the overall agent's 'conduct'.

Finally, "actions may have several degrees of complexity", ranging from a smile to navigation from a location to another one.

2.3.3 Silverman's Agent Architecture

Silverman's team in (Silverman et al., 2002) tried to "integrate human behavior models from a range of ability, stress, emotion, decision theoretic, and motivation literatures into a game-theoretic framework appropriate for representing synthetic asymmetric agents and scenarios. Our goal is to create a common mathematical framework (CMF) and an open agent architecture that allows one to research and explore alternative behavior models to add realism to software agents." The authors claim that "recent theories identify emotions as vital to the decision-making process and to manage competing motivations. [...] integrating emotion models into our agents will yield not only more believable decision-makers, but also more realistic behavior by providing a deep model of utility." The authors created an agent architecture that is composed of five sub-components, as shown in Figure 2.6.

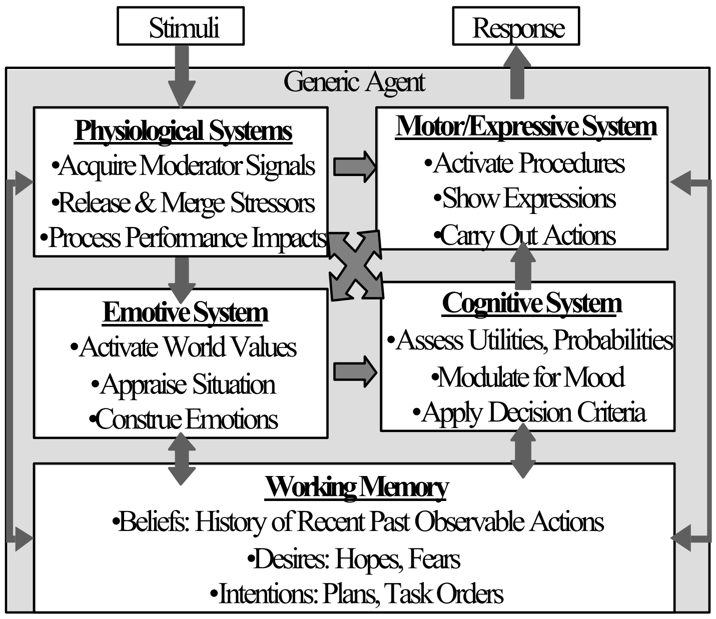


Figure 2.6: Silverman's Agent Architecture (Silverman et al., 2002)

Silverman's agent is based on the beliefs-desires-intentions (BDI) model. Beliefs are simple statistical models of the actions around the agent; desires are the "future-focused affective states of hope and fear as generated by the emotion system; intentions are the planned actions and sets of orders that the agent is seeking to carry out."

The physiological sub-system includes all sensory apparatus and several reservoirs. A reservoir is defined as a physical process that can be depleted and replenished, such as energy, sleep, and light impacts. This sub-system reacts to a set of stimuli that are perceived from the environment and possesses stressors that moderate the agent's capacities and sends alarms to other sub-systems (e.g. when pain occurs). The physiological sub-system has been designed for the easy addition and deletion of reservoirs depending on the given study or environment. For example, it is possible to add a reservoir to model the effects of severe heat vs. cold winter weather, depending on the scenarios to be modelled. An important contribution of this sub-system is that it combines several low level stressors and performance moderator functions into a single stress level. Silverman integrated "iSTRESS" as a function based on "three prime determinants: 1) event stress, which tracks agents' adverse and positive events, 2) time pressure, which is a normalized ratio of available vs. required time for the tasks at hand, and 3) effective fatigue, which integrates a normalized metric based on current level of many of the physiological reservoirs." The iSTRESS value is used to change the "coping style" of decision-making agents. Because Silverman could not find in the literature precise threshold values for "iSTRESS", he uses logic rules to combine the three factors.

The emotion sub-system receives stimuli from the physiological sub-system. Also, it is deeply connected to the agent's memory. These stimuli and other internally recalled stimuli work with an alterable set of activation / decay equations and parameters for a variable number of emotions. It is also possible to combine multiple emotions into a utility estimate for a given state. Thus, it becomes possible "to capture the behaviors of different 'types' of people and organizations and how differently they would assess the same events, actions, and artifacts in the world."

The cognitive sub-system is the point where the diverse emotions, stressors, memories, and other factors become integrated into a decision for action and transition to a next state. "While one can argue against the idea of aggregating individual emotions, this summation is consistent with the somatic marker theory."[5]. The agent's utility is derived dynamically at each iteration for each candidate action based on the agent's importance-weighted concern ontology. Silverman also included a discount factor that more heavily weights goal achievement, the closer the agent is to the end of the 'current stage' of a scenario. "Thus an agent might be conservative and construe survival as more important early in the game, yet be willing to make more daring maneuvers near the end point."

The motor / expressivity sub-system contains libraries of stored procedures that allow the agent to interact with the environment and that allow him to display his motor and expressive outputs. Based on stimuli from the other subsystems, the motor sub-system performs the actions intended to reach the next state. The sub-system performs up to the imposed physiological limits and also serves as a stimulus to the other systems. "For example crouching for a long period might cause fatigue, pain, emotive distress, and so on." The motor sub-system would not perform as well afterwards (the agent may walk more slowly).

Silverman and his team draw very interesting conclusions from their experience. Key points from these conclusions are reproduced below.

- The literature is helpful for improving the realism of behavior models
- There are benefits (and costs) of modeling stress-emotion-decision processing as an integrated topic
- Concern ontologies are vital but require ontological engineering
- Emotion models are useful for utility and decision making not just for expressivity

2.3.4 Silverman's Integrative Framework for PMFs

A Performance Moderator Function (PMF) is a mathematical model that guides an agent's behavioural model by adjusting his capacities according to various inputs such as stress, fatigue, and injuries. Isolated PMFs are used in many simulations, for example to reduce shooting

[5]The somatic marker theory proposes that emotional processes, through neural substrates that regulate homeostasis, emotion, and feeling, can affect behaviour, especially decision-making.

accuracy of a military officer under heavy stress, but no integrative framework exists to combine several PMFs into one agent model. The authors of (Silverman et al., 2003) mention that their goal is "to identify performance moderator functions (PMFs) and related human behavior models from within the research literature and (1) identify and properly abstract them, (2) assess their internal validity, and (3) prepare the best of them for implementation and reuse. This would make it easier to (re)utilize PMFs and thus to improve the realism of human behavior in models and simulations." The framework has been named PMFServ and its agent architecture is presented in Figure 2.7.

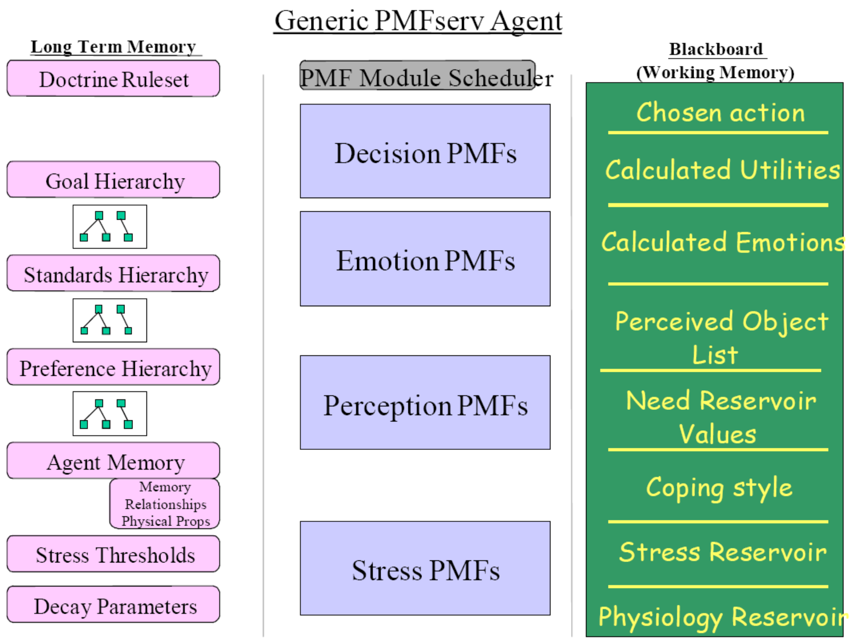


Figure 2.7: PMFServ Agent Architecture (Silverman et al., 2003)

"PMFserv is built around a 'blackboard' data structure that loosely corresponds to a short-term or working memory system. Sensory data about the world flows into the lower layers of the blackboard structure, as constrained by stress and other factors described below. Modular PMF subsystems then manipulate data contained in the blackboard and in a long-term memory store." "Moving up the blackboard from the bottom reveals the decision cycle of a single agent. Physiological data across a range of measures (including PMFs for arousal, exertion, hunger, thirst, injury, etc.) are combined to set the levels of a series of stress reservoirs. [...] Each reservoir keeps track of both the current level of the stimulus in the environment and any stress that results from that stimulus. There are a large number of stressors that moderate an agent's ability to perform up to capacity. In some cases, these produce alarms. For example, alarms may occur when there is pain or when a critical threshold is exceeded (e.g., hunger, fatigue, panic, etc.). An important criterion for such a module is that it should support study of common questions about performance moderators (e.g., easy addition or deletion of reservoirs such as pain or stress), individual differences in reacting to particular stressors, and/or how to model reservoir behaviors linearly (PMFserv approach) or non-linearly, such as with bio-rhythms."

The authors mention concerns about testing, verification and validation. One important question is “how to reliably determine whether each agent is operating according to specification. That is, verification is necessary to ascertain that agent behavior is (1) consistent with respect to individual PMFs; (2) complete with respect to the collected set of all PMFs being implemented; and (3) somehow coherent with respect to their own goals, standards, and preferences in the scenario.” Unfortunately, this task is quite tedious. “One way to verify that the PMFs are working properly is to separately examine each agent and each PMF as the scenario unfolds.” The authors tested PMFServ on military scenarios with a set of visual PMF interfaces. These interfaces contained tabs that revealed the agents’ “current physiology (mild exertion, noise), coping mode (Defensive Avoidance), emotions (e.g., liking or disliking specific aspects of the situation)”.

The authors mention also that “verification that multiple PMFs work in concert is not the same as validation. The latter requires one to evaluate how well scenario outcomes correspond to real world or historical events. Historical recreations are challenging because participants’ thoughts, motivations, and stress levels can be known or estimated only at a general level.” Silverman and his team suggest a few approaches. “As a qualitative approach, one might ask knowledgeable observers to compare the simulated and historical outcomes. A more quantitative approach would be to quantify events along a timeline and/or quantify outcomes by type of participant and determine correlative relationships between real and simulated events and outcomes. Of course, it is also possible to combine qualitative and quantitative efforts to evaluate correspondence.”

Once again, the authors provide a useful set of conclusions and recommendations, which are reproduced below.

- Integrated models will improve the realism of simulated agent behavior
- Value sets are vital but require significant engineering
- Emotion models are useful for culture-based utility and decision making
- Shift attention from the development of automatons to the development of realistic agent behavior
- Assemble a reusable, easily-adapted library of realistic digital casts and avatars to populate a wide array of scenarios encountered by soldiers and police
- Reduce, by at least an order of magnitude, the effort needed to introduce human performance modeling components (PMFs, AI, A-life, etc.) into simulators

2.3.5 Silverman’s Unified Architecture

(Silverman et al., 2006b) is based on the previous two sub-sections because it unifies in Figure 2.8 the agent architectures presented in Figure 2.6 and in Figure 2.7. The article draws from the results of six years of research on ways to enhance the realism of synthetic agents. The architecture proposed PMFs on physiology, stress, personality, cultural, emotive processes, perception, social processes (relations, identity, trust, nested intentionality), and cognition.

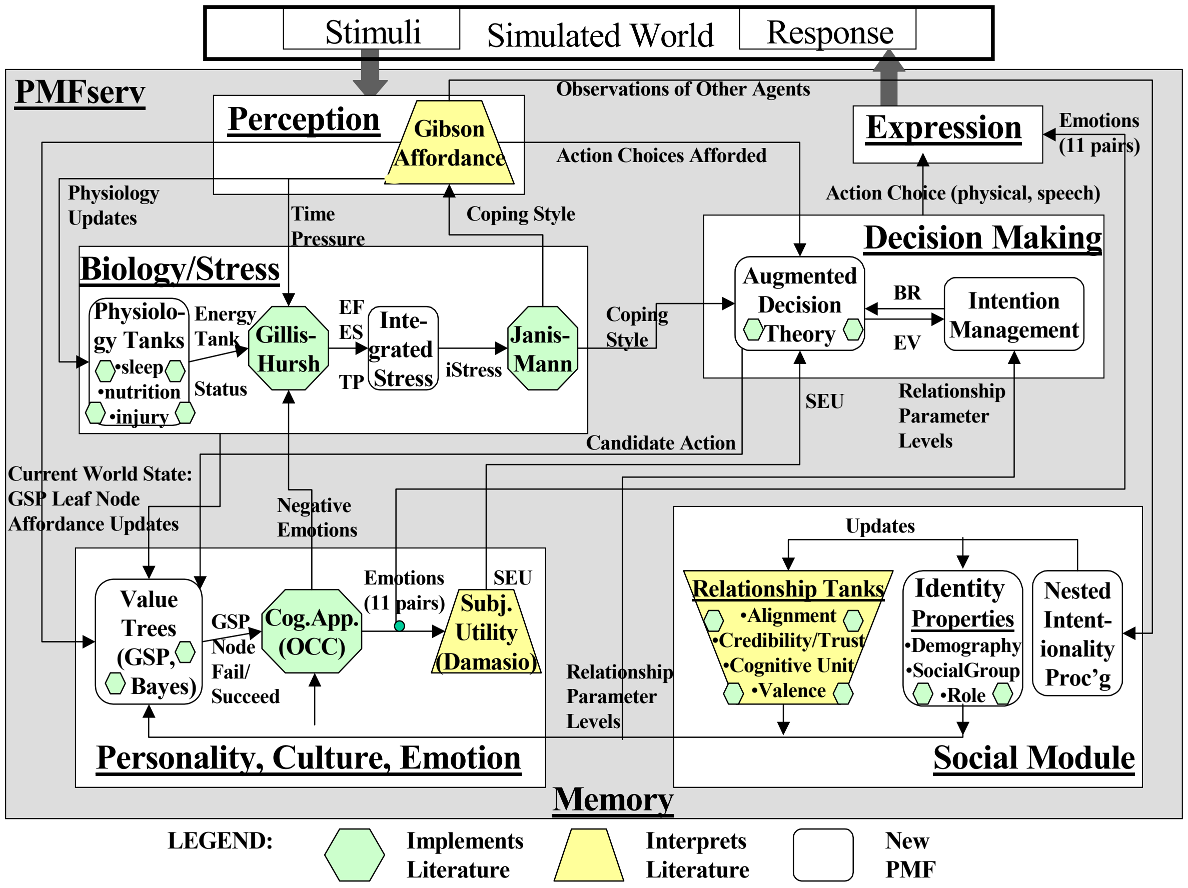


Figure 2.8: PMFserv Implementation of the Unified Architecture (Silverman et al., 2006b)

The architecture presented in Figure 2.8 is based on several social theories. For the sake of brevity, they will not be detailed here. The description of the main building blocks of agent architecture is similar to the one presented in Sub-Section 2.3.3. The article also presents several use case studies (asymmetric warfare, civil unrest, and political leaders) that seem to be very similar to those presented in earlier articles. Finally, Silverman and his colleagues conclude with interesting lessons learned.

2.3.6 Pan's MASSEgress Agent Model

In (Pan, 2006), Pan incorporated diverse human behaviours into a Multi-Agent Simulation System for Egress analysis (MASSEgress). "Social behavior is simulated through modeling of individual behavior and interactions among individuals. Competitive, queuing, herding, and leader-following behaviors are modeled. MASSEgress is a computational framework; its modular design allows easy extensions to include additional behavior types." "Comparisons of MASSEgress with other evacuation models have been performed to demonstrate its capabilities as well as to validate the computational framework with prior results. Simulation to replicate a historical event [fatal fire at a nightclub in the USA] has also been carried out.

Finally, an application of MASSEgress to simulate emergency evacuation of a multi-story university building is performed to illustrate the potential utilization of the simulation system for egress design analysis."

Theoretical Propositions

An individual's behaviour is the outcome of his decision-making process. Pan conjectures that an individual's decision-making process follows three basic conventions: 1) instinct, 2) experience, and 3) bounded rationality. "An individual may select one or a combination of these basic conventions when faced with emergencies, depending on the specifics of the situation. These three conventions are explained below."

"Instinct refers to inborn patterns of behaviour responsive to specific stimuli. Executing an instinct does not require conscious thought process." "Examples of human instincts are fear, death and survival. When there is a need to make decisions under high stress, following one's instincts is the most primitive way that an individual relies on in making instantaneous and quick decisions."

"An individual often relies heavily on his/her personal experiences in making decisions." "Decision-making in terms of following experience is usually straightforward and quick. The process typically follows three basic steps: (1) recognize a situation that is the same as or similar to an experience in the past; (2) retrieve the routines that were successful according to prior experience; and (3) carry out the routines."

"Rational decision-making assumes that decisions are based on evaluation of alternatives in terms of their consequences for preferences. The process involves four basic steps: (1) search for possible options; (2) anticipate consequences of each option; (3) weigh each consequence against preferences; and (4) choose the most favourable option. Such a decision process is bounded because typically, not all options are known, not all consequences are considered, and not all preferences are evoked simultaneously."

Since MASSEgress has been developed for evacuations, non-adaptive crowd behaviour is very important. It can be defined as "the type of crowd behavior that does not adapt to an emergency situation and often leads to destructive consequences, which range from clogging at exits, to stampede, pushing, and trampling, etc." Pan shows a process model of the emergence of non-adaptive crowd behaviour in Figure 2.9.

Implementation Details

Each MASSEgress agent is based on an 'Individual Behavior Model', which "is composed of three subsystems: a Perception System, a Behavior System, and a Motor System", as shown in Figure 2.10. "These subsystems implement how each agent senses the situation and the environment, makes decision and acts according to its behavior model."

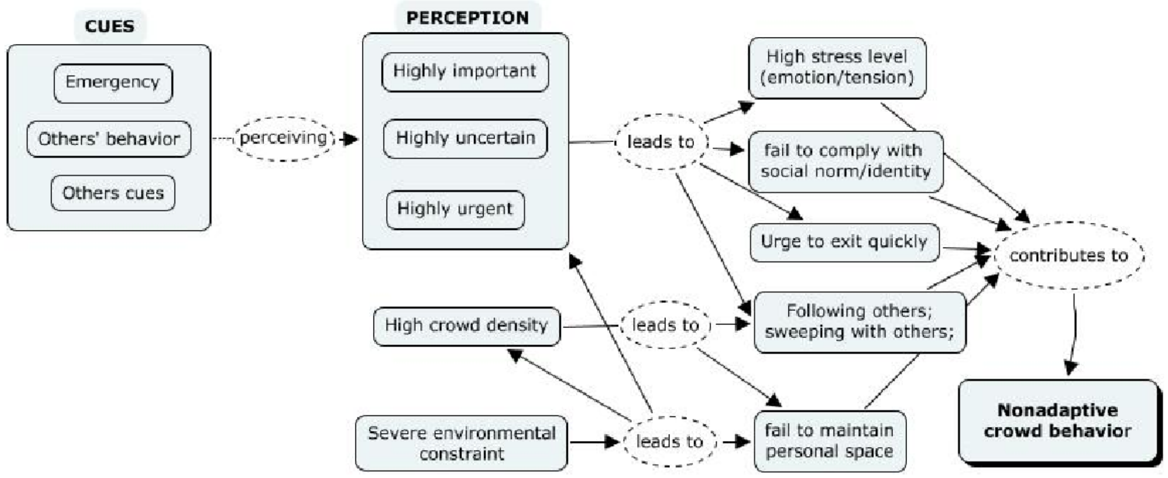

CUES
Emergency
Others' behavior
Others cues
perceiving
PERCEPTION
Highly important
Highly uncertain
Highly urgent
leads to
High stress level (emotion/tension)
fail to comply with social norm/identity
Urge to exit quickly
Following others; sweeping with others;
contributes to
Nonadaptive crowd behavior
High crowd density
leads to
Severe environmental constraint
leads to
fail to maintain personal space

Figure 2.9: A Process Model of the Emergence of Non-Adaptive Crowd Behaviour (Pan, 2006)

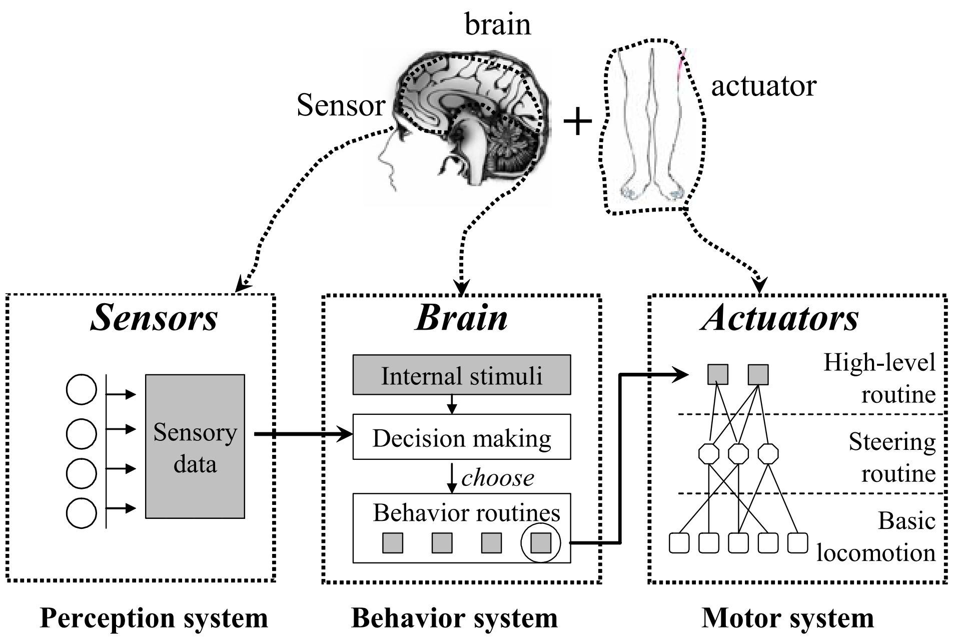


Figure 2.10: MASSEgress' Agent Architecture (Pan, 2006)

“The perception system of an agent consists of one or more sensors. “The sensory mechanism is an algorithmic procedure that processes the input parameters and then produces sensory data for further processing by the behavior system.” “There is multiple sensory information such as visual, audio, emotions or tensions, and others that could affect individuals' decision making”. “We have adopted the concept of view volume which is a visual cone defined by a perception range and a view angle. The view volume represents the basic constraint of an agent's visual perception — an object is visible only if it falls within the view volume and is not occluded by any obstacle.” This cone is presented in Figure 2.11.

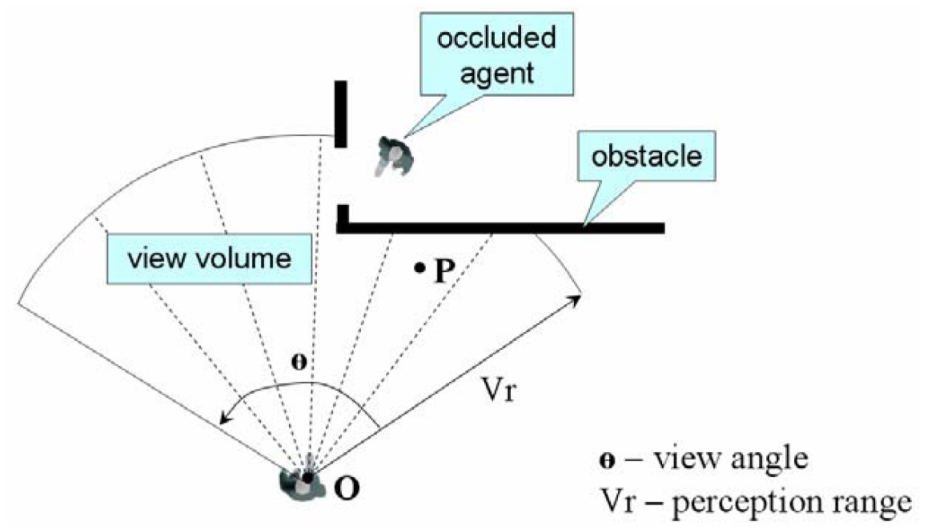


Figure 2.11: MASSEgress' Agent Perception (Pan, 2006)

"Based on the sensory data received from the perception system, the agent then takes into consideration of some internal stimuli (i.e., psychological and sociological factors) and makes specific behavior decisions. The primary components of the behavior system are decision-making rules which are organized as decision trees. A set of complex decision-making rules can be organized as a single decision tree". The psychological and sociological factors involved in a decision-making process in evacuation situations are different from those in public demonstrations and protests, but they are nonetheless interesting to mention. The following factors are considered in MASSEgress: familiarity (with the location and exits), decision-making type (personality, experience), urge to exit, stress, and herding factor (tendency to follow others when not knowing what to do).

"Due to the modularity of MASSEgress, it should be pointed out that the decision-making rules and behavior routines can be tested independently before being integrated into the system — a decision-making rule or a behavior routine can be assigned explicitly to an agent for testing purpose. The design has the flexibility and extensibility that allows modeling the psychological and sociological aspects of an agent's behaviors and incrementally incorporating them into the system."

"Many decision trees can be constructed to simulate personality differences among agents and to simulate the influences of various stress levels on agent decision-making." "Constructing a decision-making tree is essentially to define the decision-making process to be undertaken by the agent." An example is shown in Figure 2.12.

2.3.7 Wijermans' Festival Simulation

In (Wijermans et al., 2008), the authors propose "an approach to model human behaviour in groups by incorporating multiple goals and situatedness in terms of limited and subjective

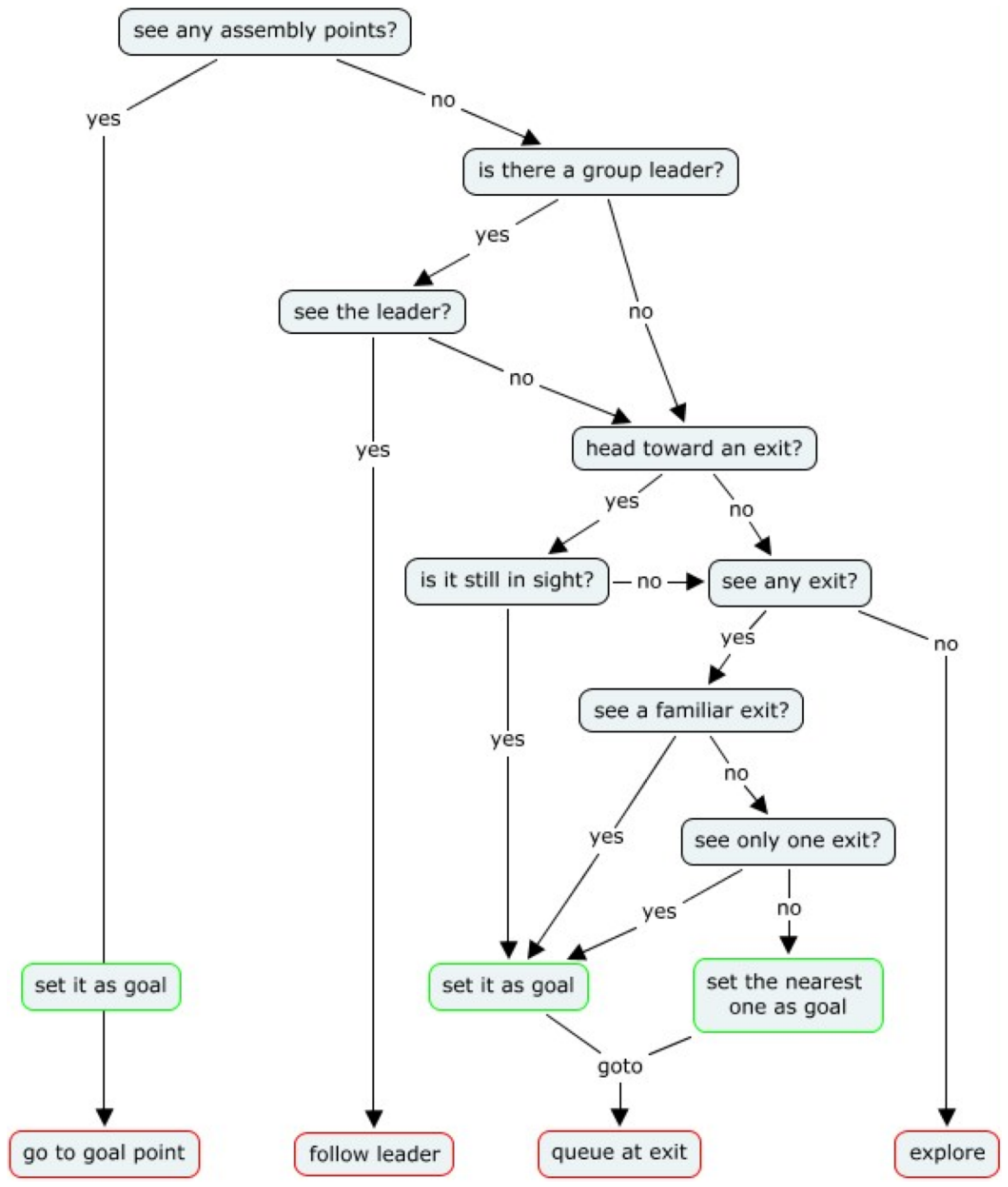


Figure 2.12: MASSEgress' Agent Decision Tree (Pan, 2006)

perception." Their "view on modelling human (group) behaviour, evolves around the idea that behaviour is a result [of] the interaction of an individual with its environment". They explain their model as reproduced below.

- **Multiple goals:** With main goals we imply that they are abstract goals that can potentially be satisfied by each behaviour, instead of being very concrete and directly linkable to one behaviour. The dominance of each of the main goals influence the chance of a possible behaviour (satisfier) to be selected. The goal dominance itself is subjected to influences given a current situation, i.e. situatedness. This dynamics between the goal dominance and thus the selected behaviour could cause perseverance or a behaviour turnover.

- **Limited perception:** The requirement of limited perception lays restrictions on how an individual is able to perceive the world it is situated in and thus how it is influenced by it.
- **Subjective perception:** The additional restrictions in perceiving the world are our earlier experiences and our current state are captured by the subjective perception requirement. This can be seen as a sort of filter on 'objective' perception.

Wijermans and her colleagues implemented their model in a festival scene, which is a grid of 40 X 40 cells on which the agents (festival visitors) are situated, as shown in Figure 2.13. "The stage is represented by the yellow block, whereas the brown blocks represent the bar and toilets". "An agent can show directed movement, such as moving to the stage, stay close to other agents or visit the bar or toilet. Each behaviour can satisfy the goals in their own way. Depending on the goal dominance at a given time, they [the goals] influence which behaviour is more likely to be chosen. These internal settings are influenced and used by two main internal processes of an agent, perception and behaviour selection."

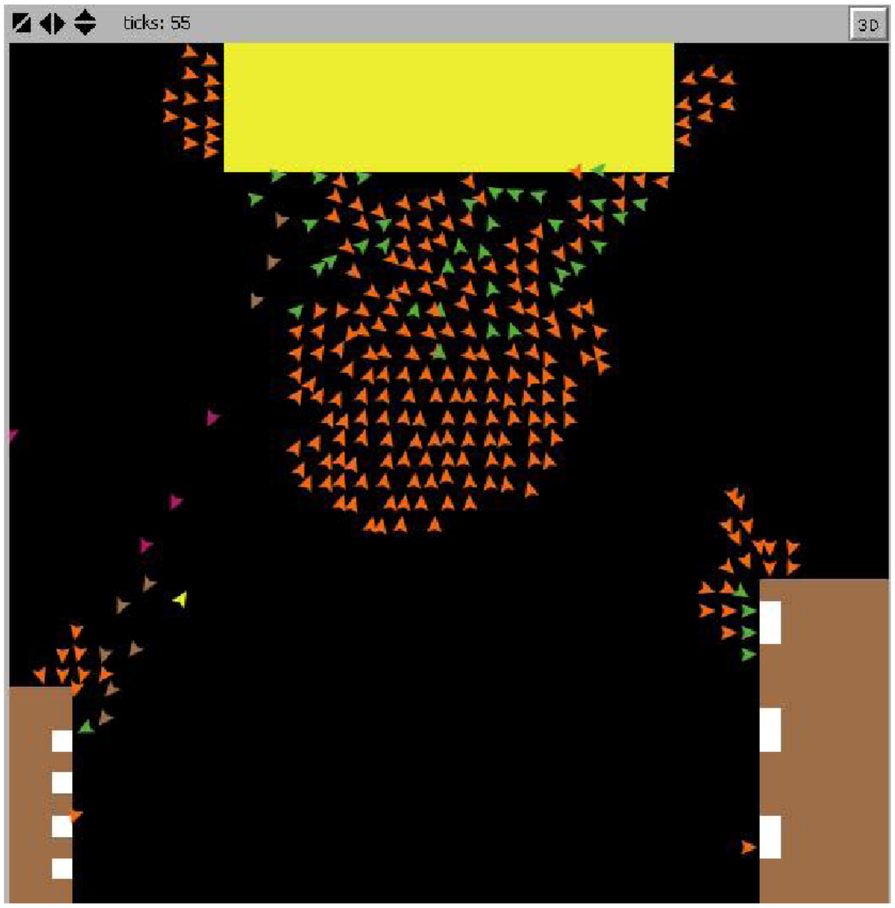


Figure 2.13: Wijermans' Festival Simulation (Wijermans et al., 2008)

In their simulation, they implemented their model as reproduced below.

- **Multiple goals:** "Concretised by four fixed general driving forces": "1.identity, 2. social, 3. safety and 4. subsistence. The first two main goals are based on the distinction between behaving as being a part of a social context (social goal) and personal drives that are not social (identity goal). The latter two are physiological rooted drives related to staying alive eating, drinking, sleeping etc. (subsistence goal) and protect our physical and mental safety (safety goal)."
- **Limited perception:** "Bounded by the current position, heading and specific limitations such as gaze-width, gaze-depth, and hearing range.
- **Subjective perception:** Influenced by "the proximity and hearing distance of a stage", the "level of density agents", and the "need to drink, or go to the toilet".

This simulation is fairly basic, but is has interesting elements that are similar to what is required in the Crowd-MAGS project. In addition, the authors offer interesting thoughts on future works. "As this is only our first raw version there is a lot of room for further development. We will continue doing so by starting with extending the social influences. This will be done by discerning between other agents, for instance in choosing to be close to others as a behaviour, this agent will choose to be close to its friends (people in its social network). Another essential extension is the time dependency for choosing a behaviour. At this point a behaviour is still chosen by calculating all utilities and choose the maximal one, this is not as it should be. Depending on the time an agent has it should calculate and compare to choose a behaviour. Just to give an idea how this framework is extended more, not to speak of the additional behaviours one can provide an agent with."

2.3.8 Kenny's Psychological Factors Affecting an Agent's Performance

The authors of (Kenny et al., 2001) developed a "model comprised of five psychological factors for understanding and assessing individual behaviour in a crowd." These factors could be used in Performance Moderator Functions (PMFs), especially since Kenny and his colleagues ordered by level of influence. "The model forms a pyramid, as shown in Figure 2.14, and the factors closer to the top have a more immediate influence on performance." *Motivation* is "an individual's ability to be committed to the crowd's cause despite fear, fatigue, opposition, and personal needs." *Confidence* is "an individual's belief in the ability to accomplish goals." *Stress* is "physical, mental, or emotional reactions to internal or external demands." *Focus* is "the ability to concentrate on those things leading to a goal." *Emotions* "guide decisions, influence reactions to situations, and determine how individuals feel".

2.4 Group and Crowd Models

Many conventional crowd control techniques are based on academic works performed over a century ago, such as Le Bon's famous book *The Crowd: A Study of the Popular Mind* from

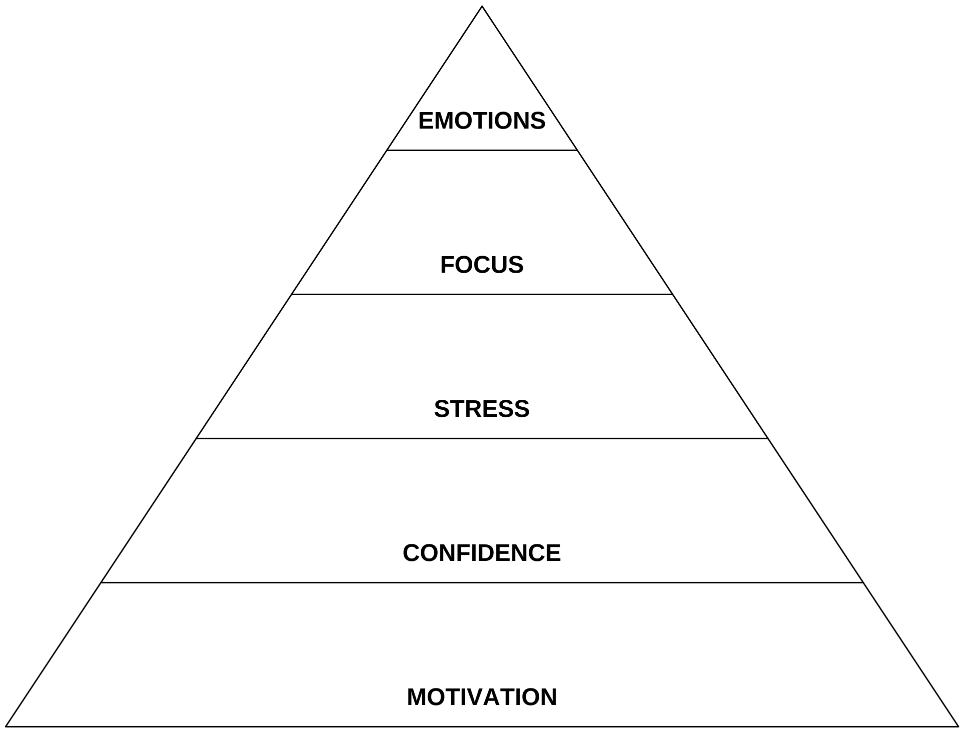


Figure 2.14: Psychological factors affecting an agent's performance (Kenny et al., 2001)

1895. These old writings fostered the beliefs that crowds are homogeneous masses whose behaviours can uniformly be categorized as active, expressive, aggressive or hostile. These writings also held that crowd participants were given to spontaneity, irrationality, loss of self-control, and a sense of anonymity. These assumptions have been seriously questioned by social scientists and some of them have gathered empirical data that adjusts the views and beliefs on crowds. A summary is reproduced below from (Kenny et al., 2001).

- Crowds are not homogeneous entities — all participants are not the same.
- Crowds are not made up of isolated individuals, but of "companion clusters", which arrive, remain and leave together.
- Crowd participants are not unanimous in motives.
- Crowd participants do not necessarily assume a sense of anonymity.
- Crowds are not given to unique emotional displays.
- Crowd participants seldom act in unison and, if they do, it does not last long.
- Crowds do not cripple individual cognition.
- Crowds are more of a process — they have a beginning, middle and end.
- Crowds are not uniquely distinguished by violence.

The authors also add that “social, political and economic factors are not consistent predictors of riot intensity.” “Additionally, a crowd has a limited duration. Its numbers are likely to diminish as individual needs take precedence over those of the crowd.” These new observations have given birth to various crowd models in the past decade; the most interesting ones for the Crowd-MAGS project are presented in this section.

2.4.1 Moulin’s Group-Focused Approach

(Moulin, 2009) suggests a simulation approach that is focused on groups rather than agents. The author starts by defining ‘purposive crowds’ as those “in which people gather for a specific collective purpose such as demonstrating against measures or regulations enforced by civil or military authorities, celebrating specific events or persons, participating in rallies to promote particular causes, and so forth. It has been observed that in a ‘purposive crowd’, most people come in groups, often small groups of friends or colleagues, acquaintances or even families.”

Next, Moulin suggests “that when modeling and simulating a purposive crowd, the most important element is not the individual, but the group! Indeed, individuals reason and make decisions on an individual basis, but their references are groups: groups of demonstrators that they recognize around them (‘in-groups’ as sociologists call them), but also ‘out-groups’ that they perceive as adversaries. A bystander may observe a crowd event as an uninvolved individual, but if she decides to join the demonstration, it is most likely that she will try to join a nearby group of demonstrators which attracts her and will offer her the opportunity to participate in collective actions. Consequently, there is a need to simulate the attraction and repelling of agents by groups.” Indeed, Moulin presents “ample evidence from sociological analyses that group dynamics greatly influence crowd situations”. He also advances that “it seems more plausible to model a purposive crowd using an approach based on ‘group dynamics’ rather than on one which is solely based on individuals’ interactions.”

Moreover, Moulin defends that “in the case of purposive crowds, we will assume that participating in collective actions is more important for agents than carrying out individual activities and pursuing individual goals. From a cognitive point of view, this is a simplification, but we think that it will not be too constraining for the types of applications that we address, that it will ease the computational load of the micro-simulation and that it is reasonable when considering the social characteristics of crowd situations.” “Putting the emphasis on collective actions, it seems natural to explicitly introduce agent groups in crowd simulations. ”

“In the proposed approach, groups play a pivotal role: 1) at a micro-level, agents are able to recognize and join groups in order to participate in collective activities; 2) at a meso-level, group interactions and relations evolve as a result of the individual activities of their members; 3) at a macro level, groups collect data about their membership, the collective activities they support, as well as their interactions with other groups in order to inform situations.” “We suggest that we need to model crowd phenomena at different levels of granularity. Obviously, we need to model agents at a micro-level which simulates individuals’ behaviours in the virtual environment as current crowd simulations do. However, agents will also need capabilities to perceive and recognize groups around them, to decide to join groups in order to participate in collective activities, but also to leave them, if needed. Hence, there will be an ‘organic’ link between this micro-level and a meso-level where models of groups and group interactions will be considered." "Finally, we emphasize that decision support requires a macro-level of observation and analysis of crowd behaviours, what we call the 'situation assessment perspective'."

2.4.2 Thalmann's ViCrowd

(Thalmann et al., 2000) introduces the concept of levels of autonomy related to groups of agents. The authors "describe methods to provide intelligence focused in a common group entity that controls its individuals." "Although some crowd simulations seem to be composed of autonomous agents, the individuals are controlled by complex groups behaviours." Thus, the authors have classified crowd behaviours in three categories, which are reproduced below.

- **Guided crowds**, with behaviours defined explicitly by the users
- **Programmed crowds**, with behaviours programmed in a scripting language
- **Autonomous crowds**, with behaviours specified by rules or other complex methods.

In (Thalmann et al., 2000) and in (Musse and Thalmann, 2001), the authors present a model (and a system: ViCrowd) that "distributes the crowd behaviours to the groups and then to the individuals", as shown in Figure 2.15. Individual agents also have a repertoire of innate behaviours, which are defined as basic 'inborn' ways to behave.

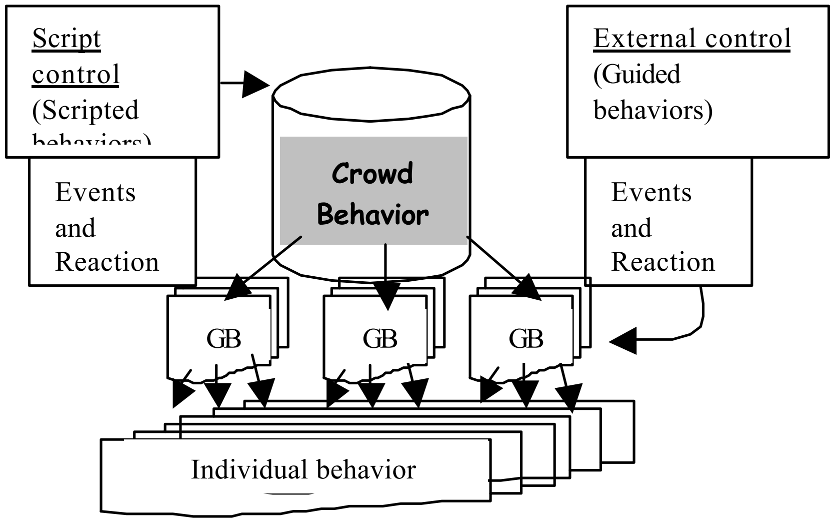


Figure 2.15: ViCrowd's Hierarchical Structure (Musse and Thalmann, 2001)

"The intelligence, memory, intention and perception are focalized in the group structure. Also, each group can obtain one leader. This leader can be chosen randomly by ViCrowd, defined by the user or can emerge from the sociological rules."

“The memory of groups is processed only by the leader of the group. In fact the memory is a structure where the leader’s perceived information can be stored and processed afterwards depending on the specified behavioral rules. The size of the memory (capacity of storage) can be pre-defined for each group of crowd.”

“Group perception concerns the information about the location of groups/agents as well as some associated parameters: knowledge, beliefs and intentions. In this way, a group can perceive the internal status of another group, for instance. As with memory, the perception is associated to just the leader of group.”

Next, the authors list eight group behaviours that are implemented in ViCrowd:

1. **Flocking:** “Group ability to walk together in a structured group movement in which agents from the same group walk at the same speed towards the same goals”
2. **Following:** “Group ability to follow a group or an individual motion”
3. **Goal Changing:** “Agents can have the intention to change groups, consequently assuming the goals of its new group”
4. **Attraction:** “Groups of agents are attracted around an attraction point”
5. **Repulsion:** “Group ability to be repulsed from a specific location or region”
6. **Split:** “This behaviour concerns the subdivision of a group to generate one or more groups”
7. **Space Adaptability:** “Group ability to occupy all the walking space”
8. **Safe-Wandering:** Procedural method “to evaluate and avoid collision contacts with agents and objects”

More details should be given here about agent-group relationships. “Basically, individuals have a more complex structure of parameters including: i) a value for the relationship with all groups (value between 0 and 1) and ii) a value for its domination status, which describes how much the considered agent is able to dominate the others (leadership ability). If the relationship with other groups is better than the current group, individuals can change groups. In addition, if the individual presents a high value for leadership ability, he/she can become the new leader of the group, which can change the group behavior too.” The splitting “behavior concerns the randomly generation of intentions to create new groups. The number of agents to be transferred to the new group is random as well as the list of agents.”

Finally, the authors demonstrate their model with the simulation of crowds in several different situations, notably citizens in an urban park, a crowd in a theatre, travellers in a train station, and a political rally. In the case of the theatre, “the user specifies directly in real time events generated by the speaker whereas the reactions are pre-programmed in the script. For example, the user can trigger an event that should generate a positive reaction, the latter being programmed in the script, e.g. ‘applaud’.”

2.4.3 Pan's MASSEgress Crowd Model

(Pan, 2006), with its MASSEgress system, presents interesting models for the crowd in addition to the agent models. Pan advances that "from the perspectives of social interaction, an individual's social behaviors are shaped by social structures through" 1) the principle of social identities, 2) the respect of personal space, 3) and the principle of social proof. These three factors are explained below.

- **Social identity.** "An individual in a crowd usually acts differently than when he/she is alone or in a small group." "Being part of a society is one essential aspect of a person. Societies are organized through various social structures. Social structures impose rules on individuals in the form of laws, regulations, cultures, and norms. Social structures are composed of diverse identities (i.e., social roles), and each identity has a set of associated rules, which define how it interacts with other identities."
- **Personal space.** "Under normal circumstances, an individual seeks social interaction with others; at the same time, the individual also tries to avoid intruding others' privacy as well as to defend intrusions." Even in crowded environments, individuals continue to attempt to regain their personal space and avoid physical contact with others. When crowd density reaches a certain magnitude [...], maintenance of personal space may become practically impossible, which could lead to nonadaptive crowd behaviors."
- **Social proof.** "Social proof is a phenomenon in which individuals, when faced with perceived uncertainty, for example insufficient information about new situations, follow the actions of others to guide behavior."

Pan adds that during a emergency situation, "regardless of the nature of an emergency, how it impacts an individual depends on the way that he/she perceives the situation and the environment, even though such a perception can be inaccurate or misguided. Varying perceptions of emergencies result in varying emotions and mental stress levels, which can in turn invoke varying decision-making mechanisms."

"Social behaviours are complex phenomena emerging from the interactions of individual agents which are in a group of autonomous agents. A single agent's behaviour is essentially nondeterministic at a microscopic level." "However, at a macroscopic level, certain behavioural patterns may be observed". "These social behavioural patterns are called emergent phenomena", and they are based mainly on four factors, according to Pan.

- **Individual behavior:** "Social behaviors are collective efforts of individuals. The behavior of each agent directly impacts the behavior of the group"
- **Group size:** "Social behavior may not emerge if the size of the group is too small"
- **Individual behavior distribution within a crowd:** "Agents may demonstrate multiple behaviors based on population type and perceived situation"

- **Geometric constraints:** "Certain social behaviors are the direct results of geometric constraints"

MASSEgress is able to demonstrate social emergent phenomena including competitive, queuing, and herding behaviours, as shown in Figure 2.16.

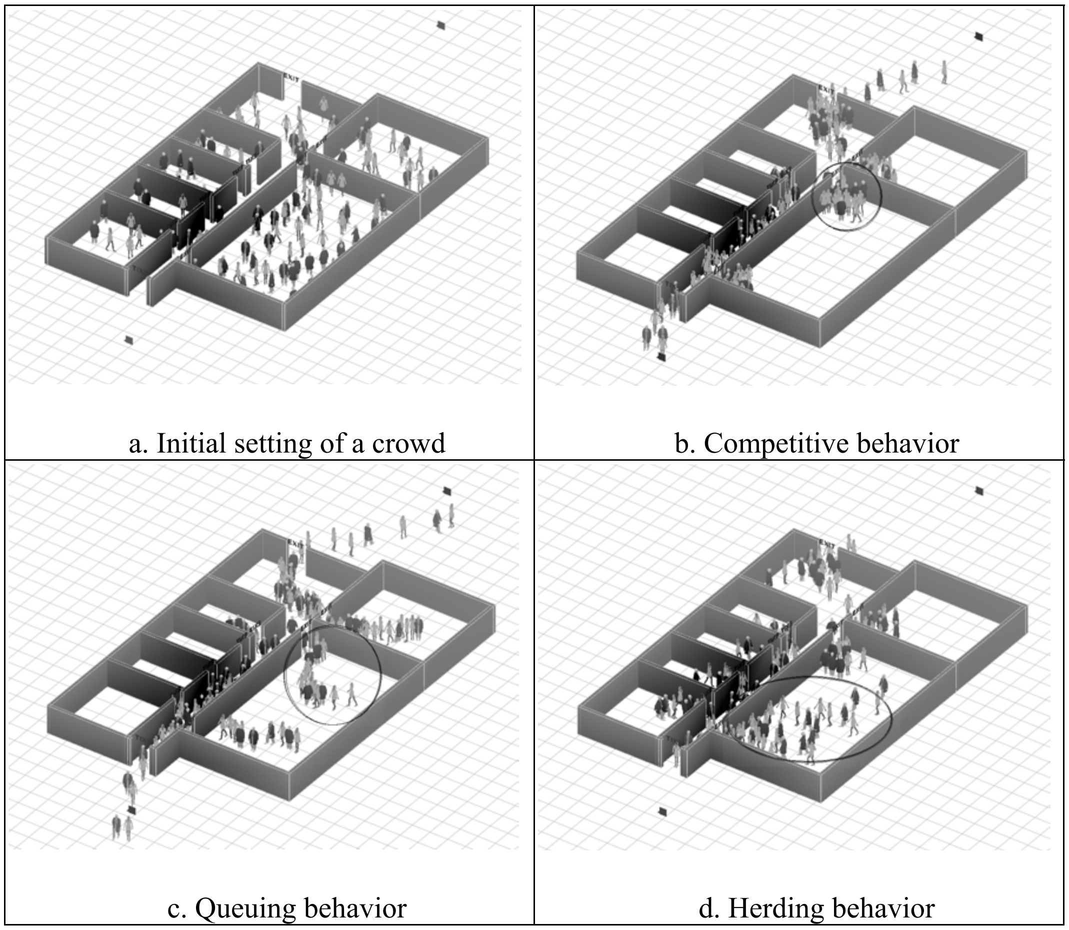


Figure 2.16: Emergent Crowd Phenomena with MASSEgress (Pan, 2006)

2.4.4 Goh's EMAS System

(Goh et al., 2006) presents a "spatial Evolutionary Multi-Agent Social Network (EMAS) to investigate the emergent macroscopic behavioural dynamics of civil violence, as a result of the microscopic interactions" between multiple goal-oriented agents. The proposed EMAS framework contains among other things a civil violence model (CVM), which consists of multiple groups of agent interacting and co-existing in a virtual world. Far from being a full-fledged 3D simulation like several other projects, this world consists of a 2D grid of cells where each cell can be empty or contain one agent. Three types of agents are specified: cops, quiescent civilians (bystanders) and actives (violent protestors).

According to the authors, “grievance and greed are modeled as the two idealized components that collectively measure the tendency of quiescent civilians to join actives in their revolt against a central authority.” Civilians become active according to the mathematical model based on grievance and greed. Cops can put actives in “jail”, effectively removing these agents from the simulation for a certain number of iterations. “Jailed agents will either be converted back to quiescent state or revert back to active state with a certain probability after release”. “A successful arrest is made when a cop wins an [iterated prisoner’s dilemma] game set against an active within the neighborhood of interaction.”

The “position of each agent at the next time episode is determined by the current position of that agent and also the current states of all neighboring cells.” Several movement strategies have been implemented and they are reproduced in Table 2.2.

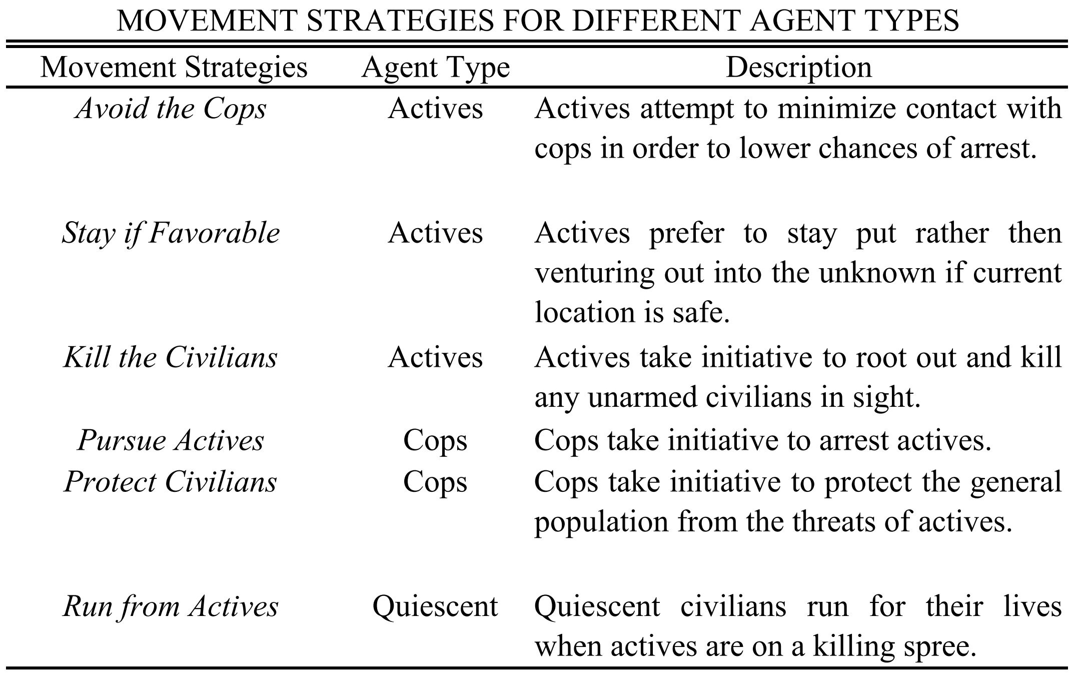


Table 2.2: Goh’s Movement Strategies for Different Agent Types (Goh et al., 2006)

MOVEMENT STRATEGIES FOR DIFFERENT AGENT TYPES

Movement Strategies	Agent Type	Description
Avoid the Cops	Actives	Actives attempt to minimize contact with cops in order to lower chances of arrest.
Stay if Favorable	Actives	Actives prefer to stay put rather then venturing out into the unknown if current location is safe.
Kill the Civilians	Actives	Actives take initiative to root out and kill any unarmed civilians in sight.
Pursue Actives	Cops	Cops take initiative to arrest actives.
Protect Civilians	Cops	Cops take initiative to protect the general population from the threats of actives.
Run from Actives	Quiescent	Quiescent civilians run for their lives when actives are on a killing spree.

“Learning is carried out independently by agents to improve their performance based on some form of heuristics that utilize domain-specific information available at hand or from previous experiences.” “Agents with weaker strategies learn from stronger ones by adopting some of the better traits. Overall fitness of each population is increased as strategies become increasingly more competent with each elapsed generation.” The authors use genetic learning with a 14-bit chromosome that is described in detail in the paper.

The authors have run several interesting scenarios and “experimental results reveal the onset of global emergent phenomena as well as interesting patterns of group movement.” For example, “an intuitively effective cop strategy is thus one that space cops in strategic positions that will prevent the formation of huge clusters.” The authors have also tried varying the number of cops in the system. The results show that “with higher cop density, actives undergo greater contemplation before deciding to show their discontent publicly.” The authors also investigated the influence of “defectors and charismatic leaders” and the influence of varying “jail terms”.

2.5 Military Simulation Models

This section presents models of control forces that have been developed for multi-agent simulations. The first sub-section starts by discussing important concepts that must be simulated, but that are not yet found in all models.

2.5.1 Important Concepts to be Simulated

Simulating control forces is simpler than simulating a crowd because control force officers tend to be less autonomous and unpredictable than crowd members. Nonetheless, modelling control forces is a complex process. Control forces are organised according to a well-defined hierarchy (e.g. commander -> squad leader -> squad member). Cooperation and communication between control force members and groups is fundamental. Orders and instructions are defined in doctrines and policies and they are communicated from commanders to executors. Commanders normally give orders to their units verbally or using hand signals.

Unlike military operations, where the goal is the elimination of the adverse party, crowd control situations require the control forces to help civilians accomplish their goals in a lawful manner. This help can come in many forms, including blocking streets, preventing violent outbursts, and escorting crowd leaders. Thus, “tactical units should follow a non-lethal weapons hierarchy, whenever possible. This means that they should initially use non-lethal measures for passive defense, to hold crowds at a distance. If violent behavior occurs, tactical units may increasingly resort to more active and greater levels of non-lethal force.” (Kenny et al., 2001) Indeed, control forces follow a force continuum that is deeply related to the context in which they are involved (e.g. size and composition of the crowd, purpose of the crowd, cooperation of crowd leaders with control forces, etc.) (Kenny et al., 2001; Lorenz, 1996). One example is shown in Figure 2.17 and another one was provided by the Québec City police department (but cannot be reproduced).

At the top of Figure 2.17 are lethal weapons that are not of interest for this thesis. The middle of the scale suggests the use of non-lethal weapons, while the bottom relates to the position and stance of the officers. Thus, in addition to communication and selection of weapons, the problem of deployment remains one of the multiple issues to examine when modelling control forces. In fact, the environment in which a situation takes place is a key element in the tactic and strategic choices of control forces. For example, crowd control formations may be employed to disperse, contain, or block a crowd. If the situation is serious, the commander may consider employing a mix of batons and tear gas. However, commanders must realize the limitations of formations, which are not the answer to all civil disturbance situations. The crowd context and the environment must be taken into account, and therefore modelled, for plausible simulation of control force decision-making.

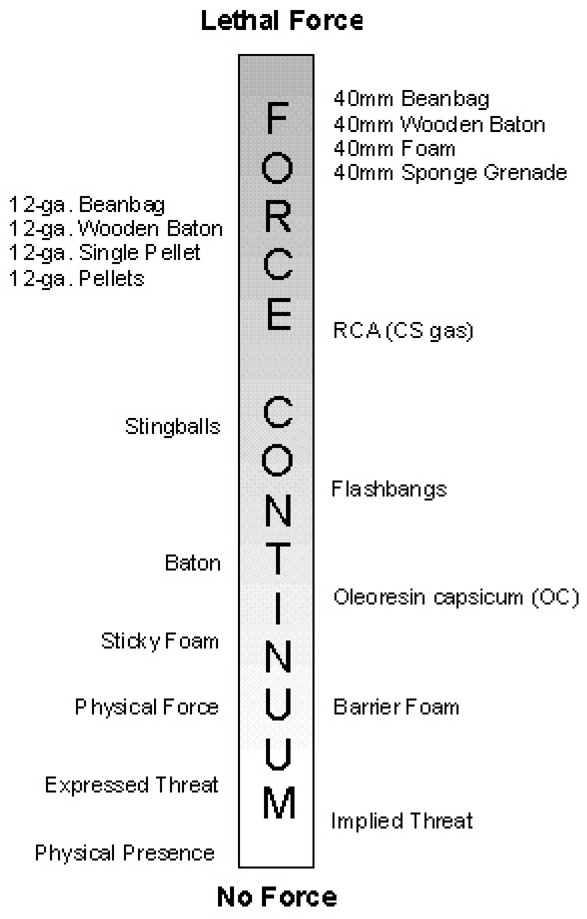


Figure 2.17: The Force Continuum (Lorenz, 1996)

Past civil disturbances have shown that frequently used formations are the line, the wedge and the echelon. The line formation is used more often because of its offensive and defensive applications. As an offensive formation, the line is used to push back the crowd. As a defensive formation, it is used to hold the crowd or to deny access to restricted areas. The wedge is an offensive formation that is used to penetrate and split crowds. The echelon is an offensive formation used to divert the crowd by moving it away from buildings, fences, and walls. The usual distance between members in formations is approximately 75 cm, although this value is adjusted for particular situations. In particular, some police forces find that a larger interval makes formation members less vulnerable to thrown objects (Moulin et al., 2007).

In addition to protecting the public, the goal behind the doctrines is to create a psychological effect on the crowd. For example, commanders may increase or decrease the cadence and adjust the distance between control forces members to impress rioters. Also, putting on full protective gear is sometimes avoided because it can appear as an aggressive move to some crowd members.

Various police departments and armies provide their doctrine to the public. A few interesting ones are (International Association of Chiefs of Police, 2001), (Oakland Police Department, 2004), and (Commonwealth Human Rights Initiative, 2005).

2.5.2 McKenzie's Crowd Federate

McKenzie and his team worked on a project (McKenzie et al., 2004, 2005b,a) that had several similarities with the Crowd-MAGS project. Their goal was to develop an application based on a set of models capable of military simulation with a credible psychological basis for the crowd behaviour. The absence of models of crowds in military simulations, and the need to integrate them in urban simulations was their main motivation. Their project was split in two phases, the first one consisting of "three parts: a requirements analysis to identify military simulation crowd modeling requirements, a literature survey to examine psychological research relevant to crowd modeling, and a design study to explore design issues in the implementation of a crowd simulation" (McKenzie et al., 2004). The second phase is detailed below (reproduced from (McKenzie et al., 2004)).

1. *Crowd federate implementation:* Design and development of a simulation that generates and controls crowd members, is interoperable with existing military simulations via HLA, and has a reconfigurable architecture to allow later replacement of its component models.

2. *Cognitive model development:* Acquisition of psychological information describing the behavior of crowds via both literature review and direct psychological research, the development of a computational model of crowd member behavior based on the psychological information, and the integration of that model into the crowd federate.

3. *Requirements analysis continuation:* Continuation of the process of identifying requirements for crowd modeling in a military simulation.

4. *Historical survey:* Study and analysis of historical incidents where crowds had a significant effect on the course or outcome of military engagements.

5. *Reference scenarios:* Development of documented, historically accurate scenarios in a military simulation of historical events involving crowds, for testing and validation of the crowd federate. This includes the development of geo-referenced terrain.

6. *Experiments:* Conduct of two experiments planned to test the crowd federate, the first one assessed the needed level of crowd behavior fidelity, and the second one tested the architectural reconfigurability of the crowd federate.

7. *PMFserv (Performance Moderator Framework) evaluation:* Independent evaluation of a psychological model based on performance moderator functions.

The Crowd-MAGS project also was divided into two phases, with tasks very similar to those in the Crowd Federate. It thus seems like a useful guide. The Crowd-MAGS research methodology, presented Section 1.4, was inspired from McKenzie's project.

(McKenzie et al., 2005b) has the potential of being highly useful for the Crowd-MAGS project for two reasons. First of all, the article defines sets of basic behaviours (actions) according to aggressiveness categories. Table 2.3 "details the aggression level continuum used to map the crowd's mental state, captured by an aggression level variable, with possible crowd behaviors." (McKenzie et al., 2005b)

Table 2.3: McKenzie's Aggression Level / Behavior Continuum Table (McKenzie et al., 2005b)

Aggression Level	**Behavior**
Neutral	Wandering
Avoidance	Fleeing Hiding
Curious non-aggressive posture	Hanging out / watching Standing / climbing on elevated structures Seeking to burning tire/smoke Chanting / flag waving
Aggressive posture	Burning tires Building barricade Swarming toward helicopter Taunting / yelling Carrying sticks / raising firearms
Aggressive non-lethal actions	Throwing rocks / projectiles Pushing / shoving Hand-to-hand fighting Dragging bodies through streets
Violent lethal actions	Firing RPGs Shooting Throwing Molotov cocktails Driving "technicos" (civilian vehicles with mounted machine-guns)

This list will be helpful in determining what types of behaviours should be modelled in the Crowd-MAGS project. Another useful part of (McKenzie et al., 2005b) is a list of stimuli that could affect agents' behaviours. This list could serve as a basis for the types of actions and events to be simulated in the Crowd-MAGS project. The list is presented in Table 2.4.

McKenzie's application, called the Crowd Federate, is implemented as a distributed simulation federate that is interoperable with existing military simulations. In particular, it supports crowd members and military police unit's interactions into a set of conflict scenarios. (McKenzie et al., 2005a) presents simulation results obtained after a series of experiments (simulation runs) in which the Crowd Federate demonstrated specific capabilities. For example, a crowd of 100 people stopped the progression of a control force vehicle. Moreover, for a crowd of 200 members, the results demonstrated an interaction with military police entities, which resulted in control forces firing into the crowd, leading to an increase in crowd aggression level.

Table 2.4: McKenzie's Stimuli (McKenzie et al., 2005b)

Stimuli Type	**Stimuli**
Sensory	See nearest member of the control force See nearest friend See nearest member of a group self is sympathetic to Sense distance to above three Hear chanting Hear militia broadcasting on megaphones
Event	See burning tire Hear gunfire Hear explosion See / smell gaseous substance See crashing helicopter See / hear firefight See someone near you get shot See someone near you hit Get hit by others in crowd Get hit by a member of the control force See downdraft of helicopter See militia demonstrating weapons See militia loading weapons
State change	Leader is shot Leader runs away Lose ability to communicate amongst CMs Appearance / actions of organized agitator group See crowd members fleeing Feel that crowd is getting "crowdier" / seeing individuals joining the crowd

2.5.3 Varner's SULNT

Varner and her colleagues worked on a project that seems very similar to Crowd-MAGS (Varner et al., 2000). Indeed, the US Defense funded research to help the Marines get properly instructed and prepared to use non-lethal weapons in peace-keeping and crowd control operations. The models and software application seem quite relevant to the Crowd-MAGS project's goals. Also, Varner's project "concluded with a successful test and demonstration" with military personnel. Unfortunately, only brief information is available about Varner's project. This sub-section presents the most relevant and precise information available.

"A prototype Small Unit Leader Non-Lethals Trainer (SULNT) has been developed that models a peacekeeping mission in a generic urban environment. The SULNT is a constructive simulation that provides a training environment for the situational assessment and decision making skills that must be used by the squad leader" dealing with a crowd control scenario.

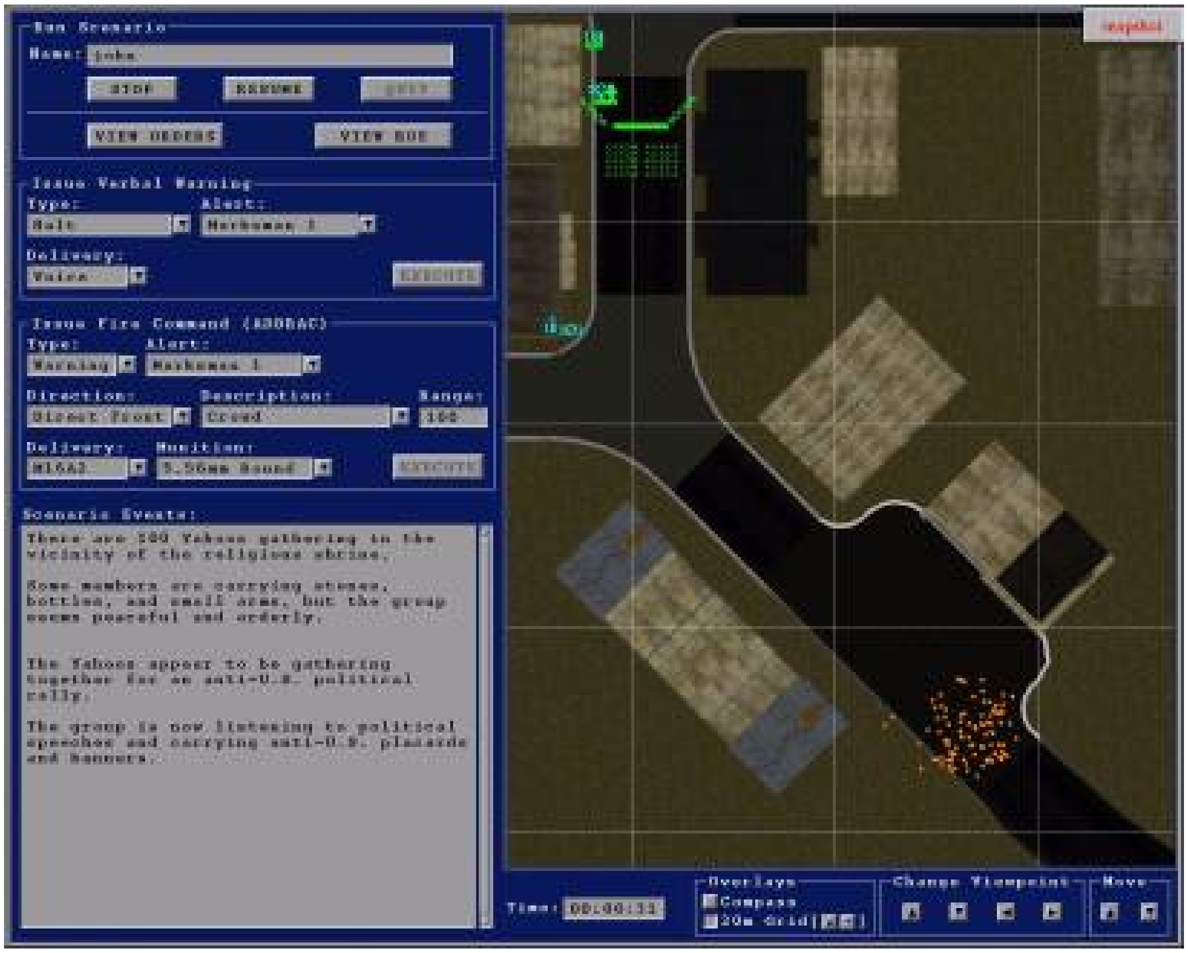
SULNT was “a visual, scenario-based computer simulation that was highly interactive. The simulation included models of the effects of specific non-lethal munitions [...] and also of crowd and mob activities.” The trainees (Marines students) “were able to demonstrate their knowledge of proper rules of engagement, the procedures for dealing with crowds and mobs, and their ability to make decisions about the appropriate level of force needed to control, contain, or disperse crowds and mobs.” The system contained “three realistic 3D models of towns and buildings” in which crowds moved “along instructor-defined pathways and responded not only to actions taken by Marines but also responded autonomously to actions by other simulated crowds and to the passage of time.” Crowds were based on “historical and generic ethnic and religious groups” and ranged from 2 to 500 members.

More specifically, “each crowd faction was characterised by the values of a series of attributes which together comprised a crowd profile. Attributes included [...] arousal state, prior experience with non-lethal munitions”, armament level, “degree of fanaticism and devotion to their [...] causes”, “and attitudes toward the Marines (fear, respect, anger, etc.), among others. The attribute profiles were linked to a behavioural model that generated crowd activities such as” loitering, celebrating, carrying signs, throwing stones, demonstrating, rioting, and dispersing. “The link came through a set of Boolean relations shaped by our understanding of the literature and validated by our experts. Crowd movement was determined along initial paths defined by an instructor. However, actions taken by the trainee and other scenario events could influence crowd movement as well.”

“In a typical scenario, the trainee would deploy his squad in fire teams, reserves, and designated marksmen.” Also, the trainees “were able to deploy model barriers and fortifications and were constrained to realistic levels of ammunition and squad strength.” “The Marines were stationary for the most part and were in a reactive mode”. However, “the trainee could issue verbal orders to the crowd or call in lethal or non-lethal fire. As in the real world, the supply of clips and ammunition is limited, range is restricted, and munitions effects are probabilistic rather than absolute. The actions taken by the Marines affected crowd behaviour in several ways. Some crowds could be dispersed by a simple verbal command, whereas other crowds would disperse only with swift and heavy action from the Marines.”

A useful scenario model surely was devised. Indeed, ‘a Marine could run the same scenario several times and each run would have a separate outcome.” “During an exercise, all of the trainee and simulated crowd actions are logged to support an after-action review (AAR) process.” This process “allowed a trainee and his instructor to replay a scenario run with all actions preserved. The replay could be paused or fast forwarded to specific elapsed time markers so that an instructor could make a teaching point. Scenarios could also be saved for re-use.”

The user interface allowed a trainee to see a “a 2D plan view of the area”, “descriptions of events, his orders and background information, and rules of engagement”, as well as “verbal feedback from an electronic speech synthesiser.” In addition, a scenario specification module was available for instructors to create and edit scenarios. An example of the run-time interface is shown in Figure 2.18.



Name: john

Type: Halt
Alert: Marksman 1
Delivery: Voice

Type: Warning
Alert: Marksman 1
Direction: Direct Front
Description: Crowd
Range: 100
Delivery: M16A2
Munition: 5.56mm Round

Scenario Events:

There are 100 Yahoos gathering in the vicinity of the religious shrine.

Some members are carrying stones, bottles, and small arms, but the group seems peaceful and orderly.

The Yahoos appear to be gathering together for an anti-U.S. political rally.

The group is now listening to political speeches and carrying anti-U.S. placards and banners.

Time: 00:00:11

Figure 2.18: SULNT's Run-Time User Interface (Varner et al., 2000)

2.5.4 Woodaman's AgentKit

(Woodaman, 2000) presents a simulation project for crowd control scenarios. The author used a software package called AgentKit. BasicCombatAgent (Figure 2.19) is a module of AgentKit and it is responsible of agent creation and animation according to behavioural models. The agents are distinguished by their roles, which are a function of the agents' behavioural rules and resources. The specified roles are *Rioter*, *Riot Leader*, and *Peacekeeper* (control force member). Woodaman modelled "the individuals in the crowd as being thermodynamic particles possessing different states as a function of their temperature. The rioters generate, emit, and absorb heat via their heat radiator effectors. Groups of rioters start in a cool state but become 'hotter' the longer they congregate until individuals 'boil over' into a hot, violent state." "The heat the agents receive from other agents is in inverse proportion to the square of the range. Thus, the rioters only got hot when several got in close proximity of each other."

According to Woodaman, rioters behave like crowd members in that they act scared when alone, but find strength in numbers. Rioters have two states: passive and aggressive. The current state is a function of the agent's temperature and thus a rioter's implied goal is to

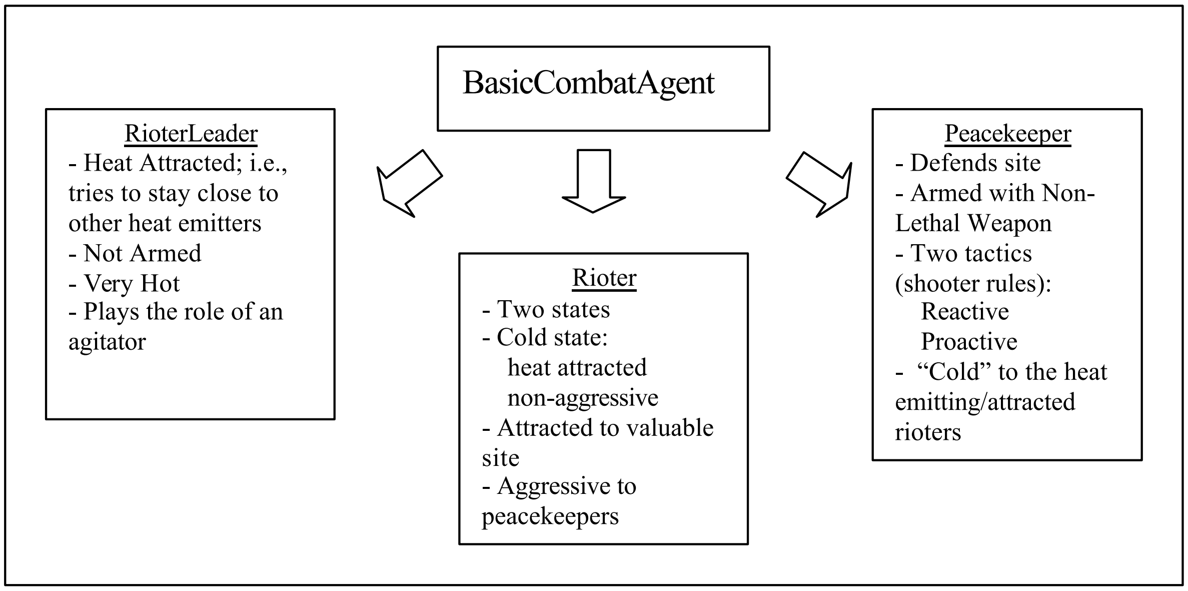

BasicCombatAgent
RioterLeader
- Heat Attracted; i.e., tries to stay close to other heat emitters
- Not Armed
- Very Hot
- Plays the role of an agitator
Rioter
- Two states
- Cold state: heat attracted non-aggressive
- Attracted to valuable site
- Aggressive to peacekeepers
Peacekeeper
- Defends site
- Armed with Non-Lethal Weapon
- Two tactics (shooter rules): Reactive Proactive
- "Cold" to the heat emitting/attracted rioters

Figure 2.19: BasicCombatAgent and the Three Specified Roles (Woodaman, 2000)

maximize his temperature when in the passive state and to keep it above the “boiling point” when in the aggressive state. Rioters engage peacekeepers with rocks with mathematical functions that define shooting accuracy and impact. Some functions modify the agent’s heat but unfortunately the values used in the functions are arbitrary. Riot leaders are always in the passive state but their heat is always at a high level. Other rioters are attracted by this heat. On the other hand, peacekeepers are “cold” and thus repel rioters. Peacekeepers can use a reactive or a proactive strategy. They are allowed to use non-lethal weapons in reactions to the rioters. They are modelled as wearing a helmet, a face shield, and a bullet-proof jacket. This protection allows them to die after five stone hits, while rioters may die more “easily”.

The author simulated the encounter between a small unit of peacekeepers and a rioting crowd in scenarios in which the control forces are guarding a valuable site against the attack of an aggressive crowd. The crowd evolves from an initial peaceful state to a violent state, in which they attack the control forces. The results show the number of hits on the three types of agents and analyses the best strategy for different types of crowd.

2.6 Other Models

Several other papers have the potential of being useful for the Crowd-MAGS but they are not directly related to crowd control simulations. For example, some models are purely theoretical and have not been implemented in a simulation project. Another example is the case of simulation of pedestrian flows where the emphasis is put on the phenomena that emerge from the interactions of a large number of pedestrians moving in a fairly constrained space, such as sidewalks and corridors. In such simulations, queuing and congestion phenomena can be studied in relation to the dimensions of pathways, the presence of obstacles, and other facilities channelling the pedestrian flows. Thus, this section presents papers that share common problems and solutions with crowd control simulations.

2.6.1 Tajfel's Social Identity Theory

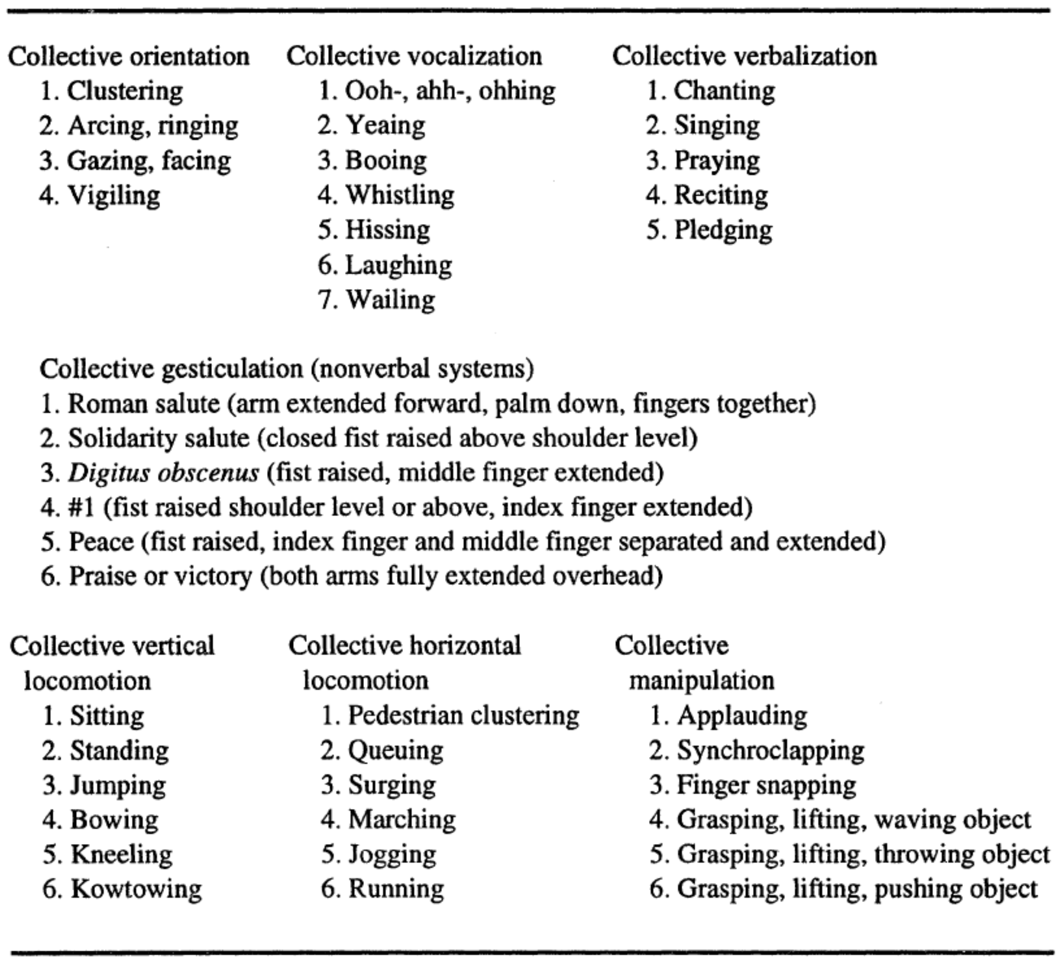
The Social Identity Theory, first introduced in (Tajfel, 1978), assumes that an individual's identity is multiple and that it constitutes a complex system rather than being unitary. A distinction has been made between *personal identity*, which refers to the unique characteristics of an individual, and *social identity*, which refers to an individual's self understanding as a member of a social category. Ample literature, both theoretical and experimental, exists to support the social identity theory. In particular, Stephen D. Reicher and John Drury have presented several influential papers over the years.

As an example, (Drury and Reicher, 2005) presents a field study of two crowd control events that emphasize empowerment through collective actions. The paper also describes the Elaborated Social Identity Model of crowds (ESIM), which involves a revision of some of the basic terms of the social identity tradition. It "suggests that newly empowered definitions of self emerge from conflictual interactions between groups. According to the ESIM, one's social identity entails an understanding of one's position within a set of social relations along with definitions of possible and legitimate action flowing from that position." "The ESIM posits two features of intergroup dynamics which are necessary for a change towards empowerment among participants in collective action: first, asymmetry in the stereotypes held of each other by movement participants and such external forces as the police and, second, an (initial) asymmetry of power relations" (e.g. the police uses excessive force and treats all crowd members in a similar manner).

On the experimental side, a typical pattern of identity change has been observed in several studies involving different types of crowd events, including football matches, student demonstrations, tax protests and environmental protests. In this pattern, moderate participants of a crowd change identity and become 'activists' as a result of police actions perceived as being illegitimate. For example, a study has shown that police officers seriously consider the social influence of their actions during crowd control events (Cronin and Reicher, 2006). Concerned about accountability, senior officers must display a balance of tolerance and discipline to avoid that too many "crowd members shift from personal to social identity in the crowd, and that control of behaviour passes from personal concerns to the norms, values, and beliefs associated with the relevant social category." Therefore, perception and interpretation of actions has an important influence on the participants' behaviours.

2.6.2 McPhail's Collective Actions Classification

In (Schweingruber and McPhail, 1999), the authors "provide a set of criteria and procedures for systematically observing and recording collective actions" in crowd control situations. They also list more than 40 elementary collective actions grouped in seven categories. The data have been created "from extensive prior observations of temporary gatherings. The data collected provide a rich record of collective action across space and time." "Although this list of elementary forms may not be exhaustive, it is empirically grounded." The list is presented in Figure 2.20. The authors also reorganized the actions around four regions of the body. The results is shown in Figure 2.21. 'Facing' refers to the general direction of the body, 'voicing' refers to the mouth, 'manipulating' refers to the hands, and 'locomotion' refers to the legs.



Collective orientation
1. Clustering
2. Arcing, ringing
3. Gazing, facing
4. Vigiling

Collective vocalization
1. Ooh-, ahh-, ohhing
2. Yeaing
3. Booing
4. Whistling
5. Hissing
6. Laughing
7. Wailing

Collective verbalization
1. Chanting
2. Singing
3. Praying
4. Reciting
5. Pledging

Collective gesticulation (nonverbal systems)
1. Roman salute (arm extended forward, palm down, fingers together)
2. Solidarity salute (closed fist raised above shoulder level)
3. *Digitus obscenus* (fist raised, middle finger extended)
4. #1 (fist raised shoulder level or above, index finger extended)
5. Peace (fist raised, index finger and middle finger separated and extended)
6. Praise or victory (both arms fully extended overhead)

Collective vertical locomotion
1. Sitting
2. Standing
3. Jumping
4. Bowing
5. Kneeling
6. Kowtowing

Collective horizontal locomotion
1. Pedestrian clustering
2. Queuing
3. Surging
4. Marching
5. Jogging
6. Running

Collective manipulation
1. Applauding
2. Synchroclapping
3. Finger snapping
4. Grasping, lifting, waving object
5. Grasping, lifting, throwing object
6. Grasping, lifting, pushing object

Figure 2.20: Elementary Collective Actions in Seven Categories (Schweingruber and McPhail, 1999)

2.6.3 Pan's MASSEgress Framework

Pan's MASSEgress system is well described in (Pan, 2006) and seems highly relevant for the Crowd-MAGS project. Its architecture is shown in Figure 2.22 and it "consists of six basic modules: a Geometric Engine, a Population Generator, a Global Database, a Crowd Simulation Engine, an Events Recorder, and a Visualizer." These modules' descriptions are reproduced below.

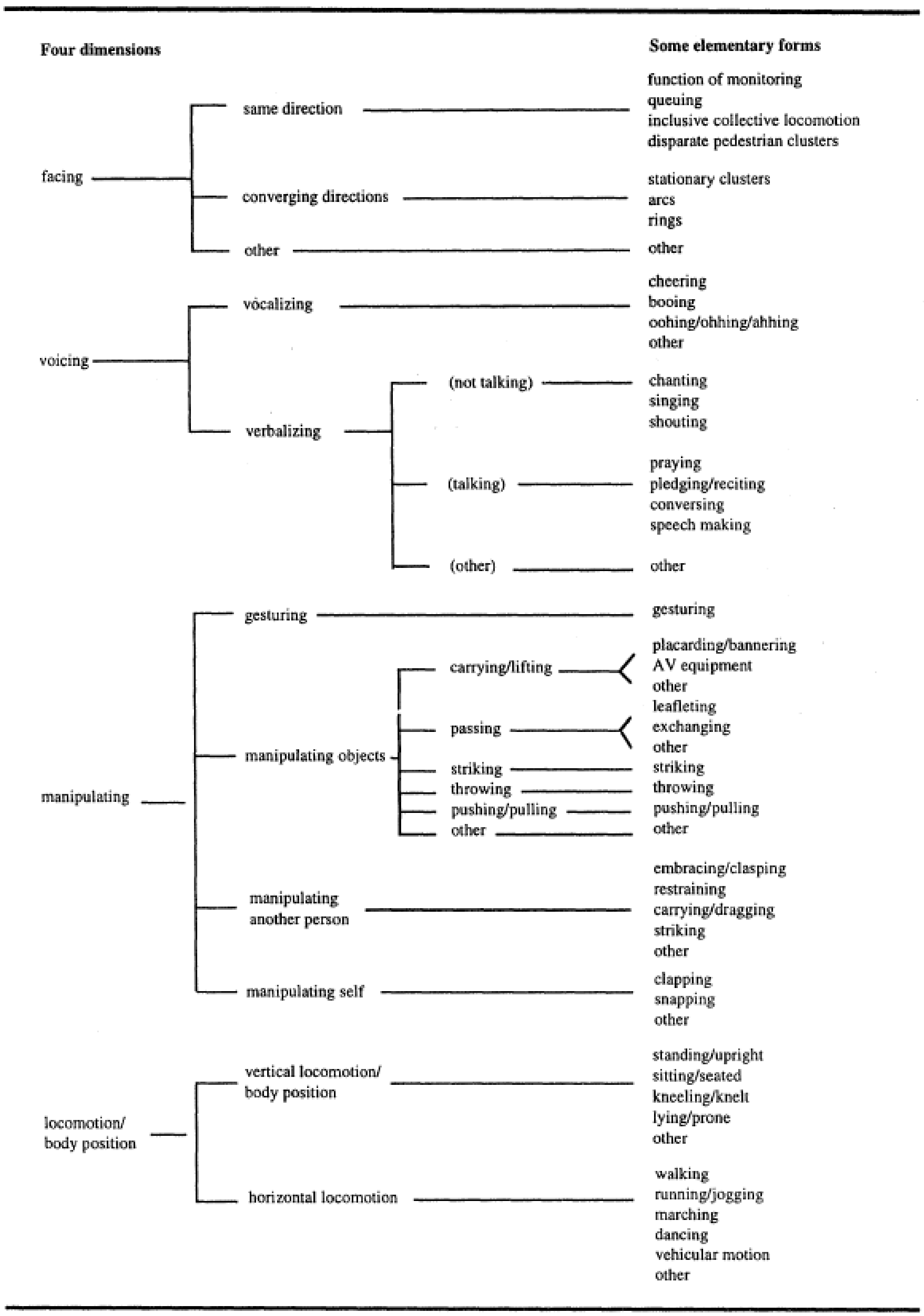


Figure 2.21: Elementary Collective Actions in Four Regions of the Body (Schweingruber and McPhail, 1999)

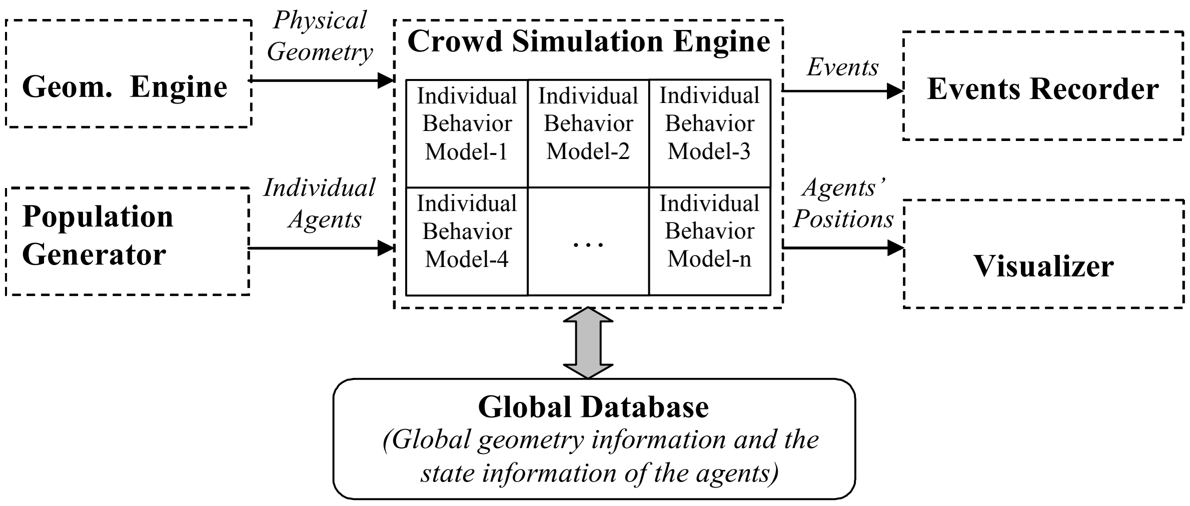

Geom. Engine
Physical Geometry
Population Generator
Individual Agents
Crowd Simulation Engine
Individual Behavior Model-1
Individual Behavior Model-2
Individual Behavior Model-3
Individual Behavior Model-4
…
Individual Behavior Model-n
Events
Events Recorder
Agents' Positions
Visualizer
Global Database
(Global geometry information and the state information of the agents)

Figure 2.22: MASSEgress' Architecture (Pan, 2006)

- The *Geometric Engine* generates the geometries representing the physical environments (e.g., a building or a train station, etc.). Spatial information, including obstacles, exits, spaces, spatial layouts, exit signs, etc., is most conveniently defined using CAD tools such as AutoCAD or Architectural Desktop (ADT).

- The *Population Generator* generates virtual agents to represent a crowd based on a distribution of age, mobility, physical size, type of facility (hospital, office building, train station, stadium, etc.) and other human factors. The population, its composition, and occupants' behavior would be different for different facility types. This module allows the user to easily generate occupants and specify space assignments.

- The *Global Database* maintains all the information about the physical environment and the agents during the simulation. It maintains the state information (mental tension, behavior level, location) of the individuals. The database is also used to support the interactions and reactions among the individuals.

- The *Events Recorder* captures the events that have been simulated for retrieval and playback. The simulated results can be recorded for further analyses, for example, to derive evacuation patterns and statistical information. The events captured can also be used to compare with known and archived scenarios.

- The *Visualizer*, which is implemented using OpenGL, receives the positions of agents, and then dynamically generates and displays 2D/3D visual images.

- The *Crowd Simulation Engine* is the key module of the multi-agent simulation system. Based on the behavior models and classified rules, each agent is assigned with an Individual Behavior Model based on the data generated from the population generator.

2.6.4 Wijermans' Rioting Factors

In (Wijermans et al., 2007), the authors have identified several factors that may influence a crowd and lead to riot behaviour.

- **Factors related to the physical environment in which the crowd event takes place:** human density, temperature, noise, and scent
- **Factors related to the social environment:** in/out group perception (tendency to selectively favor the in-group and diapprove of the out-group), friendship (unique relations that an individual has with his social surroundings), leadership (people exerting a larger influence on others), and social network (an individual's connections and ties)
- **Physiological factors that affect the individual:** arousal, energy, and the use of alcohol and drugs
- **Functional factors that affect the individual:** saliency (relates to the phenomenon where something, such as norms or identity, is in focus and therefore has more influence than other norms/identities), activation level of some memory elements (implies that in showing aggressiveness/violence, the elements representing these behaviours have to be highly activated), and insecurity (relates to a situation where no behavioural rules is known in terms of experience)

Although not mentioned directly, these factors have probably influenced the development and the experimentation of their crowd simulation, presented in Sub-Section 2.3.7.

2.6.5 Silverman's EthnoPolitical Game

(Silverman et al., 2006a) presents a socio-cultural 'game' of inter-group competition for control of resources. The 'health' of the groups can be measured with: 1) political goods (e.g. jobs, money, training, healthcare), 2) level and type of security available to impose will on other groups, and 3) popularity and support for the leadership as voted by the group's members. Group leaders must interact with the following groups: loyal in-group, resistant out-group, and those "undecideds" that might be turned into allies. "Also, if there are other groups, they are examined to determine how they might be enlisted to help influence or to defend against the outgroup and whatever alliance it may have formed." A set of actions are available to leaders, such as *motivate*, *threaten*, and *form pact*. "Followers' actions are to support their leader's choices or to migrate toward another group they believe better serves their personal value system." In the end, "leaders who choose successful policies will remain in power, provide benefits to their followers, and ward off attackers." Analysts and trainees play the role of a leader and have similar constraints as 'agent-leaders'. Each agent-leader is modelled within the PMFserv framework (introduced in Sub-Section 2.3.4). It should be noted that this model has not been applied to crowd simulation.

2.6.6 Jager's Clustering and Fighting Simulation

(Jager et al., 2001) presents a multi-agent system to simulate clustering and fighting behaviours of two-party crowds. The authors created a 2-D environment composed of a 100 X 100 cells of 1 m^2 that can each contain a maximum of one agent. Three types of agents have been defined: "hardcore, hangers-on and bystanders. These different types of agents scan their environment with a different frequency". Simulation iterations represent one second and simulations last 1.5 hours. The authors also provided agents with simple rules based on the recognition of own-party agents and other-party agents. "The agents' internal state is called their 'aggression motivation', i.e. their tendency to approach or avoid and, in certain cases, their tendency to start a fight."

The goal of the simulation was to study the effects of group size, size symmetry and group composition on clustering and fights. Jager and his colleagues are "aware of the simplicity of the approach-avoidance rule that guides the behaviour of our agents. It is however surprising that with so modest a psychological process, the outcomes so much resemble real world phenomena." The three main conclusions are presented below.

- "Larger groups tend to cluster more than small groups"
- "Clustering develops faster and to a greater extent for asymmetrical groups than for symmetrical groups"
- "The risks of fights going out of control is greatest in situations where the crowd is large and the two parties have a different size. Here the number of hard-core group members seems to play a less important role than the sheer fact of the large numbers and asymmetry."

The authors also present four interesting recommendations for further research.

- "Enrich the psychological makeup of our agents. It would for instance be possible to equip the agents with different needs (e.g., subsistence, protection (safety) and identity (status)). We could try our hand at modelling fatigue"
- "Study how the shape of the space interacts with the behavioural dynamics " because "in real life situations the space in which riots take place is much less simple. There are streets, stadiums, buildings and various obstacles, none of which appear in our simple square."
- Study the impact of pre-event organization of the parties: "another real life factor that did not enter our simulation is the fact that quite often one of the parties is much more organised than the other", such as the case of control forces.
- "Validate the resulting behavioural dynamics against existing empirical data." The authors recommend collecting empirical data on the stages preceding aggressiveness in riots, and on "the motives and decision processes of people to engage in a fight or not. Such data may be very valuable in improving the simulation model, both for improving our knowledge on the behavioural dynamics in crowds, as for the development of intervention techniques that avoid the emergence of incidents."

2.6.7 Osaragi's Pedestrian Flow Model

(Osaragi, 2004) presents a model of pedestrian flow that examines "relationships between pedestrian behavior and pedestrian-usable space within a modeled space". "The repeated appearance and disappearance of these walk-groups is an important feature of pedestrian flows." Osaragi's system can automatically extract pedestrian groups from his simulations by using relative velocity, relative angle and relative distance. Simulations have been run for each pedestrian behaviour attribute (e.g. males, females, adults, elders) under high-density conditions for 30 seconds.

2.7 Simulation Tools and Platforms

Many models were presented in the previous sections and it is legitimate to ask if they can all be integrated into a single framework. In fact, some of these models are only theoretical (e.g. Kenny's psychological factors) while some others are integrated in systems of various sizes (e.g. Thalmann's ViCrowd, Silverman's PMFServ, Pan's MASSEgress). Moreover, other simulation systems exist, although they might not be directly related to crowd control simulation.

In order to select an appropriate development platform for the Crowd-MAGS project, (Moulin et al., 2007) presents a comprehensive review of platforms, systems, and tools that could be useful in building a system that meets the requirements of the Crowd-MAGS project. These technical requirements are listed below.

- **Modelling the environment:** data have to be realistic and obtained from a GIS, specific objects (e.g. fences, urban furniture, non-lethal weapons) may be modelled, the simulated environment may be modified dynamically
- **Modelling individuals:** several types of agents may be modelled (e.g. bystanders, crowd participants, control force officers), characteristics of individuals may be modelled (e.g. physical and psychological aspects), and agents may interact with objects.
- **Modelling groups:** different kinds of groups may be modelled (e.g. sub-groups, social groups, emerging groups)
- **Modelling interactions:** between individuals, between individuals and objects, between individual and groups, and between groups.
- **Modelling scenarios:** different populations may be specified, events may be simulated (e.g. explosion, fire)
- **Data collection:** information for assessing the efficiency of the intervention strategies may be collected

The solutions examined have been classified into five categories, as presented below.

- **Military simulation:** FLAMES, Sextant Framework
- **Visualization and animation:** Massive, Crowd IT
- **Academic and open-source tools:** Crowd Simulation framework (CSF), Delta 3D, MAGS, SDKsim, Crowd Federate, PLAMAGS
- **Pedestrian flow simulation:** Myriad II, Exodus, PedGo, Pedroute, AENEAS, SimWalk
- **Others:** Pedsim, Horde 3D, Brahms, SeSAm, Pedestrian simulator, Anylogic

The review showed that none of these tools fulfilled the requirements mentioned above. However, some tools that showed a good potential for further development or enhancement were identified. These tools are CSF, SDKsimMAGS (based on MAGS), Sextent (and AI.Implant), SPIROPS, Anylogic and PLAMAGS.

2.8 Conclusions

Today, numerous and diverse organizations rely on simulation to help them to better understand their domain or to increase their efficiency. More specifically, many control force agencies are now interested in simulation to enhance the safety of their personnel and of non-adversaries, such as civilians. These agencies are becoming more and more aware of the possibilities of multi-agent geo-simulation, such as 3D visualization, sophisticated behaviour models, and micro-analysis possibilities. Since computer hardware is now fast enough to support hundreds of agents and animation packages can represent the world and the agents in a highly realistic manner, simulation users are now seeking more behaviourally and cognitively realistic agent models.

Plausible agent models must be grounded on the behavioural literature if they are to be well validated. Although very rich, the behavioural literature can unfortunately not be directly encoded into useful models and agent architectures in most cases. Some studies may reveal how certain situations may affect one aspect of a person, such as stress or fatigue; however, none provides an integrative framework that would be sufficiently detailed for multi-agent simulation. For example, integrating perceptive and cognitive capacities, behavioural decision theory, and emotions “rapidly departs from grounded theories and enters into the realm of informed opinion” (Silverman et al., 2002).

The models and frameworks developed by Silverman’s and Thalmann’s teams seem to be the most interesting ones presented in the literature. These authors assert that crowds are not the entities that classical researchers have implied. Rather, crowds are a gathering of a multitude of individuals and small groups that have temporarily assembled. Moreover, Silverman’s team has summarized recent models useful for crowd simulation to determine how they might be integrated into a common framework, and to implement and assess the value of such a framework. Efforts to model stress, emotion, and decision processes as integrated factors (as they are in real human beings) present new possibilities for improving and expanding realistic agent behaviour based on the interplay of multiple factors and settings.

Unfortunately, no approach explicitly models groups as autonomous entities that have physical and psychological impacts on agents. Indeed, although several systems are able to simulate certain aspects of the dynamics of groups in a crowd, they essentially simulate the group dynamics in a 'kinematic way'. These systems only take advantage of the geometric properties of agents moving in groups, such as distance between group members, orientations, personal space. However, there is a need for more elaborated models integrating both the individual's characteristics (psychological, emotional) and social rules/behaviours in order to explain why agents may join a group or dissociate from a group, why perceiving and interpreting the actions carried out by the members of a group may induce an agent to change behaviour or even a change of 'social identity' as some sociologists call it (Drury and Reicher, 2005). It is this need that motivated the elaboration of the model that was proposed in (Moulin, 2009) and that is the theoretical basis of this thesis.

With regards to control forces, no system convincingly models agents and groups and their interactions with CF agents. In fact, the models are too simplistic. For example, in Crowd Federate (presented in Sub-Section 2.5.2), soldiers are modelled as 'player-controlled characters'. Thus, these agents have no autonomous behaviours because their movements, orientation and speed are externally controlled by users. In fact, they can be considered as avatars rather than autonomous agents. With AgentKit (presented in Sub-Section 2.5.4), CF agents are autonomous agents but they can use only two strategies. Thus, hierarchy, formations, and a certain force continuum are not modelled at all. For the Crowd-MAGS project, developing convincing models of a crowd and control forces will require taking into account the social interactions between the different in- and out-groups and how they influence changes of individuals' attitudes and behaviours.

Some authors have described their research methodologies and offered recommendations, which could be highly relevant for the Crowd-MAGS project. For example, the phases followed for the development of Crowd Federate by McKenzie's team allowed them to grow their models and software organically as requirements evolved and data was gathered. Moreover, Jager ran several experiment types to discover the factors that could have an influence on crowd behaviour. A replication and continuation of these experiments would certainly benefit the crowd modelling literature.

The following chapter will present an overview of the solution created for the Crowd-MAGS project. In many respects, the models and methodology used are based on the conclusions and recommendations put forth by the authors whose works were presented in this chapter.

Chapter 3

Overview of the Proposed Solution

Before diving directly into this thesis' proposed models, this chapter presents an overview of the solution proposed to fulfill the objectives presented in Section 1.3. First, Figure 2.1 is revisited in relation to the Crowd-MAGS project; Figure 3.1 will be used as a reference throughout this chapter to show how the concepts to be introduced interact with one another.

Section 3.1 starts by introducing the types of events to be simulated and details about how they can be specified for the Crowd-MAGS project. Next, Section 3.2 presents the software platform and the underlying technologies. Afterwards, Section 3.3 discusses the type of information to be gathered and analysed as well as how the models have been developed with analysis in mind. Finally, Section 3.4 introduces many concepts necessary for the understanding of the models presented in Chapter 4.

The solution proposed in this thesis puts forth contributions in all four phases of the simulation methodology (presented in Figure 3.1) although the major advances are the agent and group models. The first phase in the figure is the scenario specification. For the Crowd-MAGS application, an interactive scenario editor was developed to allow users to drag-and-drop components directly into the 3D environment that they chose. Scenarios are stored in a XML format that is human- and machine-readable. Section 3.1 and Section 4.6 will present details about these features.

The next phase is the scenario execution, and the Crowd-MAGS application in itself is the most important contribution to this phase. As mentioned in Chapter 2, this application is the first one to integrate crowds, control forces, and non-lethal weapons in one simulation system. Section 3.2 will present more details about the application.

In Figure 3.1, the dotted lines towards the SD model represent an optional path. For the Crowd-MAGS project, a SD model was developed by DRDC team members and ran in parallel with the Crowd-MAGS system. The macro-level behaviour of the crowd was compared using both systems.

The following phase in Figure 3.1 is analysis. It consisted of collecting data produced by

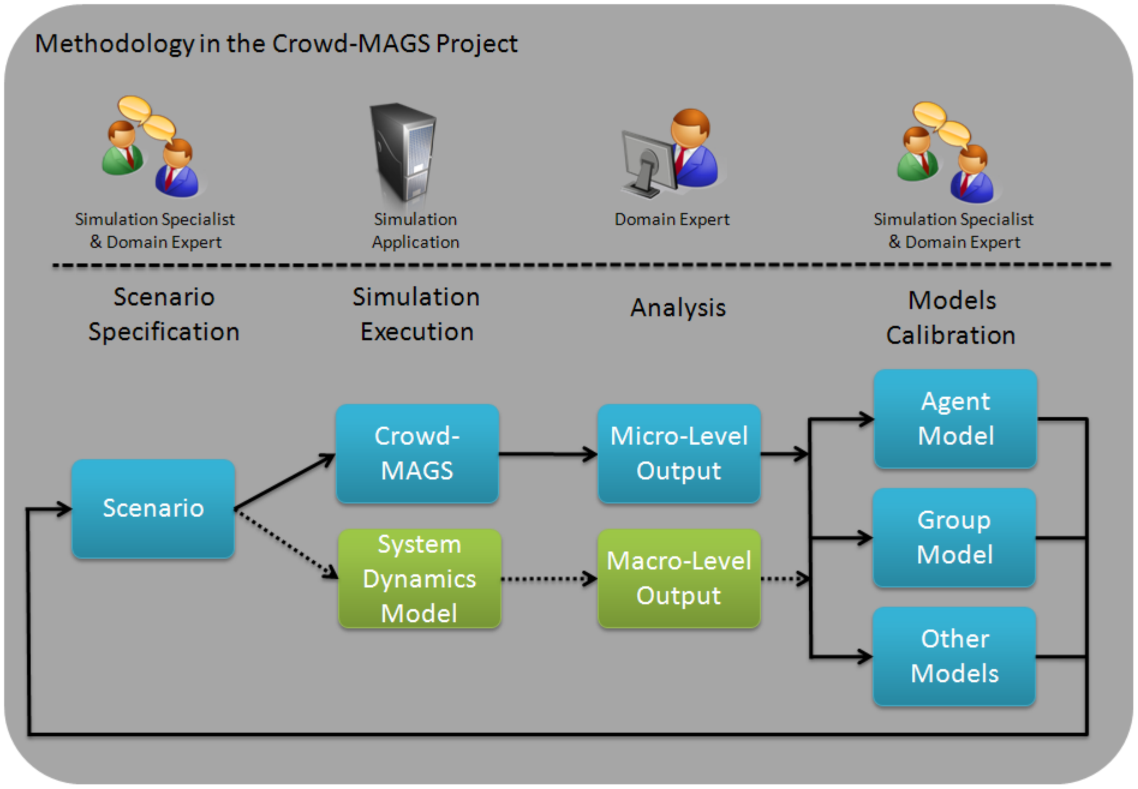


Figure 3.1: Methodology in the Crowd-MAGS Project

various scenarios and analyzing it. Comparisons between different runs of a given scenario were performed, as well as comparisons between scenarios and with the SD system. In this project, the output coming from the optional SD model was used to help analyze, validate and calibrate the Crowd-MAGS models. Section 3.3 will present more details about output analysis.

Finally, the step of models calibration is shown in Figure 3.1. The figure shows an arrow going back to the first step because simulation is an iterative process. Following a first batch of scenarios and analysis, and once the models were well enough calibrated to give a certain level of confidence, more complex scenarios were elaborated and run. Further calibration and analysis was performed, until improvement became less significant. Section 3.4 will discuss model development and calibration.

As mentioned above, this thesis addresses all four phases of the simulation methodology. Here are summarized the thesis' objectives along with the related phases.

- *Scenario Specification:* The calibration of the simulation platform (scenarios and models) to be readily used by end-users. A scenario specification and calibration tool is needed.
- *Simulation Execution:* The development of a software platform that would allow simulating the models with control forces, crowds, and non-lethal weapons.

- *Output Analysis:* The creation an information model to collect data during simulations to assess the efficiency of intervention strategies.
- *Models Calibration:* The elaboration of a model for explicitly modelling emerging groups and an agent model that would interact with these groups.

3.1 Scenario Specification

In Crowd-MAGS, a scenario is a specification of:

- the agents and other components that will be in the environment;
- the planned events that will happen during the simulation;
- parameters related to the crowd, the control forces, and other aspects;
- and finally other details like the camera viewing angle or the time of day at which the scenario will start.

As mentioned in Chapter 2, no simulation model that combines control forces, crowds, and non-lethal weapons exists. The Crowd-MAGS scenario allows the specification of all of these elements and more in a single file. This possibility allows domain experts and analysts to directly go from the idea of a scenario to a complete scenario specification file.

In order to assess the effectiveness of different intervention strategies for the control forces, many scenarios are required to analyze the positions and interactions of control forces and the crowd in different situations. In some cases, it may be required to evaluate a situation where a small police force is overwhelmed by a large aggressive crowd. At other times, it may be necessary to control a large passive demonstration with a limited number of officers. Once a basic scenario has been designed, it may be run many times, possibly with different intervention strategies every time (e.g. placing fences at different locations, using tear gas or not, changing mobilization level). Running a given scenario multiple times gives different results due to the stochasticity of some parameters and the autonomy of all agents. However, the overall assessment of the events remains similar between different runs of the same scenario.

The Crowd-MAGS application offers a scenario specification module that allows a user to quickly create scenarios and to edit them in detail without requiring any technical knowledge. Section 4.6 and Sub-Section 5.1.9 will discuss the scenario specification model in complete details; Sub-Section 5.5.5 will show the scenario editor.

3.2 The Crowd-MAGS Application

The Crowd-MAGS application is a software tool aimed at simulating the interactions of crowds and control forces in the presence of non-lethal weapons. The application was developed along-side of the proposed agent and group models, and thus these models represent a significant part of the application's core. Other models also are integrated, such as the ones for knowledge representation, non-lethal weapons, and scenario specification.

3.2.1 Application Features

In addition to executing the simulation, the application offers a scenario specification module and a graphical user interface (GUI) with widgets that enable a user to interact with the agents and the simulation while it runs. In fact, agents and groups possess a history of events, which logs two types of events: those from the system and those from the user (e.g. a user dragging an agent). All system and scenario parameters can be adjusted in the application or directly in the scenario file.

Section 3.3 explains the analysis that was performed on simulation outputs. At first, generic applications such as Vensim and Microsoft Excel were used and the need for other analysis tools never arose. Thus, one of the features that were not developed in the Crowd-MAGS system is a set of specific tools for analyzing the simulation data.

Chapter 5 will present the application in more depth: technical details about the application, the user interface, scenario specification, calibration, and the generation of simulation data.

3.2.2 Programming Languages Used for the Application

This sub-section presents the main three programming languages that were used to build the Crowd-MAGS system. The commercial languages are briefly introduced while the in-house language, PLAMAGS, is described in more details.

Java

Most of the Crowd-MAGS application was developed in Java. More specifically, the user interface, the scenario editor, the simulation engine, and many smaller components are built in Java. This language was selected because it integrates easily with PLAMAGS, which is used to specify and run the agent behaviours. Java is not the most efficient language, but the performance gains with respect to other languages such as C++ were deemed almost insignificant compared to the performance gains that can be achieved by improving the application's models and components. Java was selected also because it is easy to learn and easy to use. In fact, a defence scientist from DRDC (the project sponsor) who had no programming knowledge was able to understand important algorithms by reading the Java code and even to make minor modifications.

C++ and PhysX

One of the most computationally-demanding modules in the Crowd-MAGS application is the management of physics in the 3D environment. More specifically, the environment engine must calculate all physical forces that apply on all components at every iteration. For a reasonable simulation of physics, each second must contain at least 40 iterations; otherwise, fast moving objects could exhibit non-realistic behaviours. Since there are usually thousands of components in the environment in Crowd-MAGS scenarios, and since every one of them is affected by physical forces 40 times per second, performance is a critical issue. Thus, this part of the application was developed in C++ with the help of the Nvidia PhysX physical simulation engine. C++ code is more difficult to create and manage but is highly efficient and can interact with Java through the Java Native Interface (JNI) technology.

PLAMAGS

Developing a simulation application is difficult but modelling human behaviour is a much more complex task. Fortunately, a programming language for the modelling of complex behaviours has been developed by the GRIC. The PLAMAGS language (Programming LAnguage for Multi-Agent Geo-Simulation) (Garneau and Moulin, 2008) is an agent-oriented language that offers several constructions available in procedural and object-oriented languages.

PLAMAGS is coupled to Nvidia PhysX, which is one of the most powerful physics engine available. Hence, realistic physics can be simulated between agents and objects in the virtual geographic environment (VGE). For example, fences are defined with their real mass and can be fixed to the ground. Agents cannot walk through them but the fences can be brought down or moved if a large number of agents push against them. Simulations can also use particle systems for various purposes, such as smoke and tear gas.

PLAMAGS is a complete development environment, but its power comes from the available constructs for the specification of agent behaviours. The most important ones are listed below.

- Agents' goals can be specified as a hierarchy of objectives (called a behavioural graph)
- Objectives can be of type *Behaviour*, *Elementary* or *Compound*
- Objectives can be associated with activation, execution, and completion rules
- Objectives may be constrained to execute only when the agent holds certain resources
- Objectives are associated with states, such as ACTIVE, IDLE, and SUCCESS
- Objectives can be executed in concurrency
- Each elementary objective can perform actions coded in Java
- Behavioural graphs can be added to and removed from agents at any time
- Agents possess predefined and adjustable mechanisms for perception and navigation

In PLAMAGS, agents can be associated with behaviours that are represented by multi-layered directed graphs composed of various objectives (Figure B.1 shows an example). Agents may possess several behaviours (graphs), which can be added and removed while simulations run (e.g. removing the navigation behaviour if the agent is detained).

Three types of objectives exist. *Elementary* objectives are at the bottom of the hierarchy and are associated with actions — often atomic — that the agents perform. *Compound* objectives are decomposable structures that can represent sub-behaviours. *Behaviour* objectives are simply *compound* objectives that are at the highest level of the hierarchy; there can be only one *behaviour* objective per graph.

A non-elementary objective starts the execution at the entry point(s). An objective is activated when its predecessor is completed, if activation rules are validated (the rules are simply if-then statements). At every iteration, an activated objective is executed only if its execution rules are validated. In addition, an objective terminates when its completion rules are validated. The objectives' states are UNINITIALIZED, IDLE, ACTIVE, SUCCESS, FAILURE, FINISH, and they are detailed in Table 3.1.

Table 3.1: Objectives' states in PLAMAGS

State	**Description**
UNINITIALIZED	The behaviour engine has not yet reached this objective
ACTIVE	The objective has been activated and it is executed at every iteration
IDLE	The objective has been activated, but it is not executed (until switched back to ACTIVE)
SUCCESS	The objective has terminated with success
FAILURE	The objective has terminated with failure
FINISH	The objective has terminated and its successors will not be activated

PLAMAGS' behaviour engine supports hierarchy and parallelism. Thus, it is natural that a 'child' objective will not be executed if its parent is not executed. Moreover, several objectives can be executed in concurrency. It is sometimes necessary to restrict two or more objectives from executing in concurrency, and the concept of resource solves this problem. Objectives can be associated with resources and priority levels. If more than one objective associated with a resource tries to execute at a given iteration, only the one with the highest priority will be executed.

Behavioural graphs are stored textually in .BDL files. BDL stands for Behaviour Description Language and its grammar and details are presented in (Garneau, 2007). To ease the comprehension of behaviours, a visual formalism was created for the Crowd-MAGS project and it is presented in Section B.1. PLAMAGS and this formalism were used to create all of the behaviours related to crowd control, which are presented in Appendix B.

It should be noted that PLAMAGS is in line with Wijermans' statement, introduced in Sub-Section 2.3.7, that states that "human behaviour is driven by a multitude of goals that underlies the range of behaviours" to be simulated. Wijermans proposes that agents hold multiple goals. Also, Thalmann's behaviour model described in Sub-Section 2.3.2 is hierarchical. "The behavioural model decomposes a behaviour into simpler behaviours which may themselves be decomposed further. Each level of this hierarchical decomposition contains one or more behaviours performed either sequentially, or concurrently." Next, Pan uses decision trees for his agents' behaviours, as discussed in Sub-Section 2.6.3. These trees could be seen as highly simplified hierarchical behavioural graphs. However, they do not allow concurrency nor activation, execution, and completion rules. Moreover, Silverman uses a goal hierarchy in his agent architecture, as presented in Sub-Section 2.3.4. Finally, PLAMAGS is in a way an evolution of the MAGS system, presented in Sub-Section 2.3.1, in which behaviours are based on automata where each node represents an objective.

3.3 Simulation Output Analysis

The Crowd-MAGS application allows visualizing the simulation in 3D as it runs. This feature provides a way of appreciating the situations, but it is not enough for a complete assessment of the scenario or comparison with another scenario. Thus, the application generates extensive data when the scenarios end. Due to the high complexity of the crowd simulation domain, this data is crucial in gaining a better understanding of how the scenario's parameters and events influence the unfolding of the events. Also, this data helps with the calibration phase, by fine tuning the models to reflect reality as closely as possible.

In order to calibrate the system so that one can extract results with confidence, the calibration phase should ideally be done with real data. In the early phases of the project, a data collection effort lasted for almost a year but resulted in little usable data. Nonetheless, macro-level information was gathered about the events of the Summit of the Americas of 2001 in Québec City. In addition, several videos of major protests from North America and Western Europe were collected from online video publishing web sites and television broadcasting stations.

Since little to no calibration could be performed from empirical data, a completely different model than the agent-based model was built so that they could be run in parallel and be compared to one another. The other model uses a systems dynamics approach and was developed by DRDC team members. This model was designed to simulate the scenarios, but of course at the macro-level only. Section 6.8 shows how the two models work together and demonstrates that they can be used in synergy to achieve a level of calibration that would otherwise be very difficult to achieve in a short period of time. The remainder of Chapter 6 will present how the system generates output data, how the analysis and the calibration processes took place, and what are the final results.

3.4 Creation of Models

The main contribution of this thesis is the combination of the agent and the group models, but several other models are necessary for the Crowd-MAGS application to simulate plausible situations. For example, the environment model defines how the environment is modelled, and how components can interact with it. The information model is a data representation scheme for anything that is tracked during simulations. This model aims to centralize all the data that is generated by various components so that it becomes organized and easily analyzable. Chapter 4 will present these models and a few others while this current section presents the notions that are necessary to understand the models.

First, Sub-Section 3.4.1 introduces the concept of a virtual environment. Next, Sub-Section 3.4.2 introduces semantics, which are used by agents to understand the virtual environment. Sub-Section 3.4.3 follows by expliciting the notion of time in the Crowd-MAGS simulation. Next, Sub-Section 3.4.4 introduces the social identity theory, which is at the heart of the agent and group models. Finally, Sub-Section 3.4.5 presents other notions that are fundamental to the Crowd-MAGS models.

3.4.1 Representation of the Environment

One of the predominant features of any geo-simulation is the VGE. All components to be simulated evolve in the environment, which is a representation of the world. In fact, only the locations necessary for the simulation are represented, and they can be the entire planet, a continent, a city, a large building, or any other location. Thus, the environment model must be well defined before other models, such as agents and groups, can be specified.

Unlike GIS, components are not necessarily arranged in layers. This difference is due to the fact that components in a GIS are used to help the user visualize the situation, whereas they are used for agent reasoning and calculation of physical forces in a simulation. Thus, components cannot be removed to help the user visualize the situation. Nonetheless, advanced simulators may “hide” components to help the user without impacting the simulation.

Like with GIS, spatial data can be represented in raster or vector format. Crowd simulators often use raster data because of its simplicity (Benenson and Torrens, 2004), for example (Goh et al., 2006; Wijermans et al., 2008), but an increasing number, such as (Pan, 2006), are using vector format for its exact representation of the environment and its flexibility towards scale. In vector format, any location in the environment can thus be represented by 2D coordinates (like GIS) or 3D coordinates.

The Crowd-MAGS’ environment is fed from a GIS file that is enriched with semantic labels associated with points, lines, or polygons. These labels are used by the agents to reason about their surroundings, as they cannot understand the 3D visualization that the user sees. Section 4.1 will present the environment model, but a few more notions are presented next before diving into the details of the model.

In order to navigate properly, agents must be able to retrieve the navigation zones and the obstacles from the environment. It is well known that raster data exhibits the problem of cells that are partly navigable and partly not. Although this problem might be acceptable for simple GIS analyses, it can cause odd behaviours when autonomous agents, and especially crowds, query such a cell. Thus, semantics must be used in the environment so that agents can conceptualise their surroundings, and thus reason and navigate properly. Semantics could help agents to understand what part of the cell is navigable, but in this case a vector representation would be more suited.

The process of finding a navigable way between two points is called path planning. This task is a very demanding one in multi-agent simulation, both from a design point of view and from an execution point of view. The modeller must carefully think about what details are important in the decision taken by an agent who chooses one path over another. Too many details are not helpful because path planning is computationally demanding and could slow down a system significantly when executed for many agents. Thus, the algorithms must be efficient and the data that they use must be aggregated to a certain level. Section 4.1 discusses options for path planning in Crowd-MAGS.

3.4.2 Representation of Semantics

In order for agents to conceptualize components and other concepts in a simulation, semantics must be attached to the concepts to be understood. This thesis proposes the use of semantic labels, which define the “known vocabulary” of the agents. The use of semantics allows the agents to reason on a qualitative representation of the environment, like real humans, instead of reasoning on purely quantitative facts, like a macroscopic or statistical computer simulation model.

It is proposed that each semantic concept be defined in two parts:

- A semantic label that defines a unique concept in the simulation, such as *fence*, *woman*, *yelling*, and *gas mask*.
- A set of semantic categories to which the label belongs. For example, *fence* and *gas mask* could belong to the category *control force equipment* and *gas mask* could belong also to the category *instigator equipment*.

To facilitate the use of semantic definitions, it is also proposed to group them into semantic collections. Such a collection contains all of the semantic definitions that are related to a specific concept. For example, a semantic collection could be created to contain all semantics related to agent actions; another one could be created for all NLW.

Semantic designation is used extensively in the Crowd-MAGS models and system. Using the proposed semantics, system and scenario designers can describe any concept in a more generic manner, allowing a higher behavioural design level and a higher output analysis level.

3.4.3 Representation of Time

The notion of time is fundamental in crowd simulation. Indeed, time is at the heart of any agent behaviour, to control the frequency and the duration of each objective. The notion of simulation step exists in PLAMAGS, but does not have any meaning with respect to time. Thus, a convention must be set so that one iteration equals a fixed amount of time. Only then will all components be able to work on the same temporal base and interact with one another.

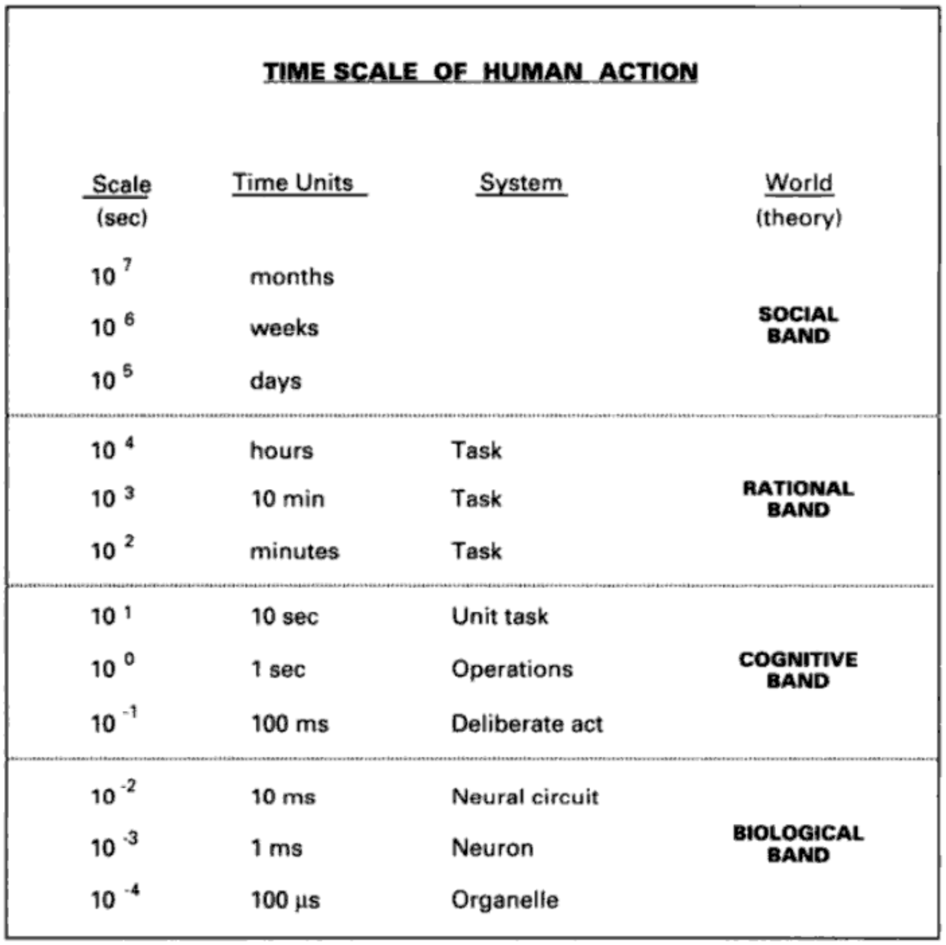
Choosing a time value for the representation of one iteration requires careful thought. A value too small would require many iterations to simulate a short lap of time; this effect would reduce performance. On the other hand, a value too large would prevent simulating objectives that are executed at high frequency. Thus, the time representation of an iteration should be the duration of the most high-frequency objective to be simulated. In the case of human agents, the smallest amount of time corresponds to the reactive behaviours, such as perceiving the surroundings and moving. Based on 1) the behaviours to be simulated in the Crowd-MAGS project, 2) Newell's time scale (shown in Figure 3.2), and 3) performance tests run with PLAMAGS (and presented in Appendix C), the duration of one iteration has been fixed at 100 ms, thus leading to the simulation of 10 iterations per second.

3.4.4 The Social Identity Theory

One of the main social theories that guided the development of the agent and group models is called the social identity theory, and it was introduced in Sub-Section 2.6.1. This theory has already inspired other simulations, such as Pan's MASSEgress system, presented in Sub-Section 2.4.3. Moulin also suggests using the social identity theory, along with placing an emphasis on groups. (Moulin, 2009) was introduced in Sub-Section 2.4.1, and it is presented here from a more applied (less theoretical) point of view.

In (Moulin, 2009), which summarizes and integrates into a theoretical framework several papers on the social identity and inter-group relations, Moulin claims that groups are the missing elements in current crowd simulations, if they are to be used for decision support in crowd control situations. In fact, Moulin assumes that participating in collective actions is more important for agents than carrying out individual activities and pursuing individual goals (in crowd control situations). This assumption is a simplification from the cognitive point of view but it is a reasonable one given the scenarios to simulate and the social characteristics of the crowds.

By placing the emphasis on collective rather than individual actions, it becomes natural to explicitly model groups. Groups evolve at a meso level of the simulation to simulate group interactions and to collect data that will be used for situation assessment at a macro level. Indeed, external observers, such as control forces' commanders, political authorities and the media, usually assess a crowd situation from a global point of view. In the proposed approach, groups play a pivotal role: 1) at a micro-level, agents are able to recognize and join groups in order to participate in collective activities; 2) at a meso-level, group interactions and relations



TIME SCALE OF HUMAN ACTION

Scale (sec)	Time Units	System	World (theory)
10^{7}	months		
10^{6}	weeks		SOCIAL BAND
10^{5}	days		
10^{4}	hours	Task	
10^{3}	10 min	Task	RATIONAL BAND
10^{2}	minutes	Task	
10^{1}	10 sec	Unit task	
10^{0}	1 sec	Operations	COGNITIVE BAND
10^{-1}	100 ms	Deliberate act	
10^{-2}	10 ms	Neural circuit	
10^{-3}	1 ms	Neuron	BIOLOGICAL BAND
10^{-4}	100 µs	Organelle	

Figure 3.2: Newell's Time Scale for Human Action (Newell, 1990)

evolve as a result of the individual activities of their members; 3) at a macro level, groups collect data about their members, the collective activities that they support, as well as their interactions with other groups.

3.4.5 Fundamental Notions

This sub-section presents notions that were necessary to develop in order to implement the social identity theory.

3.4.5.1 The Notion of Social Identity

An agent who represents a crowd participant should be able to change its behaviour depending on the way in which he interprets the surrounding situation. In line with Reicher's Extended Social Identity Model (Drury and Reicher, 2005), the Crowd-MAGS model proposes that an agent be associated with a fundamental social identity (mainly composed of its personality traits) and different social identities that may be adopted when participating in collective activities. The concept of social identity is used to aggregate data, objectives, and behaviours related to a certain identity in order to create a repertoire of social identities from which agents can choose. During a simulation, an agent may change his social identity at any time, but according to a few rules that are based on his fundamental social identity and on the surrounding situation. The complete social identity model is presented in Sub-Section 4.2.6.

3.4.5.2 The Notion of Spatio-Temporal Group

In some current simulations, group behaviours are viewed as patterns emerging from pedestrian flows (Osaragi, 2004; Goh et al., 2006; Musse and Thalmann, 2001). In contrast, groups must be explicitly modelled in the case of purposive crowds to be able to manage the dynamics and the interactions of groups and agents (Moulin, 2009).

In Crowd-MAGS, it is proposed that a spatio-temporal group (STG) emerges around a 'seed': an agent that is the origin of the STG. This agent creates the STG by broadcasting messages around him to attract other agents. However, a STG may be pre-created, such as a line-up in front of a bank machine (the STG is initially empty but still exists). As a reasonable simplification, it is suggested that a leader agent, when one exists, provides the STG with basic characteristics and directives that are useful to coordinate the STG and the collective actions carried out by its members. Some examples of these characteristics are the desired type of social identity (demonstrators, instigators, etc.), the desired level of aggressiveness of the collective actions (observing, chanting, forcing fences, etc.), the organization of the members around the leader (tight formation, straight line, loosely arranged, etc.). The leader also decides where the group moves. Other agents are autonomous and thus they are free to follow the leader and observe the directives; however, they may decide to do otherwise, in which case they will likely leave the STG. Each agent's possesses a history of individual actions and the STG possesses a history of collective actions. These 'logs' can be used by any agent to decide if he wants to join or leave the STG. The complete STG model is detailed in Section 4.3.

3.4.5.3 The Notion of Projected Image

In reality, different crowd members may observe the same situation and react to it in different ways. In fact, the way that an individual interprets a crowd situation (essentially the perceived behaviours of other individuals or groups) significantly influences his decisions and behaviour changes (Silverman et al., 2006b). Hence, it is important to model this interpretation process in order to plausibly simulate phenomena such as social identity changes and adhesion to groups. To this end, the notion of projected image was proposed (Moulin, 2009) and implemented in Crowd-MAGS.

A projected image is a data structure that can be associated with different kinds of entities such as agents, groups and other objects located in the environment. This data structure contains the information made available by the entity to other agents who perceive it. For example, an agent's projected image may contain data about his appearance and age category while a group's projected image may contain data about the number of group members and the collective actions being performed. The agent's projected image model is discussed in Sub-Section 4.2.5 and the group's projected image model is discussed in Sub-Section 4.3.4.

3.4.5.4 The Notion of Interest Point

As mentioned in Sub-Section 3.4.1, the VGE's data structures must allow agents to access information about their surroundings through perception activities. Agents must be able to perceive objects, other agents, groups, and non-lethal weapons such as tear gas clouds. Agents must also be able to identify certain points or areas that carry specific significations or where certain activities are carried out. For this purpose, the notion of interest point (or area) has been introduced. An interest point (or area) is a location (or area) in the VGE with associated 'semantic characteristics' (category, attributes, allowed or prohibited activities, etc.) that agents are able to perceive and interpret. An interest point (area) may be stationary or mobile. Typical examples of stationary interest points (areas) are: meeting area for a demonstration, itinerary stops for a peace march, fences, etc. Mobile interest points can be used to characterize the position of a group of agents, such as instigators or a police squad. Interest points are described in more details in the environment model, which will be presented in Section 4.1.

3.4.5.5 The Notion of Resource

Several agent behaviours related to crowd events require resources to be executed. For example, instigators need a free arm and a rock in order to throw a rock to the control forces. A gas mask can also be considered as a resource that an agent may own or give to another agent. Some resources are limited and agents may compete to acquire them. Other resources are internal to the agent, such as the cognitive charge and the ability to navigate. Hence, resources must be explicitly managed in the agent model to affect the agent's decision making. Crowd-MAGS's resource model is based on the agent's internal resources presented in (Lamarche and Donikian, 2002). Internal resources will be used to manage an agent's concurrent behaviours, and consequently to establish priorities between behaviours that require the same resource at a given moment. External resources will be used mostly to prevent certain behaviours from happening, such as throwing a tear gas can when no cans are left. Complete details about resources are given in Sub-Section 4.2.1.

3.4.5.6 Non-Lethal Weapons and the Notion of Health

The concept of agent health is still rarely modelled in crowd simulations ((Silverman et al., 2003) does it through its physiology module and Performance Moderator Functions (PMF) reservoirs). Yet, it is crucial in crowd control events because wounds can affect an agent's perception, navigation, memory, and behaviours. The objectives of the Crowd-MAGS project called for the simulation of a few non-lethal weapons: fences, plastic bullets, and tear gas. It should be noted that it is not necessary to deal with agent death since lethal weapons are not modelled (only NLW). In addition, behaviours have been developed to make the agents flee when they detect high danger or when they become critically wounded.

Following the users' needs and the limitations of the behaviour engine, the computational power available, and the modelling complexity, a simple health model was developed. Each agent possesses a level of health, which is modelled as a quantity comprised between 0 (dead) and 1 (fully healthy). Localized damage, which would allow independently considering damage to different parts of the agent body is not modelled due to complexity. Falling health does not affect the behaviours' execution sequence, but rather how well objectives get performed. This choice is in line with the PMFs that Silverman proposed. For example, Crowd-MAGS agents move more slowly and perceive other agents and groups less clearly when injured. An example from (Silverman et al., 2006b) is that a high level of fatigue, stress, or injury might lead to agent to panic.

Only NLW and fighting can affect an agent's health level. Plastic bullet rifles are represented as resources that control forces may use, while the bullets are also resources. Tear gas cans can be thrown by officers and kicked back by agents. The cans themselves cannot hurt agents, but they emit tear gas clouds that decrease the agents' health. Megaphones are used to send messages to the crowd while fences are used to restrict movement in certain areas. Fences can be fixed to the ground, but they can be brought down or moved if a large number of agents push against them.

More details about NLW are presented in Sub-Section 5.1.10 and the complete health model is detailed in Sub-Section 4.2.8.

3.4.5.7 Appreciation of Aggressiveness

In crowd control events, violence is a major concern: peaceful demonstrators want to avoid violence, instigators may seek opportunities for violent actions and control forces want to limit violence and disruption of public order. Consequently, collective actions should be qualified on an aggressiveness scale. (Moulin, 2009) proposes that this scale range from -1 (very peaceful) to +1 (very aggressive). Estimating a ranking of collective actions on such a scale is not a very difficult task. For example, McKenzie and his team (McKenzie et al., 2005b) qualified the actions of individuals and groups in a crowd using an aggressiveness scale. Once an acceptable set of collective actions have been defined, the agents' appreciation of aggressiveness can be defined.

In order for agents to plausibly make decisions based on the collective actions carried out in their surroundings, it is hypothesized that agents must express a certain appreciation for different levels of aggressiveness (Moulin, 2009). For example, a bystander might highly appreciate chanting and probably does not appreciate instigators throwing projectiles. On the other hand, instigators might appreciate throwing projectiles and may find chanting too 'passive'. Due to the complexity of simulating these 'feelings', each agent cannot interpret in his own manner the actions carried out around him. Thus, it is suggested that agents be characterized by *appreciation profiles*, which can be adopted by multiple agents (Moulin, 2009).

By observing the collective activities that take place around him, an agent may become excited and feel an urge to participate in the collective actions. Conversely, an agent may be disapproving the collective activities that he observes and become reluctant to participate in them. To this end, the notion of enthusiasm was introduced (Moulin, 2009). Basically, it represents the overall appreciation of the collective actions being carried out around an agent. Thus, enthusiasm takes its values in [-1, +1] with +1 expressing an extreme enthusiasm (or excitation), 0 being neutral and -1 expressing a complete reluctance to participate in collective actions.

Complete details about appreciation of aggressiveness and enthusiasm are given in Sub-Section 4.5.5.

3.4.5.8 Adhesion to Spatio-Temporal Groups

It was already mentioned that Crowd-MAGS agents seek to participate in STGs based on the collective activities being carried out. In addition, if an agent is already participating in the collective activities of a STG, he needs to decide if he will continue or leave the STG. Thus, these decision rules must be modelled taking into consideration the collective actions. Because all STG members are autonomous, the collective actions being performed could change quite rapidly. Current members might not appreciate the actions of some new members, but they may not necessarily want to leave the STG immediately. In consequence, the notion of support is necessary. It can be defined as the long-term appreciation of the STG's collective actions. Because appreciation of aggressiveness takes its values in [-1, +1], support will be constrained to the same range. Coming back to the previous example, the new members' actions might bring down the other members' support values towards the STG, but the recent actions will bear only a certain weight with respect to the history of actions that have been performed before. Thus, some members might eventually leave, when their support towards the STG drops below a certain level. Agents who want to join a STG might also wait before their support goes above a certain threshold. More details about these decision rules are presented in Sub-Section 4.5.6.

3.4.5.9 Adoption of a Social Identity

In line with the social theories presented above, Crowd-MAGS agents possess a fundamental social identity and may adopt different ones during the simulation. The decision to adopt a new social identity is based on the assessment of the situation. Since the notion of enthusiasm represents an assessment of the situation based on the collective actions, it seems natural that enthusiasm be used in the decision rules for social identity changes (Moulin, 2009). More specifically, each agent has access to a set of candidate social identities based on his fundamental social identity. The agent will evaluate in which social identity he would be the most enthusiastic, and adopt this identity. For example, a bystander who is surrounded by aggressive demonstrators might be more enthusiastic about the situation if he ‘behaved’ as a demonstrator, so he might adopt this identity. More details about this algorithm are explained in Sub-Section 4.5.7.

3.5 Conclusions

This chapter was meant to be an introduction to the Crowd-MAGS models as well as the Crowd-MAGS project in general. First, the methodology was presented and divided into four main phases: scenario specification, simulation execution, output analysis, and models calibration.

The types of events to be simulated were introduced, as well as the possibilities and limitations of specifying them in the Crowd-MAGS system. Next, the software application was discussed, especially regarding the technologies used to build the system. Afterwards, the need for output analysis was explained and techniques of validation were discussed. Finally, several notions necessary to the understanding of the models were presented. For example, the concept of a VGE, semantics, and time were discussed. The social identity theory, which guided the development of the models, was explained along with concepts necessary to implement it in Crowd-MAGS.

Now that all of the fundamental notions have been introduced, the next three chapters can be presented. First, Chapter 4 dives into the details of the Crowd-MAGS models. Next, Chapter 5 shows how these models are used in the Crowd-MAGS application. Finally, Chapter 6 discusses the experimentations that have been performed with the application.

Chapter 4

Crowd Simulation Models

Models are at the core of this thesis and hence this chapter presents all of the models used in the Crowd-MAGS simulation. First, Section 4.1 presents the environment model, on which other models are based. Next, the prominent models are discussed: the agent model in Section 4.2 and the group model in Section 4.3. Afterwards, Section 4.4 details the information model. Next, generic behavioural models and those specifically related to crowd control are covered in Section 4.5. Finally, Section 4.6 discusses the last model, which defines how scenario specification are to be written. In addition, Section 4.7 discusses possible improvements to the models, and Section 4.8 presents conclusions about all of the Crowd-MAGS models.

4.1 Environment Model

The environment representation is a very important aspect of crowd simulations. In fact, agents evolve in the environment and several other models depend on the environment, such as the agent model. Hence, this section introduces aspects of the 3D environment provided by PLAMAGS, as well as other notions necessary for the agents to interact with the environment.

The PLAMAGS language (Garneau and Moulin, 2008) allows the creation of 3D environments and components, as shown in Figure 5.1. This feature is useful for the users of the simulators, as it allows them to view in real time what is happening in a scenario rather than only interpreting it from numerical output data.

However, 3D visualization is not useful for the agents to understand their environment. In order to navigate properly, agents must be able to retrieve the navigation zones and the obstacles. Thus, semantics must be used in the environment so that agents can conceptualise their surroundings, and thus reason and navigate properly. Semantics can be assigned manually or automatically to different components in the environment. Even though a complete and automatic topologic representation of the environment is not possible with PLAMAGS, it is possible to manually assign semantic labels to components in the VGE in order to create high-level abstractions of the environment. Since the scenarios to be simulated in the Crowd-

MAGS project are located in small open-space areas, this solution is sufficient. Specifically, an agent would know what zones are walkable, forbidden, interesting, scenic, and any other semantic concept that the environment designer chose.

The model proposes to consider the entire environment as walkable except for buildings. These obstacles will be managed by PLAMAGS' obstacle avoidance algorithm at the time of the navigation, and not in advance by path planning. Agents will therefore always plan to move between locations in a straight line, and go around obstacles when they are blocked. For every component other than buildings, the model proposes to use *interest shapes*, which represent situated components, not necessarily embodied (they may be invisible and occupy no volume), that contain semantic information to be used by the agents. Based on the GIS notion of *shape*, interest shapes can be *interest points*, *interest lines*, and *interest polygons* (hereafter called *interest zones*).

Interest Shapes

An interest shape is composed of several properties:

- **Location:** Since an interest shape is a situated component, it contains a property relating to its location: a point (X,Y) for an interest point, two points for an interest line, and three or more points for an interest zone.
- **Semantic definition:** Interest shapes store in their semantic definition the concept that they represent. For example, an interest zone representing a fast-food restaurant could have the semantic definition *fast-food restaurant*, which could be in the semantic categories *restaurant*, *building*, and *private property*. The semantic definition allows agent behaviours to reason spatially and qualitatively to determine a destination.
- **Type:** Interest shapes can be static or dynamic. In the former case, shapes' points cannot be moved and the shape can never disappear, such as parks, buildings, and protesting areas. In the latter case, shapes' points may be moved (displacement, stretching or distortion) and the shapes can disappear, such as the area occupied by a group of people that may eventually be disbanded.
- **Perceptibility:** This property indicates if the interest shape can be perceived by the agents. For those that cannot be perceived, the scenario designers must store them into some of the agents' memories, usually at the time when the agents are created. For example, the interest points where the control forces will deploy their squads are not perceptible, but all control force agents know these points (i.e. these agents do not need to perceive the interest points).
- **Unique identifier:** This property identifies an interest shape in a unique manner, allowing for very situated behaviours, such as going to a specific bus stop to meet another agent.
- **Additional properties:** Additional properties can be added at any time. For example, an interest point representing a pile of rocks for instigators could contain an additional property representing the number of available rocks.

Class Diagrams

In order for behaviour designers to easily use the VGE and its semantic representation, a certain level of genericity is required. In addition to interest shapes, several other concepts must exist and must integrate seamlessly. Thus, the following class diagrams explain the basic architecture to be respected when creating any component related to the geometrical and geo-referencing aspects of the simulation.

First, the most generic classes of the environment model are shown in Figure 4.1. To start, two categories of objects exist in the model. First, *ISpatial* components are those that can be characterized by Cartesian coordinates. Secondly, *IOrientation* objects those that can be associated with an orientation (e.g. 15° east of North). Defining an orientation for a line or a polygon is often ambiguous, and thus *IZone* and *ILine* are not oriented. *ILocation*, which represents a point in space, has no orientation either.

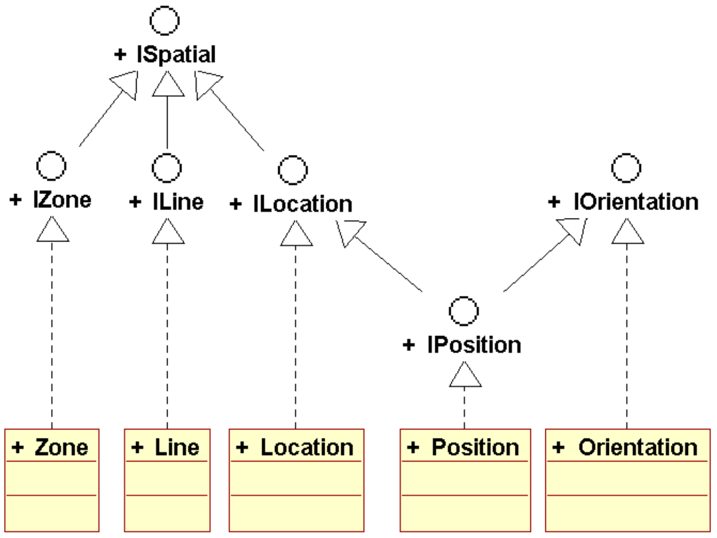


Figure 4.1: Environment Model: Main Interfaces

IOrientation components obviously possess an orientation, but they are not necessarily located at any particular place in the environment; thus, they are not spatialized. Components that are both spatialized and oriented are called *IPosition*. This interface can be used, for example, when a squad of officers is sent to a particular point and arrange in formation; orientation is necessary or the squad could be facing a wall instead of the protestors. Concrete classes are shown on the diagram for all interfaces because these classes are used by many components that need to implement the interfaces.

Figure 4.2 shows all of the components from Figure 4.1, in addition to interest shapes. Three types exist: interest points, interest lines, and interest zones. All of them are spatialized, but in different ways. For example, an interest point is spatialized through a position and an interest zone is spatialized through a zone.

Figure 4.3 shows how components compose one another. The *InterestZoneVertex* class

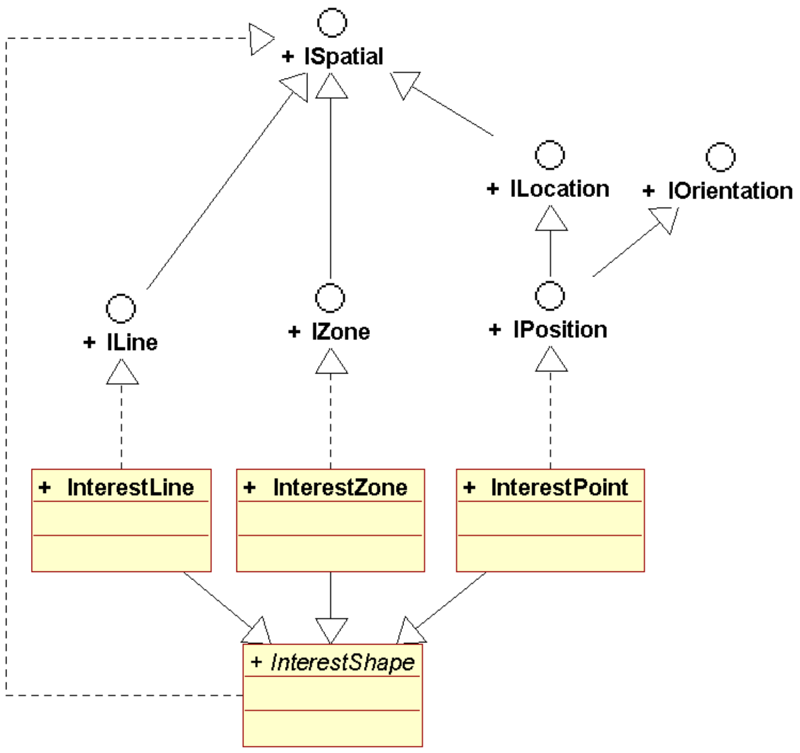


Figure 4.2: Environment Model: Main Interfaces and Interest Shapes

obviously represents the vertices composing every interest zone, and each interest zone must contain at least three vertices.

Lines are defined by their two end-points and thus lines contain two instances of ILocation. An interest line contains a line and some semantics, as presented above. In a similar fashion, each interest point contains a position and each interest zone contains a zone. It is important to notice that a zone contains multiple ILocation instances while an interest zone contains multiple interest zone vertex instances, which themselves each contain an ILocation. Although this design may seem overly complicated at first, it allows for a more versatile use of the classes. In addition, it allows for more information about the components to be available from any component. This feature is particularly useful when an agent perceives an interest zone vertex, since it can get access to the interest zone and all other vertices. Note that no memory is wasted since the set of ILocation instances contained in the zone object is the same set that is contained in the interest zone through its vertices.

In addition to these models, it is convenient to attach positioned components to targets. For example, delimiting a square zone around an agent would consist in attaching four vertices to the agent: the vertex would then move along with the agent. Vertices, as well as interest points, can be attached to any object that has a position. The vertex would then move along with the agent, changing the shape of the zone. It should be noted that zones cannot be attached, as they do not possess one single location. Instead they possess many points, each of which can be attached individually. If each point is attached to a different target, then the zone's shape may change.

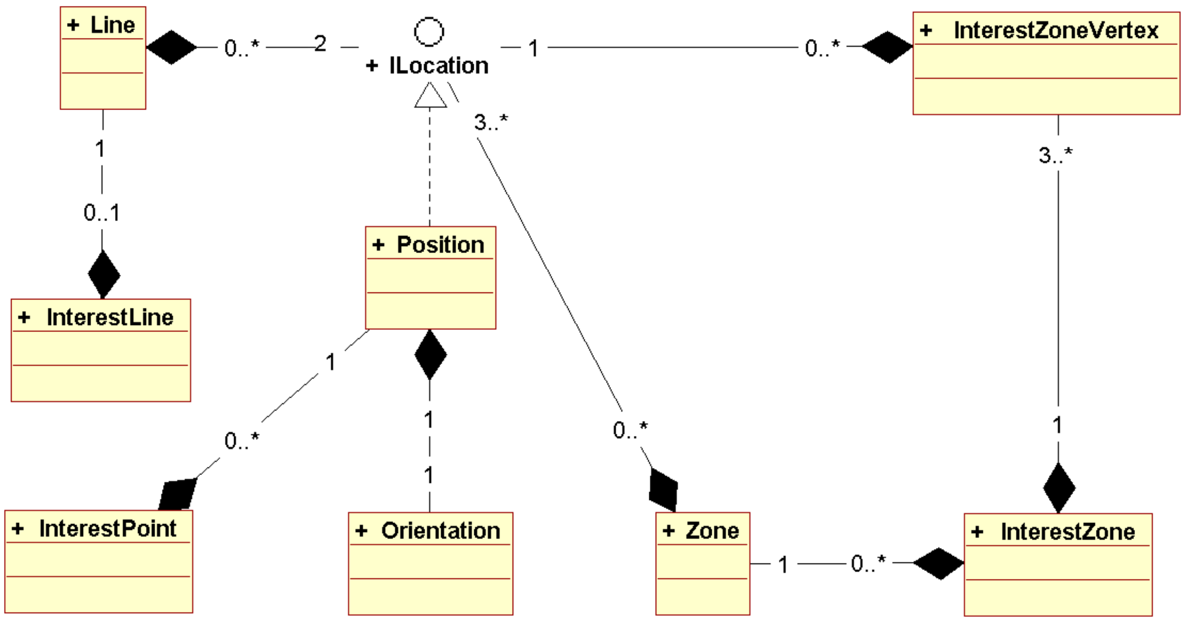


Figure 4.3: Environment Model: Geometry Components

4.2 Agent Model

The Crowd-MAGS project needed to model several types of agents, such as instigators and police squad leaders, but all of them are based on the same agent model, which defines the architecture and the common components shared by all agents. In fact, this model is generic enough to be used in a variety of projects, not necessarily related to crowd control. Most parts of the conceptual model presented here have been implemented but some elements have not been realized in the course of the project due to lack of time. Chapter 5 presents the Crowd-MAGS application, and more specifically Section 5.1 details how the models presented here have been used.

As mentioned in Sub-Section 2.2.1, there is no universal definition of an agent. However, several components are similar between most agent models for crowd simulation. Indeed, Figure 4.4 tries to integrate the agent models and notions discussed by Pan, Silverman, Thalmann, and Wijermans and presented in the literature review. In particular, Thalmann suggests that agents be able to perceive objects, agents, agents' actions, and agents interacting with objects. The environment, through dangerous events and control forces' weapons, affects the agents' health and emotions. Silverman suggests that an integrated stress function affects the agent's behaviour. The environment also models agents with 2D or 3D representations. Groups are located in the environment and they also may impact the agents' behaviours, as Wijermans and others propose. The memory is composed of different parts, which may contain behaviours, norms, goals, elements that have been observed, etc.

The Crowd-MAGS agent model, shown in Figure 4.5, is an extension of the generic model to include the notions of projected images, STGs, social identities, and a few others. All

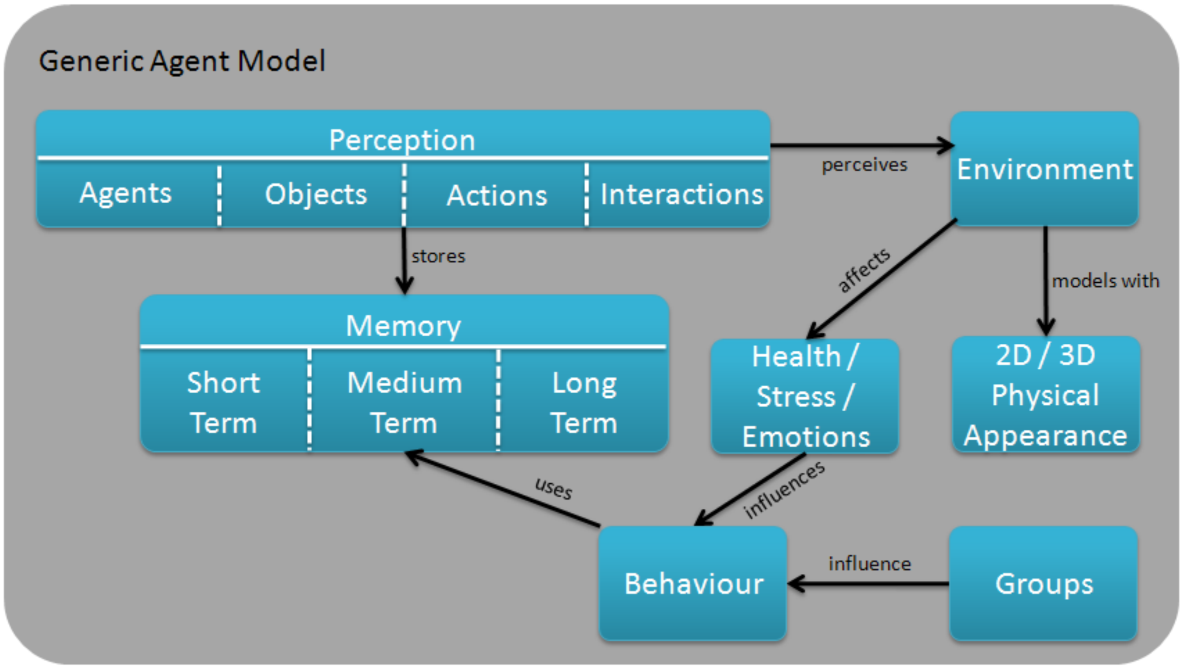


Figure 4.4: Generic Agent Model

elements will be discussed in this chapter except two. First, the PLAMAGS environment, which is a separate entity with which the agent model interacts; secondly, the agent's external properties, which are discussed in Section 4.4 and Sub-Section 5.1.5.

Section 4.2 is organized around Figure 4.5. The components on which other components rely are presented first. More specifically, the first component to be discussed is the concept of resources. Sub-Section 4.2.1 explains what resources are and how they are used by nearly all components. For example, navigation needs resources to execute and it is presented in Sub-Section 4.2.2. Perception is the following component to be discussed. Sub-Section 4.2.3 explains that data is gathered automatically in four ways and stored into memory. Sub-Section 4.2.4 goes into more details about the three types of memory. Next, internal, projected and perceived images are discussed in Sub-Section 4.2.5. Social identities and profiles are detailed in Sub-Section 4.2.6. Related to social identities is the concept of social group, which is discussed in Sub-Section 4.2.7. Next, Sub-Section 4.2.8 presents the health model, which allows agents to react plausibly to non-lethal weapons and injuries in general. Finally, Sub-Section 4.2.9 introduces the communications model.

In addition to all of these 'sub-models', agents possess properties of which they are not aware. Only the system uses these values, mostly for calculations related to physics and the 3D environment. These properties are listed here:

- Size (bounding cylinder)
- Mass
- Physical force (i.e. ability to push heavy objects)

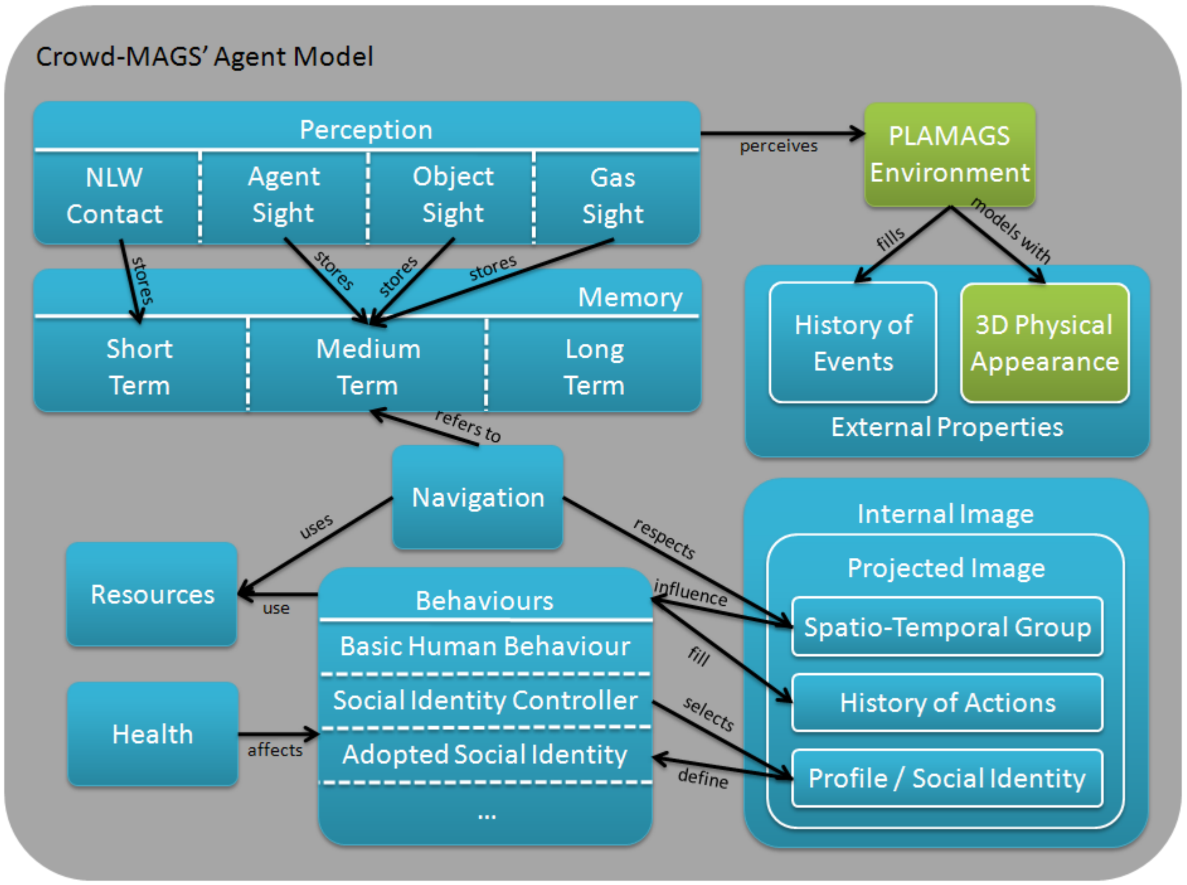


Figure 4.5: Crowd-MAGS' Agent Model

- 3D model (color of clothes, accessories like banners, etc.)
- Fields of view (agents, interest shapes, tear gas)
- History of events (Agent created, agent dragged by user, agent removed, etc.)

4.2.1 Resources

In a very broad sense, resources are one of the fundamental aspects of everyday life. For example, driving requires the resource *car* and walking requires the resource *legs*. The same requirements are true also for Crowd-MAGS agents. For example, an agent must be able to speak the local language to discuss with a control force officer and throwing a projectile requires a projectile of some sort.

In Crowd-MAGS, all of these requirements to the execution of tasks are managed through the concept of resources. A task can be specified as a set of objectives, each one possibly requiring resources. A resource can represent a capability (speaking English, memorizing locations), an internal entity (eyes, legs), or an object owned by the agent (banner, gas mask). Resources are also used to manage the concurrency of objectives; an agent who wants to chant and also yell at an officer simultaneously can only do one of these tasks because they both require the resource *mouth*. Thus, only the objective with the highest priority will be executed. Inspired from (Paris, 2007), the resources model contains three basic types, as shown in Figure 4.6.

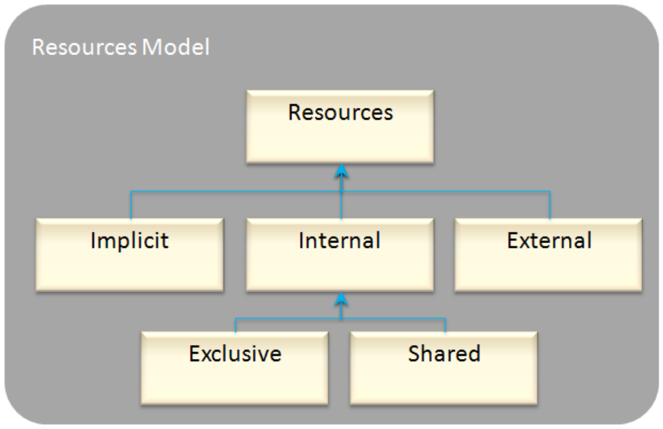


Figure 4.6: Resources Model

Implicit Resources

Implicit resources represent general capabilities, such as speaking English. This type of resource cannot be gained, modified, or lost. It is assumed to be present and constant from the time at which the agent is created until he is removed from the simulation (i.e. an agent cannot improve his English skills during a simulation, or forget how to speak it). Implicit resources are named in this way because they are implicit in the behavioural graphs and in the agent specifications. For example, a graph that makes an agent yell at a control force officer implicitly gives the agent the *English speaking* resource.

Internal Resources

Internal resources represent parts of the human body or mind, such as legs (for navigating) and the brain (for concentration on tasks). Like implicit resources, agents are created in the simulation with internal resources; however, agents may lose these resources temporarily or permanently. For example, an intoxicated agent may lose the *brain* resource temporarily and an agent who receives a plastic bullet in the eye may lose the *sight* resource permanently.

Two types of internal resources are modelled and they differ in their usage. Exclusive resources can be used by a single objective at a time (e.g. the *mouth* resource can be used by either *chant* or *yell*, but not both simultaneously) while shared resources can be shared among objectives (e.g. the *brain* resource can be used to plan a path between two locations and to look for a friend inside a crowd).

When creating an objective, the designer plans a list of all required resources, each one with a numerical priority. At run-time, if two objectives are depending on the same resource, only the objective(s) with the highest priority(ies) will be executed. In the case of an exclusive resource, only one objective will execute and possibly interrupt an objective that had requested the resource with a lower priority. In the case of a shared resource, a more complex algorithm with pre-emption is necessary because multiple objectives may already have been using the resource. Only a finite ”amount” of a resource is available (e.g. the *cognitive charge* resource can be shared by a few objectives, but only to a point where the overall behaviour is plausible).

External Resources

External resources represent objects that agents use to accomplish objectives. Agents may be created with external resources, but they may also gain them, use them, or dispose of them during the simulation. Examples are a banner, a megaphone, a gas mask, and rubber bullets. These resources are the only ones for which agents may possess a countable quantity. For example, an agent may have only one *cognitive charge* but he may have many *plastic bullets*. Thus, the objective *Shoot* might execute a few times and stop after a while when there are no more *plastic bullets*. More resources can be obtained from resource suppliers located in the environment, such as police trucks.

Resources can also be used by components other than just the objectives. Agents are self-aware of their resources, and thus any algorithm within the agent can check for the existence or availability of resources. For example, a path planning algorithm might yield different results depending on the agent possessing the *gas mask* resource (i.e. avoiding an area with tear gas).

4.2.2 Navigation

Of all human behaviours, the ability to move is probably the one that is the most often simulated, even though it is highly complex. As explained in (Paris, 2007), navigation is composed of two main mechanisms: first, path planning is a cognitive behaviour that globally evaluates the movement in the environment, searching an approximate path between the current location to a given destination; second, reactive navigation is a reactive behaviour that locally evaluates the movement, mainly to avoid collisions with other agents. Without the first component, agents would go from A to B in a straight line, avoiding other agents on the way. Without the second component, agents would take the best path from A to B but walk into other agents along the way.

For several reasons, the proposed navigation model does not contain a path planning algorithm. First of all, PLAMAGS does not provide any path planning algorithms. These algorithms are complex and time was lacking to implement one during the course of the project. Secondly, the PLAMAGS environment is based on GIS files, which do not contain any semantic information that agents may exploit. Thus, any path planning would be based only on obstacles in the environment, and not on interest points such as places to rest or to get resources. Thirdly, path planning algorithms can take a significant amount of computing resources, slowing down simulations with large numbers of agents. PLAMAGS is already slow beyond a few hundred agents, and adding a path planning algorithm would most likely render performances unacceptable. Finally, the situations to be simulated in the Crowd-MAGS project are localized in small wide-opened locations, making navigation in a straight line less problem-prone than in larger areas or in areas with many obstacles.

Figure 4.7[1] shows an example of path planning in different environments. Going from A to B is problematic because of the shape of the City Hall and the many other buildings around. However, going from C to D is not a problem because the straight line algorithm barely encounters an obstacle, at which point reactive navigation can slightly alter the path.

Reactive navigation is handled automatically by PLAMAGS. The algorithm allows an agent to avoid other agents and components that are in his field of view, but not to avoid environment obstacles, such as buildings and steep hills. As shown in Figure 4.8, the algorithm computes angles of approach for each obstacle with respect to the initial path and picks the smallest one. The new diverged path is stored in the agent because it is not calculated at every iteration; rather, it is invalidated after a fixed number of iterations or after the agent has stopped moving. (Marcotte, 2007)

Based on the needs and constraints of the Crowd-MAGS project, the proposed navigation model (shown in Figure 4.9) uses a novel architecture where individual agent decisions can be superseded by the physical simulation of the environment. Thus, an agent may try to move 30 cm to the right, but the environment might make it move only 10 cm because a moving obstacle was in the way (agent, fence, car).

Figure 4.9 shows that several high-level behaviours may be concurrent to choose the destination of the agent. This concurrency is managed by the reservation of the *navigation* resource, which is available to only one behaviour at a time. The successful behaviour selects the destination in the environment by providing a low-level destination represented by Cartesian coordinates. Then, the agent's low-level navigation mechanism defines the desired movement for the current iteration, which is directly oriented towards the destination (oriented in a straight line), while trying to avoid obstacles when encountered. This mechanism produces a directive to the environment with a collision potential movement (since obstacles may also be in movement). Next, the physics engine (PhysX) reads in this directive as well as those from other agents, reads in global information about the environment, and provides movements for all components that follow the laws of physics — components may have blocked each other or bounced off each other.

The *navigation* resource is the key to the high-level moving behaviours. Since the agent can move to only one destination at a time, the *navigation* resource is exclusive and any behaviour that needs to set the agent's destination must request the use of this resource. For

[1]The map was produced by the City of Kitchener, Ontario, Canada. Accessed 2009-05-19. `http://www.kitchener.ca/pdf/downtown_districts_11X17.pdf`.

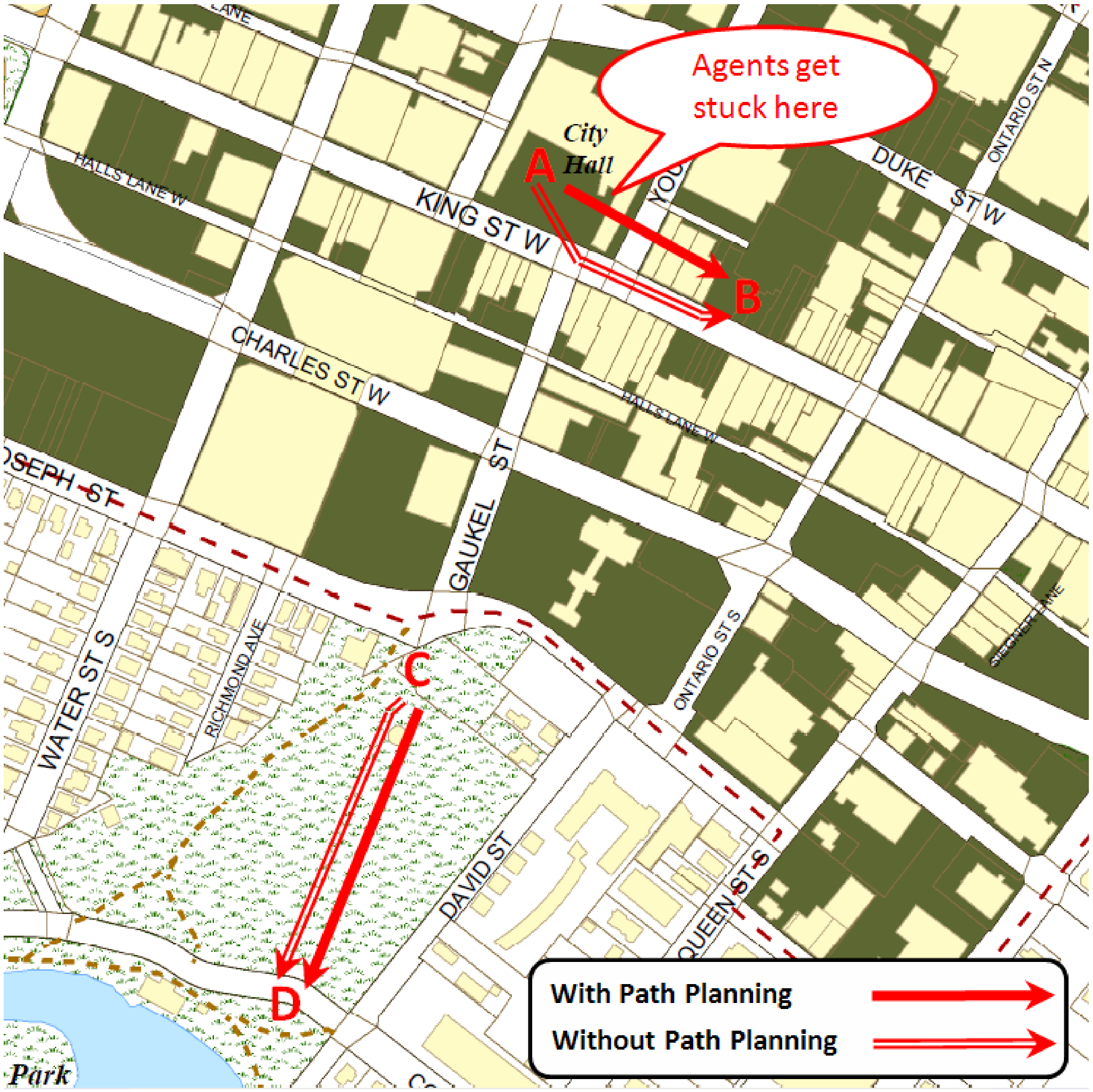


Figure 4.7: Navigation With and Without Path Planning

example, one objective could be to go eat and another one could be to riot. The control over the agent's movement must be managed, and only the highest priority objective will be allowed to set the destination.

The navigation model's architecture allows agents to select destinations without needing to reason on low-level details. This advantage increases the system's performance and keeps down the complexity of creating of high-level behaviours. Performance is increased also because the collision management algorithm is centralized in the PhysX engine, which is a highly-optimized commercial engine.

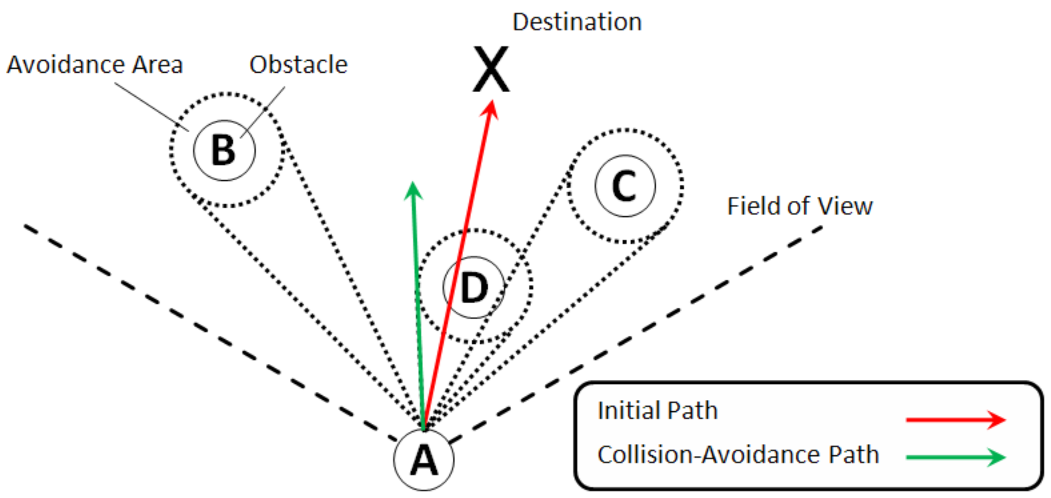


Figure 4.8: Collision Avoidance in PLAMAGS

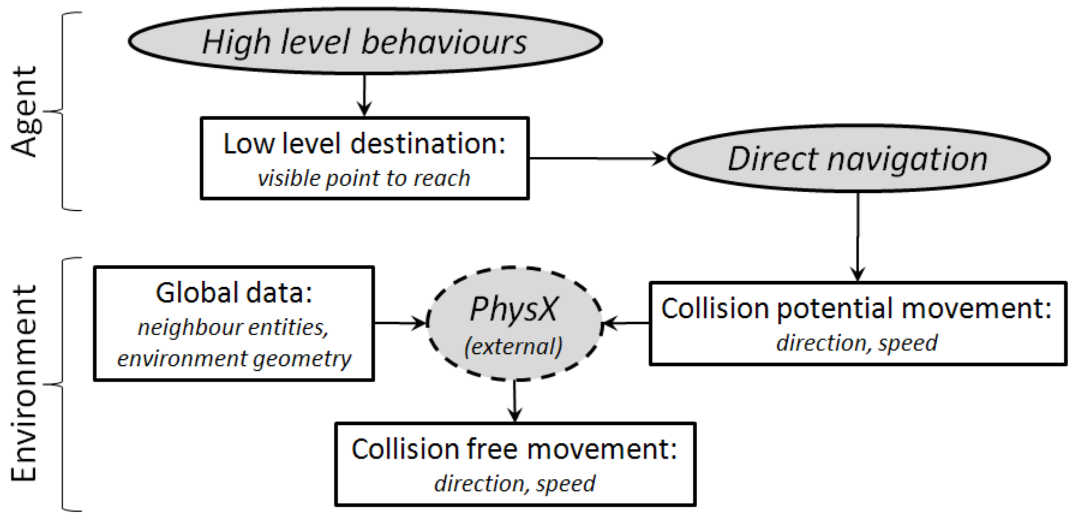

Agent
High level behaviours
Low level destination: visible point to reach
Direct navigation
Environment
Global data: neighbour entities, environment geometry
PhysX (external)
Collision potential movement: direction, speed
Collision free movement: direction, speed

Figure 4.9: Navigation Model

4.2.3 Perception

One of the most important and complex human abilities is the perception and comprehension of the surrounding environment. Necessary for most behaviours, perception is considered to be an active and conscious mechanism (Paris, 2007). The agent's perception capabilities (sensors) are presented here and the active mechanism (algorithm) will be detailed in Sub-Section 4.5.3.

The goal of perception is to extract and organise data from the environment. Some of these data trigger reactive behaviours, like running away when caught in tear gas, while other data is stored for later utilization. Most of the perceived data, however, are filtered out in order to avoid overloading the decisional processes (e.g. while driving, the brick colors on houses and buildings are filtered out although perceived).

In realistic models of perception, the perceived data are simple but they need to be semantically and conceptually identified in order to be understood by the behaviours. This process, demonstrated in Figure 4.10, is described in the memory model of (Atkinson and Shiffrin, 1968), which shows that external stimuli go through the sensorial memory to be identified and transmitted as concepts to the working memory). In an effort to keep model complexity down, this kind of perception is not used directly in Crowd-MAGS, but rather the agents can perceive external data directly in a conceptual manner (illustrated in Figure 4.11), thanks to the semantic labels and categories attached to environment components.

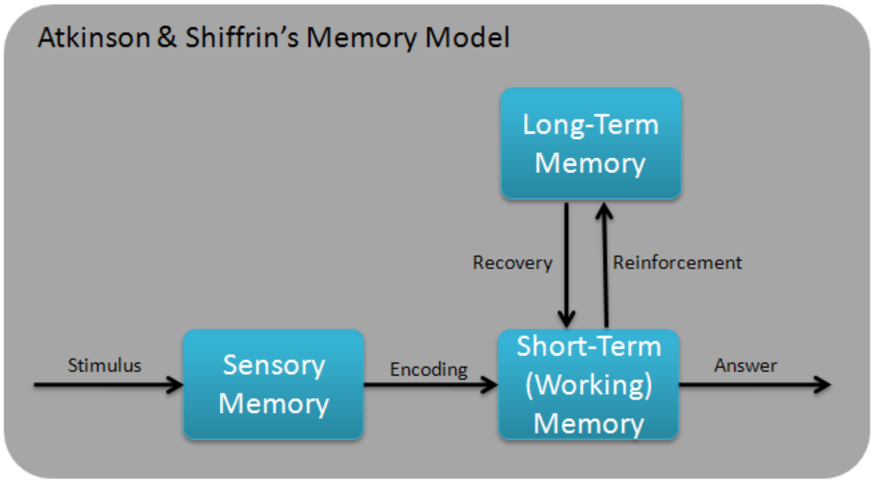


Figure 4.10: Atkinson & Shiffrin's Memory Model

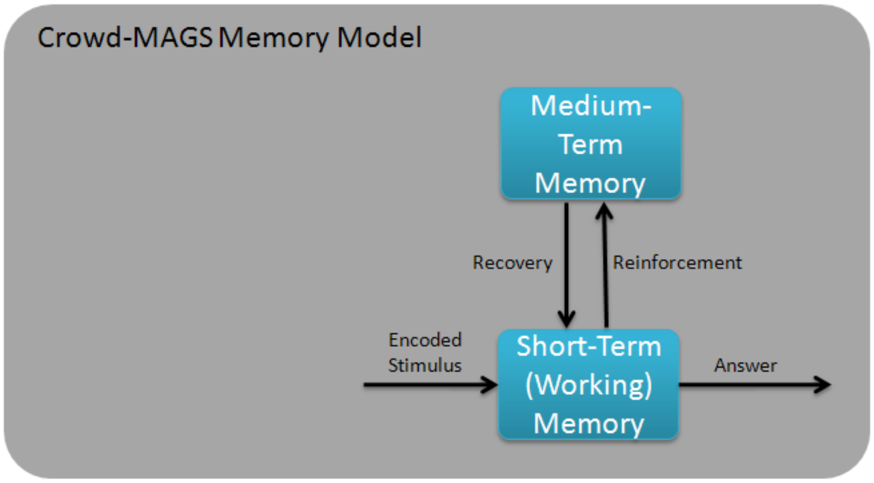


Figure 4.11: Crowd-MAGS's Memory Model (Overview)

Two types of perception are generally defined for humans: automated perception, which automatically gathers information from the environment, and directed perception, which gathers more precise information on a small focused area of the environment. Once again, model complexity is kept low and therefore only the automated perception is used in the Crowd-MAGS agent model.

Automated perception is sensible to two factors. First, the distance between an agent and the stimulus affects how well the stimulus is perceived. Secondly, the agent's cognitive charge affects how well a stimulus is understood. In other words, the more an agent is focused on a task, the less he will perceive and understand surrounding stimuli. The Crowd-MAGS agent model does not model the cognitive charge but it does simulate the impact of distance on perception. Distance defines the ”clarity” of each perceived component. Clarity is best when the perceived component is at the agent's location, and decreases to zero when the component is at the maximal perception distance.

External stimuli can be perceived by any of the five human senses:

- **Sight:** Visual perception retrieves what can be seen by an agent. This perception is sensible to an angle of sight: an agent can see entities only within a given angle around his viewing direction. Also, this perception is sensible to occluding obstacles: an agent can see entities only if they are not hidden by other entities.
- **Touch:** Tactual perception retrieves what can come in contact with an agent. This perception can be voluntary (e.g. an agent touches a fence) or incurred (e.g. an agent receives a rock that was thrown).
- **Hearing:** Auditory perception retrieves what can be heard by an agent. This perception is not implemented in the Crowd-MAGS agent model, but it is simulated so that the crowd can ”hear” commands from the control forces, as explained in Sub-Section 4.2.9.
- **Smell:** Olfactory perception retrieves what can be smelled by an agent. This perception is not implemented in the Crowd-MAGS agent model.
- **Taste:** Gustatory perception retrieves what can be tasted by an agent. This perception is not implemented in the Crowd-MAGS agent model.

In Crowd-MAGS, visual perception (illustrated in Figure 4.23) handles three types of components: perceived agents and STGs, interest shapes, and tear gas clouds. Tactual perception also perceives tear gas, but only when the agent is in contact with the gas (i.e. the agent is caught in a gas cloud). In addition, this perception handles hits from plastic bullets. The perception mechanisms are detailed in Sub-Section 4.5.3.

4.2.4 Memory

This section presents in a detailed manner the memory model, which was shown in Sub-Section 4.2.3 in relation to perception. Three types of memory are presented, short-term, medium-term, and long-term, as well as a functionality offered by medium-term memory. Figure 4.12, which shows the complete memory model, is explained throughout this sub-section.

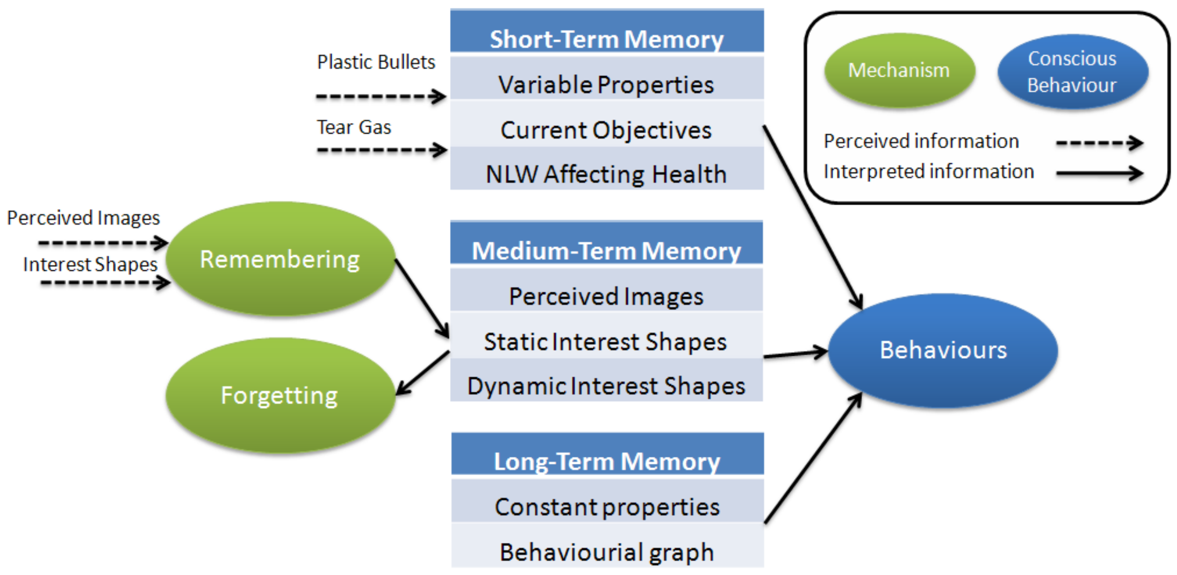


Figure 4.12: Memory Model

4.2.4.1 Short-Term Memory

Every Crowd-MAGS agent possesses an internal image and a projected image (detailed in Sub-Section 4.2.5) that can both store values, which can vary or stay constant. Thus, one part of the short-term memory is implemented by allowing every agent to set values in its internal image as the simulation runs. Examples are the current position, the health level, the destination, and the appreciation of surrounding STGs, to name a few. Values that never get modified represent long-term memory.

Another type of short-term memory is an agent's behavioural graph. The graph is considered as a type of memory because an agent can access the current objective, the previous one, find out if any objective failed or succeeded, etc. Thus, an agent knows what he is doing at any time.

Finally, agents keep in their short-term memory the harm that is currently being done to them by NLW. Thus, behaviours can find out how many bullets just hit the agent, but not how many hit the agent a few minutes ago. If more complex behaviours were developed, harm done to the agent should go in the medium-term memory to simulate behaviours like grudge, vengeance, and fear of reprisal.

4.2.4.2 Medium-Term Memory

The medium-term memory is the most utilized and complex one. Indeed, it stores more information than the short-term and the long-term memories and thus requires more advanced storage and retrieval mechanisms. In addition, the data is accessed and modified more often than in the long-term memory, and thus it must be computationally efficient. Figure 4.12 shows that the medium-term memory is informed by a mechanism (autonomous behaviour) called *Remembering*. Three types of items are stored in medium-term memory: perceived images (of agents and STGs), static interest shapes and dynamic interest shapes. Perceived images are projected images of agents and STGs stored with a clarity value, which depends on how well the agent was able to perceive. Moreover, clarity decreases with time as a result of the *Forgetting* mechanism. Perception, remembering, and forgetting mechanisms are detailed in Sub-Section 4.5.3, Sub-Sub-Section 4.5.4.1, and Sub-Sub-Section 4.5.4.2 respectively. Next, static interest shapes can be points, lines, or zones that will never disappear, such as parks, fences, and protesting areas. Finally, dynamic interest shapes are similar to static ones except that they may disappear, such as the area occupied by a group of people that may eventually be disbanded. The *Forgetting* algorithm removes the shapes that have disappeared.

The problem of multi-agent crowd simulation is a difficult one partly due to the large number and high complexity of agents to be simulated. In order to simulate advanced memory mechanisms at low computational costs, it is proposed to avoid storing perceived images inside each agent's memory. Rather, only a reference to the component that was perceived should be stored in the perceiving agent's memory.

This approximation of reality does not reduce much the plausibility of the model if agents use filtering techniques to access other agents' projected images. When an agent needs to access another component's image, he builds a perceived image on the fly according to the corresponding perception clarity. Filtering can also be used to prevent access to certain data in a projected image to certain types of agents. Filtering can be extended to the temporal dimension, preventing an agent from accessing images that he has not perceived for a long time. Overall, this technique allows for advanced customizable memory mechanisms at very low space and time costs. It also is in line with Wijermans' subjective perception, presented in Sub-Section 2.3.7.

4.2.4.3 Long-Term Memory

To comply with Atkinson's memory model (Atkinson and Shiffrin, 1968), long-term memory must be modelled. Because of the relatively short duration of crowd control situations, this type of memory is not needed *per se*. Indeed, this type of memory is used by the learning mechanisms of human beings; thus, it is reasonable to consider long-term memory as permanent and not alterable for Crowd-MAGS agents. For example, if an agent knows how to speak English, then its knowledge of the language will not improve or degrade over the course of a few minutes or hours.

Crowd-MAGS' long-term memory contains two types of information. First, the constant properties of the agent's internal image represent immutable knowledge. Secondly, the behavioural graphs represent the agent's know-how. These are the only implementations of long-term memory.

4.2.4.4 Providing Queries for Stored Information

As mentioned above, the medium-term memory is the one that is the most solicited. More specifically, most queries relate to finding perceived images or interest shapes with specific semantic names or categories. In order to enable agents to find and recognize components in the virtual environment, each element is defined by a semantic definition, which can be part of semantic categories. The memory model has been designed to answer queries that take one of two types of criteria: a semantic name or a semantic category. Thus, the medium-term memory allows an agent to find a specific component based on criteria that can be very generic or very specific. In fact, these criteria can be used by any high-level behaviour or algorithm that needs to find a location, such as a destination.

As an example, if a behaviour wants to find the closest interest shapes in the category "public building", the medium-term memory's query would look for all interest shapes, keep only the ones with semantic definitions in this category, and then find the closest one to the agent. Another example would be requesting the semantic name "Tourny fountain", in which case only one component should be found. The destination found by the query could be a specific location, a zone, or nothing. In the case where the query returns a zone, the agent could, for example, select a location inside the zone. If nothing has been found, then the objective that requested such a destination may be unable to execute as planned.

4.2.5 Internal, Projected, and Perceived Images

In crowd situations, the way in which an individual interprets behaviours of other individuals or groups (in the crowd or in control forces) significantly influences his decisions and behaviour changes. Hence, it is important to model this interpretation process in order to plausibly simulate various crowd phenomena. Based on Bakker's cognitive approach (Bakker, 2000), and in line with cognitive sciences, which specify that anyone has his own symbolic representation of the world, and of himself, the proposed agent model contains two sets of properties.

First, the internal image is the agent's self-representation, and so it is available only to him. Second, the projected image is the agent's externalised representation, and so can be perceived by other agents. Conceptually, the projected image contains many properties that are also in the internal image, such as the weight and height of the agent. Moreover, a third concept is necessary to customize how the projected images are perceived. A perceived image corresponds to the interpretation of a projected image by an observing agent, as illustrated in Figure 4.13.

These properties are usually constants (height, age, gender, etc.) and variables (fatigue, fear, aggressiveness level, etc.). However, properties could also represent any complex object such as a set of favourite locations or a set of demonstrators that the agent finds unpleasant. Note that components other than agents could have such properties. For example, STGs possess projected images, which will be detailed in Sub-Section 4.3.4

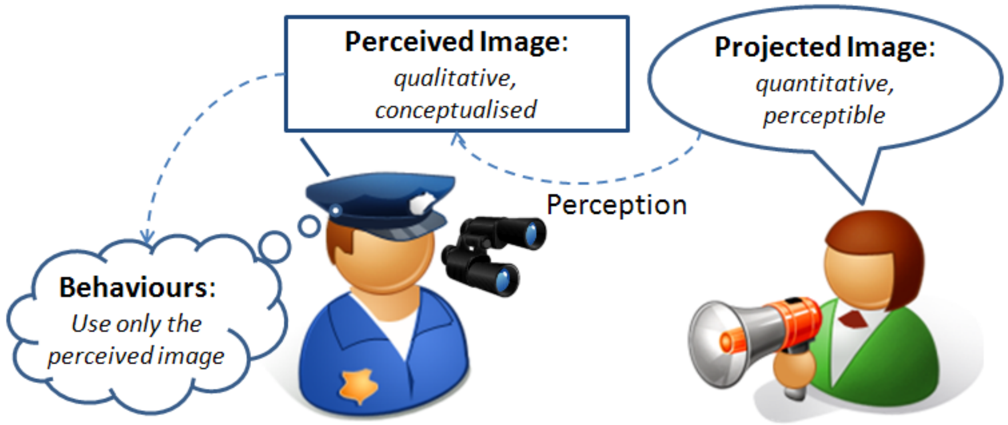

Perceived Image:
qualitative, conceptualised
Projected Image:
quantitative, perceptible
Perception
Behaviours:
Use only the perceived image

Figure 4.13: Projected and Perceived Images

4.2.5.1 Internal Image

The internal attributes of an agent are not directly or exactly perceptible by others. Most of these attributes are quantitative and they are internally used by the agent behaviours. The following list suggests properties that should be included in the internal image:

- Resources
- Fundamental profile
- Memory
- Membership to a social group
- Membership to a STG
- Current destination (where the agent is going)
- Ultimate destination (where the agent would eventually like to go)
- Itineraries
- Required resources
- Support for known STGs
- Enthusiasm

4.2.5.2 Projected Image

The projected image of an agent is directly perceptible by other agents. This set of properties is mainly used to influence the behaviours of other agents. The attributes remain quantitative because they are not conceptualised yet. This step is performed by the observing agent. The following list suggests properties that should be included in the projected image:

- Adopted profile
- Current speed
- History of actions
- Membership to a STG

4.2.5.3 Perceived Image

The projected image of an agent contains a list of observable properties, but they are not directly accessible by other agents. However, observing agents can "reason" on these properties to conceptualise and utilize them. For example, the height of an agent, which is already one of his internal attributes, can be added to his projected image. Others agents can perceive this height, but as in real life, no one can evaluate someone's height in an exact manner by simply looking at them. Instead, the height may be conceptualized by the observing agent using discrete size concepts, such as *small*, *standard*, and *tall*. Another more skilled agent might use categories such as [1.5 m - 1.6 m], [1.6 m - 1.7 m], etc. The manner in which the raw quantitative data will be conceptualized depends on the perception clarity and the filtering technique (introduced in Sub-Sub-Section 4.2.4.2). The technique may depend on psychological and physiological properties of the observing agent, as well as any tools that he may be carrying. This topic could lead to further research.

4.2.6 Social Identity

The social identity is a very high level behaviour that defines the way in which an individual behaves towards others (in crowds or society in general). This kind of behaviour is generally influenced by the surrounding situation (context), and more precisely by the perception of surrounding agents and groups. The social identity is relatively stable for an individual, but may change in specific situations. A distinction must be made between *personal identity*, which refers to the unique characteristics of an individual, and *social identity*, which refers to an individual's general behaviour a member of a social category.

In the proposed model, personal identities are represented by the basic behaviours that agents may possess. In addition, agents exhibit more complex behaviours since they are placed in social settings. All behaviours related to society (interaction with other agents) are organized in social identities, which are implemented with behavioural graphs. Each agent starts with a social identity, which is tagged as his fundamental social identity (FSI). This identity contains the fundamental social properties of the agent and he may keep it for the whole simulation or adopt a different identity based on the perceived situation and on his internal state. The agent normally has a set of candidate social identities that represent plausible changes from his FSI (i.e. a crowd demonstrator might adopt the instigator SI but he cannot adopt the squad leader SI). The SI that an agent has adopted, whether it is the fundamental one or another one from the possible ones, is called his adopted social identity (ASI). People eventually go back to their FSI when the social situation ends, but this behaviour is not necessarily true for the agents in a short simulation.

Figure 4.14 shows two agents and a set of four SI. Each agent can adopt only three SI, one of them being the FSI. The left agent's FSI is *SI 1*, and he has not adopted a different one. The right agent's FSI is *SI 3*, but he has adopted *SI 2*. As a convention, the agents are shown with shirt colors matching the colors of their ASI. For example, if the right agent adopted *SI 3*, his shirt should turn from orange to green.

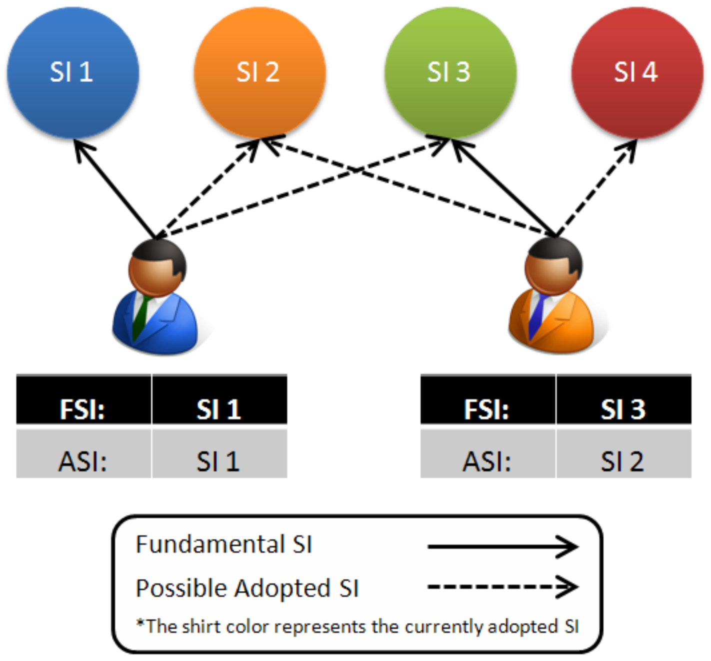


Figure 4.14: Social Identity Model

In order to create crowds that are refined further than what is possible with social identities, the concept of SI profile is introduced. A profile is a simple data structure that contains information (mostly parameters) that influences the agents' behaviours and mechanisms. Each profile is immutable and is linked to a specific social identity. For example, two profiles for the social identity "squad leader" could be created: tolerant and proactive; the latter would have a lower tolerance for aggressive actions and use counter-measures more promptly. It is suggested that profiles alter behaviours slightly at most; if two agents with the same SI but different profiles act drastically differently, then they probably belong to two distinct social identities.

4.2.7 Social Group

A social group represents a set of agents who know each other and who are linked by a social relation. For example, such a group can represent a family, a volunteer association or a police squad. Each agent in such a group plays a role, which is represented by a specialized social identity. Unlike STGs, these groups are not perceptible (i.e. agents who are not part of the group do not know that the group exists, even if they perceive the members). In addition, the members are not necessarily located close to one another. However, the members of a social group can also be in a STG (that contains exactly the same members) as for example police squads.

For simplicity's sake, the concept of role is modelled using social identities. When an agent joins a social group, he must choose a role. The group can define as many different roles as required, as shown in Figure 4.15. While playing this role, the agent drops his current social identity to adopt a new one, which normally has very specific purposes (e.g. *squad leader* social identity). Members of the group can change their roles, such as a squad deputy leader who temporarily becomes the squad leader. It is possible that some roles in a group are not filled by any agent. In addition, an agent cannot belong to more than one social group — this issue is discussed in Sub-Section 4.7.4.

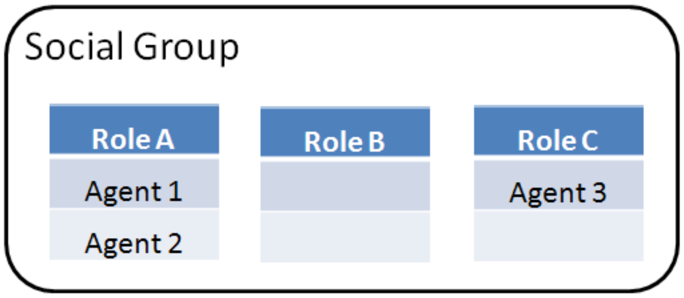


Figure 4.15: Social Group Model

4.2.8 Health

The concept of agent health is still rarely modelled in crowd simulations. It is a complex issue that affects many other parts of the agent model, such as perception, navigation, memory, and behaviours. Analysis of crowd control videos from various events in the Western culture was performed during the Crowd-MAGS project and revealed that one of the most obvious consequences of a protestor getting hurt by plastic bullets or tear gas is change in his behaviour. Consequently, the agent model should reflect this impact.

Development of the Model

The first idea that was examined was to consider health as a resource, so that it could prevent the execution of certain behaviours. The agent would possess a certain level of health, which could be modelled as a quantity comprised between 0 (dead) and 1 (fully healthy). This approach raised two main problems. First, PLAMAGS' behaviour engine cannot handle resources with quantities but only resources that are either available or not. Secondly, the *health* resource would have to be represented by a new type of resource that does not get reserved by the behaviours that use them. In other words, if two behaviours require a health level of at least 0.6 and the health level of the agent is 0.8, then both behaviours should be able to execute. However, the PLAMAGS engine, if it could handle such a resource, would reserve 0.6 unit for the first behaviour, leaving only 0.2 for the next one, which could not execute. In consequence, modelling health as a resource to be handled by the PLAMAGS engine was not a feasible approach.

Based on the practicality of viewing health as a resource and on the PLAMAGS engine' limitations, it was decided to implement the health concept as a new type of "resource" that does not affect the behaviours' execution sequence, but rather how well objectives get performed (e.g. agents move slower when hurt). This idea is inspired from Silverman's performance moderator functions (PMFs), introduced in Sub-Section 2.3.4. For Crowd-MAGS, it was proposed that an agent's health starts at 1 and can go down to 0, at which point the agent is dead. Localized damage was not modelled but rather a concept of global health was selected. Localized damage would allow independently considering damage to different parts of the agent body: eyes, mouth, arms, hands, legs, etc. Although such a model would be more realistic, it is highly complex and bears numerous impacts on the design of the agent behaviours.

Factors Impacting Health

Agents can get hurt by four different types of non-lethal weapons. First of all, each plastic bullet received by an agent reduces its health by a fixed percentage. Next, each second spent in tear gas reduces an agent's health by a varying percentage. The exact value varies with the gas concentration around the agent. Both of these weapons cannot kill an agent because their impact is relative to the agent's current health, creating functions with asymptotes at $health = 0$. On the other hand, the other two weapons cause absolute impacts on the agent's health. The first one is the Molotov cocktail, which reduces health by a large amount, regardless of the current health level. The second one is fighting, where each punch reduces health by a small amount multiplied by the aggressor's health. The actual values can be changed in the scenario file and need to be calibrated using scientific studies to reflect injuries that could happen in a real situation.

For increased realism, the notion of *protection against injuries* was implemented. For example, officers and instigators may wear gas masks, eliminating the health impact from gas clouds. This accessory also changes the instigators' behaviour, who do not run away from gas clouds anymore. Officers' armours protect them against different types of attacks (punches, Molotov cocktails, etc.) at various degrees, which require calibration.

Impacts of Health

Silverman's integrative PMF framework is based on serious literature analysis and has been designed to be adaptable to various projects. However, due to the relatively short time available to develop the health module during the Crowd-MAGS project, only simplified algorithms, inspired from Silverman's works, are used in Crowd-MAGS to reflect the impacts of health on the agents. Specifically, two major and one minor behaviours are influenced by health.

First, an agent's speed is proportional to his health. Thus, an agent who normally walks at 1.4 m/s slows down to 0.84 m/s when his health drops to 0.6. He can still run (for example, if he is fleeing from tear gas), but his running speed will still be only 60% of what it would normally be. The second behaviour that is affected by health is the forgetting algorithm (presented in Sub-Sub-Section 4.5.4.2). Projected images are always perceived with a certain clarity, and the forgetting algorithm slowly decreases this clarity until it becomes zero, at which point the perceived image is removed from memory. The speed with which clarity decreases is inversely proportional to health, making an injured agent forget STGs as well as other agents relatively fast. Finally, the behaviour that makes instigators fight with CF officers receives an impact that is proportional to the agent's health. This concept simulates the fact that an injured person throws weaker punches and kicks than a healthy person.

Another impact of health is that agents will eventually leave if they are injured enough. Tests have shown that agents who get hurt too seriously amidst a large crowd leave very slowly because they cannot walk very fast. Consequently, they often get stuck or even get pushed back, which frequently leads them to getting more injured by tear gas or other sources. Eventually, their health drops to zero and thus their walking speed too. From that point, these agents are "paralyzed" since they cannot move themselves. In consequence, it is proposed that agents be removed from the simulation when their health falls below a critical level — until advanced behaviours that would make agents help each other are designed. This level should be customizable and depend on the agent's adopted SI. For example, a bystander would be removed from the simulation when he would be slightly hurt but an instigator would stay until he would be critically wounded.

4.2.9 Communication

As mentionnned in Sub-Section 4.2.3, hearing perception has not been implemented. Therefore, modelling one-to-one and broadcast communication is quite difficult. Technically, no type of communication has been truly implemented; however, agents have the possibility of broadcasting verbal messages in their surrounding environments through different means. In fact, nothing happens externally when an agent broadcasts a message, but the action is added to his history of actions. Since no auditory perception is modelled, agents cannot "hear" the message through an unconscious mechanism, like visual perception. Rather, they must look in the history of actions of other agents to determine if these agents sent a message. This process is an active conscious behaviour, which is inefficient and thus used as seldom as possible. Consequently, the behaviours have been designed so that communication does not play a major role.

4.2.10 Discussions

The perception model presented above could be seen as an extension of more 'standard' models, such as those proposed in (Pan, 2006; Thalmann et al., 2000; Wijermans et al., 2008). Indeed, Pan and Thalmann suggested that agents be equipped with multiple sensors, such as visual, tactile and auditory ones. In addition, Pan and Wijermans proposed that visual perception be available in a cone that may get obstructed by environment elements, such as objects and agents. Moreover, Thalmann insists that agents be able to perceive actions from other agents. He mentions that if an agent's perception is restricted to the objects and other agents (their actions not being taken into account), then the behaviours will be limited because "only the presence and the characteristics of an object or an actor are implied in selecting a behaviour." All of these ideas and recommendations have been integrated in the Crowd-MAGS model. In fact, visual perception can be compared in Figures 2.11 and 4.23.

The concept of perceived images also is similar to Wijerman's works. She mentions that perception is subjective, due to the agent's previous experiences and current state. "This can be seen as a sort of filter on 'objective' perception.' This technique has been modelled in Crowd-MAGS and it was explained in Sub-Section 4.2.5

It is also interesting to notice that in Crowd-MAGS, profiles are associated with agents. In SULNT, presented in Sub-Section 2.5.3, profiles are associated with crowd factions. The agents inherit characteristics and behaviours based on their faction's profile. This approach could be considered as top-down, whereas Crowd-MAGS uses a bottom-up approach: agents get their characteristics and behaviours from their personal profiles and then the crowd exhibits behaviours that emerge from the agents and their profiles. This technique can lead to more precise and accurate definitions of crowds, but it probably requires more lengthy calibration. It should be noted that crowds in Crowd-MAGS can be characterized by profile distributions that adjust the proportions of agents that get assigned to certain profiles.

4.3 Spatio-Temporal Group Model

Emerging groups are ubiquitous in crowd situations and thus in Crowd-MAGS simulations as well. It was observed that in real crowd control situations, emerging groups can drastically change the course of events. For example, a small group of instigators, who may not even know each other, might assemble to throw projectiles at control forces. The officers might then decide to wear heavier protection, which may anger many pacific protestors. It was observed also that the most aggressive actions performed by instigators are performed in small groups (3-5 people), which assemble and disassemble quickly and frequently. Consequently, the agent model must work in pair with the group model and the notion of STG will be used to model these emerging groups. The STG model is described in detail in this section, starting with Figure 4.16 which shows an overview. It is important to keep in mind that this model is very generic, and does not represent any specific type of STG, such as a group of instigators or a police squad. In fact, the STG type is a property that defines the social identity of the controller agent, influencing the types of agents who are willing to join the STG.

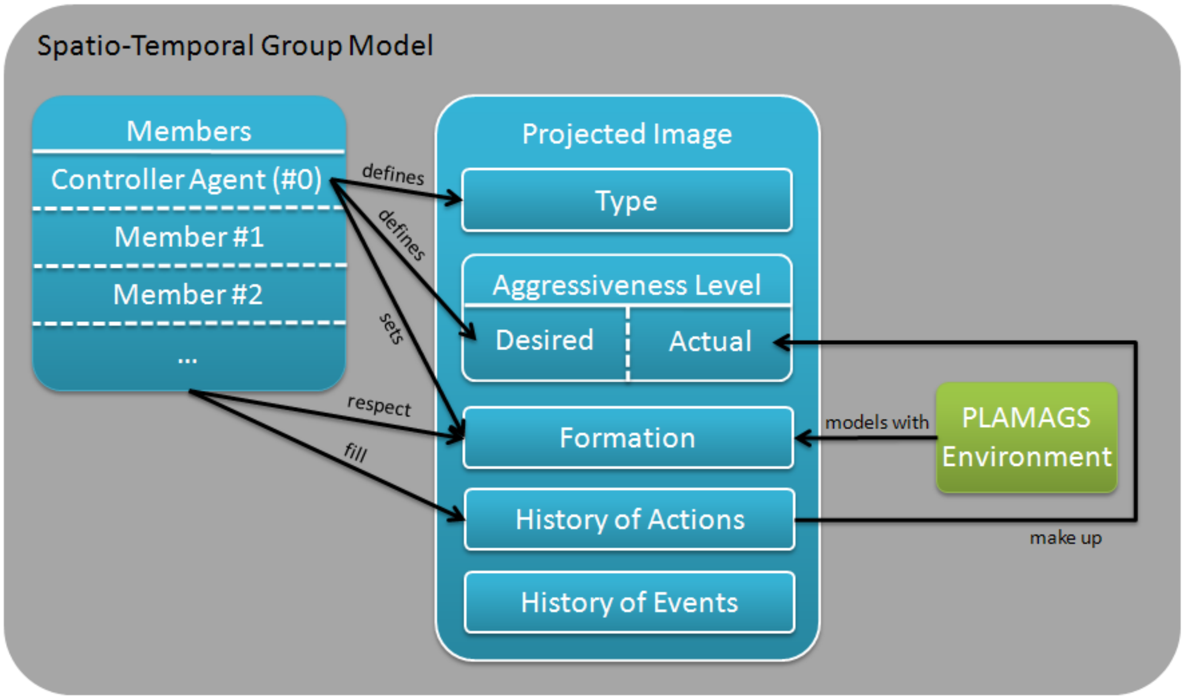


Figure 4.16: Spatio-Temporal Group Model

A spatio-temporal group is a number of agents who are gathered in a certain way around a position or around an agent. The members do not necessarily know each other and have no special roles or hierarchy. One exception exists for the controller agent, who defines how the group will evolve. More details are discussed in Sub-Section 4.3.1. Like agents, STGs project an image that is perceptible by all agents; details are given in Sub-Section 4.3.4. Two components of the projected image are important enough to be discussed separately. The aggressiveness level, which is a composite measure reflecting the behaviour of the members, is discussed in Sub-Section 4.3.3. Finally, the formation is a geometric definition of the STG and is detailed in Sub-Section 4.3.2.

4.3.1 Organization of Members

The STG model has been designed to allow any agent to join or leave any STG at any time. Thus, members in a STG are not necessarily related to each other. In fact, they may even belong to different social groups and may have never perceived each other in the environment. Agents do not have to follow any rules when joining a STG because there is no concept of roles. However, the foundation of STGs is that agents are spatially regrouped. Thus, agents must respect the formation prescribed by the STG as long as they belong to the STG. The behaviour that handles this task, a basic human behaviour that is part of all agents, is detailed in Sub-Sub-Section 4.5.2.3. However, agents will temporarily stop respecting the formation if a behaviour with higher priority takes the *navigation* resource. A common example is an agent who gets caught in tear gas and flees from the gas cloud. The priority of the behaviour to flee from tear gas, detailed in Table B.27, has been designed to be higher than the one to maintain formation. When the agent is not in the gas cloud anymore, the fleeing behaviour terminates and thus releases the *navigation* resource; in turn, the formation behaviour takes back the resource. The agent then walks back to its prescribed position in the formation.

No role is defined within a STG, but one special agent may exist: the STG controller agent. This agent is not constrained by the formation and thus navigates freely in the environment. Other agents of the formation take their positions with respect to him. When no controller agent exists in a STG, agents get their positions with respect to the hot-spot of the formation, as explained in Sub-Section 4.3.2. Controller agents may decide to dissolve the group or force a member to leave at any time.

A notion of rank among member agents exists, but it does not refer to hierarchy or roles. Ranks are merely used to assign positions within formations. Often, rank zero is reserved for the controller agent so that the formation gets organized around this agent. A list of members is kept in the STG. New agents who join a STG may be inserted at any rank, and agents may see their ranks changed at any time.

STGs may also be used in conjunction with social groups (detailed in Sub-Section 4.2.7). For example, families and police squads are social groups, and their members can be spatially close to one another. Thus, a convenient way to deal with such situations is to enable the "leader" of the social group (such as the squad leader) to create a STG as soon as possible and ask the other members of the social group to join the STG. The combination of both types of groups is novel and significantly increases the plausibility of the social phenomena observed in Crowd-MAGS simulations.

4.3.2 Formation

The formation is the most prominent feature of a STG, because it ensures the spatial cohesion of member agents and allows other agents to perceive the STG. In addition, the formation helps the user to visualize the STG as it evolves through the simulation. Certain formations can also be interpreted in various ways by other agents, such as CF squad formations.

The formation is simply the geometrical shape of a group of agents. It is always defined relative to a hot-spot, which is a position that is the reference point for the formation. The hot-spot can be located anywhere, but it is often located at the centre or at the front of the formation. Hot spots can remain motionless for the whole simulation, or they can be attached to an agent and move with him. It is frequent to attach the hot-spot to the controller agent of the STG, so that the hot-spot always moves with this agent. Formations can be strict or non-strict. In the former case, an exact position is assigned to a member based on his rank (Figure 5.4d is an example). In the latter case, the formation merely requires that the agents stay within a few meters of the controller agent, with no other indication (Figure 5.4c is an example). Formations may be changed or modified at any time by any agent or process, but the controller agent normally handles this task.

It is proposed to implement the formation concept using an interest zone (defined in Section 4.1) to which a hot-spot is added. Figure 4.17 shows two examples of STGs with their interest zones; the agents should stay within the coloured zone. With this architecture, the formation benefits from all of the features implemented by the interest zone. For example, the formation can answer geometrical queries such as indicating if it intersects with another formation. This particular feature is useful to find out if an aggressive STG intersects with a squad in formation. A strict formation provides a strict position to a member based on his rank, and a non-strict formation provides a random position within a range, ignoring the member's rank. In addition, the formation allows a member to check if his current position respects the formation, in order to reduce the number of times where members request new positions.

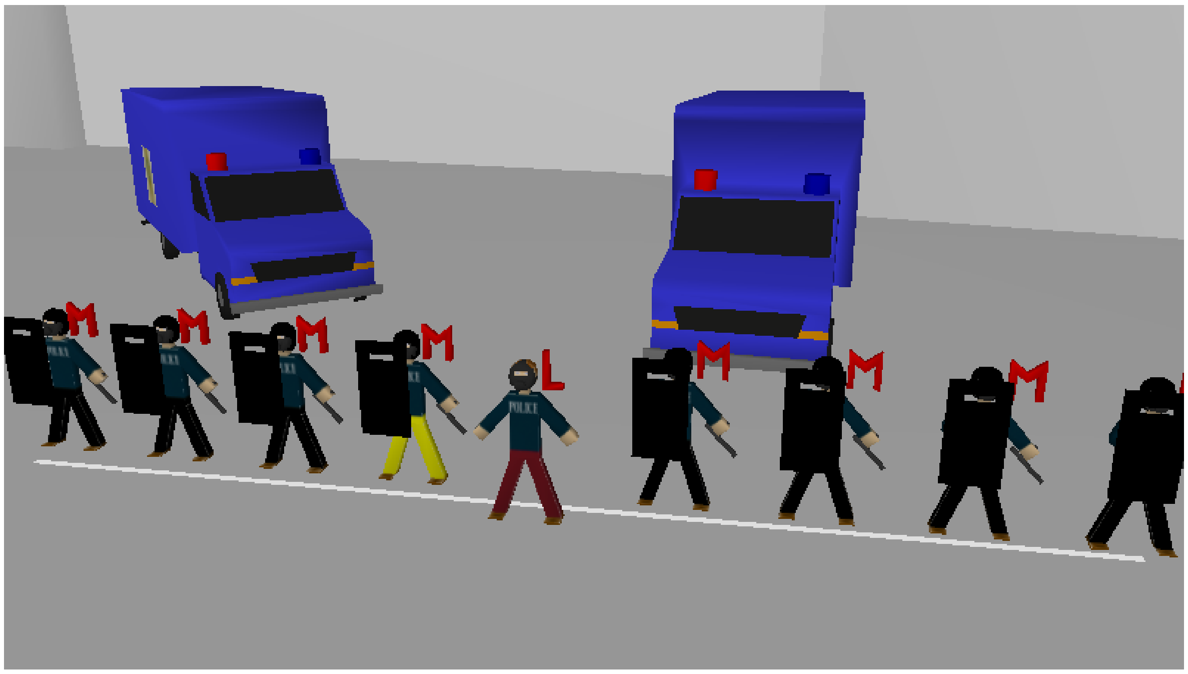
Before instantiating a specific formation, a formation description must be created. The description must contain a type name and specify if it is strict or non-strict. The algorithms for the queries mentioned above must also be implemented in the description. Finally, for every rank, the position must be stored as both a lateral and a forward displacement with respect to the hot-spot. For example, a wedge formation could specify that rank 0 has no displacement, that odd ranks (1,3,5,...) go *rank* meters to the left and backward of the hot-spot (rank 0), and that even ranks (2,4,6,...) go $rank - 1$ meters to the right and backward. These values can be planned in advanced and hard-coded, in which case no structural modifications are allowed as the simulation runs. Another option is to use a customizable formation, which is shown in Figure 4.18.

At first, a basic customizable formation is defined with only rank 0. Thus, the formation can contain a maximum of one agent whose position will be the hot spot. Then, positions can be specified one by one for as many ranks as desired. This procedure is performed by giving for each desired rank number a lateral and a forward displacement. For example, Figure 4.18 defines a wedge formation for five agents. The circles represent the position of each rank (as viewed from above). The displacements in X and Y have no units, and thus this formation can be stretched or shrunk as desired at run-time. In addition, the displacements can be modified at run-time, allowing the wedge to become narrower, for example. The main drawback of the customizable formation is that the number of members in the STG is limited by the number of ranks that have been defined.

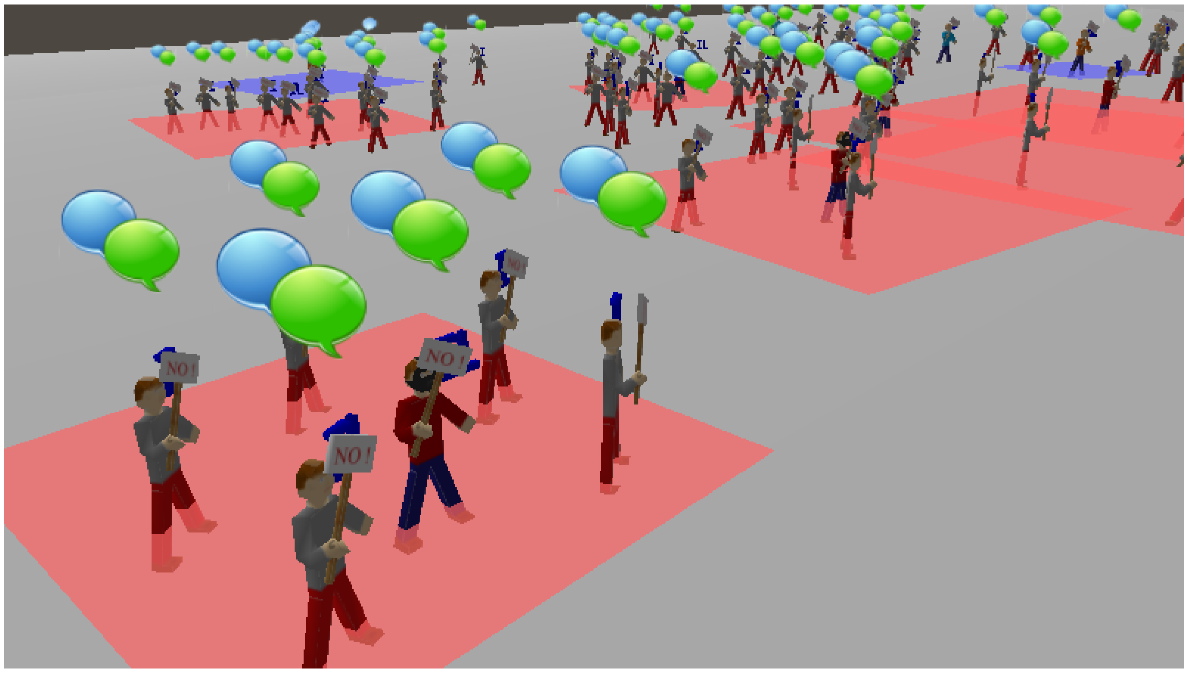
4.3.3 Aggressiveness Level

The aggressiveness level of a STG is a conceptual measure that represents the average aggressiveness expressed by the STG's members. Based on the social identity theory, agents assess their enthusiasm towards a situation periodically. The algorithm that performs this task, presented in Sub-Section 4.5.7, would be computationally infeasible if each agent inspected all actions of all other agents. Thus, the STG's aggressiveness level was introduced, among other reasons, to speed up calculations of enthusiasm. Instead of inspecting all other known agents, an agent simply looks at the aggressiveness level of all known STGs. Because the number of known STGs tends to be much smaller than the number of known agents, the algorithm executes quickly.

It is the aggressiveness of the members' actions that makes up the overall aggressiveness of the STG. Since there is no hierarchy or roles, the STG's aggressiveness level is simply



a) Sideways Line Formation (Performed by a Squad)



b) Loose Formation (Performed by Instigators)

Figure 4.17: Formations with their Interest Zones

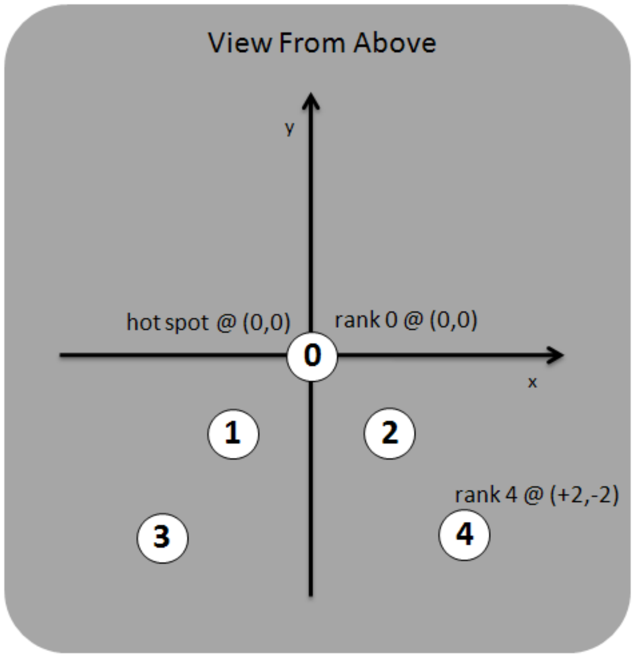


Figure 4.18: Customizable Formation (Example Viewed From Above)

the average of each member's aggressiveness level. If there is no member in the STG (such as a waiting line where no one has arrived yet), then the level is set to zero. Tests with this initial model revealed a problem related to the fact that agents try to join groups that display a certain level of aggressiveness. When a crowd leader invites other agents to join his STG, he has not yet started performing aggressive actions (which gives his STG a low aggressiveness level) and thus other agents may not be interested in joining his STG. To counteract this phenomenon, a desired aggressiveness level, set by the STG controller, was added to the STG's projected image. When agents browse STGs to find an interesting one, they can look at the desired aggressiveness level, rather than the actual one. The desired level is somewhat a target that the member agents should respect. Thus, an agent who seeks a highly aggressive STG may join a STG with a low actual aggressiveness level but with a high desired aggressiveness level expecting that the members will eventually be more aggressive.

In addition to the STG joining process, the STG's aggressiveness level is used for other reasons. First, it allows members to evaluate if they want to stay in the STG (e.g. maybe an agent is looking for something else, or maybe other agents changed their behaviours so drastically that the agent does not like the STG anymore). Also, the aggressiveness level is used by CF agents to assess the crowd members' actions, for example when selecting a target for a tear gas can or deciding where to dispatch squads. In fact, the STGs' aggressiveness levels can even help a user to understand the situation as a scenario is run. Because the aggressiveness level is an aggregated measure (of agents' aggressiveness levels), it is very useful when performing scenario analyses.

4.3.4 Projected Image

The projected image of a STG is similar to the one in the agent model and consists of a set of properties that agents perceive. The STG projected image is constantly accessed by agents who are evaluating the STG for various reasons. The projected image contains the STG type, a history of actions and a history of events (both presented in detail in Section 4.4), as well as the zone covered by the formation. Like the agent projected image, additional properties may be added to the STG projected image.

4.3.5 Discussions

It should be noted that this group model is quite novel because it differs significantly from other proposed group models. For instance, Thalmann proposed a model where the intelligence, memory, and perception are centered on the group structure, as explained in Sub-Section 2.4.2. Indeed, the group's perception and memory are associated to only the leader. In Crowd-MAGS, agents maintain their own perception and memory capabilities, no matter if they are in a STG or not. Also, STGs have no 'memory' *per se*; however, they do project an image containing a history of events and a history of actions. Another difference is that due to Thalmann's memory structure, he must impose a capacity limit, whereas Crowd-MAGS's memories are unbounded. Finally, Thalmann proposes that groups be able to perceive other groups' internal statuses. In Crowd-MAGS, a clear distinction is made between internal and projected images, so that others can perceive only 'public' information. Moreover, STGs do not possess perception capabilities; only agents do.

Although the Crowd-MAGS group model does not include reasoning, perception, or memory, it is able to recreate quite a range of crowd behaviours. Thalmann listed eight behaviours that are implemented in his system, as presented in Sub-Section 2.4.2. Also, Pan showed three crowd behaviours, which are presented in Figure 2.16. The following list explains how some of these behaviours can be recreated in Crowd-MAGS. More details on some of the behaviours will be given in Section 4.5.

1. **Flocking / Herding:** With a loose formation, agents will flock around the STG's controller (an example is shown in Figure 4.17b)
2. **Following:** An agent's destination may be set to another agent
3. **Queuing:** Not implemented, although a formation that assigns positions so that agents line up one behind another could be created easily.
4. **Attraction / Repulsion:** Agents' behavioural graphs may (positively or negatively) influence the agents' desires to reach certain locations (or agents or groups)
5. **Splitting:** An agent who does not support his group will leave to join or create a new one
6. **Space Adaptability:** Not implemented

7. **Safe-Wandering:** PLAMAGS' navigation system makes agents go around obstacles
8. **Competitive Behaviour:** If multiple agents try to go through a tight passage simultaneously, they all try to go first, often resulting in bottlenecks

4.4 Information Model

Multi-agent simulations, and most specifically crowd simulations, are systems that are more complex than analytic models, especially since they are not deterministic. In fact, they even obey principles of Chaos Theory. (Ferber, 1995; Gleick, 1987; Moss and Edmonds, 2005) It is therefore crucial to gather information in an intelligent and purposeful manner while the simulation runs so that it can be analyzed in-depth ulteriorly. The lack of useful information could lead analysts to observe initial conditions and final results, but not be able to explain the phenomena that take place between the two.

This section proposes a model to organize data generated by a crowd simulator. First, Figure 4.19 shows the overall classes necessary to keep the collected data in the most versatile format. Aiming for versatility prevents some optimizations for time and space, as they would reduce the usability or precision of the data. In addition, some optimizations cannot be planned since specific analysis objectives are undefined. Thus, this model stays at a generic level, but it can be applied directly for specific purposes, such as evaluating the efficiency of NLW in crowd control situations.

Figure 4.19 shows the *Data Collector*, which is the centerpiece of the information model. This class contains two sets of STGs: one for those currently in the simulation, and one for those that have been removed. The Data Collector contains also two sets of agents: one for agents currently in the simulation, and one for those that have been removed. In fact, agents are "created" but never "destroyed" because they are always kept in the Data Collector.

As mentioned before, every agent possesses a history of events and a history of actions, whose names intuitively explain their functions. However, the internal structures are not trivial. The following two diagrams (Figure 4.20 and Figure 4.21) show the basic building blocks necessary to record all actions and events in the system. These structures store everything that could be foreseen as useful, since specific analysis objectives are undefined.

To begin, Figure 4.20 shows that any event contains at least three elements: a semantic definition, a time stamp, and an all-purpose container. The semantic definition is used to identify and classify events, a technique that is used for efficient retrieval in the analysis phase. The time stamp represents the simulated time at which the event happened. The "extraInfo" component is often empty, but anything can be stored in it, for specific analysis purposes. In fact, if multiple items need to be stored, they can be placed in a collection so that "extraInfo" refers to the collection.

Next, two types of events are modelled. A non-observable event represents anything that happens to an agent or a component, such as being created or dragged by the user, that is not observable to the agents. Thus, neither the agent itself nor other agents know about

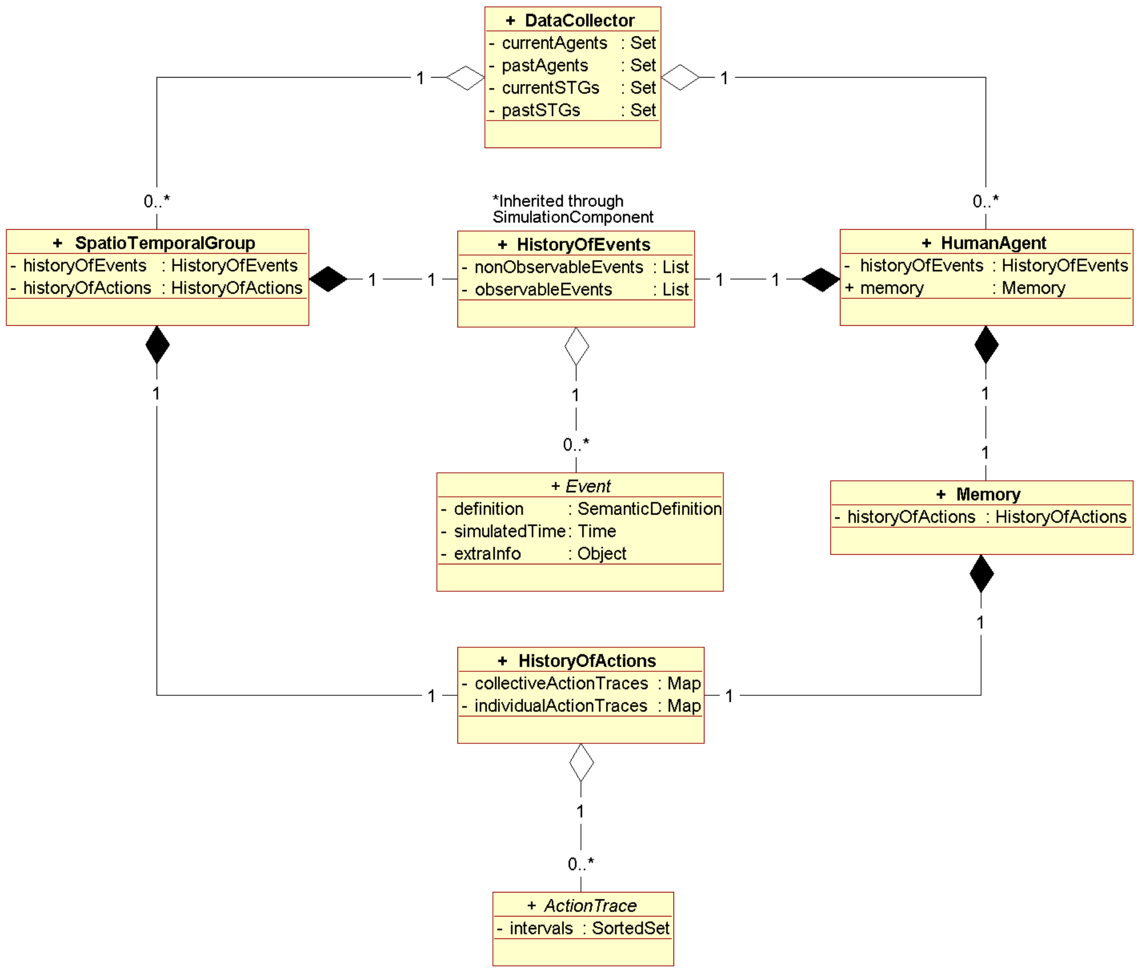

+ DataCollector
- currentAgents : Set
- pastAgents : Set
- currentSTGs : Set
- pastSTGs : Set
*Inherited through SimulationComponent
+ SpatioTemporalGroup
- historyOfEvents : HistoryOfEvents
- historyOfActions : HistoryOfActions
+ HistoryOfEvents
- nonObservableEvents : List
- observableEvents : List
+ HumanAgent
- historyOfEvents : HistoryOfEvents
+ memory : Memory
+ Event
- definition : SemanticDefinition
- simulatedTime : Time
- extraInfo : Object
+ Memory
- historyOfActions : HistoryOfActions
+ HistoryOfActions
- collectiveActionTraces : Map
- individualActionTraces : Map
+ ActionTrace
- intervals : SortedSet
1
0..*

Figure 4.19: Information Model: Overview

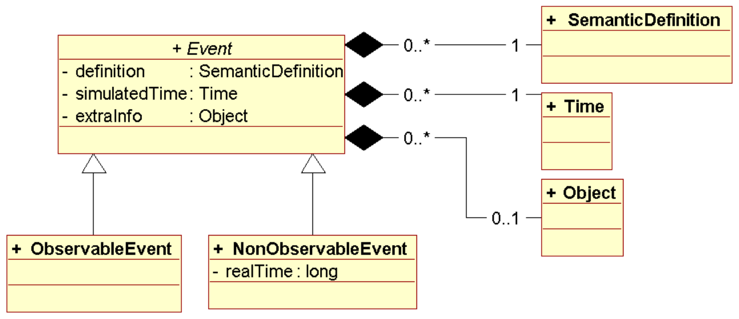


Figure 4.20: Information Model: Events

non-observable events. They can be used for debugging purposes as well as runtime and post-simulation analysis. These events also contain another property: the real time at which the event happened — which is unrelated to the simulation time. On the other hand, observable events do not contain this property and can be observed and analyzed by other agents. An example is an agent whose health reaches a critical level; any agent can observe this event.

Figure 4.21 shows the diagram for action traces. Actions are normally triggered by an agent, in contrast to events that are imposed by the user or an external entity. Before explaining the action traces themselves, it is important to understand what the ActionDefinitions are. First, a separation between the action traces and action definitions was preferred because action traces are far more numerous than action definitions (i.e. 100 agents yelling create 100 traces, but only one definition).

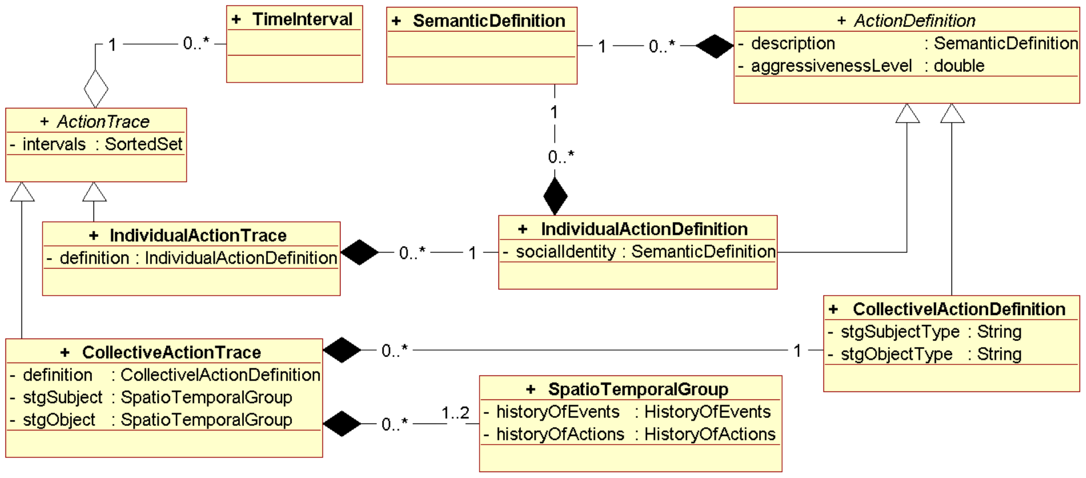

+ TimeInterval
+ SemanticDefinition
+ ActionDefinition
- description : SemanticDefinition
- aggressivenessLevel : double
+ ActionTrace
- intervals : SortedSet
+ IndividualActionTrace
- definition : IndividualActionDefinition
+ IndividualActionDefinition
- socialIdentity : SemanticDefinition
+ CollectiveActionDefinition
- stgSubjectType : String
- stgObjectType : String
+ CollectiveActionTrace
- definition : CollectiveActionDefinition
- stgSubject : SpatioTemporalGroup
- stgObject : SpatioTemporalGroup
+ SpatioTemporalGroup
- historyOfEvents : HistoryOfEvents
- historyOfActions : HistoryOfActions
1
0..*
1..2

Figure 4.21: Information Model: Actions

An action definition is a very simple structure that contains a description and an aggressiveness level. Every instance of an action definition must be either an individual action definition or a collective action definition. The only difference is that an individual action additionally stores the social identity of the agent who performs the action, while a collective action stores the types of the STG performing the action and the type of the STG which is the target of the action, if there is one. The aggressiveness level of each action definition depends on various factors. For example, a collective action with description “Yelling” would be more or less aggressive depending on the STG types involved. If instigators yell at a CF squad, then this action could be perceived as slightly aggressive. However, if pacifists yell at another group of pacifists, then the situation would be abnormally aggressive. This architecture allows for fine-tuning during the calibration phase and powerful search, classification, and aggregation during the analysis phase.

Next, an action trace records that an agent performed a certain action (represented by its definition) at certain points in time. An action trace contains a set of time intervals, which represents periods during which the agent was performing the action (e.g. [1m04s-1m25s], [3m00s-4m53s]). Like the action definitions, the action traces are divided into individual and collective ones. Collective action traces store additional information: complementing the two STG types stored in the definition, references to the actual STGs are kept. This information allows for powerful queries, such as determining how many projectiles were exchanged between two specific STGs, and for monitoring the evolution of the relationship between two specific STGs.

4.5 Generic Behaviour Models

All of the models presented so far are static because they propose ways to structure data, components, and agents without discussing how these data structures will evolve and interact. Now that the reader is familiar with all necessary models, this section presents eight basic behaviours that make use of the data structures presented above. More specifically, the first five are low-level behaviours (or mechanisms) that apply to all human agents, while the last two are high-level conscious behaviours that apply to only the crowd agents (i.e. not CF agents or others like media and paramedics).

These behaviours are based on the perception-decision-action loop representation (Lord and Levy, 1994), which has proven its suitability to model human behaviour. The decision process of this loop, which corresponds to the five behavioural stages of a human being, has been detailed in (Newell, 1990) with the well known behavioural pyramid (Figure 4.22).

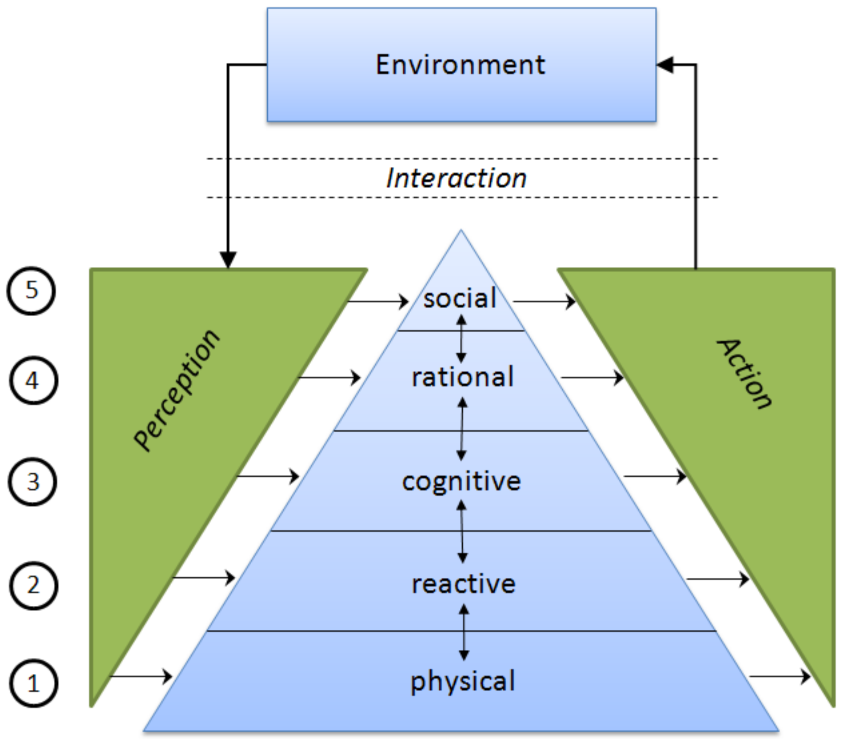


Figure 4.22: Newell's Behavioural Pyramid

Each behavioural stage is explained below with a short description.

- **Physical:** This bottom layer corresponds to reactions of the body to the laws of physics, like bleeding after being cut and get hungrier as time goes by. These behaviours should be simulated at the highest frequency of the simulation engine (ten times per second in the proposed environment model).
- **Reactive:** These behaviours are automatically performed by human beings because

they are normally unconscious behaviours (e.g. running away from tear gas when it burns). However, a person can take conscience and even take control of these behaviours. For example, a person could choose to stay in a cloud of tear gas. Alternatively, some rational behaviours can become reactive with training, like driving a car though a known path. These behaviours also should be simulated at the highest frequency and they are always simulated as unconscious behaviours.

- **Cognitive:** This type of behaviour is completely conscious and it requires the human to focus on the corresponding task. These behaviours are usually segments of a long-term rational process. For example, a person who is reading and memorizing a street sign while trying to find his way in an urban area. These behaviours should be simulated at a lower frequency, such as once per second, because they do not require a high precision in time.
- **Rational:** This layer of behaviours organizes cognitive tasks in order to accomplish more complex objectives. For example, planning to gather objects in a given area to assemble them and build a barricade. These behaviours are often complex and related to long-term goals, and thus they should be simulated at a low frequency, such as once per ten seconds.
- **Social:** This last behavioural level is more loosely defined than the other ones. It corresponds to the influence of society (and membership to groups) on the other behavioural levels. For example, it could influence a pacific crowd member to eventually become aggressive and throw projectiles at control forces. This influence spans a long period of time and thus these behaviours should be simulated at a very low frequency, such as once per minute.

The behaviours presented in this section cover four of the five stages. The bottom three stages can be considered to be similar among all humans, at least for the kind of scenarios to be simulated in the Crowd-MAGS project. For example, two individuals might go pick up a projectile from the ground in different manners, but the difference would have no significant impact on a simulation. Thus, the behaviours at these levels are generic enough to be presented here. The top most level refers to social influences and behaviours, which have not been modelled much in multi-agent simulation. Thus, this thesis proposes social behaviours that are applicable to all crowd member agents. Control force agents only perform their duty during the span of a simulation and thus no specific social behaviours have been implemented for them. In fact, the STG and formation mechanisms are sufficient for plausible social behaviours.

Non-Generic Behaviours

The more complex behaviours that the agents need to perform, such as planning a peace march or selecting a location for throwing a tear gas can, are situated at the rational level. These behaviours are much more elaborated, and obviously differ for different types of agents. For example, an instigator's rational behaviour has nothing in common with a squad leader's rational behaviour. These behaviours are much more subject to continuous improvement and extensions. For these reasons, they are presented in Appendix B. In fact, it should be noted that the Crowd-MAGS system, similarly to the MASSEgress system presented in Sub-Section 2.3.6, allows the behaviours to be added to and removed from agents easily. Hence, these behaviours have been tested independently before being incrementally integrated into the system.

Choices needed to be made to select what behaviours, and most specifically, what collective actions would be modelled. It is natural that the most frequently observed actions in real situations be modelled first, since they are the ones with the most significant impact on the simulation. In the early stages of the project, hours of riots and more peaceful crowd events were watched, in particular to identify common collective actions. These key findings are presented in Section A.1. Before selecting the behaviours to be implemented, this list was compared to others from the literature. In particular, Table 2.3 shows a list from (McKenzie et al., 2005b); this list is organized by aggressiveness level. Other lists from (Schweingruber and McPhail, 1999), which have been presented in Figure 2.20 and Figure 2.21, classify actions by category and by region of the body. Finally, an unorganized series of actions is listed in (Varner et al., 2000), including loitering, celebrating, carrying signs, throwing stones, demonstrating, rioting, and dispersing. Many actions were common among the four lists, and they were deemed to be the most important ones.

4.5.1 Resources Acquisition

As mentioned in Sub-Section 4.2.1, the concept of resources has been created to be generic and used in several different ways. In fact, resources are extensively used in Crowd-MAGS behaviours. Thus, a dedicated behaviour in charge of acquiring needed resources has been developed. This behaviour is at the *cognitive* level of Newell's pyramid because it is a conscious behaviour that is accomplished as part of a larger rational behaviour. In short, each agent possesses a list of required resources to which any behaviour requiring resources may add what is required. Then, the agent automatically starts looking for appropriate resource suppliers. Once a supplier is found, the agent goes towards it and obtains the resource. This section details this resource acquisition behaviour.

Many cases are possible when an agent is looking for a resource supplier. For example, the best supplier could be at hand's reach, could be at a great distance, or could be unreachable. A resource supplier could have an infinite number of resources (e.g. water fountain), a limited but sufficient number, an insufficient number (the agent needs more than what is available), or could possibly not provide the required resource. In the proposed algorithm, the agent goes to the closest supplier that has at least one available unit of the most required resource. However, it is possible that no unit is left by the time that the agent reaches the supplier. At this point, the agent will go towards the supplier that is the closest to its current location. If there is none, then the agent will stop looking for this resource until it is requested again. More cases exist when an agent has obtained all of its desired units at a supplier. First, the agent checks if it could obtain other resources at this supplier. If so, it will bump these resources to the top of the list, one after the other. Thus, this behaviour changes the priorities and the ordering of the list. If no other resource can be obtained from this supplier, the agent checks if it requires any other resource. If so, it heads towards the closest appropriate supplier. Otherwise, this behaviour is completed (until a new resource is required). This behaviour must also handle the cases where required resources get added or removed, or get their priorities changed, while going to or using a supplier.

A precise implementation of this behaviour is presented below. The implementation is divided into two parts. The first part is an objective coded in BDL that sets the agent's destination to the location of the most appropriate resource supplier. Of course, other behaviours also may want to set the agent's destination so this objective requires the *navigation* resource. The objective's priority is equal to the priority of the most required resource. Thus, when there is no required resource, the priority is zero and the objective does not execute (i.e. another behaviour gets the *navigation* resource and sets the destination). When the priority for a required resource is high enough, the behaviour starts executing and sets the agent's destination. The second part of the resource acquisition behaviour represents the algorithm described in the previous paragraph. One of its main tasks is to set in the agent's memory the location of the appropriate resource supplier (if any). The other main task is to interact with the supplier to get the resources. Of course, this part executes only when the agent has reached the appropriate supplier. This behaviour is shown below in pseudo-code.

```
GET_NEEDED_RESOURCES(a: Agent): void
const MIN: double // Minimum distance to use a supplier
var r: PrioritySetOfResources
var s: ResourceSupplier
var w: RequestAnswer // GIVEN, MUST_WAIT, NONE_LEFT
r := a.requiredResources
while( true )
   if( r.isEmpty )
      return
   s := SET_BEST_SUPPLIER(a)
   if( s = NULL )
      return
   if( a.distanceTo(s) > MIN )
      return
   w := USE_SUPPLIER(a, s, r.first)
   if( w = MUST_WAIT )
      return
```

```
SET_BEST_SUPPLIER(a: Agent): ResourceSupplier
var r: PrioritySetOfResources
var rr: RequiredResource
var s: ResourceSupplier
r := a.requiredResources
while( true )
   rr := r.first
   if(rr.priority < 0)
      a.memory.bestResourceSupplier := NULL
      return NULL
```

```
    s := a.memory.bestResourceSupplier
    if(s != NULL& s.doesSupply(rr.name) & s.hasAnyAvailable(a, rr.name))
        return s
    s := FIND_BEST_SUPPLIER(a, rr.name)
    if(s = NULL)
        r.changePriorityByScale(rr.name, -1)
        continue
    a.memory.bestResourceSupplier := s
```

```
FIND_BEST_SUPPLIER(a: Agent, r:String): ResourceSupplier
var v: Set // Stores the valid suppliers
for(s: Supplier in agent.memory.getInterestShapes("Resource Supplier"))
    if(s.doesSupply(rr.name) & s.hasAnyAvailable(a, rr.name))
        v.add(s)
return agent.memory.findBest(v) // Finds the closest one
```

```
USE_SUPPLIER(a: Agent, s: ResourceSupplier, rr: RequiredResource): RequestAnswer
var r: PrioritySetOfResources
var w: RequestAnswer // GIVEN, MUST_WAIT, NONE_LEFT
r := a.requiredResources
while( true )
    w := rs.request(a, rr.name)
    if( w = MUST_WAIT | w = NONE_LEFT )
        return w
    if( rr.quantity > 1 )
        rr.quantity := rr.quantity -1
        continue
    r.remove(rr.name)
    for(otherRR: RequiredResource in r.getActive()) // Positive priority
        if(s.doesSupply(otherRR.name) & s.hasAnyAvailable(a, otherRR.name))
            otherRR.priority := rr.priority
            rr := otherRR
            break
    return NONE_LEFT
```

4.5.2 Navigation Management

This section presents three mechanisms that are related to agent navigation. They are situated at different levels of Newell's pyramid.

4.5.2.1 Physical Displacement

The agents' physical movements are managed through an objective that requires the *navigation* resource. At every iteration (ten times per second), the agent moves towards its destination if it has one or he stays stationary otherwise. When setting a new destination, the walking speed can be selected (as a factor of the normal walking speed). This mechanism is the only one that physically makes the agent move, and thus it is at the reactive level of Newell's pyramid. Behaviours that need to control the agent's movements will only set the destination, and this mechanism will make the agent move towards that destination.

4.5.2.2 Wandering Around

When an agent does not have a destination set, an objective that makes the agent walk around gets activated. This objective makes the agent walk randomly in any direction at half his normal walking speed. Wandering around requires the *navigation* resource with the lowest possible priority; thus, it is activated only when no other objective uses this resource. In real life, people walk around while waiting for the bus, for example, without consciously thinking about their every step. Thus, the behaviour is hard-wired in their brains and it can be considered to be reactive.

4.5.2.3 Maintaining a STG's Formation

A behaviour is necessary for an agent to maintain his assigned position within a formation, since he is not "forced" to stay in formation. This behaviour is at the cognitive level since an agent must explicitly think about staying in position with respect to the other STG members. When an agent is in a STG, he asks the formation at every second what position he should take. If the position towards which the agent is going (or already stands) respects the formation, then no action is performed. Otherwise, the agent asks the formation what should be his new position, given his rank in the formation. This position becomes the agent's new destination. Some formations, such as the circle formation shown in Figure 5.4c, ignore the rank and assign positions randomly.

The controller agent of a STG is the reference point for other agents, so this agent never asks the formation for a position; this agent can choose his destination. Normally the behaviour that selects the destination is related to a group activity. Otherwise, unrealistic actions would happen. For example, if a controller agent decided to go to the bathroom, then the other members of his STG would try to stay in formation around the controller agent. This potential side effect is important to keep in mind when designing behaviours. One possible solution is to assign an interim controller when the real controller needs to perform a personal activity. It is also up to the designer of a new formation to implement the algorithms that assign positions based on ranks and to check if a position respects formation.

During the system's calibration, a problem was found with the implementation of the formation concept. Since the controller agent is free to choose his destination, he always goes directly towards it. However, the other members do not know what the controller's destination is, these members go to their target positions within the formation, which may change every second if the controller agent moves. In consequence, the other members are unable to catch up with the controller agent unless this one stops frequently to wait. To solve this problem, a factor of the normal walking speed is used so that all agents who are trying to reach their position within the formation walk slightly faster.

4.5.3 Perception Mechanisms

Perception is the implicit behaviour that scans an agent's surrounding environment to extract potentially interesting items. This behaviour is at the physical level of Newell's pyramid and it runs every second. Because of this high frequency and because of the large number of elements that can be perceived, the perception algorithm is prone to become a bottleneck in the system. To improve the simplicity and efficiency of the algorithm, the environment model has been developed so that only four types of items get perceived. Any component that a scenario designer would want to be perceivable must therefore fall in one of the four categories, illustrated in Figure 4.23. The first three types of perception allow an agent to perceive anything inside a cone in front of him. The fourth type detects any particle within the agent's bounding cylinder.

First, an agent is able to perceive other agents. Thus, an agent can always access a list of the projected images of the agents surrounding him. As explained in Sub-Section 4.2.3, agents access information about other agents through their projected images. The perception algorithm provides a clarity value that decreases as the perception distance increases. More specifically, the clarity is calculated with the formula presented below where *distance* represents the Euclidean distance between the observer and the observed and *maxDistance* represents the length of the field of view of the observer.

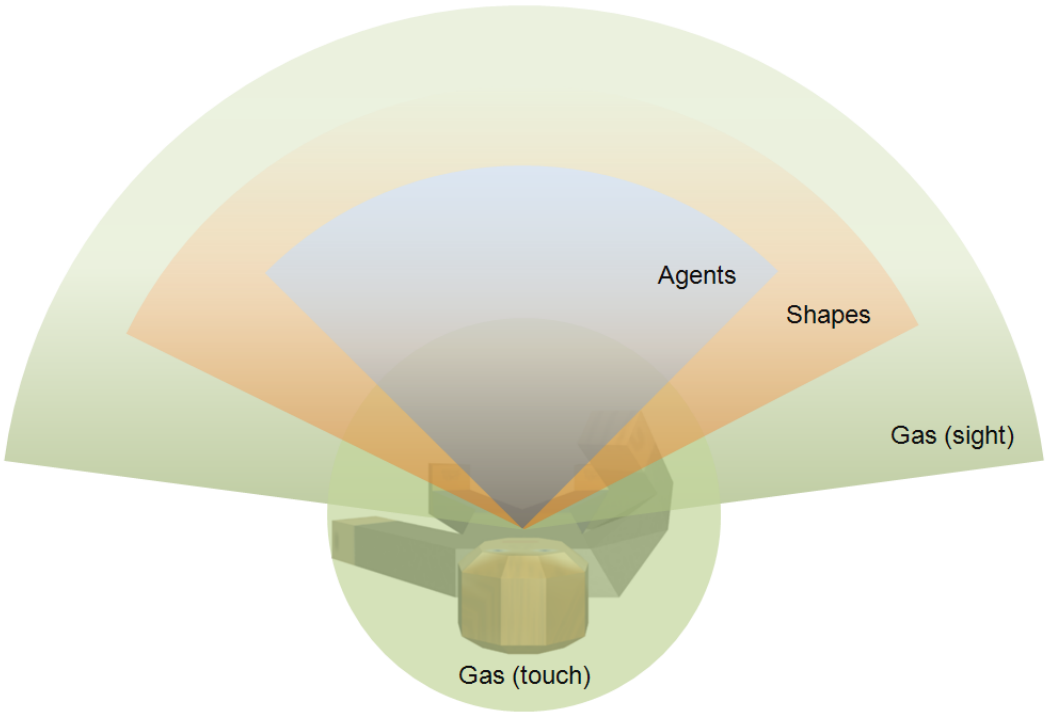
$$clarity = 1 - (distance/maxDistance)$$

Clarity is thus best when the perceived component is at the agent's location, and decreases to zero when the component is at the maximal perception distance.

Secondly, agents perceive interest shapes. Since STGs are perceivable, the zone covered by a STG's formation is implemented with an interest zone (as shown in Figure 4.17), allowing agents to perceive to STGs through the normal perception mechanism for interest shapes.

Next, agents perceive gas as whole clouds. More specifically, agents perceive individual gas particles, which link to an emitter (typically a tear gas can). The radius of the gas cloud can be accessed from the emitter, allowing from more advanced behaviours, especially when the agents try to escape from the cloud. Since gas clouds are usually ephemeral, the perception mechanism stores this information in the short-term memory.

Finally, agents perceive contact with non-lethal weapons: gas particles and plastic bullets. In the first case, the agent can get the gas concentration at his location, for more advanced behaviours (e.g. low concentrations might not bother hardened instigators). If the agent is



Agents	Shapes	Gas (sight)	Gas (touch)
Agent 1	Zone 1	Gas 1	Gas 1
Agent 2	Zone 2	Gas 2	...
...	Point 1	Gas 3	
	Zone 3	...	
	Point 2		
	...		

Figure 4.23: Perception Mechanism: Four Perception Types

caught in multiple clouds of gas, then the concentration will likely be higher. In the case of plastic bullets, the agent only knows how many bullets have hit him since the last time that he checked. Gas and bullets are stored in short-term memory, and this information is read by only the health monitor.

It is the PLAMAGS simulation environment that automatically fills the four lists for each type of perceived components. The perception mechanism can store some information directly into the short-term memory, but it is the remembering algorithm (presented next) that fills the medium-term memory by iterating through the interest shapes and perceived images lists.

4.5.4 Memory Management

This sub-section presents the two algorithms used for managing the agent's memory: remembering and forgetting. As mentioned in Sub-Section 4.2.4, long-term memory elements are usually stored at when a scenario starts or on-demand during a simulation. In addition, they are never forgotten. Short-term memory elements are stored (and overwritten) periodically, and thus they do not need an explicit mechanism for remembering and forgetting. Consequently, the algorithms apply to medium-term memory only. The memory mechanisms are at the physical level of Newell's pyramid, as this process is automatic and cannot be prevented.

4.5.4.1 Remembering

The remembering algorithm is divided into two parts. The simplest part concerns interest shapes, which are simply inserted into memory when perceived, if they were not already present. If an interest shape were already in memory, then the memory would not be altered. The second part is related to projected images. When perceiving a projected image, an agent must create a perceived image, which contains a reference to the projected image as well as a clarity value. When an image is perceived for the first time, the clarity of perception provided by the perception algorithm (detailed above) is used. Otherwise, the maximum value between the old clarity and the one from perception is multiplied by a reinforcement factor; this value becomes the new clarity. The remembering algorithm is executed every second to offer a balance between simulation realism and system performance.

4.5.4.2 Forgetting

The forgetting algorithm performs the opposite task of the remembering algorithm. Once again, it is divided into two parts. The first part merely consists in iterating through the dynamic interest shapes in order to delete the ones that have been removed from the simulation. The second part iterates over all perceived images and decreases clarity by a fixed amount. The agent's health affects this value: every "unit of damage" on the agent is added to the fixed value. For example, an agent with 80% of its full health has 0.2 "unit of damage", since health is constrained to $[0 - 1]$. Assuming a forgetting value of 0.1, the total forgetting amount would be $0.1 + 0.2 = 0.3$. Thus, the clarity of each perceived image, which is also constrained to $[0 - 1]$, is reduced by 0.3. When the clarity of a perceived image becomes 0, it is removed from memory. Forgetting is a slower process than remembering, and thus this mechanism is called every ten seconds.

4.5.5 Appreciation of Aggressiveness

This sub-section and the next two present some of the most novel algorithms proposed in this thesis. They are used by agent behaviours to assess levels of appreciation towards actions, other agents, groups, and any situation in which the agents may be. These assessments have been designed to be used by social behaviours that would make agents join groups, participate in crowd activities, and alter their own social identities. Therefore, the behaviours presented in Sections 4.5.5, 4.5.6, and 4.5.7 are all at the *social* level of Newell's pyramid. More details about the social aspects of these algorithms are presented in (Larochelle and Moulin, 2009).

To begin, an agent's appreciation towards something is defined by a function that takes as an input a level of aggressiveness (i.e. $apprec = f(aggLvl)$). Thus, the appreciation towards any concept that possesses an aggressiveness level (an agent, a STG, etc.) is calculated in the same manner. Since an aggressiveness level has been defined as a value in $[-1, +1]$, only the domain $[-1, +1]$ of the appreciation function will be useful. In this range, no restrictions are imposed except that it must be continuous and that its image must be in $[-1, +1]$.

A three-parameter triangular-shaped function is proposed for its simplicity, its rapidity of computation and its expressivity. The three parameters are the minimal, optimal, and maximal levels of aggressiveness that an agent appreciates. Figure 4.24 shows the appreciation functions for a few different agent profiles. The parameter *optimal* is the point on the curve where the appreciation is at its maximal value of $+1$ (the top of the triangle). The parameters *minimal* and *maximal* are the points where the curve hits $apprec = -1$ on the left side and on the right side respectively. An aggressiveness level of -1 is very passive while $+1$ is very aggressive. An agent does not appreciate at all when the appreciation level is at -1 and he appreciates completely when the level is at $+1$.

Figure 4.24 shows examples of triangular appreciation functions that could be created. For example, a peaceful crowd member would not like very passive actions — he would find them uninteresting. In fact, he has an *optimal* parameter that reflects actions that are neither very passive nor very aggressive. On the contrary, an aggressive crowd member would prefer actions that are highly aggressive. Another example is a control force member, who prefers the most passive actions and who does not appreciate at all actions that are highly aggressive.

The appreciation functions are called many times per iteration for many agents and thus their computation must be as efficient as possible. The efficiency of the triangular function was one of the major arguments for the selection of this function. Other functions could be used without any difficulty. For example, a normal or Poisson probability density function could possibly reflect human thinking better, but would be much too slow to compute and too difficult to specify for each profile or social identity.

4.5.6 Adhesion to STGs

As mentioned before, the social aspects of the models presented in this thesis are strongly based on groups. Agents choose to join, stay in, or leave STGs based on the members' actions. For example, a pacific bystander would be more likely to join a group of agents who are cheering than a group that is trying to tip over a vehicle. Also, the bystander might eventually leave his group if other members become highly aggressive. It is clear that an agent cannot evaluate a group based only on current actions being performed, but rather on all actions that have been performed prior to the evaluation. For this reason, (Moulin,

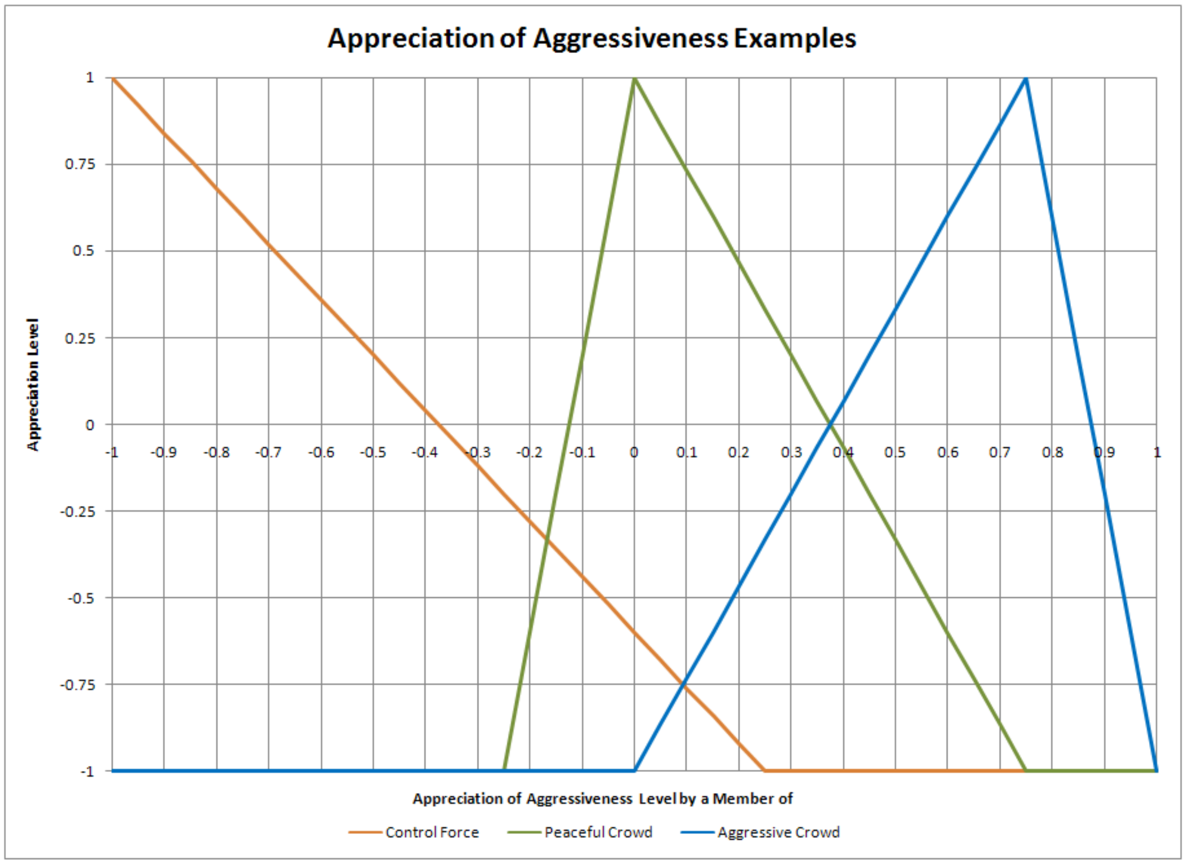


Figure 4.24: Appreciation of Aggressiveness Examples

2009) introduced the concept of "emotive" support. More recent actions will bear a heavier weight in the evaluation; actions performed in the past will still have an impact. Thus, the support that an agent feels toward a STG is a simple function that cumulates the agent's appreciation towards the actions performed by the STG's members. The degree to which past actions influence an agent's support towards an STG can be customized. Thus, some agents can quickly change their support for a STG while others will be very slow and reluctant to change.

The behaviour that calculates support towards STGs is at the *social* level of Newell's pyramid, and thus it should be executed at a low frequency. However, STG members' behaviours may change quickly so an agent must be able to change its support towards STGs quickly. A frequency of ten seconds is proposed for agents to update their support towards all known STGs. The following algorithm represents this behaviour. The parameter `a.profile.importanceOfPastEvents` is in the range $[0, 1[$: 0 means that past events will have no influence on the new support value whereas $0.\overline{9}$ means that new events will have very little influence and thus the support value will change much more slowly. The function CALCULATE_APPRECIATION takes in an agent (and thus his appreciation function) and a STG (and thus aggressiveness level) to evaluate how much this agent appreciates the actions performed by this STG's members.

```
UPDATE_SUPPORT(a: Agent): void
for all (s: STG in agent.memory)
    var apprec := CALCULATE_APPRECIATION(a, s)
    var old := a.support[s]
    var imp := a.profile.importanceOfPastEvents
    a.support[s]:= imp * old + (1 - imp) * apprec
```

Once support values have been calculated for all STGs that an agent knows, this agent must use these values to decide if he wants to join a STG, leave his current one, or create a new STG. The behaviour that manages these tasks is complex because the agent could be forced to leave his STG by external events, such as receiving a plastic bullet or having the STG controller disband the group. In such cases, the behaviour must first find why the agent is not in a STG anymore, and then take action. Because the algorithm must handle unpredictable external events, it has to be executed at a fairly high frequency. However, it is a behaviour at the *social* level of Newell's pyramid and thus a frequency of once per second is reasonable. This frequency is the same as the one that makes the agent respect his STG's formation, as mentioned in Sub-Sub-Section 4.5.2.3. It is important to keep in mind that this behaviour is executed by crowd members only and not control forces agents. For example, squad members do not change group frequently based on their desires; they must of course obey orders and stay in the group to which they are assigned. The pseudo-code for the algorithm is given at the end of this section.

The first step of the algorithm is to check if the agent is in a STG. If not, then the step is completed and the reason for not being in a STG is unknown. Otherwise, the algorithm checks if the agent should leave his STG. If the agent is the controller of the group, then he cannot leave. On the other hand, if he is a regular member of his STG and his support for this STG falls below the *adhesionSTGMin* threshold, then he leaves the group and either creates a new STG or joins another one. Similarly, if the support is higher than *adhesionSTGMax*, the agent "feels" a strong desire to control his STG. Thus, he will become the controller agent of his STG if there is none, and will leave his STG to create a new one otherwise. If the support is between both thresholds, the agent stays within his STG and the algorithm terminates.

When trying to join a STG, an agent looks through all STGs that he knows, that are of the same type as his adopted social identity, and for which the support is at least the *adhesionSTGMin* threshold. The agent joins the STG that he supports the most, if there is one. Otherwise, the algorithm terminates and the agent will search again the next time that the algorithm is called.

Two details were skipped to keep the description of the algorithm simple. As discussed in Sub-Section 4.3.3, controller agents in a STG broadcast recruitment message to gather agents. The first detail that was skipped is that agents will not leave a STG shortly after a recruitment message was sent. This restriction was introduced to give a chance to STG controllers to gather a large group. The second detail that was skipped is that each agent has a Boolean variable that tells if it is allowed to engage in social activities. When this flag is set to false, the agent cannot create or join any STG. However, it does not force him to leave his current STG. This feature is useful for example when an agent is fleeing from tear gas. He is temporarily not allowed to join or create STGs, which is a plausible situation.

```
MANAGE_STG(a: Agent): void
var reason: LeavingReason
if a.stg = NULL
   reason := UNKNOWN
else
   if a.stg.controller != a
      reason := MANAGE_LEAVING(a)
   else
      return
if a.stg = NULL & a.canJoinSTG
   MANAGE_JOINING(a, reason)
```

```
MANAGE_LEAVING(a: Agent): LeavingReason
var support := a.support[a.stg]
if support < a.profile.adhesionSTGMin & !stg.isRecruiting
   a.stg := null
   return LOW_SUPPORT
elseif support > a.profile.adhesionSTGMax & !stg.isRecruiting
   if a.stg.controller != NULL
      a.stg := null
      return HIGH_SUPPORT
   else
      a.stg.controller := a
      return NULL
else
   return NULL
```

```
MANAGE_JOINING(a: Agent, reason: LeavingReason): void
if reason = HIGH_SUPPORT | a.profile is leader
   a.stg := Util_STG.CREATE_STG(a)
else
   JOIN_BEST(a)
```

```
JOIN_BEST(a: Agent): void
var max := -8
var best: STG
for all (s: STG in agent.memory)
   if s.type = agent.socialIdentity
      if a.support[s] > max
         best := s
         max := a.support[s]
if best != null & max > a.profile.adhesionSTGMin
   a.stg := best
```

4.5.7 Social Identity Selection

The behaviour that manages social identity changes, one of the most novel elements of this thesis, is based on the social identity theory. The notion of support was introduced in the previous section, and it is used to determine how an agent “feels” towards a STG. This support may make the agent join or leave the STG, but it will not be influential enough to make the agent change his social identity. Rather, it is the assessment of the overall crowd situation that could make an agent adopt a different social identity. By observing the collective actions that take place around him, an agent may be “excited” and “feel an urge” to participate in the collective activities. Conversely, an agent may be disapproving the collective actions if they go beyond his personal set of norms for acceptable actions and thus become reluctant to participate in them. Modelling an agent’s enthusiasm is very complex when taking into account the all sources of excitement that may influence him in a crowd situation. In addition, an agent’s enthusiasm towards the overall crowd situation depends on his adopted social identity, his profile, his “mood”, and may change over time. For these reasons, a clever integration of the concepts of appreciation, social identity, profiles, and STGs is proposed to model enthusiasm at low computational costs but with high realism.

Enthusiasm is defined as the weighted sum of the appreciation towards all known STGs and is thus in the range $[-1, +1]$. The weight given to each STG is the number of members divided by the distance to the agent. Thus, larger STGs that are close to the agent have a much larger impact on the overall enthusiasm than small STGs that are far from the agent. Hence, a group of a few very aggressive protestors will not bear too much of an impact on enthusiasm, although the agent’s support towards their STG may be very low. Enthusiasm thus gives a more aggregate assessment of the crowd situation around the agent. Because enthusiasm is based on appreciation towards STGs, enthusiasm will vary in time as well as according to the agent’s social identity and profile.

This behaviour is at the *social* level of Newell’s pyramid and the social theory defends that social identity changes are infrequent, compared to other human behaviours. Thus, this behaviour will make the agent consider changing his social identity once per minute. Tests have shown that although agents consider changing every minute, actual changes are much less frequent. The pseudo-code of the behaviour is shown at the end of this section. For each SI that the agent could adopt, he calculates what his enthusiasm would be (since enthusiasm is based partly on SI). The candidate SI with the highest enthusiasm value is retained. If this value is greater than the current enthusiasm (by at least a certain threshold), then the change is performed. The reason for the threshold is to prevent an agent from oscillating between two SIs that make him more or less as enthusiastic. The threshold, called *enthusiamGap* in the algorithm, is stored in the agent’s current profile. Changing back to the fundamental SI is no different process, as the fundamental SI is always one of the candidate social identities. The function CALCULATE_APPRECIATION takes in an agent, a STG, and a profile to evaluate how much this agent would appreciate the actions performed by this STG’s members if he adopted the given profile (which is linked to a social identity). This function is the same as the one presented in Sub-Section 4.5.6, except that the version presented above used the agent’s adopted profile instead of a given one. In order to skip low level details, profiles and social identities are used interchangeably in the algorithm descriptions below.

```
EVALUATE_SI_CHANGE(a: Agent, csi: Set of SocialIdentity): void
var cur: CALCULATE_ENTHUSIASM(a, a.profile)
var highest: a.profile
var highestValue = cur
for all (si: SocialIdentity in csi)
    var e := CALCULATE_ENTHUSIASM(a, si)
    if e >= highestValue
        highest = si
if highestValue >= cur + a.profile.enthusiamGap
    a.profile := highest
```

```
CALCULATE_ENTHUSIASM(a: Agent, p: Profile): number
var enthusiasm := 0
var numSTGs := 0
for all (s: STG in a.memory)
    if s is not a CF STG
        var weight := s.numMembers / distance(a,s)
        var apprec := CALCULATE_APPRECIATION(a,s,p)
        enthusiasm:= enthusiasm + apprec * weight
        numSTGs := numSTGs + 1
if sumSTGs = 0
    return 0
else
    return enthusiasm / numSTGs
```

4.5.8 Simulating Emotions

The previous few sections have made references to human emotions in describing how agents react to aggressiveness. In fact, it was more meaningful to relate to emotions than to talk about ranges of values on an appreciation scale, for example. Indeed, the Crowd-MAGS' agent model is powerful enough to "simulate" emotions, even though they are not used explicitly. Discussing all the possible emotions that an agent may feel while observing STGs' collective actions would be too elaborate for this thesis, but a few examples are presented below. A more complete discussion is available in (Moulin, 2009).

In order to keep the agent model simple and computationally efficient, agents' "feelings" are stored as numerical values ranging from 0 to 1. Nonetheless, these simple values can relate to emotions expressed by the agents. For example, the enthusiasm value can be mapped to the following range of emotions.

- *Enthusiastic*, if $enthusiasm \geq 0.5$
- *Neutral*, if $-0.5 < enthusiasm < 0.5$
- *Bored*, if $enthusiasm \leq -0.5$

The support value towards STGs can also be mapped to emotions.

- *Proud*, if $support \geq 0.75$
- *Satisfied*, if $0.25 < enthusiasm < 0.75$
- *Neutral*, if $-0.25 < enthusiasm < 0.25$
- *Dissatisfied*, if $-0.75 < enthusiasm < -0.25$
- *Ashamed*, if $enthusiasm \leq -0.75$

In addition to the current values of these parameters, the changes in value could be used. For example, an agent could be *joyful* if his support towards his STG has increased significantly and he could be *resentful* if his support dropped significantly.

The Information model also can be useful in simulating emotions. For example, an agent could be *afraid* if his history of events mentions that his STG was the target of an aggressive action. The agent could also be *angry* at the perpetrator of this action. Another example could be a bystander who would be *supportive* of an instigator who kicked back a tear gas can towards the control forces.

The possibilities of simulating emotions with the Information model are limitless. If the proper actions are defined and performed by the agents, nearly any emotion could be represented. These emotions could eventually be used to help behaviour developers or even users to create new behaviours based on these emotions rather than on numerical values.

4.5.9 Health Monitoring

Agents' health is managed through a behaviour called the *health monitor*, which is at the physical level of Newell's pyramid. Its main task is to monitor all aspects of the environment that could influence the agent's health; it runs every second and decreases health due to tear gas, plastic bullets, fights, and Molotov cocktails. The secondary task of the health monitor is to remove the agent from the simulation when his health falls below his critical health level. The health monitor is generic and thus is used by all agents ranging from bystanders to squad leaders.

Noci-perception (the part of perception that detects pain) detects when the agent is in contact with tear gas or receives plastic bullets hits. This information gets stored in short-term memory, which is accessed only by the health monitor.

4.6 Scenario Specification Model

"For experts who work in the application domain [...], writing a scenario to control a group of agents is not easy, because most [domain experts] are not computing professionals. As the agents' functions and environments become more complex [...], the difficulty of writing scenarios causes a serious problem." (Murakami et al., 2003) Thus, this thesis proposes a scenario specification model that helps domain experts to translate scenario ideas into scenarios that are runnable on a crowd control simulator. The proposed model allows, in particular, the specification of control forces, crowds, and non-lethal weapons.

In an effort to create a human- and machine-readable scenario specification model, the XML language was selected over others, such as BDL or Java. XML can be used by anyone, even without any computer programming knowledge. Each scenario is described in a XML file, which contains no code but only scenario specifications, such as the location of the events, the agents to be created and parameters related to the usage of NLW.

Each XML file contains two main sections: parameters and components. The former section contains a variety of parameters that affect anything from the effects of NLW to the agent behaviours. The latter section contains the components that are added to the simulation. The following is a minimalist example of a scenario file. A more elaborate one is presented in Section A.2.

```
<Scenario>
  <Parameters>
  </Parameters>
  <Components>
  </Components>
</Scenario>
```

The main advantage of the proposed model is the flexibility brought forth by the XML language. Indeed, what goes into the two main markers (*Parameters* and *Components*) does not need to be defined in advance. In fact, each scenario can specify custom markers. The next two paragraphs describe the markers that are proposed for crowd control scenarios, most specifically related to the Crowd-MAGS project.

The *Parameters* section contains nine basic sub-sections but some scenarios contain also specific sub-sections. The following list explains the basic sub-sections.

- **Scenario:** Global scenario parameters, such as the date, time, and weather.
- **Cameras:** Cameras that will be setup by default. Cameras help the user to visualize the scene in the 3D environment.
- **Agent:** Global agent parameters, such as mass, walking speed, and perception distances.
- **Profiles:** Profiles (related to social identities) that will be available in the scenario.
- **Profile Distribution:** How the profiles will be distributed among the crowd.

- **Squad:** Global control force squad parameters, such as protective equipment, NLW, and formations.
- **Tear Gas Can:** Physical parameters such as mass, range, and duration.
- **Plastic Bullet:** Physical parameters such as speed, range, and inaccuracy.
- **Health:** Parameters related to the impact of NLW and protective equipment.

Other sub-sections in various scenarios specify ranges for crowd members using projectiles, delays before attacking the control forces, costs of control forces and NLW, and specifications of organized groups among the crowd.

The *Components* section is simpler because it simply lists components to be added to the simulation. Such components can be agents, interest shapes, police trucks, and more. Each component must specify at least five parameters, and some define additional specialized ones. The following list explains the basic parameters.

- **Name:** Unique identifier for this component
- **Appearance Time:** Time at which the component will appear in the scenario (not all components need to present when the scenario starts).
- **Position X:** Coordinate representing the position in the X-axis
- **Position Y:** Coordinate representing the position in the Y-axis
- **Orientation:** Orientation of the component (e.g. 90° for North-facing components)

In order to facilitate the development of several similar scenarios, it is possible to organize the XML files in a hierarchical manner. For examples, two similar scenarios where the only difference is the crowd distribution on the scene differ in only one section. It is possible to create a "parent scenario", which lists all parameters and components that are identical to both "child scenarios". Then, the child scenarios can be completed with what was not listed in the parent scenario. In addition, a child scenario can overwrite parameters already defined in a parent scenario by simply listing the new value. Scenario hierarchies can be as wide and as deep as necessary. This approach can considerably speed up scenario development and analysis since the common aspects are written once and the child scenarios are reduced in size and complexity.

4.7 Future Improvements

This section presents critics about the model and improvements that could be made to increase the plausibility, the performance, or other aspects of the models in general. Some of these aspects were planned but not implemented due to lack of time, while others are simply ideas that other researchers could pursue.

4.7.1 Improving the Environment's Semantic Representation

As mentioned in Section 4.1, a manual creation of semantic labels is proposed rather than an automatic topologic representation of the environment. This design decision does not lead to optimal results with respect to the displacement of agents and crowds, but it does provide some advantages.

The low number of semantic elements that can be created by hand ensures that nearly all semantic elements are useful to the agent behaviours and that the system performance stays quite good. Indeed, searching through a collection of elements that contains a few tens of interest shapes compared to a few hundreds of thousands of topologic elements is much faster — the time gain can become considerable when the number of simulated agent grows.

Another advantage of the proposed model is that not all semantic elements are accessible to the same agents. In a standard topologic representation, all elements are accessible to all agents; it then becomes difficult to simulate locations that are "private", such as a meeting point between two agents. With interest shapes, the scenario designer can make some elements not perceptible, or use additional properties to make the points visible or understandable by only specific agents or types of agents.

Tests have shown that for the scenarios necessary for the Crowd-MAGS projects, the proposed model is sufficient. However, if more complex scenarios were to be simulated, such as flows of people in an entire neighbourhood, a real topologic representation of the environment would be necessary for true path planning. Ideally, an informed virtual geographic environment like the one described in (Paris et al., 2009) would be used in combination with manual semantic labels.

4.7.2 Improving the Agent's Memory's Destination Finding Algorithm

Sub-Sub-Section 4.2.4.4 explained that the agent's memory is able to answer queries to find a specific environment elements from a semantic name or a semantic category. Although these queries are not complex, they are used by nearly all agents at a high frequency. Thus, improving them would bear a great impact on the overall simulation.

The first proposed improvement would be to discriminate between possible destinations based on a better heuristic than just distance. Currently, if several locations satisfy a query, then the location closest to the agent is returned. This solution poses a problem, as agents sometimes try to walk through tear gas or fences to get to a destination because it is slightly closer than another one that is easily accessible.

If the best location is in fact a zone, then the algorithm chooses a random point inside the zone. Another improvement would be would be to choose a location inside the zone based on plausible criteria instead of a random selection. These criteria could include crowd density, proximity of undesirable elements (e.g. control forces), number of obstacles to avoid, and several others.

4.7.3 Developing a Model for the Interpretation of Projected Images

As discussed in Sub-Sub-Section 4.2.5.3, agents perceive each other's projected images with a certain clarity. Depending on the skills of the perceiving agent and the clarity, the data will be obtained either very clearly, or in an approximate manner. Due to lack of time, all data is currently obtained clearly. Therefore, some abstraction model must be developed to render this data less clear.

Categories of data that can or cannot be obtained clearly must be defined. For example, clothing color could be obtained clearly at all time without decreasing the simulation's plausibility, whereas perceiving the health level should never give the exact value. Types of data must be defined such as physical properties (*small*, *medium*, *large*) and social behaviours (*passive*, *moderate*, *aggressive*). It must also be decided if different agents can conceptualize the same value in different manners. For example, agent A whose height is 1.83 m could be perceived by a child agent who understands only *small*, *medium*, and *tall* and could be perceived also by a squad leader who understands *very small*, *small*, *medium*, *tall*, and *very tall*. In this example, the squad leader is more skilled at conceptualizing the value 1.83 m. A balance between model simplicity, expressivity, and performance must be found.

4.7.4 Improving the Social Group Concept

Sub-Section 4.2.7 mentioned that a social group represents a set of agents who know each other and who are linked by a social relation. Due to lack of time, the model was developed so that each agent can belong to only one social group at a time. Unfortunately, the social group concept is of little use with this restriction. For example, if two groups are created, crowd and control forces, then agents belong to one or the other. Thus, they cannot belong to a social group representing a squad, a group of friends, or a support group for a cause. It is imperative that agents can belong to multiple groups if the concept is to be used to its full potential. However, a model that defines how the agent will handle the multiple roles (each agent has a role in a social group) will be necessary.

4.7.5 Developing More Health-Related Behaviours

Sub-Section 4.2.8 explained that the health concept was implemented in a basic manner and many improvements could still be made, especially in relation to social aspects. For example, agents should react when observing other agents being hurt. In fact, the health level is available in each agent's projected image, but no behaviour was yet developed to use this value.

On the other hand, an agent does react to its own health level. For example, an agent's moving speed decreases with his health. However, an improvement would consist of extending the behaviours that could be affected by health. For example, reducing the agent's perception field so that he gradually sees only the agents, objects, and gas clouds really close to him. Also, an agent should become less enthusiastic in protesting aggressively as he becomes severely injured. At a critical point of health decrease, the agent could drop his resources such as banners and leave dangerous zones. The agent could eventually start looking for a place to get healed. Other agents around him should react, either in providing healing, protection, moral support, or any other plausible social behaviour.

4.7.6 Developing a Communication Model

Although many types of perceptions are implemented in PLAMAGS, no mechanism still exists for auditory perception — as explained in Sub-Section 4.2.9. Thus, it is currently impossible to easily simulate voice communications, such as the CF asking the crowd to disperse. Ideally, both one-to-one and broadcast communication should be implemented. Communications would allow control force squads to cooperate, as they currently work independently. Communications would also help STG controllers in recruiting members.

4.7.7 Improving the Model for the Calculation of the Agent's Enthusiasm

Sub-Section 4.5.7 explained that agents based their enthusiasm on the appreciation of STGs around them. Unfortunately, tests have shown that this model fails to recreate plausible behaviours in specific cases. Indeed, if there are few STGs around an agent, his enthusiasm will be a biased evaluation of the overall situation because it will be based on too few STGs. This situation could lead the agent to adopt a different social identity. A naive solution to this problem is to avoid having zones where too few STGs are present. However, a better solution would be the improvement of the enthusiasm algorithm to possibly include individual agents or to use a certain threshold. More research and experimentation are necessary in this area.

4.7.8 Improving the Use of the Information Model

Section 4.4 discussed the information model and mentioned that an action definition is a simple structure that contains an action description and an aggressiveness level. This level is based on the agents performing and "receiving" the action. This architecture allows for fine-tuning action definitions during the calibration phase, but it also requires that each action be assigned an aggressiveness level for each possible combination of performing and receiving agents. For example, assuming a system composed only of bystanders, instigators, and police officers, the action yelling would need nine action definitions (with nine aggressiveness levels):

- Bystanders yelling at bystanders
- Bystanders yelling at instigators
- Bystanders yelling at officers
- Instigators yelling at bystanders
- Instigators yelling at instigators

- Instigators yelling at officers
- Officers yelling at bystanders
- Officers yelling at instigators
- Officers yelling at officers

As an example, it may seem pointless to create a definition for instigators yelling at bystanders because they would have no reason to do so. However, tests have shown that with agents changing social identities during simulations, unexpected behaviours happen. For example, instigator A yells at instigator B, who adopts the *bystander* social identity. At the same moment, instigator A performs the action "instigator yelling at bystander" because he has not had time to observe the social identity change. Thus, the system would be unable to find the appropriate action definition if it had not been specified in advance. This example shows that the task of specifying appropriate action definitions is colossal, especially when the number of social identities is high. As a reference, the social identities and action definitions used in the Crowd-MAGS project are listed in Section 5.1. The information model should provide a mechanism to handle cases where the action definition is undefined.

4.7.9 Allowing the Archival of Simulations

If the Crowd-MAGS system is to be used for training purposes, it would be convenient to have the possibility of pausing, restarting, saving, and replaying simulations. Varner's SULNT system, presented in Sub-Section 2.5.3, offers such features. In fact, all user actions get logged to support review sessions. The Crowd-MAGS information model records all actions and events performed on agents, and prints this information into log files. In addition, the information model logs macro-level information, such as overall crowd aggressiveness. However, this model would need to be extended to record users' actions, such as ordering the squads to perform certain tasks, and environmental events, such as fences falling and tear gas cans starting to emit gas. With all of necessary information recorded, it would even be possible to pause a simulation at some point, try a strategy, come back to a desired point, try a different strategy, and so on. The user interface would need to be extended to support such features. In addition, a file format would need to be modelled to store all of the recorded information.

4.8 Conclusions

In this chapter, models representing all aspects necessary for crowd control simulations were presented. First, the environment model was detailed and it was decided that a topologic representation of the environment was desirable but unnecessary for the scope of this project. Next, the complete agent model was presented, from physical aspects up to social aspects. Next, the model for STGs was presented so that agents can interact socially. Finally, a model to structure the agents' knowledge and information was proposed. Once these models were presented, behaviours to use them were detailed. In particular, the behaviours presented dealt with the relationships with STGs. Afterwards, a model for specifying scenarios was discussed, followed by critics and improvements on all models.

Although the models presented in this chapter have been developed for crowd control purposes, the models are generic. In fact, little information on how these models were used in the Crowd-MAGS system was given. The following chapter will present the Crowd-MAGS application starting by showing how the models were used and integrated into an operational system. Among others, Chapter 5 will present the five social identities that have been created and show the six formations that can be used by STGs. Thus, Chapter 4 is a theoretical presentation of the models while Chapter 5 is a practical application of the models.

Chapter 5

The Crowd-MAGS System

A great deal of models was presented in this thesis but the application of these models into an operational software is another significant contribution. This chapter thus covers the practical aspects of the Crowd-MAGS aspects and it transitions smoothly from Chapter 4 starting with Section 5.1, which revisits most models to present how they are used in the Crowd-MAGS application. Section 5.2 explains mechanisms that are necessary (in addition to the models) for the simulation to function properly. Next, Section 5.3 lists all input parameters that are available to customize the system and scenarios. Afterwards, Section 5.4 explains the structure of the system by discussing the source code and programming techniques used. Next, Section 5.5 shows the main user interface windows that are used in the application. Finally, Section 5.6 presents a summary of this chapter.

5.1 Usage of the Models

This section is a follow-up to Chapter 4: it explains how the models presented in the previous chapter have been customized and used in the Crowd-MAGS application. First, Sub-Section 5.1.1 shows the 3D environment and its contents. Next, Sub-Section 5.1.2 lists the resources that have are available to the agents. Sub-Section 5.1.3 presents the social groups for the crowd and for the control forces. Then, Sub-Section 5.1.4 presents the roles and social identities related to the social groups. Sub-Section 5.1.5 discusses the physical appearance of the agents. Afterwards, Sub-Section 5.1.6 gives more details about STGs and the formations that have been implemented. Sub-Section 5.1.7 lists the actions and events logged by the information model. Next, Sub-Section 5.1.8 gives more details about agent behaviours. Then, Sub-Section 5.1.9 discusses the XML scenario files. Finally, Sub-Section 5.1.10 presents the non-lethal weapons that have been implemented in the system.

5.1.1 Environment

This sub-section presents the VGE that was used in the Crowd-MAGS. The location selected was the neighbourhood of the Québec parliament in Québec City. Although several sites could have been selected, the open space in front of the parliament was chosen because it is easier for agents to navigate in such a space without advanced path planning algorithms, as explained in Sub-Section 4.2.2. Figure 5.1 shows a global view of the area covered by the available GIS files (with the Québec City parliament in the middle).

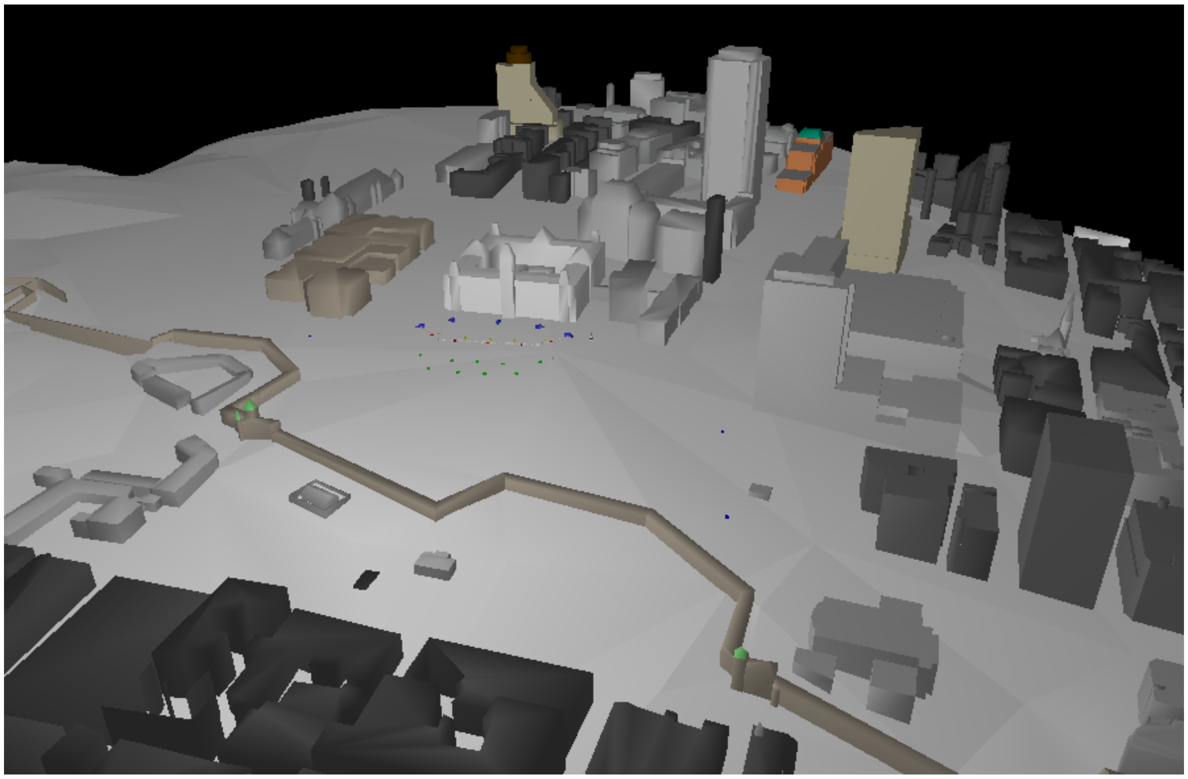


Figure 5.1: VGE: Québec City

As mentioned in Section 4.1, PLAMAGS does not offer features for creating topologic information. Thus, the agents would not be able to extract any information about their surroundings, were they placed in the initial VGE. In consequence, they would not find where to go protest, how to leave the simulation, where the control forces are located, etc. Thus, a few interest points and components were manually added to the VGE and stored in the parent scenario so that all child scenarios can benefit from this basic set-up. Figure 5.2 shows the area in front of the parliament with the key components. Figure 5.3 shows a close-up of the area around the fences.

The markers shown in Figure 5.2 are normally invisible to the user (although always perceptible by the agents) but they are shown here for the sake of explanation. All of these markers are interest points and thus they contain semantic labels. Seven types are shown here, but a few more are used in specific scenarios.

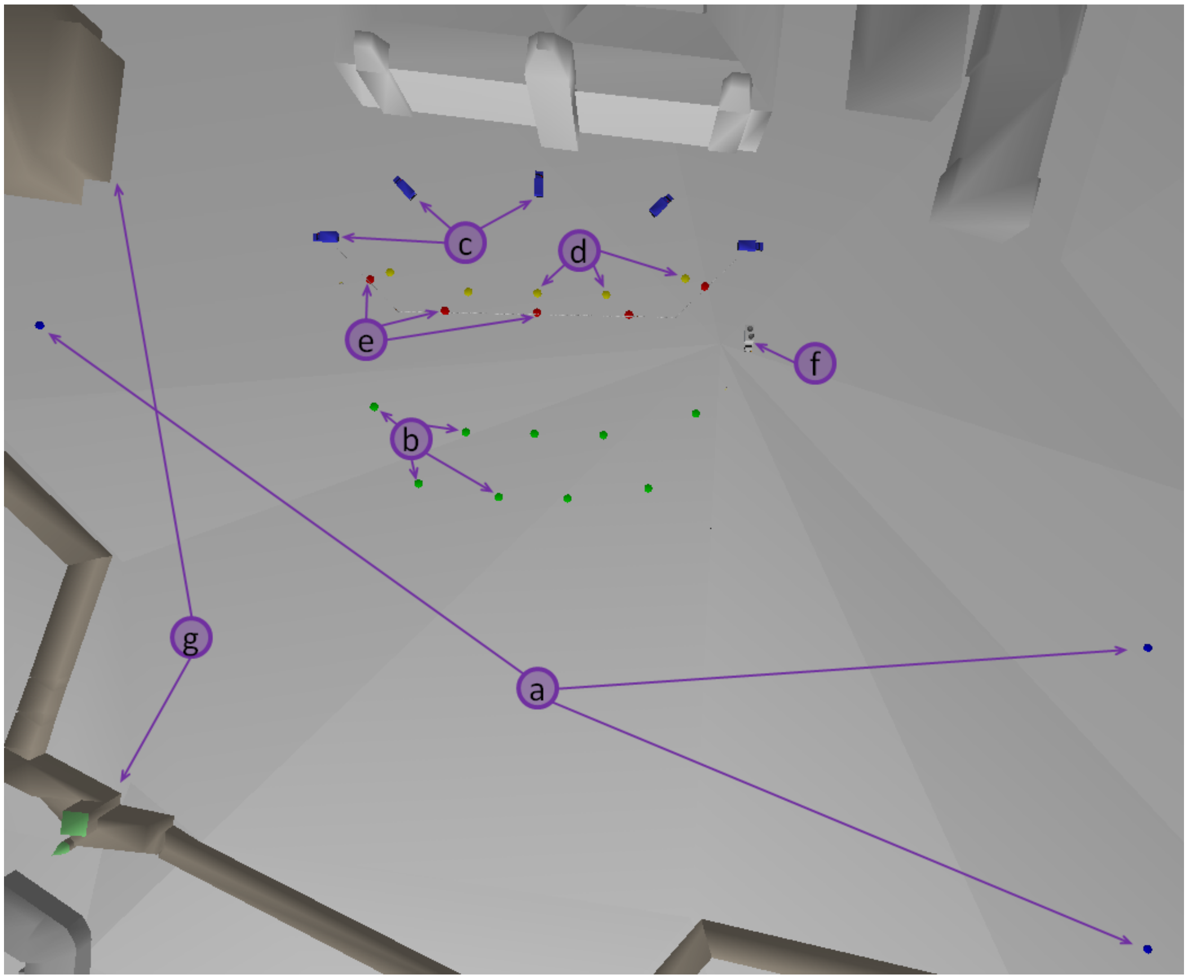


Figure 5.2: VGE: The Area in Front of the Parliament and the Key Components

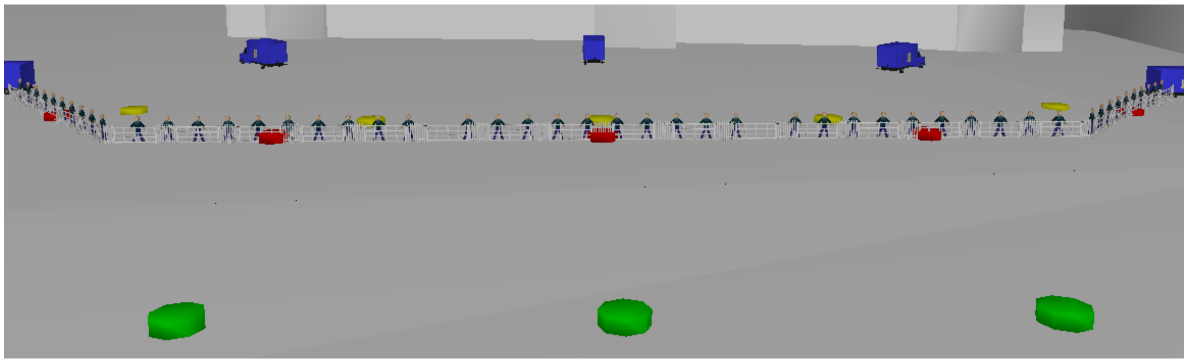


Figure 5.3: VGE: The Fences and Control Forces

a) **Exit interest points.** These interest points (represented by blue markers) are not embodied by anything in the simulation but are rather locations towards which agents go when they want to leave. They are located out of the way so that the users do not normally see agents disappearing in the middle of the action.

b) **Demonstration interest points.** The second set of interest points (represented by green markers) defines spots that are ideal for demonstrators to create STGs. Thus, these interest points macroscopically define the demonstration area, although STGs can technically be created anywhere and move afterwards.

c) **Squad trucks interest points.** The squad trucks are embodied interest points, and play the role of resource suppliers for the control force members. Indeed, these interest points represent positions for control forces to regroup and they can provide the squad members with various equipment such as body armour, shields, and helmets. These trucks can also represent targets for instigators.

d) **Deployment interest points.** These interest points (represented by yellow markers) represent the positions that the squads will take when they will try to firmly show their presence to the crowd. These points are positioned quite far from the fences to avoid provoking protestors (by moving too close to the fence).

e) **Crowd dispersion interest points.** This fifth set of interest points (represented by red markers) indicates the positions that the squads will take when they try to disperse the crowd. The instigators will try to reach these interest points when they protest aggressively.

f) **Media truck interest point.** The media truck is simply another target for the instigators. It could serve other purposes if more social behaviours were developed.

g) **Touristic site interest points.** These interest points can represent restaurants, monuments, historic sites, etc. Thus, some protestors might want to attack these targets while bystanders might just want to observe them.

In addition to all of these interest points, many other spatial references are specified in the parent scenario. For example, Location objects (introduced in Section 4.1) were created for key street intersections, key observation spots for control forces, and other such purposes. These locations can then be used to create itineraries for various agents, such as a bystander who wants to visit a few attractions (while staying on the main streets) and a control force commander who wants to survey the area.

Itineraries

After developing a few basic scenarios, it became evident that agents could not merely have one destination at a time but that they rather needed the ability to follow itineraries. At a glance, itineraries are defined as a list of locations that agents visit in a fixed order. More precisely, the locations can be any component that is spatialized (Section 4.1 gives the exact definition). Thus, steps of an itinerary could be large open spaces, stop signs, and even agents. Since agents are spatialized, they can be included in itineraries even if they move. An agent following such an itinerary would simply try to reach the other agent before moving on to the next step. Such a generic design for the itinerary concept allows it to be used by complex tasks, such as a CF commander who would want to go and talk to all of his squads.

An itinerary could be created where all destinations would be the squad leaders. No matter where the squad leaders would go, the commander would reach them one by one.

A few options are available in the itinerary concept. First, pauses can be included between any two destinations. These pauses make the agent stay at the destination once arrived. Next, itineraries can be defined as one-time events or as continuous "tracks", in which case agents would always go back to the first destination after reaching the last one. This feature could be useful for CF officers who would want to patrol an area. Another option, which can be customized for every agent if desired, is to set the proximity limit. This value tells the agent how close he needs to get to his destination before considering that he has arrived. For example, a very precise itinerary might require that the agent gets as close as 10 cm to each destination while another itinerary may need the agent to get only within 5 m of each location. Agents can be assigned as many itineraries as desired, but only one can be selected as the current one at any point in time. Instead of querying the agent's memory, a behaviour may simply make an agent follow his current itinerary. It is also possible at any time for an agent to restart his itinerary.

Marches

In order to ease the development of some crowd behaviours, an extension of the itinerary concept was created. A march is defined as an itinerary that starts based on a pre-defined condition and that ends with a public speech. Two possibilities are available for starting conditions: waiting for a certain period of time or gathering a certain number of people. The first condition is used in cases where a march is an organized rally that waits for example for 30 minutes for people to arrive; the second condition is used more often for impromptu marches where the leader wants to gather at least 50 people, for example. Once the starting condition is fulfilled, the march works like a regular itinerary. When arrived at the final destination, the leader can give a verbal address of a pre-defined length. Agents who follow the march are normally in the same STG and the STG's formation reflects the number of members and the purpose of the march. Because the march concept is more complex than the itinerary concept, a dedicated behaviour was created to ensure that agents wait for the starting condition, stay in formation during the march, and participate in the final speech.

5.1.2 Resources

The following is an exhaustive list of all resources that have been specified in the Crowd-MAGS application.

- Navigation
- Arms
- Hands
- Mouth

- Banner
- Molotov cocktail
- Megaphone
- Armour
- Gas mask
- Shield
- Helmet
- Baton
- Tear gas can
- Rifle

5.1.3 Social Groups

The concept of social groups has been implemented but it is not used to the extent that was planned. More specifically, only two types of social groups exist: the crowd and the squads. Ideally, smaller social groups would be created to represent families and organized groups of instigators, for example. However, this feature would not be highly useful since agents cannot yet belong to multiple groups, as explained in Sub-Section 4.7.4. Currently, one instance of the 'Crowd' social group is created at the start of every scenario, and every crowd member belongs to this group. No roles have been defined for this group.

On the CF side, three instances of the 'Squad' social group exist, because three squads are specified (left, front, and right of the parliament). Three roles exist, squad leader, squad deputy leader, and squad member with specializations among the squad members. Details are listed below.

- **Squad Leader:** Controls the movement of the squad, the formation, uses the megaphone, and dispatches orders
- **Squad Deputy Leader:** Acts as a squad member until the squad leader leaves, at which point the deputy acts as the squad leader
- **Squad Member:** Stays in formation
- **Squad Chemical Officer:** Squad member who has an extra behaviour to make him use tear gas cans
- **Squad Rifle Officer:** Squad member who has an extra behaviour to make him use a plastic bullet rifle

Squads in the developed scenarios always contain a leader and a deputy leader and usually contain 3, 5, or 7 other members. Squads always adopt one of four mobilization levels, which define the level of involvement, required equipment, aggressiveness of the NLW used, formation, and location with respect to the crowd. Details are listed in Table 5.1. The levels are based on the force continuum that was introduced in Sub-Section 2.5.1 and on the ones provided by the Québec City Police and by DRDC.

5.1.4 Social Identities

In Crowd-MAGS, three key elements define how agents behave: the SI (defined by a behavioural graph), the SI profile, and the SI controller, which defines the set of candidate SI for a SI change. A list of all SIs, SI profiles, and SI controllers that have been specified is presented at the end of this sub-section.

The Social Identity Controller

The PLAMAGS engine allows agents to hold several behavioural graphs at once, as well as to add and remove any of them at any time. This feature allowed the implementation of an easy solution for the adoption of social identities by the agents: one behaviour (the SI controller) handles the task of adding and removing other behaviours (adopted SIs). Thus, each fundamental SI is associated with a designated controller. A SI controller is always implemented as an objective that uses no resource and that is implemented in Java rather than in PLAMAGS. Thus, the PLAMAGS behaviour engine has no knowledge of the existence of the SI controller. This situation is ideal because the SI controller is set on the agent upon creation and it is never removed (i.e. PLAMAGS does not need to know about the SI controller).

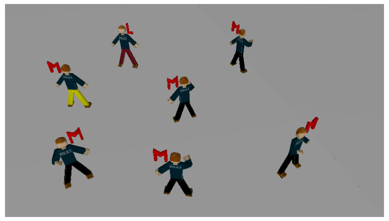
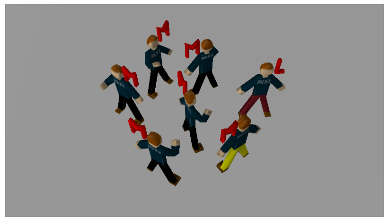
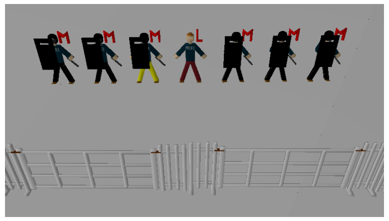
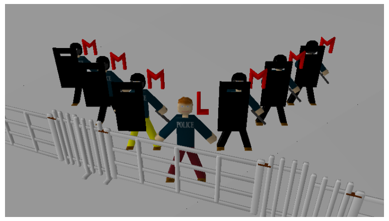
When an agent is added to the simulation, the SI controller finds the appropriate SI and makes the agent adopt it. This SI is then considered as the fundamental SI and the adopted SI. Afterwards, the controller may start changing the adopted SI. Each fundamental SI requires its dedicated controller because of the great differences in all of the agents involved in a simulation. For example, CF members rarely change their social identities and they do so based on strict orders. On the other hand, crowd members change social identities based on the social algorithm presented in Sub-Section 4.5.7.

Social Identity Profiles

A SI profile is a simple data structure that contains information (mostly parameters) that influences the agent's behaviours and mechanisms. Each profile is immutable and is linked to a specific social identity. For example, a profile called 'very aggressive teen' specified for the instigator SI could not be used with any other SI, such as the bystander SI.

The following elements are specified in each profile. They have been mentioned in this

Table 5.1: Squad Mobilization Levels

Mobilization Level	**Full Protection Required**	**Typical NLW**	**Location**	**Formation**	**Illustration**
Uninvolved	No	None	Hidden / Far from crowd	Loose Group	
Physical Presence	No	Fences	Far from fences	Tight Group	
Force Demonstration	Yes	Megaphone	Around fences	Line	
Crowd Dispersion	Yes	Tear gas and plastic bullets	At fences	Wedge	

thesis but they are explained in detail in (Larochelle et al., 2009).

- profile name
- associated social identity
- enthusiasm gap
- walking speed
- acceleration to stay in formation
- importance of past events
- minimal appreciation
- optimal appreciation
- maximal appreciation
- preferred STG type
- minimal support for STG adhesion
- maximal support for STG adhesion

List of Specified Social Identities

For all of the scenarios to be simulated for the Crowd-MAGS project, eight social identities were specified. Among the CF, squad leaders, squad deputy leaders, and squad members are the available SIs. For crowd members, bystanders, demonstrators, demonstrator leaders, instigators, and instigator leaders have been specified. Due to lack of time, only neutral profiles have been created for CF and crowd leaders. However, passive, moderate, and aggressive profiles have been specified for the other three SIs. When changing SI, an agent stays with a similar profile whenever possible (e.g. a moderate bystander would become a moderate demonstrator).

Table 5.2 presents an exhaustive list of all SIs, SI profiles, and SI controllers that have been specified in the Crowd-MAGS application. Note that the squad deputy leader adopts the squad leader SI or the squad member SI immediately after being created.

5.1.5 Agents' Physical Appearance

Because it quickly becomes difficult to analyze crowd events as the number of agents increases, visual metaphors have been introduced for the user's convenience. Rather than using visually appealing 3D models of human beings, the models were kept simple and a strict color code was adopted: the color of an agent's clothes represents his fundamental social identity. In addition, capital letters above his shoulder show his adopted social identity. Next, a resource

Table 5.2: List of Specified SIs, SI profiles, and SI controllers

FSI	**Profiles**	**SI Changes**
Bystander	Passive, Moderate, Aggressive	Bystander <-> Demonstrator <-> Instigator
Demonstrator	Passive, Moderate, Aggressive	Demonstrator <-> Instigator
Instigator	Passive, Moderate, Aggressive	Instigator <-> Instigator Leader
Demonstrator Leader	Neutral	Demonstrator Leader <-> Instigator
Instigator Leader	Neutral	
Squad Leader	Neutral	
Squad Deputy Leader	Neutral	Squad Deputy Leader -> Squad Member <-> Squad Leader
Squad Member	Neutral	

is shown above the agent's head when he is going towards a resource supplier to get the resource. Also, some actions are represented graphically above the agent's head, like chanting and yelling. Finally, some resources (such as shields or banners) are shown in 3D on the agent when he possesses such equipment. Table A.2 shows an exhaustive list of all 3D models and icons used in the Crowd-MAGS application.

5.1.6 STG

The notion of STG was presented in quite a generic manner in the previous chapter but a few more details are shown here, as well as visual examples.

Formations

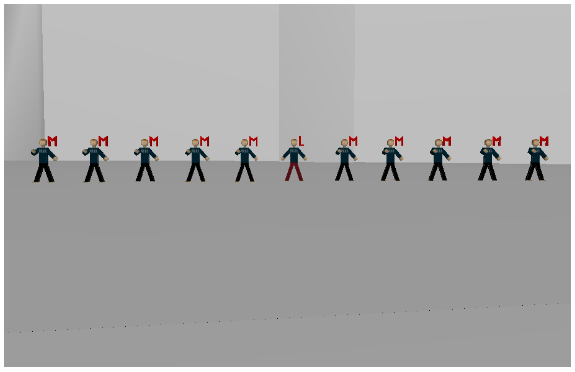
As mentioned in Section 4.3, the concept of a STG's formation is generic and thus not related to any specific type of group. Figure 5.4 shows six examples of formations that have been developed for the Crowd-MAGS scenarios. Note that the letters on the agents' shoulders are related to their social group (not their STG) and thus they can be ignored. The choice of the formations was partly based on Sub-Section 2.5.1 of the literature review. Figure 5.4a and Figure 5.4b show agents standing in line. In the first case, they stand side by side while they stand one behind the other in the second case. Figure 5.4c shows a loose formation where agents only have to stay within few meters of the controller. Figure 5.4d is the wedge formation where members are arranged in a 'V' shape with the controller at the front. Figure 5.4e shows a formation where all agents stand in front of the controller (e.g. to watch him do something). Finally, Figure 5.4f shows a hypothetical formation that is shaped like the letter B (rotated 90° to the right).

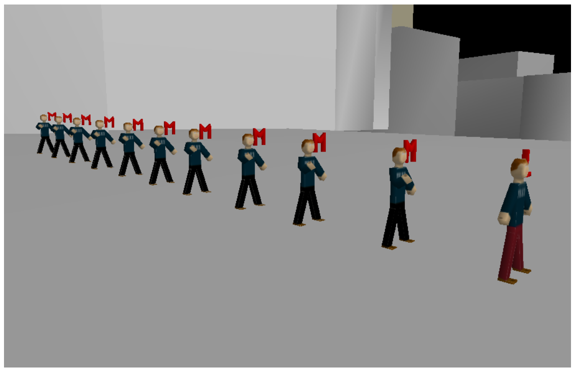
In the system, the formation concept is implemented using an interest zone (introduced in Section 4.1). Thus, it benefits from all the features implemented for any interest zone. In particular, the formation can be displayed with any color and it can answer geometrical queries such as indicating if it intersects with another formation. This is particularly useful to find out if an aggressive STG intersects with a squad in formation. Another useful feature is asking for a random point within the zone. This function is commonly used by formations that do not assign strict positions. Another useful query allows the agent to check if his current position respects the formation, before asking for a new one.

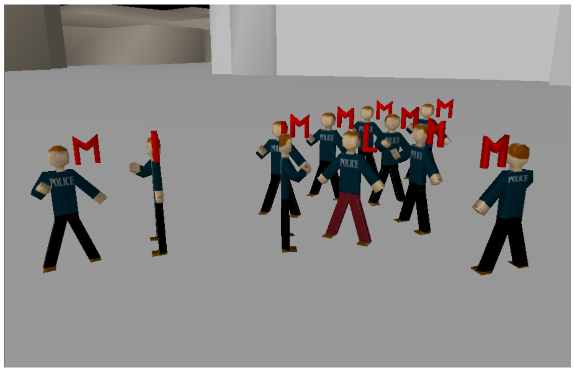
Aggressiveness Level

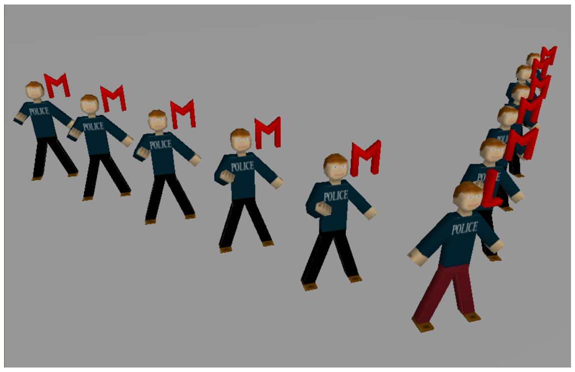
As mentioned in Sub-Section 4.3.3, a STG's aggressiveness level represents the average aggressiveness expressed by the STG's members' actions. For the needs of the Crowd-MAGS project, the calculation is performed every ten seconds so that it does not slow down performances. The value always takes into account actions performed during the past ten seconds.

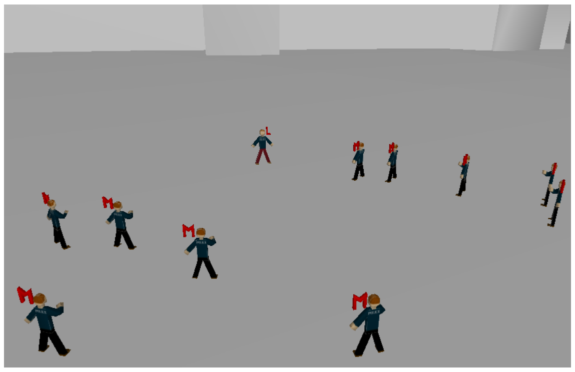
Tests have shown that one problem arose with the STG's aggressiveness levels. When


a) Sideways Line Formation


b) Forward Line Formation


c) Circle Formation


d) Wedge Formation


e) Observation Formation

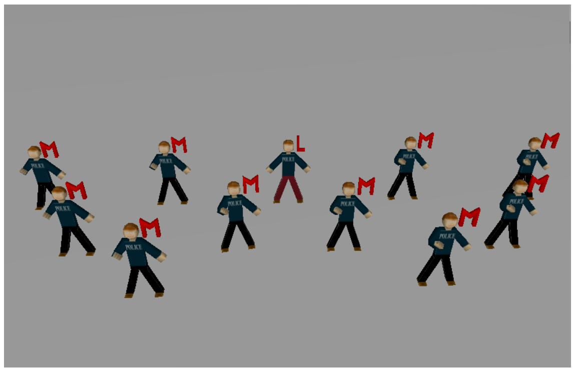

f) Custom “B” Formation

Figure 5.4: Examples of Six Formations

an agent creates a STG and tries to recruit members, he usually has not started performing aggressive actions yet. Thus, the STG’s aggressiveness level is low and other agents might not be interested in joining such a STG. Sub-Section 4.3.3 mentioned that a STG’s controller sets a desired aggressiveness level, and this concept is useful to solve the problem at hand. When agents want to join a STG, they assess them according to a weighted aggressiveness level, which is composed of the actual level and the desired level.

When the controller agent makes a recruitment effort, usually by asking agents around him to join his STG, the weighted aggressiveness level is reset to the value of the desired aggressiveness level. Then, the weighted level gradually goes back to the actual level during the following minute, as defined in the mathematical formula below. Hence, controller agents get a chance to recruit other agents but must start performing actions within a minute, at the risk of losing interest from current and potential members. The value of one minute, represented as 60 seconds in the formula, could be adjusted if required.

w: weighted aggressiveness level
d: desired aggressiveness level (set by controller agent)
a: actual aggressiveness level (average of all members)
s: seconds since last recruitment effort (limited at 60)
$w = ((60 - s) * d + s * a)/60$

5.1.7 Information Model

This section presents exhaustive lists of all actions and events that have been specified for the Crowd-MAGS application.

Agent's Actions and Events

The following is a list of all actions that can get recorded in a STG's history of actions.

- Wandering Around
- Chanting
- Showing banner
- Insulting
- Fighting
- Yelling
- Throwing a Molotov cocktail
- Ordering crowd to disperse
- Leading March
- Created STG
- Joined STG
- Left STG
- Observing

The following is a list of all events that can get recorded in an agent's history of events.

- Created
- Added to simulation
- Removed from simulation
- Dragged or turned
- Evaluated a SI change
- Changed SI
- Left STG because support for own STG was too low
- Left STG because support for own STG was too high
- Became controller of own STG
- Received Molotov cocktail
- Health reached critical level

STG's Actions and Events

Here is a list of all actions that can get recorded in a STG's history of actions.

- Ordered the crowd to disperse (performed only by squads)

Here is a list of all events that can get recorded in a STG's history of events.

- Created
- Added to simulation
- Removed from simulation
- New controller set
- New formation set
- Member joined
- Member rank changed
- Controller removed
- Member removed
- Members cleared
- Aggressiveness level calculated

- Recruitment effort made
- Received Molotov cocktail
- Set requested mobilization level
- Set mobilization level
- Changed tear gas can rules
- Changed plastic bullet rules

5.1.8 Generic Behaviours

Although each SI profile is theoretically independent from others, Sub-Section 5.1.4 mentioned that all profiles associated with a given SI had some similarities. More specifically, it was decided that all agents with the same social identity would have the same "minimal" and "maximal" parameters, but may have different "optimal" parameter values depending on their profile. This choice ensured a clear distinction between different social identities both in the theoretical sense and during the process of calibrating the profiles. Figure 5.5 shows an example of a SI and three associated profiles. This approach is used for the bystander, demonstrator, and instigator SIs. The demonstrator leader and instigator leader profiles use the same values as the moderate profile.

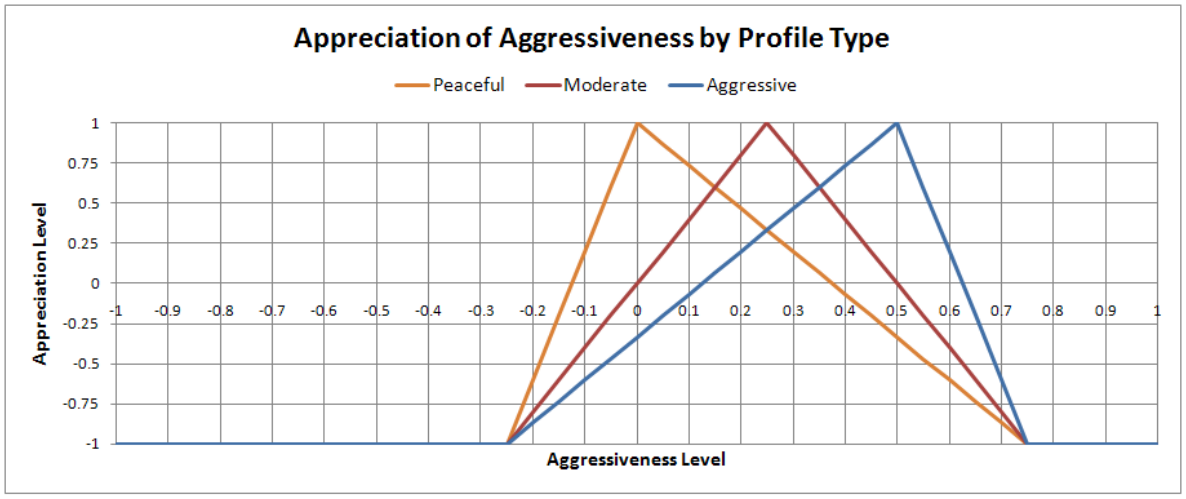


Figure 5.5: Appreciation of Aggressiveness by Profile Type

5.1.9 Scenario Specification

The scenario specification model presented in Section 4.6 is used to specify scenarios in the Crowd-MAGS system, but it is completed by other methods. For instance, the PLAMAGS library provides a way of specifying the VGE (3D map, coordinate system, physics system, etc.) in the BDL language. This feature is used because it is sufficient for the project's needs and redeveloping similar features for XML would take too long. Since all scenarios for the Crowd-MAGS project happen in Québec City, the same BDL file is used for all scenarios.

Specifying all scenarios in XML is fast and simple, but it does not allow for complex sophistications of the scenarios. For example, a scenario could contain an event where three specific agents gather at a specific time at a specific location and throw Molotov cocktails to the control forces. Such an event cannot be specified in the XML file and thus each scenario is also defined in a Java file. Like the XML files, the Java files are organized in a hierarchical manner. Thus, a considerable portion of the more complex code can be placed in parent scenarios, keeping most Java scenario files simple. Despite requiring a certain knowledge of Java, this solution has shown during the project to be a good compromise.

Unfortunately, the mechanism that supports hierarchy in the XML scenarios has not been implemented due to lack of time. Thus, all parameters and components must be specified in each XML file, rather than just once in the parent scenario. The lack of this mechanisms impacts in no way the functionality of the system, but it merely makes writing scenarios more tedious.

Because of these two adaptations of the theoretical model, each scenario is described in three files rather than just one, as shown in Figure 5.6.

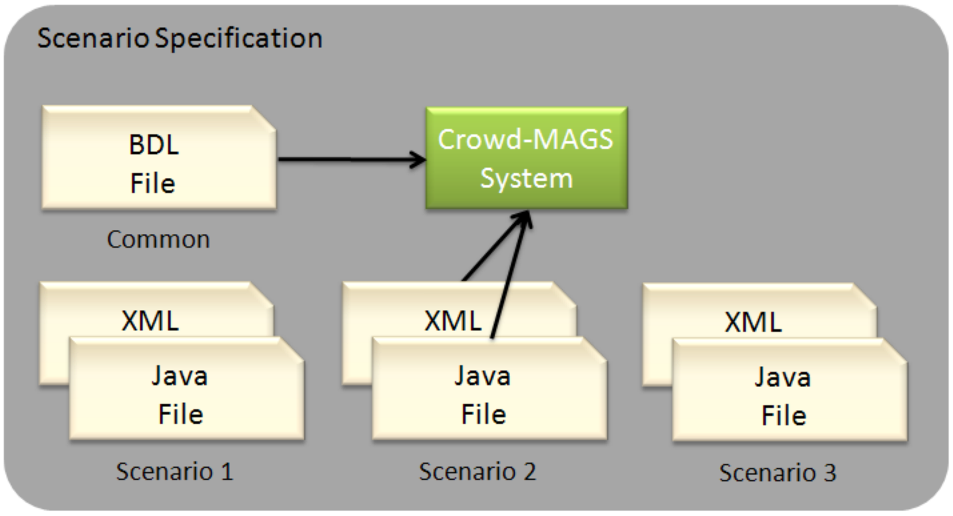
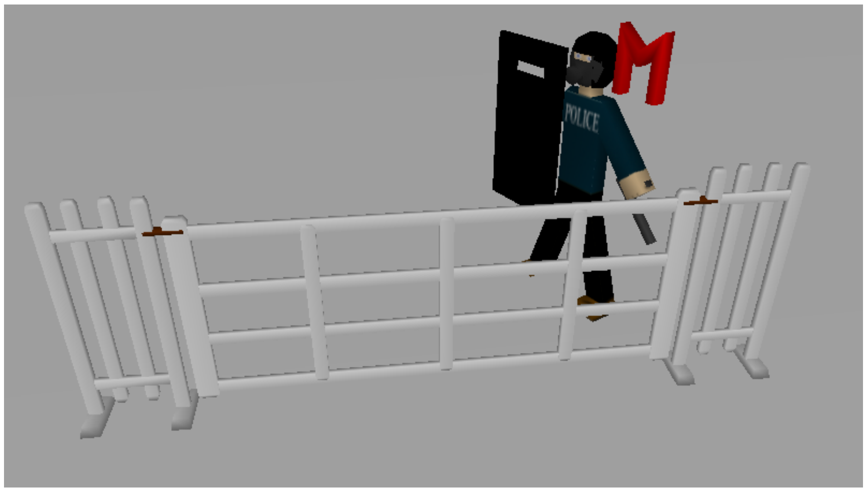


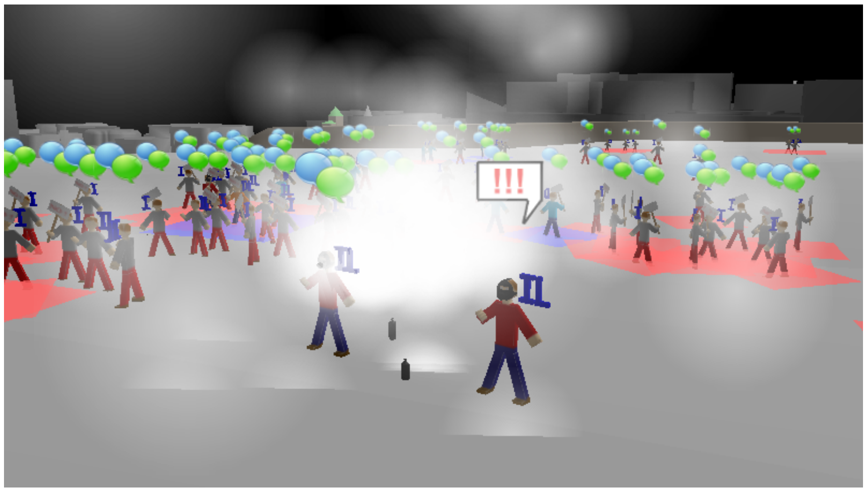
Figure 5.6: Scenario Specification in the Crowd-MAGS system

5.1.10 Non-Lethal Weapons

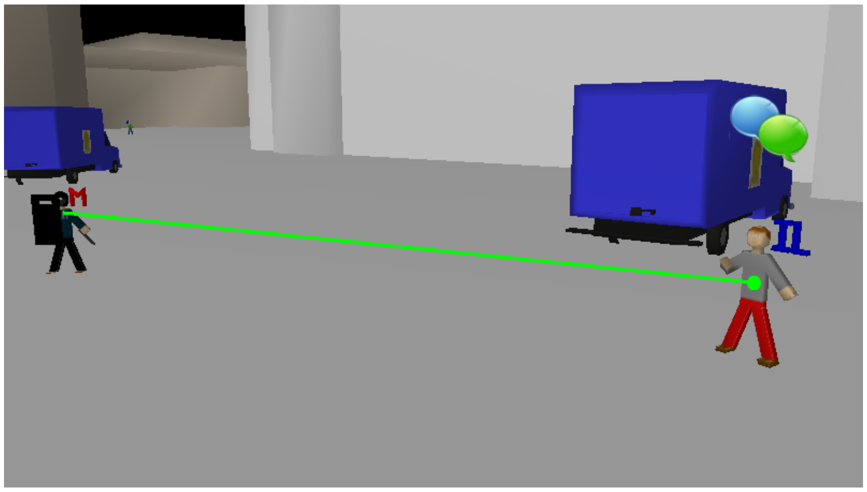
Although no new model was created specifically for NLW, they have been specified using a variety of existing generic models and classes presented in Chapter 4. The NLW are explained individually and they are illustrated in Figure 5.7.



a) A Fence



b) Tear Gas Cans



c) A Plastic Bullet

Figure 5.7: Illustrations of non-lethal weapons

5.1.10.1 Megaphone

In Crowd-MAGS, the megaphone is the only way for control forces to communicate with the crowd. It is not really a non-lethal weapon but it is part of the control forces' arsenal. So far, only squad leaders use megaphones, but more advanced behaviours could be developed. The squad leaders use the megaphone when they are at the *Force Demonstration* mobilization level or when the user requests it through the application's user interface. In the specified scenarios, megaphones are stored in the police trucks near the parliament, so leaders go get a megaphone if they do not already have one. Unfortunately, the use of the megaphone is limited because no communication model has been developed, as indicated in Sub-Section 4.2.9.

5.1.10.2 Fence

Fences are the simplest type of non-lethal weapons in Crowd-MAGS. They are simply obstacles in the VGE, but they do not contain interest points. Thus, they are not perceived by agents, who act as if there were no fences: agents do not avoid them and try to walk right through them. Ideally, more sophisticated behaviours would be developed. Nonetheless, the physics system is very useful for fences. In fact, agents get blocked by fences until enough of them are pushing in the same direction at which point the fence's anchor breaks and it can be pushed fairly easily by any agent.

Figure 5.7a shows a fence and a squad member for size comparison.

5.1.10.3 Tear Gas

Tear gas in the PLAMAGS environment (and thus in Crowd-MAGS) is implemented with tear gas cans that emit particles. The cans are very similar to reality, and can be fully customized. It is possible to change the amount of time for which a can will emit gas, the emission speed and angle, the size and acceleration of the particles, and more. Tear gas cans can be provided by resource suppliers (police trucks) and agents can carry them (as resources). Gas cans are hand-thrown (not launched from a rifle) by the chemical officer of a squad. Cans can be thrown at a reasonable distance (which is customizable). Cans bounce off obstacles and can be pushed or kicked by agents. Thus, instigators can quickly kick a can back before too much gas spreads around them.

Figure 5.7b shows that two gas cans have been launched to disperse the crowd. All agents run away from the gas except for two instigator leaders who are wearing gas masks.

5.1.10.4 Plastic Bullet

Similarly to tear gas cans, the rifle officer can obtain plastic bullets and a rifle from police trucks. Bullets are managed by the physics engine: they are simply shot from an origin point and move in a straight line until they hit an agent or an environment object, such as a fence, a building, or the ground. Bullets never bounce and disappear as soon as they touch anything.

Figure 5.7c shows that an instigator leader got close to a police truck. Since he is isolated, he is a safe target for a plastic bullet.

5.2 Mechanisms

In addition to using the models, the Crowd-MAGS application needs other components — unrelated to crowd control — to make all models work together. Thus, this section presents mechanisms that are necessary for the simulation to run. Figure 5.8 shows the components that support all mechanisms as well as the relations between these components.

At the lowest level of all components sits the PLAMAGS simulation engine, which can be viewed as the "master" component. The engine creates the PLAMAGS environment to handle physical interactions of agents and it manages on its own the behavioural aspects of the agents. In fact, it is the simulation engine that starts counting iterations at 0 and increases the iteration number after all components in the iteration have been executed. The Crowd-MAGS' architecture simplified this process by ensuring that only two types of components would be executed by the simulation engine at every iteration. First of all, the behavioural graphs, introduced in Sub-Section 3.2.2, are executed for each agent. A complete explanation on how this execution is performed is given in (Garneau and Moulin, 2008). Also, the simulation engine notifies the Scheduler, which executes the components that need to be executed at this particular iteration (since not all components are executed at every iteration).

The following sub-sections describe in detail the mechanisms shown in Figure 5.8. First, the Scheduler's inner workings are presented in Sub-Section 5.2.1. Next, the resource supplier, which agents use to gain the resources needed by their behaviours, is presented in Sub-Section 5.2.2. Afterwards, the agent creator is discussed in Sub-Section 5.2.3 because it is an important mechanism that allows the creation of a varied crowd of agents from simple specifications. Finally, the Data Collector, which collects data from agents and their histories of actions, events, STGs, and more, is detailed in Sub-Section 5.2.4.

5.2.1 Scheduler

The Scheduler is one of the most generic and important mechanisms in the Crowd-MAGS system. As mentioned before, not all components need to execute code at every iteration. For example, the navigation behaviour needs to be called ten times per second but the behaviour that regulates social identity changes is called only once per minute. Thus, a mechanism is necessary to manage these different requirements. Consequently, the Scheduler was created. The simulation engine notifies the Scheduler at every iteration, which in turn notifies only the components that need it.

The Scheduler was designed to be highly generic. Therefore, the only constraint on a

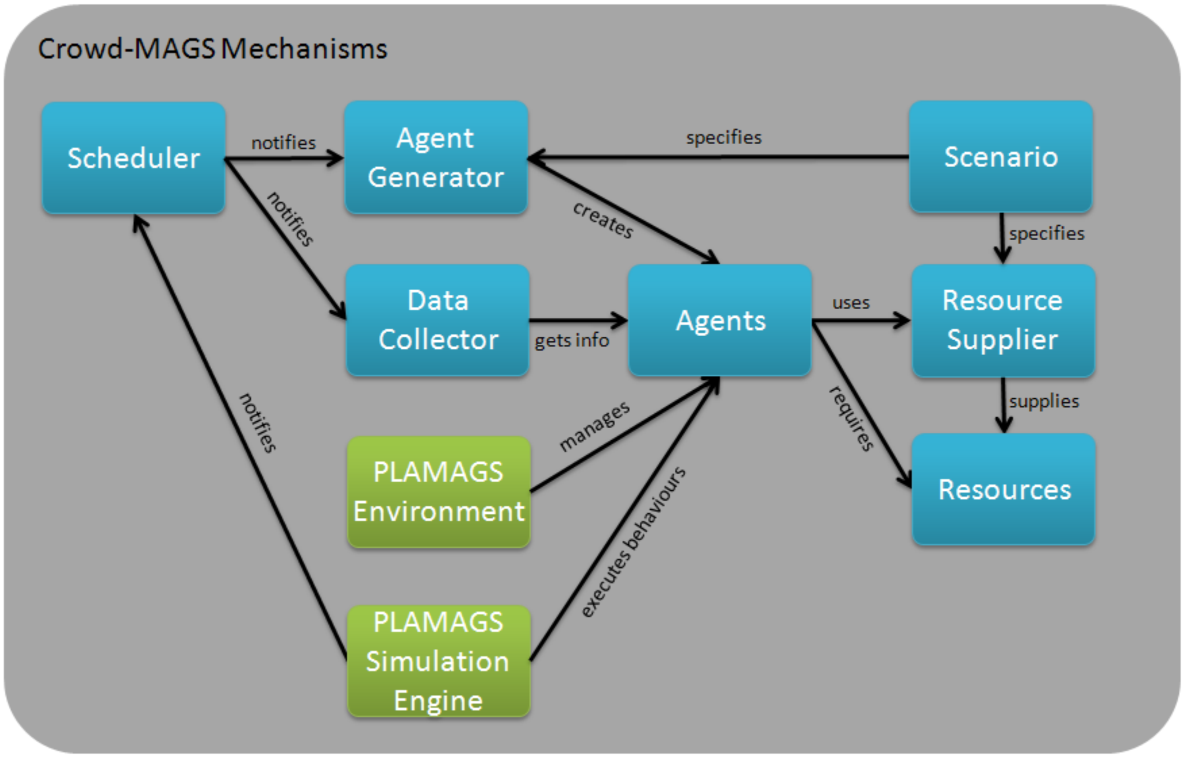


Figure 5.8: Crowd-MAGS application's Mechanisms

component that needs to be scheduled is that it implements a Java method called *execute*. This method can perform any task that the component desires. An important functionality in the Scheduler is to allow a single component to be called at different frequencies for different purposes. This features calls the *execute* method with a special "tag": a concept that is analogous to wake-up calls in hotels. The tag is the phrase that the clerk should say when calling. A client staying for a week could request, for example, to be called once a day with the phrase "Good morning" and three times a day with the phrase "Bon Appétit". The phrase is used to understand the purpose of the call. In this example, the client would receive four calls every day. The Scheduler works in the same manner, except that the phrase is the tag. For example, each agent needs to be called every 10 seconds to update his support towards surrounding STGs and every minute to consider a social identity change. When registering for the 10-second calls, the agent gives a certain tag. When registering for the 1-minute calls, he gives a different tag. Thus, ten seconds after registering, the Scheduler will call the agent's *execute* method and give the 10-second tag as a parameter. The same process will happen ten seconds later, and so on five times. The sixth time corresponds to one minute after registration. Thus, the Scheduler will call the *execute* method with the 10-second tag and will call it again with the 1-minute tag. Hence, the agent will know what he should execute at each call. Tags can be of any Java class, such as *String* or *Object*. Some components need to register for only one frequency so they may use the null object as the tag. Of course, components can unregister when necessary.

Although the Scheduler reduces the number of unnecessary calls, tests have shown that the Scheduler needed to be improved due to performance problems. In order for the simulation to run at a constant speed (as seen by the user), every iteration must take the same time to simulate. For example, if all iterations took 100 ms to simulate, then 10 iterations would happen every second. Unequal iterations would happen if not all iterations took exactly 100 ms. If some took 10 ms and others 500 ms, then the simulation would sometimes speed up and sometimes slow down. Since all agents are independent entities, it is impossible to guarantee that all iterations will take exactly the same amount of time. However, an effort should be made to balance the workload among the available iterations. For example, the behaviour that monitors changes in social identity is executed every minute. If it were executed at the same iteration for all agents, then that iteration would take an enormous amount of time. Thus, it is better to spread the execution of this behaviour among the 600 iterations that happen during each minute. Without a good distribution of tasks, bottle-necks are created at "common" iterations such 10, 20, 30, 100, 1000 etc. To eliminate this problem, the concept of shifting needs to be introduced.

When an object registers with the Scheduler to be called at every 10 iterations, it is normally called at iterations 0, 10, 20, etc. In this case, the shift is 0. If the shift were 3, for example, then the component would be called at iterations 3, 13, 23, etc. Thus, the frequency does not change but the calls are made in iterations that are less loaded. Figure 5.9 illustrates this example. The green agent has a shift of 0 and the blue agent has a shift of 3. When registering, components may choose a precise shift (including 0) but they normally request a random value. Overall, the system is much more balanced since the components are spread out statistically evenly over the range of iterations. In addition, the simulation is much more plausible with respect to less frequent events. For example, without shifting, all social identity changes would happen at the same iteration for all agents, then there would be one minute with no change, and then another number of changes, etc. With random shifting, social identity changes happen seemingly sporadically.

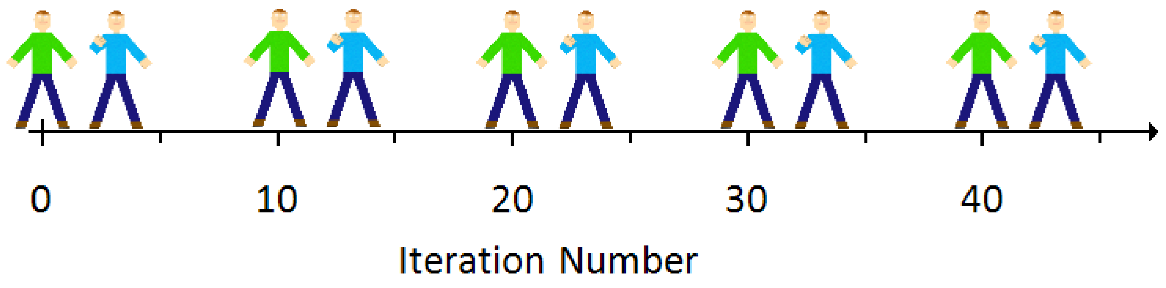


Figure 5.9: Scheduling With and Without Shifts

Finally, the Scheduler can be used to register punctual events rather than periodical events. For example, gas cans last for only a short moment once they are set off. Thus, when thrown, gas cans register with the Scheduler to be called once to start the emission of gas and once to stop it. This feature also is generic since the concept of tags is still used, and the same *execute* method is called. The only difference is that a precise time for the call is required instead of the frequency of calls.

5.2.2 Resource Supplier

The resource supplier is a fairly complex concept. In fact, it was built to be highly generic and extensible for future projects. A resource supplier component can provide agents with one or more specific resources.

Concept Details

The resources provided by a resource supplier may be anything from a physical object (e.g. banner, tear gas can, etc.) to less concrete concepts (e.g. “relaxation” and “healing power” to aid agents who have been hurt). Thus, a resource supplier can represent any concept from a pile of banners left on the ground to a first aid tent. In fact, a resource supplier does not even need to be embodied in the VGE (e.g. a zone such as a park could provide “relaxation” to bystanders).

Analogously to a vending machine that must be filled with the snacks that it provides, a scenario designer must indicate what resources can be provided by each resource supplier. Three parameters are required as inputs: the name of the resource to be provided, the quantity available (0 to infinity), and the time necessary to obtain the resource. Specifying that an infinite amount of a resource is available is a way of representing concepts such as a water fountain that can provide “thirst-relief” units or a bed that can provide “energy” units. The obtention time simulates resources that cannot be obtained instantaneously, such as food in a restaurant or anti-riot equipment that is stored inside of a police truck. For example, CF agents could require 25 seconds to obtain their armour, helmet, gas mask, shield and baton. Officers who can use plastic bullets could need an extra 5 seconds to get their rifle.

Interaction with Agents

The interactions that the resource suppliers can have with the agents consist mainly of requesting and cancelling resources. Agents may request multiple resources at one time from the same supplier. For example, if an agent requests three resources that each have an obtention time of 15 seconds, then he can get his three resources after 15 seconds, not 45. For example, it may take 15 seconds for an officer to get his helmet from the back of a police truck, and it would take approximately the same time to get a helmet and a baton. After an agent has made a request, he may cancel it, or cancel all of his pending requests. Resource suppliers can answer queries with respect to what resources they provide and to how many are left. Suppliers can also be refilled with more units of certain resources whenever necessary.

The algorithms concerning the interactions between resource suppliers and agents are presented at the end of this section. These algorithms are also explained below.

To obtain a resource, an agent must call the *request* method on the desired resource supplier. This action is performed by the resource acquisition behaviour, which was detailed in Sub-Section 4.5.1. If the method returns true, then the agent gets one unit of the resource and he can move on with his activities. If the request method returns false, then either there is no resource left, or one unit has been reserved for the agent. The first case rarely happens because the agent can check if there are sufficient resources left before requesting. In the second case, the agent has to wait. He can keep requesting the same resource, and the method will return false until the obtention period is over.

The algorithm for handling requests is implemented in the resource supplier with two helper methods as shown below. The parameters are an agent a and a resource name rn. The resource supplier contains a set of provided resources PR. First, the supplier gets the information about the appropriate resource (line 3). Then, it tries to give one unit if there is no delay to obtain this resource (line 5). If there is a delay, the supplier checks if this agent has already requested this resource (line 6). In the positive case, it returns false if it is too soon to give the resource (line 11) and returns true otherwise after removing the request (line 9). If the agent has no pending request for this resource (line 12), then the supplier tries to reserve one and sets the obtention time to the current time plus the obtention delay specified for this resource (line 14). Whether the reservation was successful or not, the supplier returns false because the resource cannot be given at that moment (line 15).

```
1  ResourceSupplier.REQUEST(a: Agent, rn: String): boolean
2  var r: Resource
3  r := PR[rn]
4  if r.obtentionTime = 0
5     return r.GIVEONEUNIT()
6  if requests[a,rn] != NULL
7     if requests[a,rn] >= current time
8        requests[a,rn] := NULL
9        return r.GIVEONEUNIT()
10    else
11       return false
12 else
13    if( r.RESERVEONEUNIT() )
14       requests[a,rn] := current time + r.obtentionTime
15    return false
```

```
Resource.RESERVEONEUNIT(): boolean
if num left = 8
   return true
else if (num left - num reserved) > 0
   num reserved := num reserved + 1
   return true
else
   return false
```

```
Resource.GIVEONEUNIT(): boolean
if num left = 0
```

```
    **return** false
**else if** num left != 8
    num left := num left - 1
    num reserved := num reserved - 1
    **return** true
**else**
    **return** true
```

5.2.3 Creation and Removal of Agents

This sub-section presents the mechanisms necessary to add agents to and to remove them from the simulation. First, low-level details about agent creation are presented. Then, the agent generator, which simplifies agent creation, is explained.

Agent Creation

To the scenario designer, creating an agent is a simple task. An agent object must be created (by calling the Java constructor) and then its *addToSimulation* method must be called. The agent's constructor requires only a unique identifier and a position in the VGE. Although very simple to use, the *addToSimulation* method triggers the entire agent initialization process. Because agents are so multi-faceted, this process is complex and involves many components.

The first step to add an agent to the simulation is to create an agent object. Then, PLAMAGS creates the component that will represent this agent in the environment. This PLAMAGS component contains mostly the physical attributes of the agent, such as its mass, its perception capabilities and its visual representation. Next, the Data Collector (detailed in Sub-Section 5.2.4) is notified that a new agent was created, so that the collector can keep track of this agent. Then, agent configurators perform initialization and configuration tasks. These configurators can be any Java object that can perform customization on the agents. Normally, one main configurator is assigned to each agent, in addition to several global configurators that are used for all agents. The instance that was assigned to the agent is always executed first; then, all of the global instances are executed sequentially. For example, a global configurator could assign the social identity controller to each agent and initialize the health level.

Each agent possesses an initialization method that he uses to adopt his fundamental social identity and profile, effectively giving it the appropriate behavioural graphs, icons, and color of clothes. The initialization method also sets the necessary variables such as destination, speed, and projected image. Finally, the initialization method registers the agent with the scheduler for all generic mechanisms and behaviours such as perception and the evaluation of enthusiasm. Once the initialization is complete, an event is added to the agent's history of events recording that he has been added to the simulation.

The agent's initialization method requires as inputs the social identity and the profile that the agent will adopt when created. In the scenarios created for the project, a random profile is selected among the available ones using a probability distribution.

Agent Removal

The removal process is almost symmetrical to the initialization process. When the *removeFromSimulation* method is called, the agent calls his uninitialization method, which makes him leave his social group and his STG if applicable and unregisters him with the Scheduler. Next, the agent configurators get called to undo what is necessary. Then, an event is added to the history of events to record that the agent was removed. Next, PLAMAGS is notified to remove the physical representation of the agent from the VGE. Finally, the Data Collector creates a file on disk with the agent's identifier and fills it with information about the agent, such as his histories of actions and events.

The Agent Generator

Although specifying the creation of one agent is simple, it is too demanding to specify every agent by hand for scenarios of reasonable sizes. Thus, the agent generator was developed. This component allows the scenario designer to specify types of agents to be generated at certain locations and at certain moments during the scenario.

An agent generator is a component in the VGE that is not embodied but that has a conceptual location. Using a generator is very simple: first it must be created and then generation requests must be assigned to it. Adding a request is analogous to adding a provided resource to a resource supplier, which was explained in Sub-Section 5.2.2, in that it indicates what types of agents get generated. The following paragraphs explain this procedure in detail.

```
public HumanAgentGenerator(String agentNamePrefix, double locationX, double
   locationY, double allowableDistanceInXorY, AgentConfigurator configurator)
```

Upon creation, the generator needs (x,y) coordinates ($locationX, locationY$) representing the center of a square zone as well as the size of this square ($allowableDistanceInXorY$). All agents will appear at random locations within this zone. In order to force all agents to originate from the exact same location, the square size can be set to zero. Another parameter provided to the generator is a name ($agentNamePrefix$) that will be given to all generated agents, followed by a unique number representing the number of agents that have ever been created by this generator. For example, giving "AgentLeftOfFence" for the name will create agents AgtLeftOfFence_1, AgtLeftOfFence_2, etc. The last parameter for the generator is an agent configurator ($configurator$), which will perform any task such as giving the agent a banner and making him join a specific STG. Once an agent generator is created, it is ready to receive requests for the creation of agents (as shown with the Java method below).

```
public void addRequest(String fsi, Time interval, int numToGenerate, int maxToGenerate)
```

Each request takes four parameters: the fundamental social identity of the agent to be generated (fsi), the interval between each generation ($interval$), the number of such agents to generate every time ($numToGenerate$), and the total number of such agents to generate ($maxToGenerate$). For example, one request may ask to generate five bystanders every minute for a total of thirty. Another one could generate one instigator every second up to a maximum of one, which would simply generate one agent one second after the request was made. The generator will automatically register itself with the Scheduler when it has requests to fulfill and unregister itself when it has no more. Moreover, each agent generator can be used for different purposes, such as promptly generating a specific type of agent or slowly generating a crowd of mixed agents. Since there can be as many generators as desired in a scenario, complete freedom is possible.

5.2.4 Data Collection

So far in this thesis, it was only briefly mentioned that data is collected to perform analyses as the simulations run. Thus, this section presents the active mechanisms that are used to collect data.

Generic Data Collection

The components presented in the environment model (detailed in Section 4.1) are heavily used by the data collection mechanisms. In fact, there is no central algorithm to collect simulation data, but rather a collection of small algorithms distributed throughout various components of the system. However, there is a central repository that stores all data collected system wide. The reason for this decentralization is that a multi-agent simulation is so dynamic that monitoring all components entails a large overhead. For example, a central "monitor" would have to maintain lists of components in the simulation and run through every element periodically to gather whatever information is necessary. Instead, all components in the Crowd-MAGS system are free to register themselves with the Data Collector and to provide any information that they want to make available publicly. Overall, most of the data collection work is done when actions and events happen to individual components, such as an agent being dragged or receiving a plastic bullet.

One of the only active tasks of the Data Collector is to write a text file about each agent when he is removed from the simulation. This file can contain basic information, the histories of actions and events, and any other information that may be useful to analysts. Similar files could be created for STGs as well.

Moreover, the Data Collector serves as the central repository of information because of the information model's architecture. In fact, the Data Collector keeps Java references to agents and STG's, but these keep references to their histories of actions, which keep references to the collective actions, which keep references towards agents and STGs, etc. Thus, nearly all information that was used to simulate agents and STGs is kept available in the same format. From this repository, specialized components can reorganize the data in a more efficient manner or perform data mining.

Specialized Data Collection

Since the data collection process is very lean and generic, a more complex procedure is run by a component that was developed specifically for crowd control purposes. The Strategy Data Gatherer was created to collect information that can be used to compare the outcomes of different scenarios from the control forces' standpoint. The frequency of collection, which is adjustable and currently set at 15 seconds, is the amount of time between two collection iterations. The system currently collects only basic information concerning non-lethal weapons and crowd agents, as shown by the following list.

- Control forces' intervention level
- Number of injured crowd members
- Number of injured CF officers
- Amount of NLW used
- Operational cost (salaries + NLW)
- Ratio between the crowd size at the moment and at the beginning
- Global crowd aggressiveness

Fortunately, the Strategy Data Gatherer can be expanded easily when analysis needs become larger and better defined. Moreover, this component does not only collect data. Once the simulation is stopped, it generates a report summarizing the collected data like the following example shows.

```
Control forces' intervention level: 13 (Offensive Move + Tear Gas + Plastic Bullet)
Number of injured crowd members: 7
Number of injured CF officers: 2
Amount of resources used
   Number of tear gas cans used: 24
   Number of plastic bullets used: 18
Operational cost
   Salaries paid to all CF employees: 1166,36 $
   Cost of tear gas cans used: 1 200,00 $
   Cost of plastic bullets used: 36,00 $
Ratios with initial crowd size:
17h00m15.0s: 1.0273972602739727 (75 / 73)
17h00m30.0s: 1.1095890410958904 (81 / 73)
17h00m45.0s: 1.1917808219178083 (87 / 73)
...
Global crowd aggressiveness: 0.17786721152062537
Average crowd aggressiveness:
17h00m15.0s: 0.15865770101931095
```

```
17h00m30.0s: 0.23025137308043542
17h00m45.0s: 0.21783029279321448
...
```

Validation with a System Dynamics Model

As mentioned in Section 1.4, a basic level of system validation was performed by comparing the Crowd-MAGS's simulation output with a system dynamics model's output. This SD system models social identities and collective actions like the Crowd-MAGS system, so both systems can "understand" each other. Differences exist in the categorization of agents into the social identities and the categorization of actions with respect to the aggressiveness level, but conversion algorithms were developed. More details will be given in Section 6.8 and a complete analysis of this validation process is presented in (Larochelle and Moulin, 2009).

The SD Data Organizer is a component that collects data periodically and stores it in the format necessary for the SD model, which was developed with Vensim. Naturally, the frequency of collection can be adjusted. The information collected is the number of agents in each adopted social identity, as well as the number of actions performed by agents of each adopted social identity, for different categories of actions.

At each collection period, the SD Data Organizer iterates over every agent and determines the category to which the agent belongs. Then, the SD Data Organizer iterates through all actions that the agent performed during the last collection period. Five categories of crowd actions have been defined: pacific / preventing violence, provocation / encouraging violence, violent action against environment, violent action against CF, and panic. Six categories exist for control forces: communications, defensive move, offensive move, tear gas, water cannon, and plastic bullets. Once the simulation stops, a report is generated in a format that is directly understandable by Vensim.

5.3 Input Parameters

This section tries to present a complete list of all parameters that can be configured in the Crowd-MAGS system. For parameters required by the PLAMAGS language itself (such as the GIS file for the map), the reader can refer to (Garneau, 2007). The parameters that can be configured in the Crowd-MAGS system are specified in the XML scenario files. Nine categories exist and they are listed below with their respective contents.

Physical Properties of Agents

- Mass of the agent
- Height of the cylinder containing the agent's 3D representation
- Radius of the cylinder containing the agent's 3D representation

- Base walking speed for a healthy agent that moves normally
- Field of view for agents and components
- Field of view for interest shape
- Field of view for gas clouds
- Factor of the base speed when the agent flees something

Specifications of Social Identity Profiles

- Profile's Name
- Social identity associated with the profile
- STG type that agents with that profile will seek to join or create
- *Minimum*, *Optimum*, and *Maximum* parameters required for the *Appreciation of Aggressiveness* function
- *Minimum* and *Maximum* parameters required for the *STG Adhesion* algorithm
- Minimum gap between current and projected enthusiasm when considering a social identity change
- Weight of past events when updating the support towards a STG

Crowd Characteristics

- Probability distribution of all profiles associated with the *bystander* social identity
- Probability distribution of all profiles associated with the *demonstrator* social identity
- Probability distribution of all profiles associated with the *instigator* social identity

Squad Characteristics

- Number of gas cans to get from the police truck when refilling
- Number of plastic bullets to get from the police truck when refilling
- Minimum aggressiveness necessary from the target to use tear gas
- Minimum aggressiveness necessary from the target to use plastic bullets
- List of required protective equipment
- Loose formation's radius for the *Uninvolved* mobilization level
- Tight formation's radius for the *Physical Presence* mobilization level
- Distance between agents when in line formation for the *Force Demonstration* mobilization level

- Distance between agents when in wedge formation for the *Crowd Dispersion* mobilization level

Health Parameters

- Amount of impact of tear gas
- Amount of impact of Molotov cocktails
- Factor of increased tolerance to tear gas from control forces with respect to the crowd
- Amount of protection for punches and kicks from bodily armour
- Amount of protection for Molotov cocktails from bodily armour
- Amount of protection for punches and kicks from helmet
- Amount of protection for Molotov cocktails from helmet
- Amount of protection for punches and kicks from shield
- Amount of protection for Molotov cocktails from shield
- Critical health level of control forces
- Critical health level of bystanders
- Critical health level of demonstrators
- Critical health level of demonstrator leaders
- Critical health level of instigators
- Critical health level of instigator leaders

Aggressive Actions Parameters

- Minimum distance between a crowd member and control forces to assault them
- Minimum distance between a crowd member and control forces to attack them
- Safe distance for throwing Molotov cocktails (minimum and maximum)
- Impact of punches and kicks on health

Tear Gas Parameters

- Mass of a can
- Safe distance for throwing a can (minimum and maximum)
- Duration of emission for a can
- Amount of time that a can stays in the air when thrown

- Inaccuracy when throwing a can

Plastic Bullet Parameters

- Velocity of a bullet
- Safe distance for shooting a bullet (minimum and maximum)
- Inaccuracy when shooting a bullet

Miscellaneous Parameters

- Scenario start time
- Time before demonstrator leaders consider the protest stagnant

5.4 System's Structure

The PLAMAGS implementation of the Crowd-MAGS project is composed of various technical packages that are somewhat independent. PLAMAGS provides a simulation engine and an environment for basic multi-agent simulations. The requirements of the Crowd-MAGS project called for a complete framework for the simulation of complex agents, such as humans, and complex phenomena, such as social interactions in crowd situations. Thus, many new structures, tools, and utilities were developed as an extension to PLAMAGS in order to simulate and analyze the necessary social behaviours. This section presents these structures and introduces the system that was built.

Architecture

Crowd-MAGS architecture is relatively simple and quite similar to Pan's MASSEgress system, presented in Sub-Section 2.6.3. In fact, Crowd-MAGS architecture can be explained using Figure 2.22. In Crowd-MAGS, the geometry engine and the visualizer are PLAMAGS' 3D environment engine that uses PhysX. The crowd simulation engine is composed of Crowd-MAGS' system code and PLAMAGS's behaviour engine. Next, the events recorder is the 'DataCollector' Java singleton object, which uses the information model. The population generator is the scenario file and the 'AgentGenerator' Java objects. Finally, the global database represents the GIS file and the Java objects in RAM, such as the agents' memory contents. For brevity's sake, the more technical descriptions of the Crowd-MAGS system are presented only in (Larochelle and Moulin, 2009).

Four modes exist in the Crowd-MAGS system, although one is used only while the system is loading at start-up. The possible transitions between these states are shown in Figure 5.10. System loading is the first state that is always activated when the system starts. This state consists of two parts: first, all semantic definitions necessary for social identities, actions, events, interest shapes, and other uses are instantiated. Next, all BDL files are read by the

PLAMAGS engine so that it can build an internal representation of all behaviours, 3D models, and other utilities. Once the system is loaded, it automatically goes in the default mode and the main interface (detailed in Section 5.5) is brought up to the screen. From this point, the user can choose to edit a scenario (the system goes in editing mode) or to run a scenario (the system goes in simulation mode). When the system is in one of these two modes, it must come back to the default mode before proceeding to any other task.

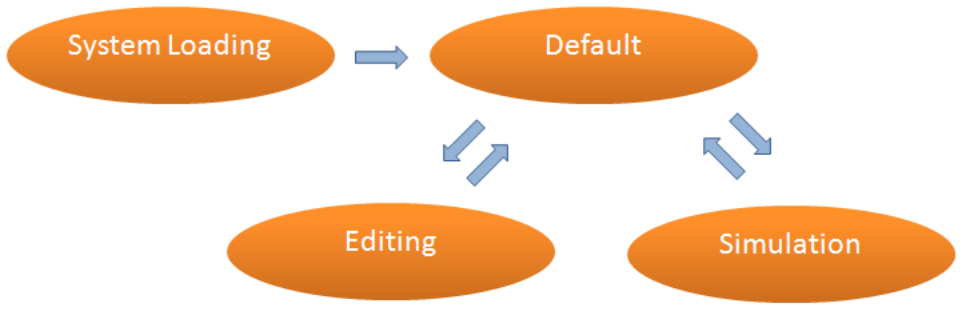


Figure 5.10: System Modes

In the default mode, only Java's basic and user interface threads are running. However, multiple threads are used for simulation and post-processing of simulation data in order to improve robustness and performance. Thus, going into the editing or simulation modes creates new threads, many of which are created by PLAMAGS and its 3D environment engine. These threads, detailed in (Garneau, 2007), are shut down when the application goes back to the default mode.

Source Code Organization

Given the requirements for a system as complex as Crowd-MAGS and the time frame available for development, the Java programming language was determined to be the best solution for the system's main language. The main advantages offered by Java for this project are that 1) it is well known by the developers, 2) PLAMAGS is based on it, 3) it provides a rapid development environment, and 4) it provides advanced functionalities such as robust multi-threading and complex inheritance structures. Moreover, the Crowd-MAGS system also takes advantage of PLAMAGS' programming language: the Behaviour Description Language (BDL). This language offers facilities for writing complex parallel and hierarchical behavioural graphs in a manner that is much simpler than any mainstream language, such as Java. Thus, many of the agent behaviours are specified in .BDL files, although some are specified in Java files. Figure 5.11 shows the source code's architecture.

The "generated" Java code is a conceptual way of thinking of the output of the BDL interpreter. However, no Java code is actually generated because the BDL compiler that was planned has not been completed. Instead of being compiled, the BDL code is interpreted: the interpreter reads the BDL code and calls Java code that is equivalent to the BDL code. Unfortunately, interpreted languages are slower than equivalent compiled languages and in this case the effect on performance is drastic. Tests with an incomplete version of a BDL

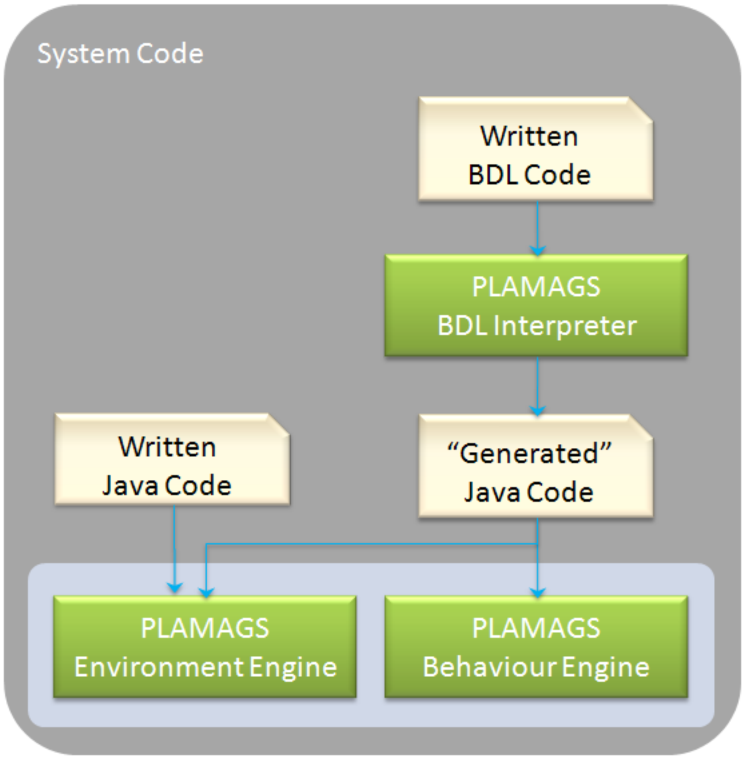


Figure 5.11: Source Code Architecture

compiler showed that the compiled code executed approximately 100 times faster. Because every agent in the simulation executes (at least partially) his behavioural graph at every iteration, the performance drain increases severely with the number of agents. Thus, the behaviours written in BDL are limited to those where the equivalent Java code would be too complex. For example, coding behaviours that require resources would be far too complicated and error-prone in Java.

Figure 5.12 presents the arrangement of the source code packages (the arrows indicate dependencies). The Crowd-Control MAGS and Crowd MAGS components represent the majority of the code that was developed for this project. The Crowd MAGS package consists of Java classes related to simulating crowds, without any relation with crowd control. The Crowd-Control MAGS package extends several classes from the Crowd MAGS package and introduces new ones to simulate control forces and crowd control concepts, such as health. The PLAMAGS 'package', which represents the entire PLAMAGS library, is detailed in (Garneau, 2007). The PhysX package is a commercial one and its documentation is available at `http://developer.nvidia.com/object/physx.html`. The Utilities package was developed to contain various abstract data types and tools to deal with computer memory and image processing. Thus, Utilities and PhysX packages can be reused in any project. They are very unlikely to change, although features may be added to them at any time.

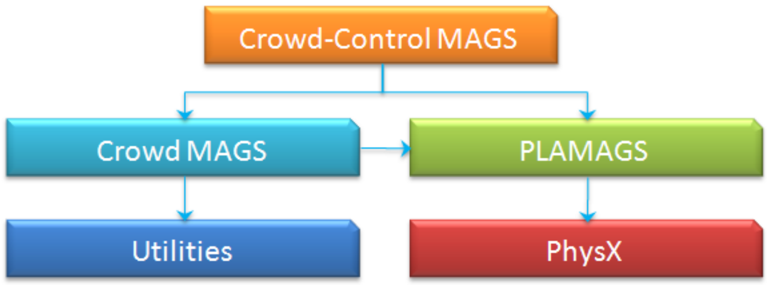


Figure 5.12: Source Code Package

Performance Limits

Many performance tests were run and are presented in Appendix C. All tests indicate that under any circumstances, the number of agents could not go over 750 agents. In fact, depending on the events happening in the scenario, the simulation usually runs slower than real-time for about half of that number. Although the exact causes of the slow-down are not identified exactly, all clues lead to think that the PLAMAGS simulation engine is very slow, most especially the behaviour engine. Thus, development efforts should be focused on the behaviour engine, should this project be continued.

The ultimate goal of the Crowd-MAGS system was to simulate up to 5000 agents. This number was a target because data collected from the 2001 Summit of the Americas was available and the real-life situations involved approximately 5000 crowd members. Unfortunately, this goal was not reached. However, the system's performance is positively comparable to other similar systems. For example, Varner's SULNT, presented in Sub-Section 2.5.3, can simulate between 2 and 500 agents. McKenzie ran tests on his Crowd Federate system (presented in Sub-Section 2.5.2) ranging from 35 agents up to 200 agents, although the actual limit is unknown. Pan's MASSEgress system, presented in Sub-Section 2.6.3, is quite out of range because it can simulate more than 2500 agents. It should be kept in mind that MASSEGress is an evacuation simulator, and not a crowd control simulator; the agent behaviours are much simpler.

5.5 User Interface

This section presents the main parts of the user interface created for the Crowd-MAGS system. It is also possible to run the simulation without any graphical user interface (from the command line). The interface was built by assembling different Java classes, most of which are shown in Figure 5.13. All of the GUI components could be easily relocated on the screen (e.g. placing the control bar at the bottom of the interface instead of at the top). Several of these components could also be reused in other projects, whether they are related or not to crowd control. The main user interface presented is used for running scenarios; Sub-Section 5.5.5 presents the user interface for editing scenarios.

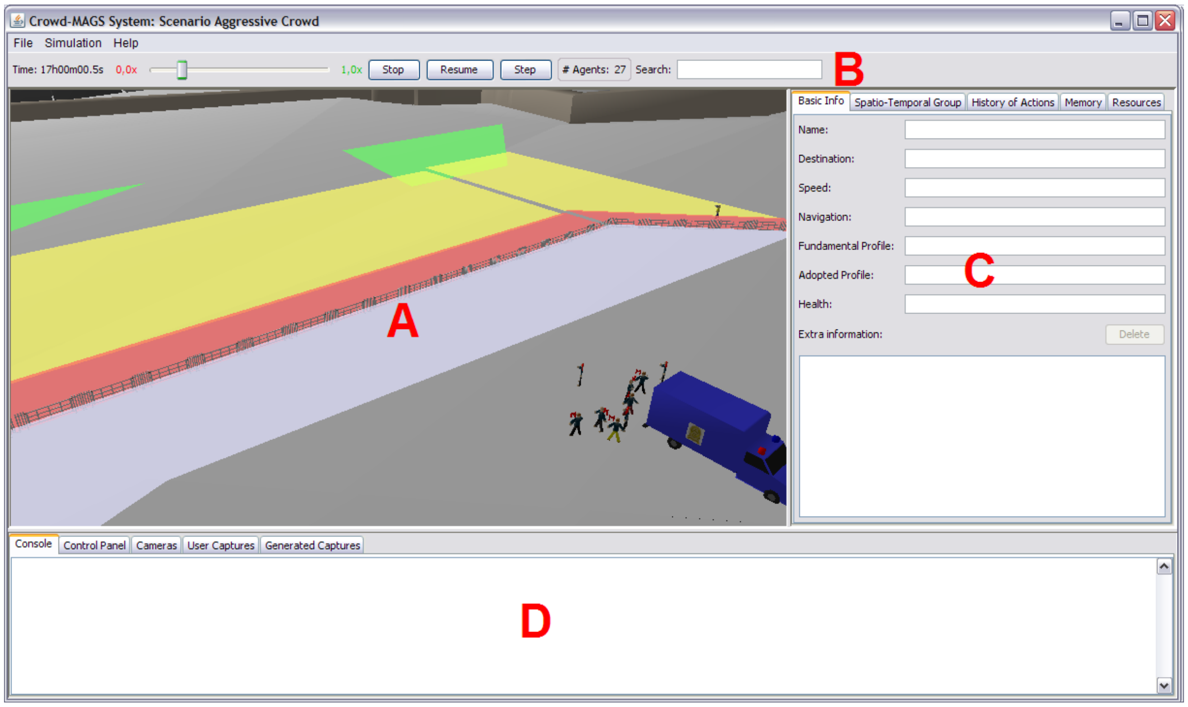

Crowd-MAGS System: Scenario Aggressive Crowd
File Simulation Help
Time: 17h00m00.5s 0,0x 1,0x Stop Resume Step # Agents: 27 Search:
B
Basic Info Spatio-Temporal Group History of Actions Memory Resources
Name:
Destination:
Speed:
Navigation:
Fundamental Profile:
Adopted Profile:
Health:
Extra information:
Delete
C
A
Console Control Panel Cameras User Captures Generated Captures
D

Figure 5.13: User Interface: The Main Window

Figure 5.13 shows the main window that is shown to the user when a simulation starts. The screen is divided into four main parts. The most prominent one is the simulation window (A), which shows the 3D animation. On top of it sits the control bar (B), which allows controlling the simulation such as starting, pausing, or slowing down. The information box (C) is on the right of screen, only providing information about the components in the simulation. At the bottom is a multi-purpose panel (D) that shows messages from the simulation, different cameras, and screen captures. Each one of these components is detailed in the following sub-sections.

5.5.1 The Simulation Window (Part A)

The simulation window is probably the component with which a user would interact the most. With this component, the user can rotate the camera by moving the mouse and move in the 3D environment by using the keyboard. Other interactions can be programmed, such as making the system react to the user clicking on agents or on the map. For example, clicking on an agent selects it, filling the information box with the agent's information and placing a large arrow on top of the agent, as shown in Figure 5.14. The simulation window is currently in the anamorphic widescreen format (16:9) but it could be resized to any dimensions. Unfortunately, PLAMAGS does not allow this screen to be resized dynamically, so it must be set before the system is loaded.

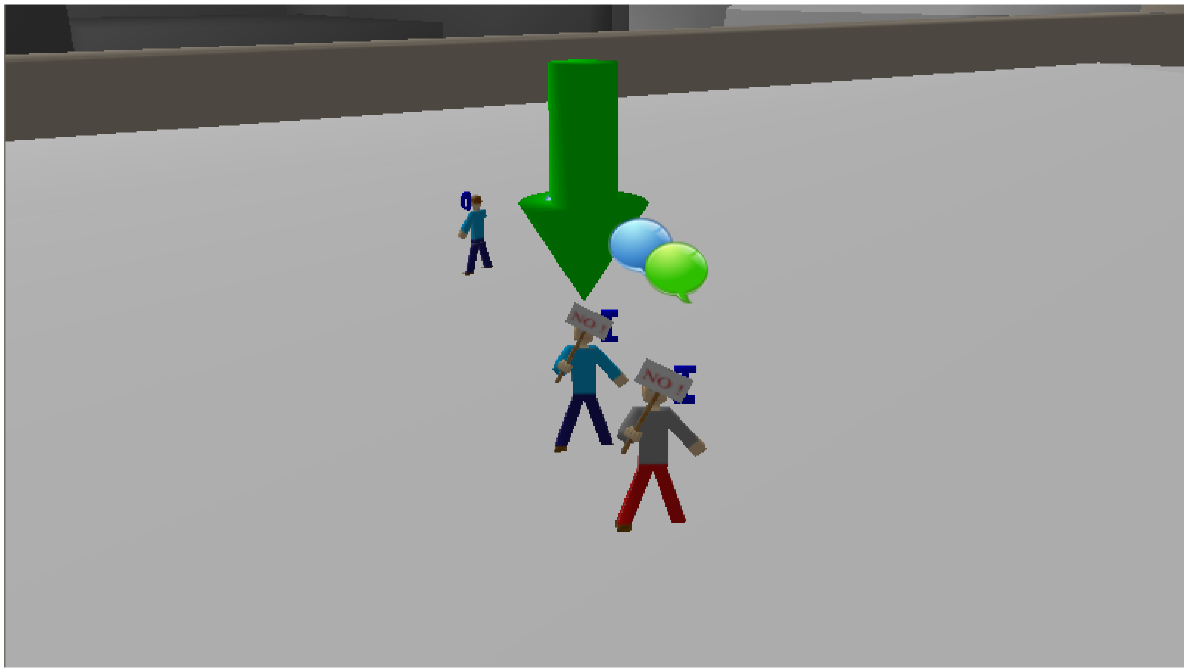


Figure 5.14: User Interface: A Selected Component

5.5.2 The Control Bar (Part B)

The control bar, shown in Figure 5.15, is a simple component that is used to control the simulation engine. Starting from the left, the bar shows the simulated time and the performance. A performance of 1X means that one simulated second takes one second of real time; a performance of 2X means that one simulated second takes half a second of real time. Next, the control bar shows a slider, which allows controlling the speed of the simulation from 0.1X (10 times slower than reality) to 5X (5 times faster than reality) in small increments. Pushing the slider to the extreme right lets the simulation run as fast as it can, so the performance is dependent only on the computer's hardware and software. To the right of the slider is a green indicator of the requested speed. The actual speed, which may be slower depending on the computer and the complexity of scenarios, is shown in red farther left in the control bar. Next, a few buttons allow controlling the simulation, such as pausing and stepping one iteration at a time. Next, the number of agents that are currently in the simulation is shown. Finally, the control bar offers a search functionality. Typing the unique identifier of any component - agent, fence, interest point, etc. - will search and select this component, if found.

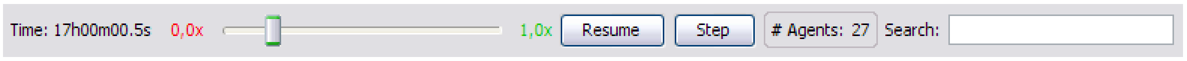
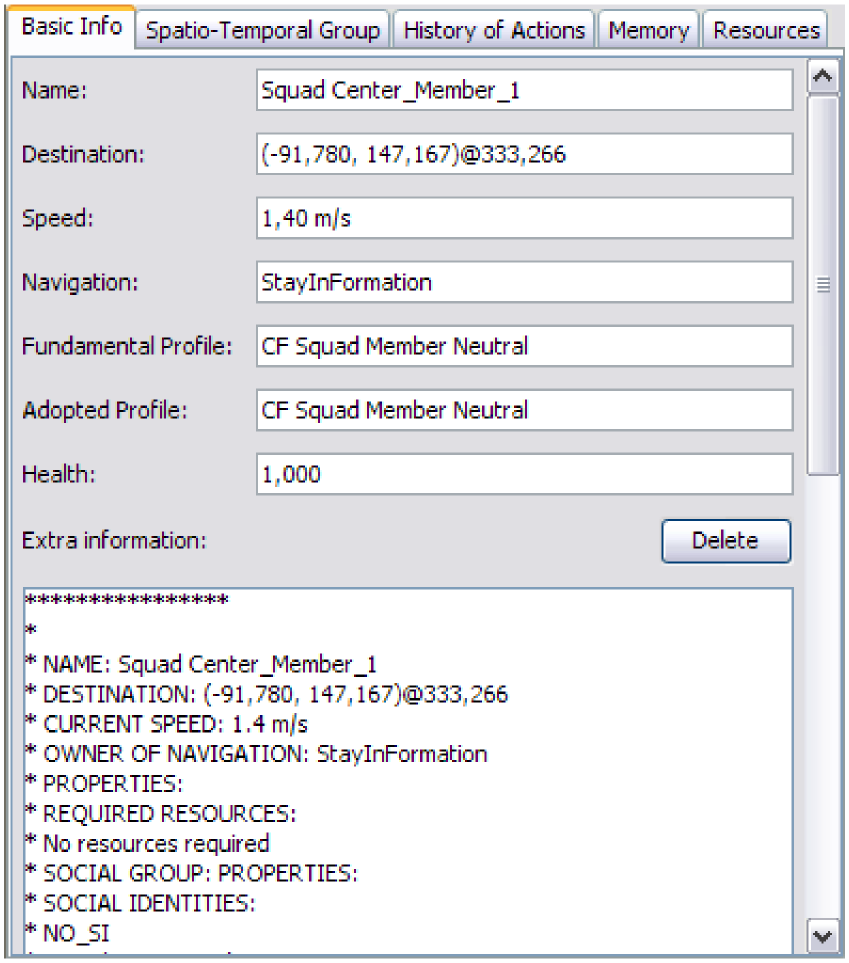


Figure 5.15: User Interface: The Control Bar

5.5.3 The Information Box (Part C)

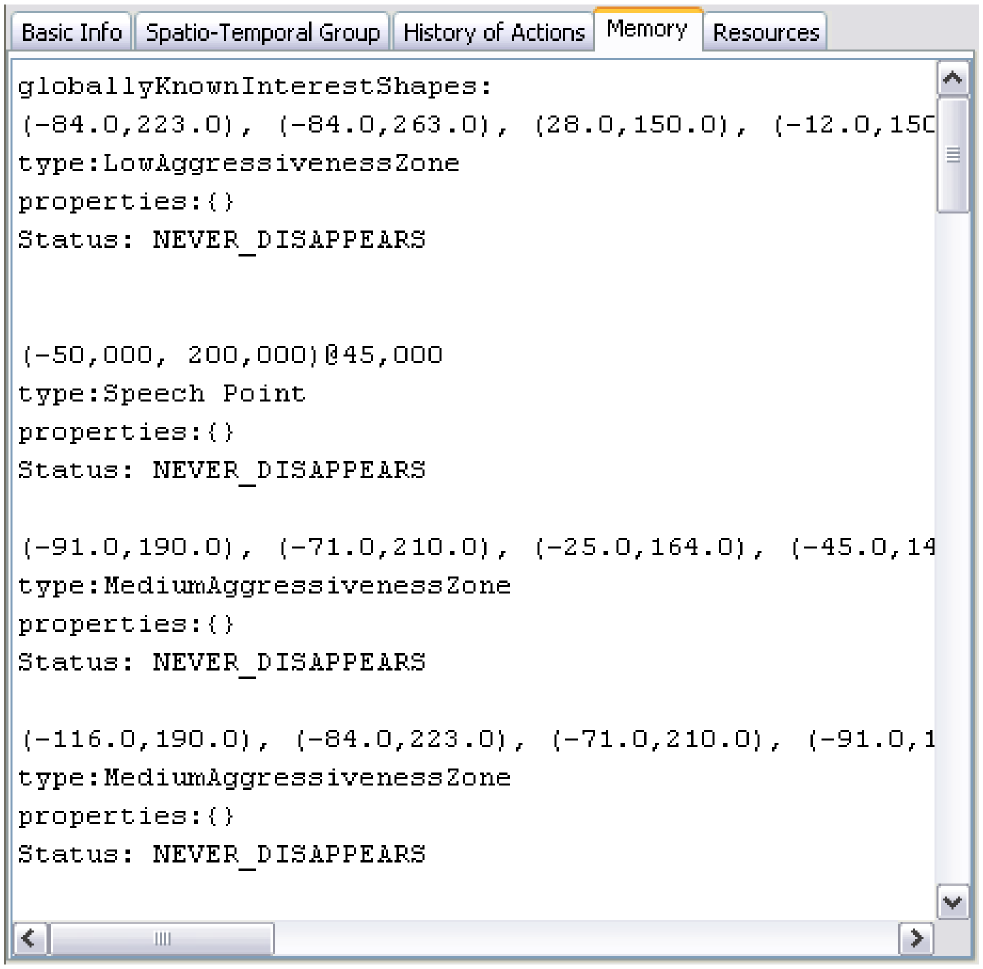
The Information Box is a component that was initially developed for debugging and calibrating the system, but it turned out to be highly valuable when tracking the events of a complex scenario. The panels included in the Information Box provide extensive information about agents and components. As shown in Figure 5.16, the "Basic Info" tab provides the most common and basic information about the selected agent. The other four panels show information about the selected agent's STG (name, location, formation, aggressiveness level, actions, etc.), the agent's actions, the agent's memory (interest points, projected images, etc.), and the agent's owned and required resources. An example of the "Memory" panel is shown in Figure 5.17. When a non-human component is selected, the five panels are replaced by one blank panel where basic information is listed.



Name:	Squad Center_Member_1
Destination:	(-91,780, 147,167)@333,266
Speed:	1,40 m/s
Navigation:	StayInFormation
Fundamental Profile:	CF Squad Member Neutral
Adopted Profile:	CF Squad Member Neutral
Health:	1,000
Extra information:	

```
****************
*
* NAME: Squad Center_Member_1
* DESTINATION: (-91,780, 147,167)@333,266
* CURRENT SPEED: 1.4 m/s
* OWNER OF NAVIGATION: StayInFormation
* PROPERTIES:
* REQUIRED RESOURCES:
* No resources required
* SOCIAL GROUP: PROPERTIES:
* SOCIAL IDENTITIES:
* NO_SI
```

Figure 5.16: User Interface: The Information Box (Basic Information)



```
globallyKnownInterestShapes:
(-84.0,223.0), (-84.0,263.0), (28.0,150.0), (-12.0,150
type:LowAggressivenessZone
properties:{}
Status: NEVER_DISAPPEARS



(-50,000, 200,000)@45,000
type:Speech Point
properties:{}
Status: NEVER_DISAPPEARS

(-91.0,190.0), (-71.0,210.0), (-25.0,164.0), (-45.0,14
type:MediumAggressivenessZone
properties:{}
Status: NEVER_DISAPPEARS

(-116.0,190.0), (-84.0,223.0), (-71.0,210.0), (-91.0,1
type:MediumAggressivenessZone
properties:{}
Status: NEVER_DISAPPEARS
```

Figure 5.17: User Interface: The Information Box (Memory Information)

5.5.4 The Multi-Purpose Panel (Part D)

The Multi-Purpose Panel, shown in Figure 5.18, provides a blank text box and three filmstrips. Of the four tabs, the first one is the only one that is simply a blank text box that gets filled with various information as the simulation progresses. Anything that may be useful to the user could go in it, such as a notice for every social identity change and an indication when a new STG has been created.

The next tab holds the Commander Control Panel, as shown in Figure 5.19. This panel is used to guide the control forces' strategy during a simulation. It is currently simple but it could easily be extended. All of the commands on this panel currently apply to all of the squads in the scenario, although it will be possible to develop a selection function to give orders to only one squad at a time. The first four buttons are used for ordering squads to adopt the respective mobilization level. The next button is used for ordering the squad leaders to use megaphones to order the crowd to disperse. Finally, two check boxes allow setting the non-lethal weapons that the squads are allowed to use. Every time that they are clicked, the

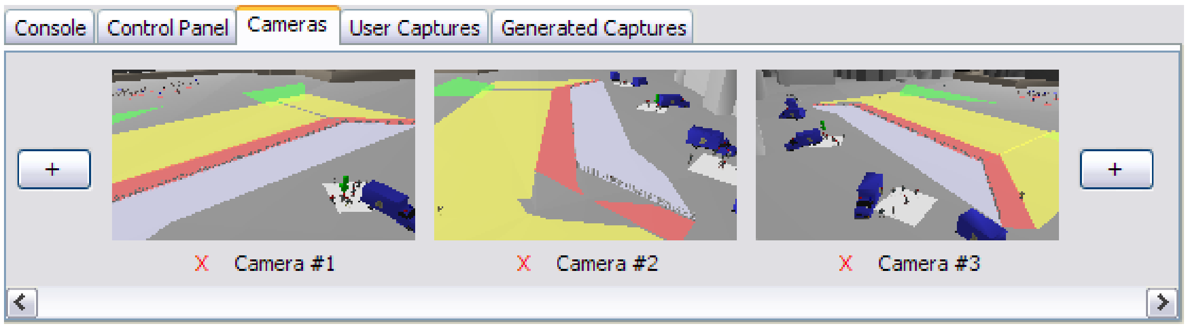


Figure 5.18: User Interface: The Multi-Purpose Panel

agents go get the necessary resources if they have none. For example, a chemical officer gets 3 tear gas cans if he has none left. Similarly, a rifle officer gets 1 rifle and 10 bullets.

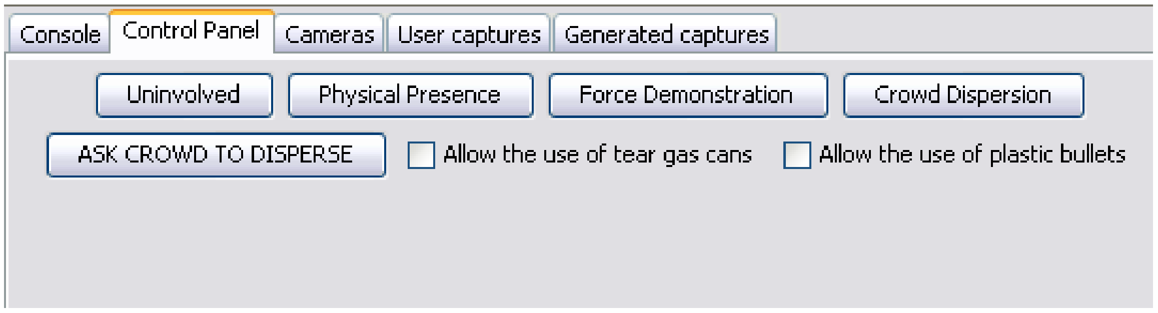


Figure 5.19: User Interface: The Commander Control Panel

The next three tabs are all based on a filmstrip component. The first tab shows all cameras that have been set for the scenario. It is possible to add and remove cameras at any time during a simulation. In the current implementation, cameras are fixed: moving around in the environment has no impact on the cameras in the thumbnails. Right-clicking on any camera will open an external window with this camera (shown in Figure 5.20). The last two tabs are very similar to each other, as they both contain a list of screen captures. In the “User captures” tab, clicking on the “+” button takes a capture of the main camera and saves it to a file along with information about the time at which the capture was taken. For examples, captures could be saved every 15 minutes to monitor a situation and perform in-depth analyses once the simulation is over. “Generated captures” are taken when specific events happen. For example, a scenario could specify that every time that a STG reaches an aggressiveness level of 0.9, a picture is to be taken. Another example would be capturing the screen every time that a CF member receives a Molotov cocktail.

5.5.5 Scenario Specification

In order to edit scenarios, a user must use different screens than the ones presented above. However, the interface for the scenario specification module has been designed to be as similar

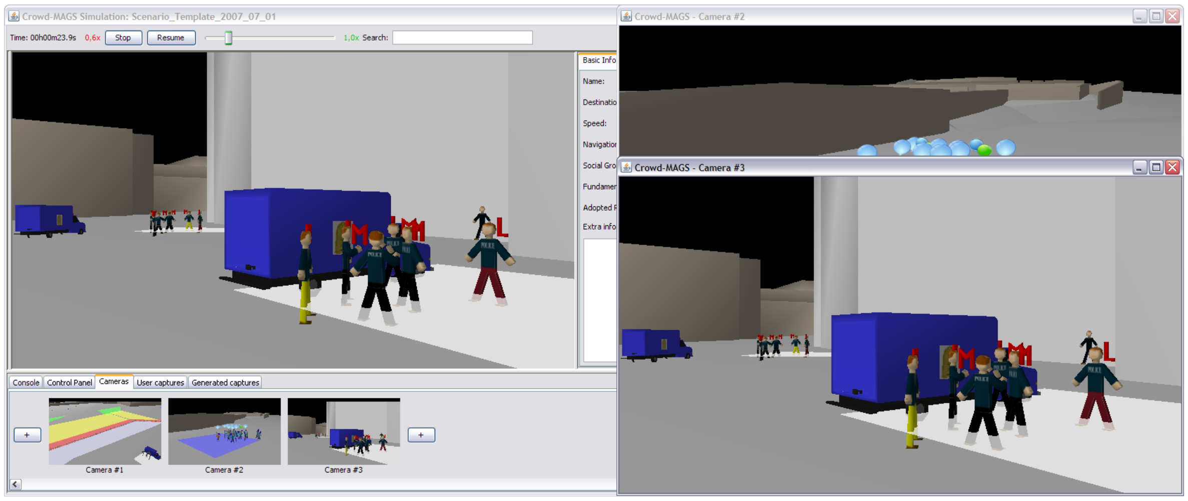


Figure 5.20: User Interface: External Camera Windows

as possible as the main user interface. Figure 5.21 shows the main window used for editing scenarios. The middle part shows the simulation window, just like in the regular interface. The only difference is that all elements are motionless, since the simulation is not running.

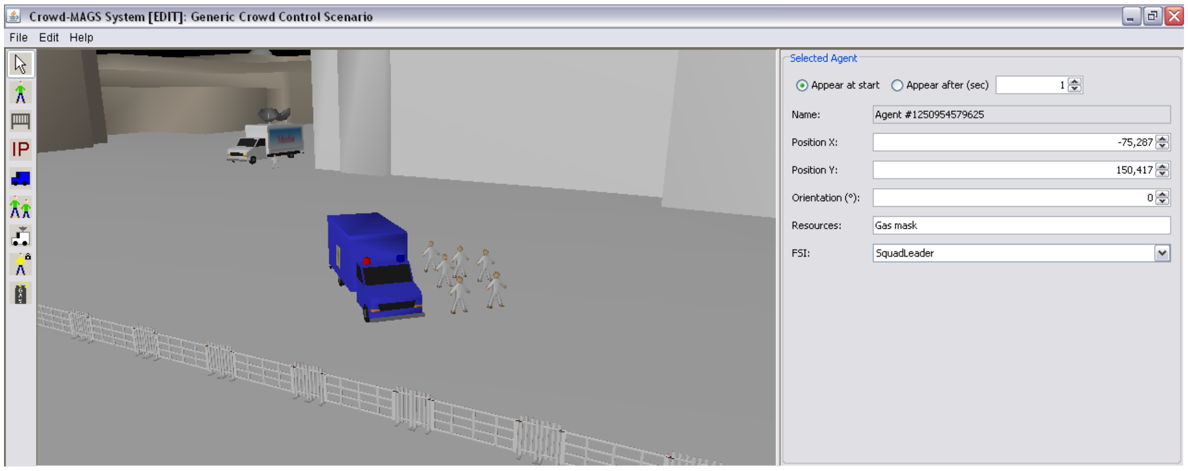
On the left of Figure 5.21 is the palette, which allows adding simulation components to the scenario with a single click. The user must first select what type of component he wants to add to the scenario, and then click on the map wherever he wants one instance of the component to appear. The available types are: agent, fence, interest point, police truck, agent generator, media truck, journalist, and tear gas can.

An information panel to edit components can be seen on the right of Figure 5.21. Options are available to select the time of appearance, the position, the fundamental social identity for agents, the available resources, and other details.

Finally, scenario parameters, such as the start time, the crowd distribution, and non-lethal weapons impacts, can be edited with the configuration panels, shown in Figure 5.22. This panel is accessible from the menus in Figure 5.21. Nearly all parameters in the XML scenario file, and listed in Section 5.3, can be edited with these configuration panels.

5.5.6 User Guide

A user guide for the Crowd-MAGS application is provided in (Larochelle et al., 2009). It explains the user interface in depth and provides details about every available feature as well as instructions on performing common tasks.

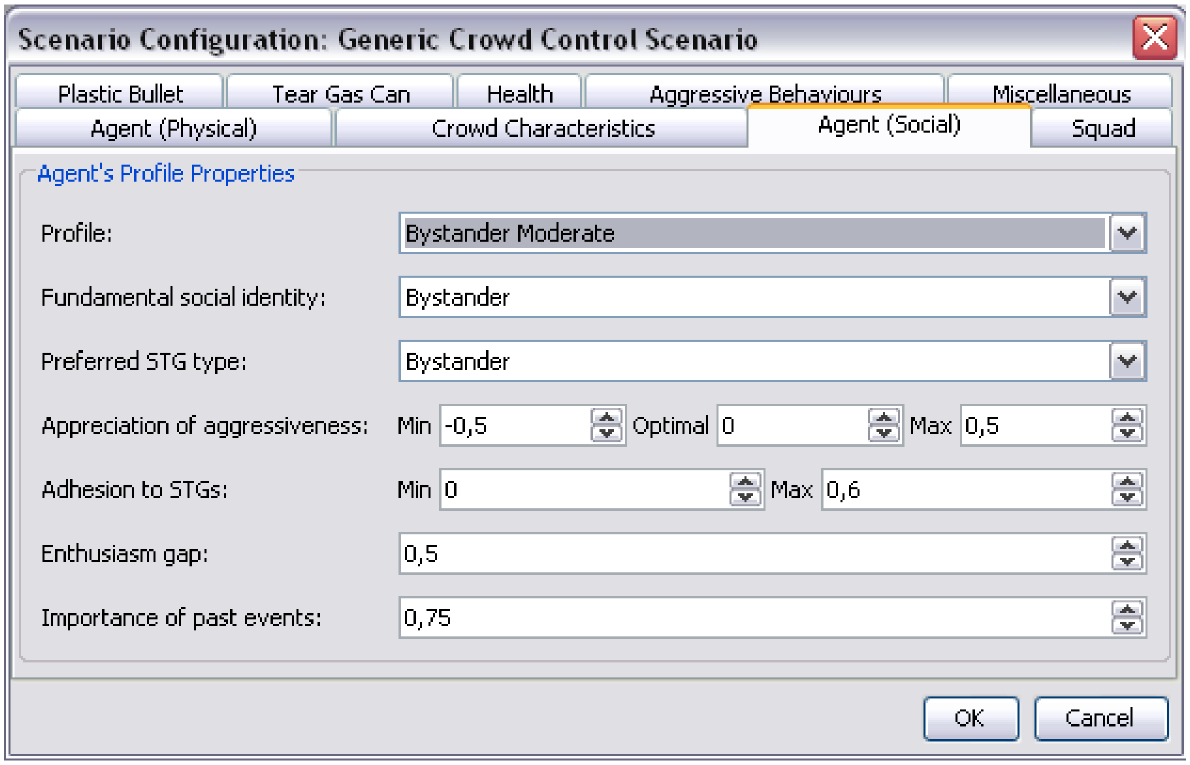


Selected Agent

[✓ Appear at start] Appear after (sec) 1

Name:	Agent #1250954579625
Position X:	-75,287
Position Y:	150,417
Orientation (°):	0
Resources:	Gas mask
FSI:	SquadLeader

Figure 5.21: User Interface: The Scenario Specification Main Screen



Agent's Profile Properties

Profile:	Bystander Moderate
Fundamental social identity:	Bystander
Preferred STG type:	Bystander
Appreciation of aggressiveness:	Min -0,5 Optimal 0 Max 0,5
Adhesion to STGs:	Min 0 Max 0,6
Enthusiasm gap:	0,5
Importance of past events:	0,75

Figure 5.22: User Interface: The Configuration Panel

5.6 Conclusions

Chapter 5 started by discussing how the theoretical models were put in application. Most models were covered, including models for the agent, STG, and non-lethal weapons. Mechanisms such as the Scheduler and the Data Collector were presented because they are crucial to the application even if they are not based social models of crowds. A list of parameters available in the system was presented to give an idea of the customization possibilities for the system and the scenarios. Finally, the technical structure of the system was discussed, as well as its user interface.

Now that Chapter 4 presented the Crowd-MAGS models and that Chapter 5 showed how these models are used in the Crowd-MAGS application, the reader should have an idea of what is possible with the system and how it can fulfill the project's objectives. Chapter 6 goes in this direction by discussing the experimentations that have been performed to assess the effectiveness of various non-lethal weapons and control force strategies.

Chapter 6

Experimentations with the Crowd-MAGS System

Once a system has been built, a thorough calibration process must be undertaken before the system can be used with confidence. This chapter presents the methodology used during the calibration phase and the results of the experimentation with the Crowd-MAGS system. Because no team member was a domain expert, the only analysis performed on output data was used to calibrate the scenarios and models, and not to draw conclusions about control forces intervention strategies. Several iterations of the four phases (shown in Figure 3.1) were executed to achieve a desirable level of calibration.

The calibration and validation methodology was partly based on the available literature. For example, (Silverman et al., 2003) insists on the need to "evaluate how well scenario outcomes correspond to real world or historical events". Silverman and his team also suggest combining qualitative and quantitative evaluations to include both opinions from domain experts and numerical validation . Furthermore, (Jager et al., 2001) offers useful recommendations. Among others, Jager and his colleagues suggest to experiment with different protective equipment (e.g. gas masks), spatial distribution of the agents, and pre-event organization (e.g. CF intervention techniques), and crowd composition. All of these suggestions and some others have been considered and are reflected by the experiments presented in this chapter.

Section 6.1 presents a scenario that will serve as a basis for the entire chapter. Section 6.2 discusses how parameters were initially calibrated while the system was being developed. Section 6.3 explains how parameters were further adjusted by qualitative evaluation of the basic scenario's results. Next, Section 6.4 lists the criteria that will be used for the full calibration of the system. Also, Section 6.5 presents the scenarios that will be used for the full calibration. Afterwards, Section 6.6 explains the actual quantitative calibration process. Then, further experimentation is presented in Section 6.7. Finally, Section 6.8 discusses further validation that was performed with a system dynamics model.

6.1 Generic Scenario

This section presents the underlying structure of the scenario that was used as a basis for all scenarios used during the calibration and experimentation. The scenario is illustrated with figures and explained thoroughly. Adaptations for different purposes will be presented throughout this chapter.

Initial Configuration

The basic scenario takes place in front of the parliament (as shown in Figure 5.2) and involves a few hundred agents. Fences are set-up in a trapeze shape in front of the parliament, with police trucks behind the fences. Figure 6.1 shows the three police squads that protect the fences: one at the left end of the fences, one in the middle, and one at the right end. Each squad has its own truck for equipment (the truck is a resource supplier) and assigned deployment points. The squads start in the *Uninvolved* mobilization level (which is detailed in Table 5.1) and they are not allowed to use tear gas or plastic bullets. Specific scenarios will change the mobilization level and NLW rules according to their needs. Crowd members are generated at various locations along the ramparts to resemble a crowd of people arriving from different locations, as shown in Figure 6.2. Specific scenarios will vary the size and composition of the crowd according to their needs. Coloured interests zones have been created so that agents know more about their surroundings. The green zone is for agents who want to observe from far; the yellow zone is for active demonstration; the red zone is for aggressive demonstrators; and the white zone is for CF. The social behaviours expect these zones to exist. For example, angry instigators will seek to go to a high-aggressiveness (red) zone to insult CF.

Crowd Participants and their Actions

Bystanders are not generated anywhere near the ramparts but only near the gates (along St-Louis and Dauphine streets). Other bystanders are generated on Grande Allée Street just west of the parliament (it is a very popular touristic area). The bystanders' main goal is to visit touristic areas (as indicated by interest shapes) and to go home afterwards (leave the simulation). However, if STGs (CF or crowd) are becoming aggressive, the bystanders will stand and observe from the distance. If some STGs become too aggressive, the bystanders will be uncomfortable and will leave the simulation. Also, bystanders leave when asked by CF.

Demonstrators do not perform many actions alone. In fact, as soon as they are created, they start looking for a STG. Once they joined one, they start chanting and showing banners. If the CF are perceived as too pro-active, the demonstrators will move near the fences to yell at and insult the CF. If the CF go into *Crowd Dispersion* mobilization level, the demonstrators will flee and exit the simulation.

Even though demonstrators may turn aggressive, their main goal is to show support for a cause. Sometimes, organized groups will come prepared and will have a leader who is

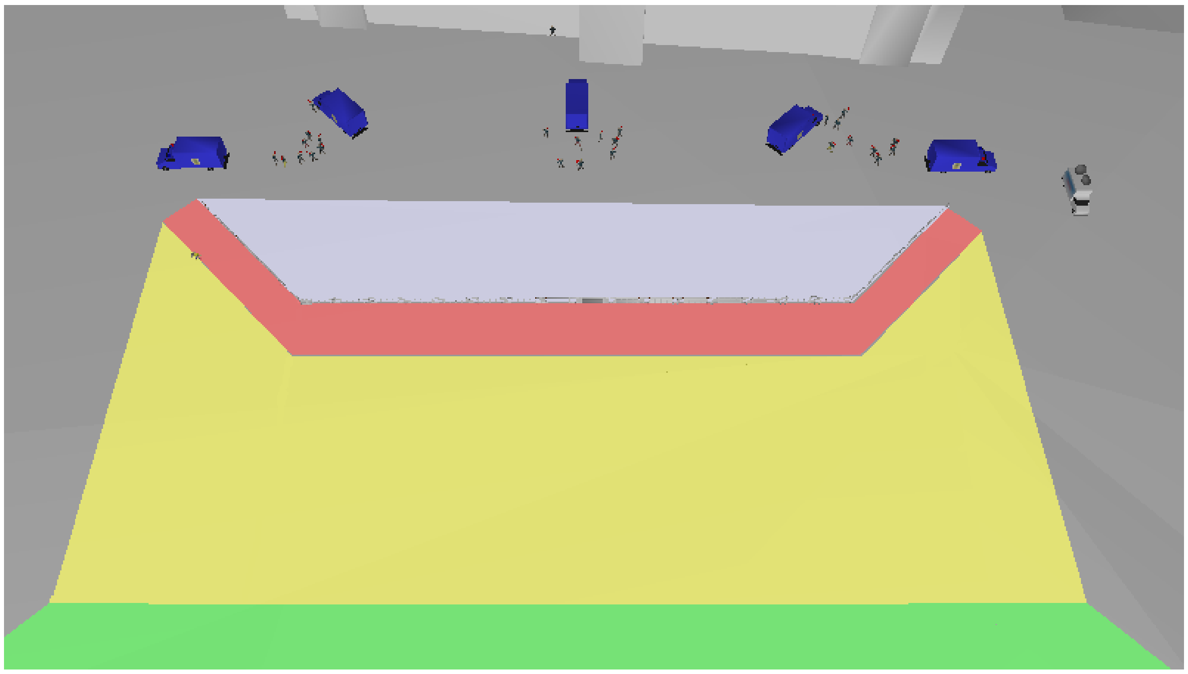


Figure 6.1: Initial Position of Squads in the Generic Scenario

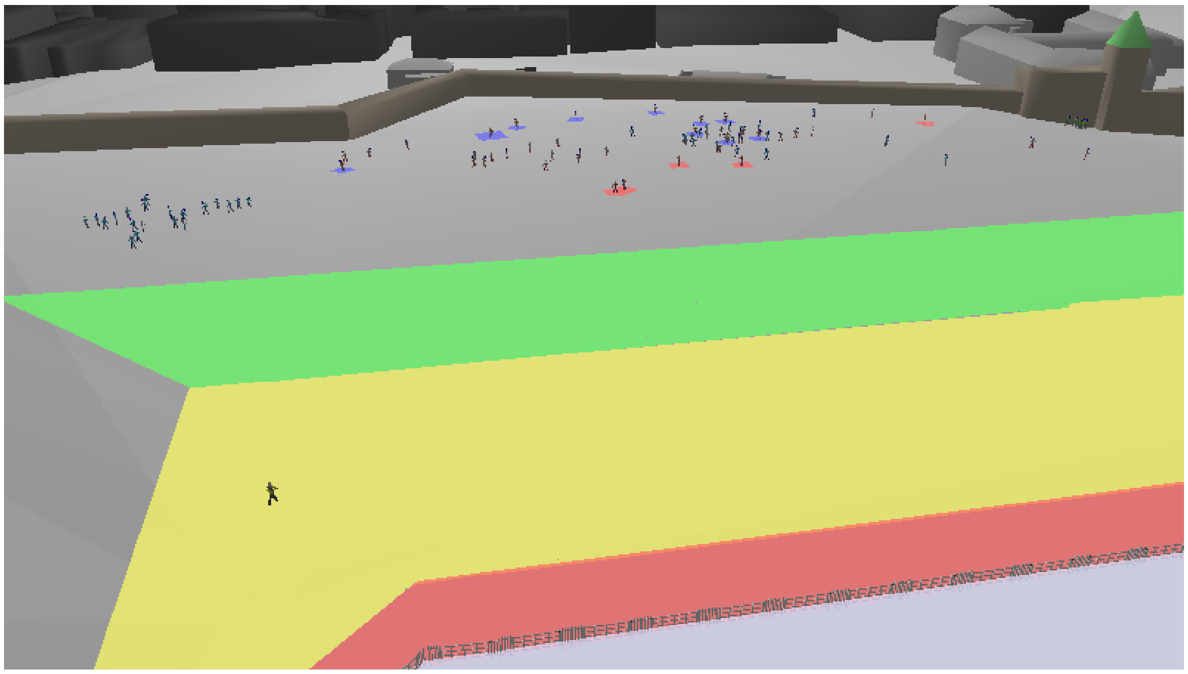


Figure 6.2: Initial Position of the Crowd in the Generic Scenario

responsible for organizing a march, a speech, and controlling his group of demonstrators. Thus, one demonstrator leader is generated on Grande Allée Street near the parliament along with several demonstrators. The leader will wait there to gather more demonstrators. When he has waited long enough, he will start a march in front of the parliament. At his destination, he will address a speech to all participants who followed him. Once completed, he will disband his STG. This action causes new STGs to be created and demonstrators thus continue protesting but in smaller STGs. If the police is perceived as too aggressive during the march or speech, then the demonstrator leader also disbands his STG. Some STGs composed of demonstrators will be formed, and other demonstrators will start protesting on their own, most likely in a more aggressive manner (e.g. in a STG composed of instigators).

Like demonstrators, instigators do not perform many actions alone. Thus, they immediately start searching for a STG. When in one, they start insulting and attacking the CF. If they are close enough to the CF, they will fight; otherwise, they will throw Molotov cocktails if they have some. The instigators will not flee if the CF go into the *Crowd Dispersion* mobilization level.

Instigator leaders are similar to instigators. In fact, they perform no different actions but simply take the lead. Instigator leaders start by trying to gather as many instigators as possible and assault the CF when their STGs are large enough. The STG starts by moving near the fences and then insults the CF. If the CF get into a pro-active mobilization level, the instigator leaders will start pushing the fences and attacking the CF.

6.2 Isolated Calibration

The Crowd-MAGS system allows the customization of a large number of parameters, some of which can be calibrated based on isolated tests and on scientific studies found in the literature. Others, mostly related to crowd characteristics, must be calibrated in plausible test scenarios before being considered acceptable for more complex scenarios.

Many parameters were initially calibrated while the system was being developed. The values were calibrated on the basis of available data and relying on the qualitative assessment of the realism of the effects of each parameter during isolated tests. Section A.4 lists each parameter and its initial value.

The following sections will start from these values and refine them using more complex scenarios. For example, it was initially estimated that agents who flee (e.g. from tear gas) would run 2.5 times faster than their normal walking speed. However, complete scenarios may confirm or infirm the accuracy of this value depending on the aggregate crowd behaviour.

6.3 Qualitative Calibration

After all parameters had been calibrated in isolation, the generic scenario from Section 6.1 needed to be run to ensure that everything is plausible and that no anomalies happened. As expected, a few problems were noted.

First of all, nearly all demonstrators near the ramparts became instigators within a minute and joined instigators STGs. This problem resulted from the fact that no demonstrator leader was generated near the ramparts and consequently the demonstrators had to join instigator groups if they wanted to be in a STG (there were no demonstrators STGs). A simple solution consisted in creating a similar number of demonstrator leaders compared to instigator leaders. Consequently, demonstrators mostly joined to demonstrator STGs and instigators mostly joined instigator STGs, although a mixture was possible depending on the spatial arrangement of the agents.

Next, the spatial distribution of agents was addressed as it was causing more severe problems than simply mixing agents in STGs. The scenario was set-up with several agent generators, each one configured for a specific social identity. Thus, agents appeared in tight "packs" around the (x,y) coordinates of their agent generators. Hence, they had difficulty moving and most importantly joining leaders, who were also stuck in other packs at different locations. The solution was implemented in two steps. First, agent generators were relocated to obtain a more homogeneous crowd. Next, the generation radius of each generator was increased so that agents would not be overcrowded when they get generated. The result was a much more plausible crowd, both qualitatively and quantitatively with respect to the aggressiveness levels.

Additionally, several smaller details were changed to increase the realism of the overall scenario. For example, each CF squad was increased to 9 total members instead of 5. The fences went through an overhaul as their width was increased, their anchoring strengthened, their mass increased, and their friction increased. In result, instigators could no longer push the fences easily to penetrate into the CF zone. Currently, several agents are required to push a fence before it moves. Finally, the squad leader was equipped with an armour and a gas mask when the simulation starts.

At this point, the scenario was ready to be used as a basis for more specific scenarios. Qualitative calibration has been performed, but no conclusions could be made with respect to the quantitative calibration due to the high complexity of the system. The next section will present the criteria that were selected for quantitative calibration. Afterwards, a list of scenarios will be elaborated based on these criteria.

6.4 Selection of Criteria for Quantitative Calibration

Performing qualitative calibration of a crowd simulator is not an easy task, but performing quantitative calibration is much more difficult. Indeed, the large number of parameters and vast amount of data generated by each scenario are impressive. In addition, it is often impossible to link a change in one input value to a measurable impact in scenario output. "In complex systems theory, systems are understood to be emergent, i.e. a small number of rules, applied at a local level and among many entities, are capable of generating counterintuitive complex phenomena, behaviors, and patterns at above individual levels. Often, these are manifested in such a way that the actions of the parts do not simply sum to the activity of the whole." (Benenson and Torrens, 2004) Moreover, it is not even an easy task to identify the key parameters to evaluate the activity of the whole.

Since one of this thesis' goals is to assess the efficiency of intervention strategies and non-lethal weapons, it should be defined what is an efficient strategy. Unfortunately, the project's sponsor (DRDC) was unable to provide an answer to this question. Consequently, multi-criteria decision analysis techniques were used to elicit candidate attributes for evaluating CF interventions. Complete details on the methodology used can be found in (Larochelle, 2008).

First, Keeney's top-down approach (Keeney and Gregory, 2005) was used to identify the fundamental attributes, such as 'Ensuring public safety' and 'Minimizing costs'. Then, these attributes were broken down in categories, sub-categories, and more until the attributes could be clearly measured, such as 'Cost of all tear gas cans thrown'.

In parallel, Roy's bottom-up approach (Roy, 1985) was used to identify consequences of various actions that could happen during the simulation, such as 'Insulting officers' or 'Chanting in a large group'. From all consequences, the elementary ones were identified. Roughly, an elementary consequence is one that is clearly positive or clearly negative, such as 'Improving the CF's image' or 'Hurting a bystander'.

Finally, both lists were combined and inspected keeping Crowd-MAGS' information model in mind. Some attributes could have been useful, but were not available from the system, such as the amount of public property damage. The combination of both lists was a lengthy process, but it partly compensated for the lack of domain knowledge since the sponsor was not able to provide help. In fact, both lists cross-checked each other so the possibility of forgetting a crucial element was reduced, especially since one method is top-down and the other is bottom-up. In the end, a list of criteria that are both available in the system and critical to the assessment of an intervention was created and it is presented below.

- Control forces' intervention level
- Number of people harmed
  - Number of bystanders harmed
  - Number of participants harmed
  - Number of CF officers harmed
- Operational cost
  - Salaries paid to all CF employees
  - Cost of tear gas cans used
  - Cost of plastic bullets fired
- Amount of resources used
  - Number of gas cans used
  - Number of plastic bullets fired
- Crowd size and ratio with respect to initial size (every 15 seconds)
- Crowd aggressiveness (every 15 seconds)

6.5 Conception of Scenarios for Calibration

In order to compare intervention strategies, a list of reasonable strategies was necessary. With the help of DRDC, and in accordance with Crowd-MAGS' models, a list of twelve levels of intervention was established.

- Uninvolved
- Force Demonstration + Communication
- Physical Presence + Tear Gas
- Force Demonstration + Tear Gas
- Physical Presence + Plastic Bullets
- Force Demonstration + Plastic Bullets
- Physical Presence + Tear Gas + Plastic Bullets
- Force Demonstration + Tear Gas + Plastic Bullets
- Crowd Dispersion
- Crowd Dispersion + Tear Gas
- Crowd Dispersion + Plastic Bullets
- Crowd Dispersion + Tear Gas + Plastic Bullets

For the scope of this project, these twelve possibilities will be the only intervention levels that will be used and compared. Nonetheless, each level may be applied in an infinite number of ways, by simply changing the location of the control forces or the amount of non-lethal weapons used. This range of intervention techniques covers well Lorenz's force continuum, shown in Figure 2.17.

As mentioned in (Kenny et al., 2001; Lorenz, 1996), control forces follow a force continuum that is deeply related to the context in which they are involved, which includes crowd composition. Thus, for calibration purposes (at least for the beginning) it was decided that the generic scenario would be used without any changes, except for the crowd composition. Since the crowd social identities have profiles in categories *passive*, *moderate*, and *aggressive*, it is natural to create three similar types of crowds. Thus, each one of the twelve intervention techniques will be matched to three types of crowds (except for the passive crowd that control forces will have to handle smoothly). Table 6.1 shows the 26 calibration scenarios to be used. In each case, the control forces stay uninvolved for the first 30 seconds and may adopt any strategy afterwards. The scenario ends after five minutes because this duration is long enough to evaluate the impact of different strategies.

The three types of crowds contain the same types of agents, but in different proportions. In fact, the total number of crowd members is always 250. Tables 6.2, 6.3, and 6.4 present

Table 6.1: Scenarios Used for Calibration

Control Forces' Strategy	**Crowd**		
	Passive	**Moderate**	**Aggressive**
Uninvolved	1a	1b	1c
Force Demonstration + Communication	2a	2b	2c
Physical Presence + Tear Gas		3b	3c
Force Demonstration + Tear Gas		4b	4c
Physical Presence + Plastic Bullets		5b	5c
Force Demonstration + Plastic Bullets		6b	6c
Physical Presence + Tear Gas + Plastic Bullets		7b	7c
Force Demonstration + Tear Gas + Plastic Bullets		8b	8c
Crowd Dispersion		9b	9c
Crowd Dispersion + Tear Gas		10b	10c
Crowd Dispersion + Plastic Bullets		11b	11c
Crowd Dispersion + Tear Gas + Plastic Bullets		12b	12c

the exact composition of each crowd. It should be noted that only *moderate* profiles have been implemented for demonstrator leaders and instigator leaders. The other values have been selected *ad hoc* in the objective that the three crowds will clearly behave differently. Unfortunately, no such data (detailed at the micro-level) exists in the literature or from empirical studies.

Table 6.2: Initial Crowd Distribution for a Passive Crowd

Social Identity	**Number**	**Passive(%)**	**Moderate(%)**	**Aggressive(%)**
Bystanders	54 (22%)	33	34	33
Demonstrators	170 (68%)	75	15	10
Demonstrator Leaders	5 (2%)	0	100	0
Instigators	16 (6%)	50	25	25
Instigator Leaders	5 (2%)	0	100	0

6.6 Quantitative Calibration

Qualitative calibration, discussed in Section 6.3, ensured a certain level of plausibility with respect to the generic scenario. However, qualitative assessment is certainly not sufficient in the case of crowd control simulation. Quantitative measures must be taken to ensure a higher level of plausibility. Thus, the quantitative calibration process aimed at fine tuning the parameters that could not be calibrated well enough in isolation or *de visu* with the generic

Table 6.3: Initial Crowd Distribution for a Moderate Crowd

Social Identity	**Number**	**Passive(%)**	**Moderate(%)**	**Aggressive(%)**
Bystanders	36 (14%)	33	34	33
Demonstrators	170 (68%)	50	25	25
Demonstrator Leaders	5 (2%)	0	100	0
Instigators	34 (14%)	25	50	25
Instigator Leaders	5 (2%)	0	100	0

Table 6.4: Initial Crowd Distribution for an Aggressive Crowd

Social Identity	**Number**	**Passive(%)**	**Moderate(%)**	**Aggressive(%)**
Bystanders	18 (7%)	33	34	33
Demonstrators	170 (68%)	25	30	45
Demonstrator Leaders	5 (2%)	0	100	0
Instigators	52 (21%)	25	25	50
Instigator Leaders	5 (2%)	0	100	0

scenario. The parameters to be calibrated are the aggressiveness levels of collective actions, the crowd distribution, and the social identity profiles.

This section is organized as follows. The parameter values are presented (in tables and diagrams), as well as the crowd aggressiveness resulting from a scenario. In fact, each scenario was run three times: with a passive crowd, a moderate crowd, and an aggressive crowd. If any implausible results were obtained, the parameters were modified and the scenarios were run once again. This process was executed until satisfactory results were obtained. Next, the scenario was replaced with a more complex one and the evaluation was started over.

It is important to note that this section presents a summary of the calibration process. Although only six trials are presented, many more were carried out. Since many of them were very similar, only the 'distinct' ones are presented here. Others can be found in (Larochelle et al., 2009), but even more have been recorded. *It must be acknowledged that the DRDC team performed a large portion the tests necessary for quantitative calibration.*

Also, all results are based on an average of three simulation runs, since crowd simulation is a stochastic process. Effectively, "a single agent's behavior is essentially nondeterministic at a microscopic level; if the system is executed multiple times with the same initial setting, the agents would not behave exactly the same way each time" (Pan, 2006) Hence, when it is mentioned that Scenario X was run, it was actually run three times and the average of the results is shown.

6.6.1 Initial Settings

This sub-section presents the state of the system as it was originally calibrated.

As seen from the videos that were collected in the early stages of the project, the following collective actions are by far the most common in demonstrations. In addition, (McKenzie et al., 2005b), who organized actions by aggressiveness level like the Crowd-MAGS project, identified these actions as some of the most common and important ones to model (as shown in Table 2.3). These actions are the only ones that were calibrated early as the other actions were too uncommon to have an impact on the overall crowd aggressiveness. The values for the following five actions were originally set considering three main factors: physical harm, psychological harm, and the severity of actions that could follow as a reply. However, it will be shown throughout this section how the values were adjusted so that the global crowd aggressiveness varies in a significant manner between passive and aggressive crowds. Otherwise, passive and aggressive crowds had similar aggressiveness levels.

Table 6.5: Initial Settings for Collective Actions

Collective Action	**Aggressiveness Level**
Chanting	0.15
Showing banner	0.15
Yelling	0.2
Insulting	0.3
Fighting	1

The following three diagrams represent the agent profiles as they were originally calibrated.

Tests were run with Scenario 1 (the first line of Table 6.1) where the control forces stay uninvolved. Table 6.6 and Figure 6.6 show the overall crowd aggressiveness after the full duration of the scenario (5 minutes).

Table 6.6: Initial Results of Crowd Aggressiveness with Scenario 1

Crowd Type	**Aggressiveness Level**
Passive	0.106
Moderate	0.119
Aggressive	0.124

The first surprising aspect of these results is the relatively low aggressiveness level for all crowds. Levels around 0.25 were expected, but none even reached 0.15. In addition, the three curves were very close to each other, which could render analyses more difficult when comparing different intervention strategies for a given scenario. One of the reasons for the low values is that a significant number of bystanders are present in Scenario 1. Their main

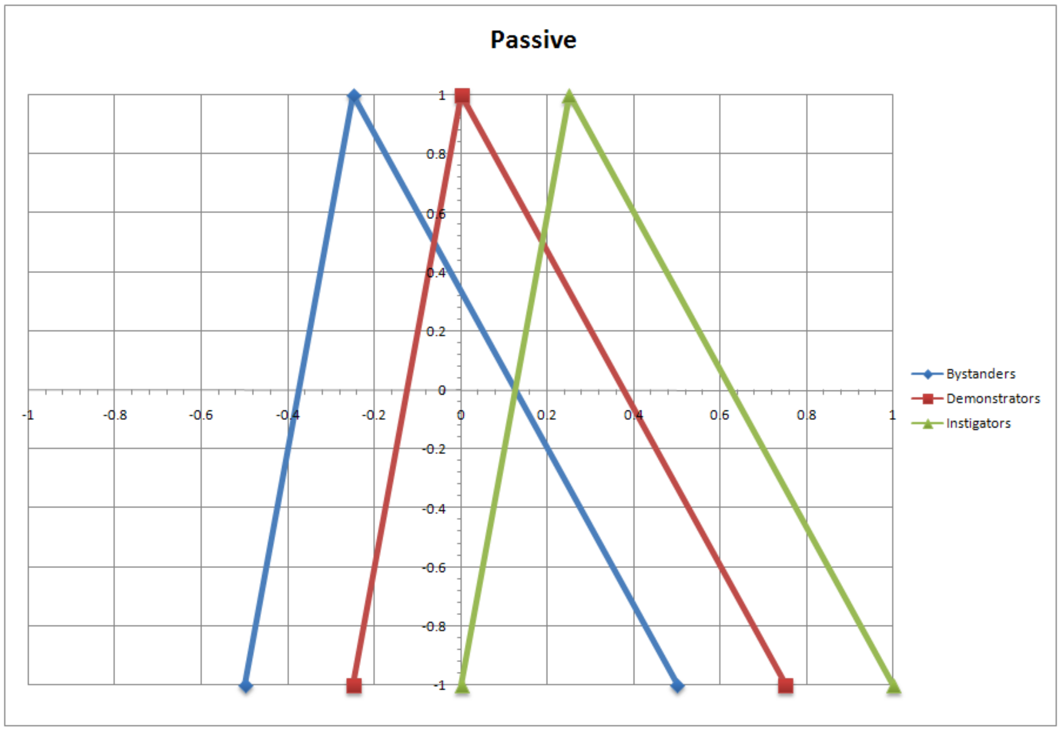


Figure 6.3: Initial Settings for Passive Crowd Members

actions consist of visiting, which is not counted in the aggressiveness level, and observing, which has a level of 0.05. However, all bystanders are counted in the number of crowd agents regardless of their actions, thus reducing the overall crowd aggressiveness compared to other social identities. One explanation for the small distance between the curves is related to the small distance between the most frequent actions which were listed above. Considering that the agents who perform these actions are only a fraction of the total number of agents, the numerical difference between the values appear even weaker in the overall crowd aggressiveness.

The following sub-section presents Trial #1, which tries to increase and differentiate the crowd aggressiveness levels.

6.6.2 Trial #1

The parameters in Table 6.7 were modified to solve the problems found in the previous sub-section.

Scenario 1 was run again and the overall crowd aggressiveness is shown in Table 6.8 and Figure 6.7.

Clearly, the situation here is worse than the initial settings. First of all, the overall aggressiveness decreases with increasing proportions of aggressive agents. This fact is due to

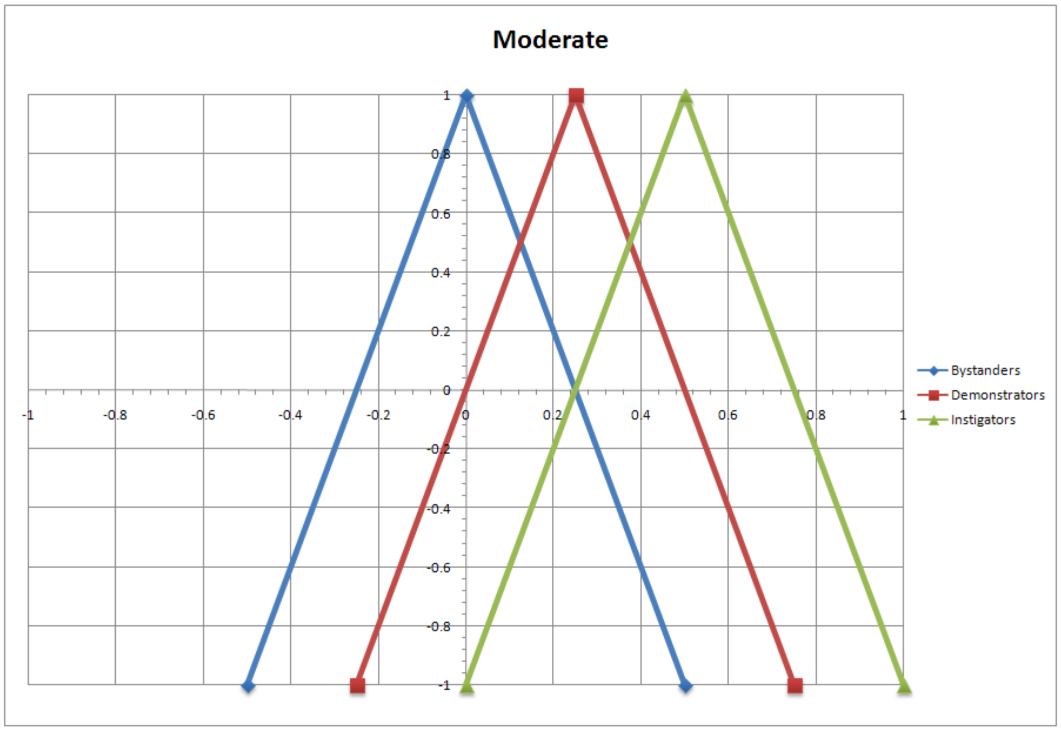


Figure 6.4: Initial Settings for Moderate Crowd Members

Table 6.7: Trial Settings #1 for Collective Actions

Collective Action	**Aggressiveness Level**
Chanting	0.25
Showing banner	0.25
Yelling	0.5
Insulting	0.6
Fighting	0.9

Table 6.8: Trial Results #1 of Crowd Aggressiveness with Scenario 1

Crowd Type	**Aggressiveness Level**
Passive	0.311
Moderate	0.258
Aggressive	0.254

the actions' aggressiveness levels being increased but the agents' profiles not being adjusted in consequence. Hence, bystanders almost always felt like the crowd was too aggressive, so they

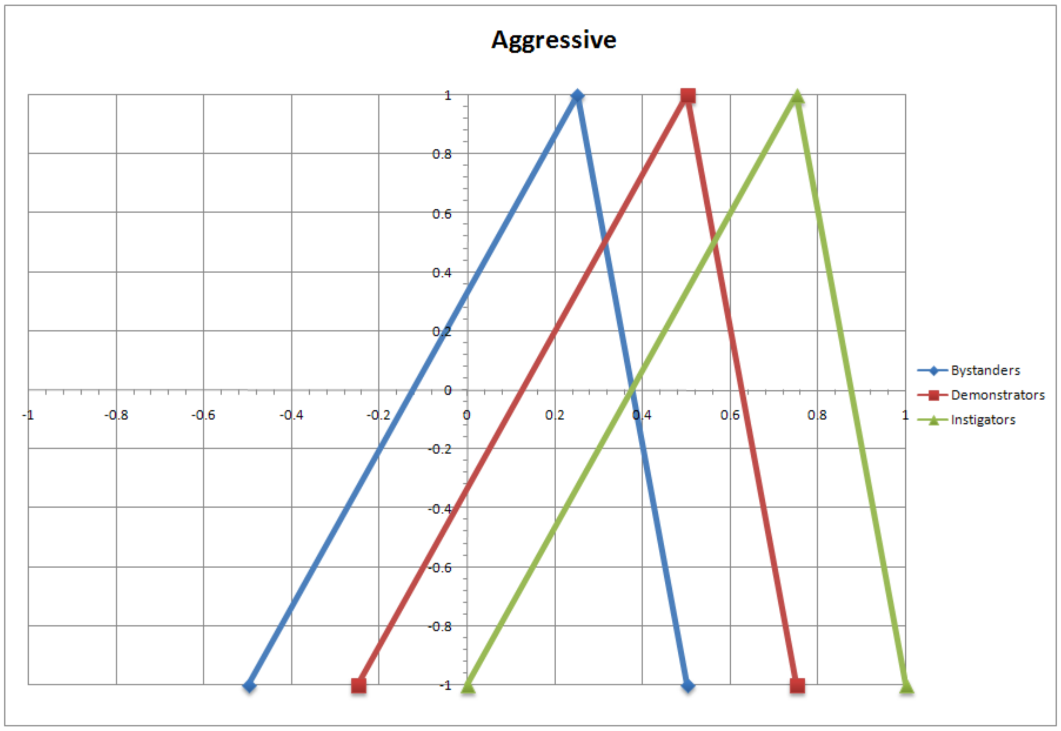


Figure 6.5: Initial Settings for Aggressive Crowd Members

preferred to become demonstrators and instigators. This created a crowd composed almost entirely of instigators. In addition, the more instigators were present, the faster bystanders became instigators. The blue curve on the diagram shows the snowball effect caused by this miscalibration.

The next trial aims at adjusting agents' profiles to reflect the changes in the actions' aggressiveness levels.

6.6.3 Trial #2

The profiles shown in Figure 6.8 were modified to solve the problems found in the previous sub-section. Only the demonstrators were modified because they are the most numerous in the crowd and the other profiles will be adjusted afterwards if necessary. The curves were shifted to the right to reduce the number of demonstrators that become instigators.

Scenario 1 was run again and the overall crowd aggressiveness is shown in Table 6.9 and Figure 6.9.

These results, which show a clear difference between the three types of crowds, seem much more plausible. Since Scenario 1 includes control forces that are uninvolved, it is safe to assume that the behaviour of the crowd (without external actions or events) is calibrated well enough.

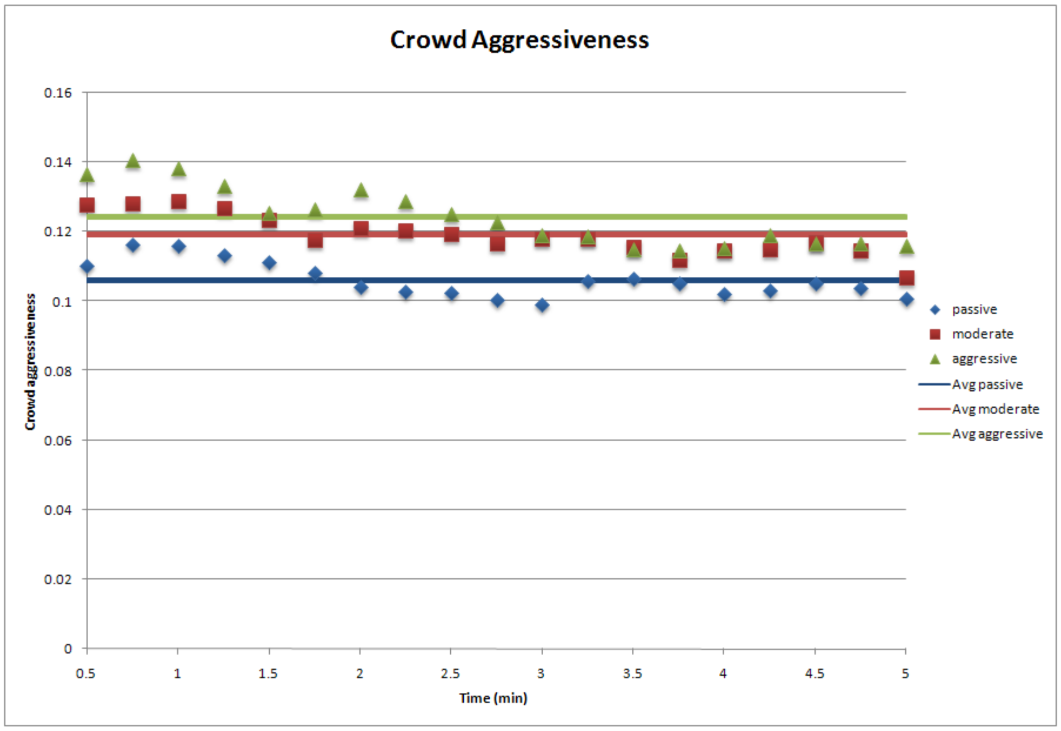


Figure 6.6: Initial Results of Crowd Aggressiveness with Scenario 1

Table 6.9: Trial Results #2 of Crowd Aggressiveness with Scenario 1

Crowd Type	**Aggressiveness Level**
Passive	0.193
Moderate	0.213
Aggressive	0.233

Next, Scenario 2 will be run. It involves the control forces changing to the *Force Demonstration* mobilization level after 30 seconds and asking the crowd to disperse. These tests could show that the crowd reacts as expected to the CF intervention or it could show glitches that need to be fixed. The actual results are shown in Table 6.10 and in Figure 6.10.

Table 6.10: Trial Results #2 of Crowd Aggressiveness with Scenario 2

Crowd Type	**Aggressiveness Level**
Passive	0.322
Moderate	0.271
Aggressive	0.297

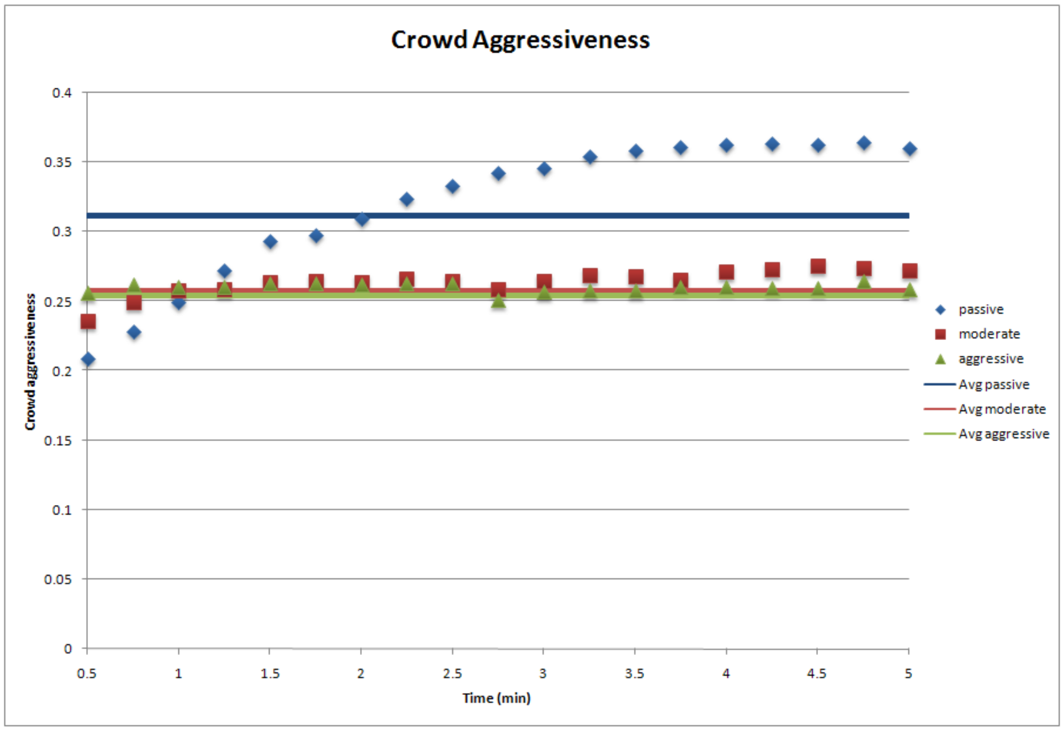


Figure 6.7: Trial Results #1 of Crowd Aggressiveness with Scenario 1

Once again, the passive crowd ends up being more aggressive than the others. This result is probably due to the bystanders changing directly into instigators (skipping the demonstrator step) because the control forces antagonized the crowd so much that it became highly aggressive. The next sub-section will try to adjust the agents' profiles that were not modified before.

6.6.4 Trial #3

The profiles in the following three diagrams were modified to solve the problems found in the previous sub-section. The curves were mostly shifted to the right.

The parameter values in Table 6.11 were modified to increase the difference between the aggressiveness of the crowds.

In addition, *adhesionSTGMax*, introduced in Sub-Section 4.5.6, was increased from 0.6 to 0.8 to prevent agents from splitting STGs too frequently. This problem was observed qualitatively although it is not clear that it has a direct impact on the quantitative analysis of scenarios. Tests with Scenario 2 returned the results shown in Table 6.12 and Figure 6.14.

Here, the passive crowd is clearly below the other ones. However, the three crowds are still very close. After having calibrated all obvious parameters, it was observed that the crowd compositions were too similar to clearly exhibit distinct aggressiveness levels.

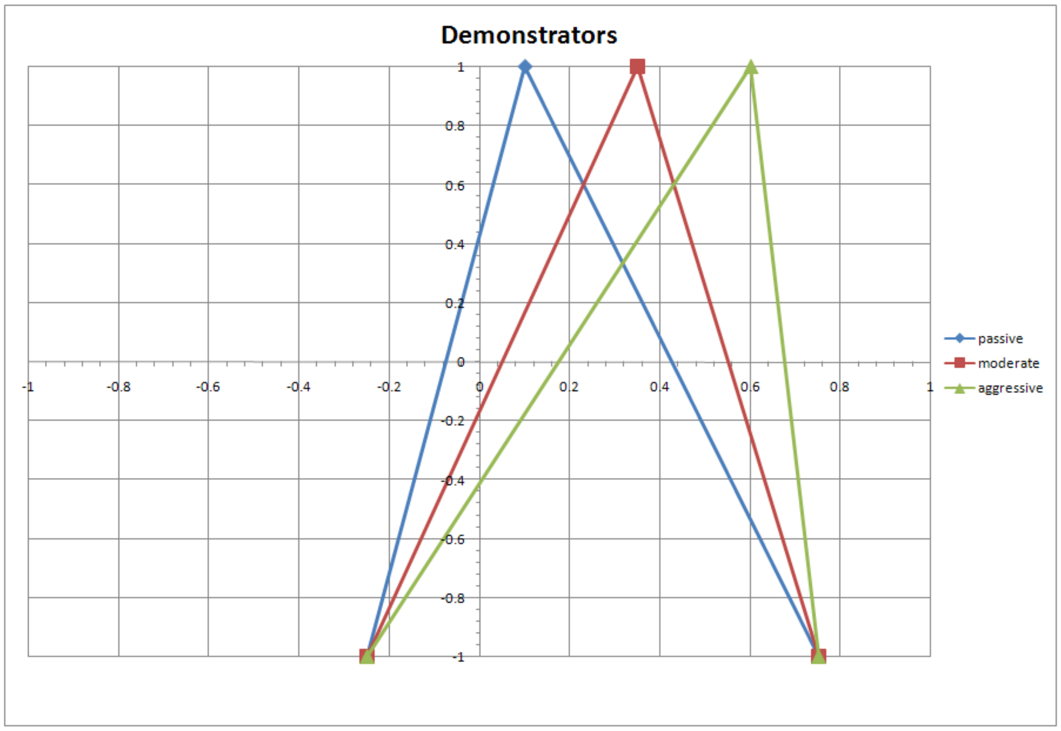


Figure 6.8: Trial Settings #2 for Demonstrators

Table 6.11: Trial Settings #3 for Collective Actions

Collective Action	**Aggressiveness Level**
Chanting	0.15
Showing banner	0.2
Yelling	0.4
Insulting	0.7

Table 6.12: Trial Results #3 of Crowd Aggressiveness with Scenario 2

Crowd Type	**Aggressiveness Level**
Passive	0.253
Moderate	0.279
Aggressive	0.287

The next trial aims at differentiating the crowds with respect to the number of agents in each social identity.

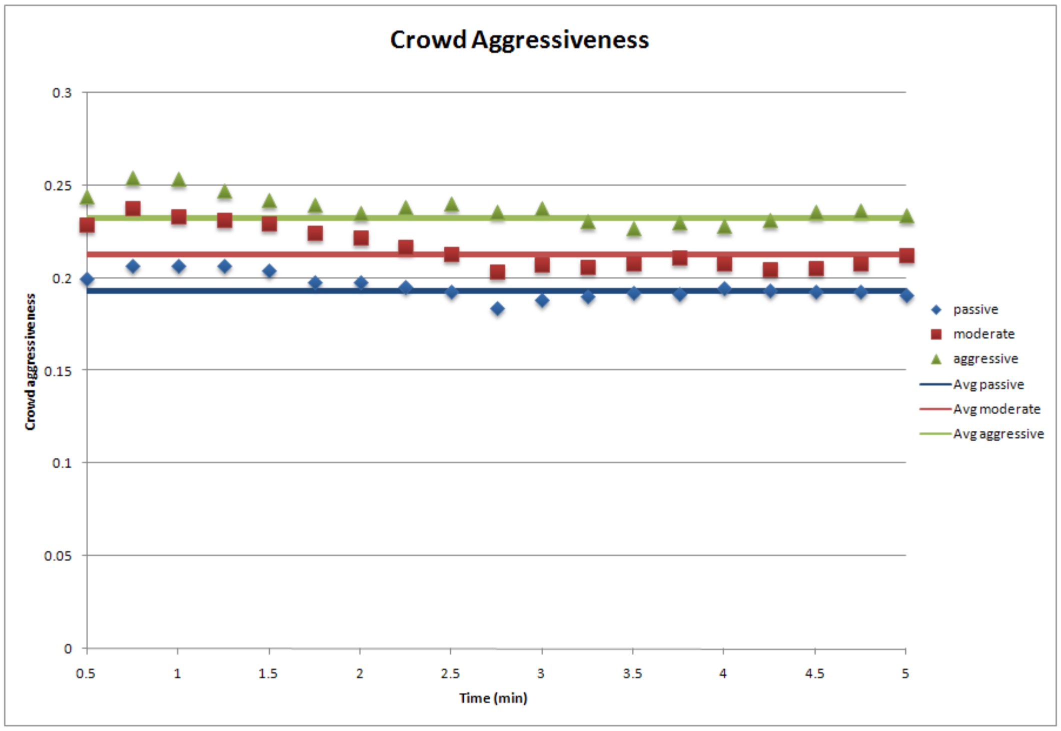


Figure 6.9: Trial Results #2 of Crowd Aggressiveness with Scenario 1

6.6.5 Trial #4

Crowd compositions were modified to solve the problems found in the previous sub-sections and they are shown in Tables 6.13, 6.14, and 6.15.

Table 6.13: Trial Settings #4 for the Passive Crowd Distribution

Social Identity	**Number**
Bystanders	63 (25%)
Demonstrators	170 (68%)
Demonstrator Leaders	5 (2%)
Instigators	9 (4%)
Instigator Leaders	3 (1%)

Tests were run on Scenario 2 and they returned the results shown in Table 6.16 and Figure 6.15.

With these settings, the passive crowd is clearly less aggressive than the others but it is practically impossible to distinguish between the other two. In fact, the aggressiveness of the aggressive crowd drops significantly four minutes into the scenario. The value even decreases below the passive crowd's level. This phenomenon is very difficult to explain, since to no tear

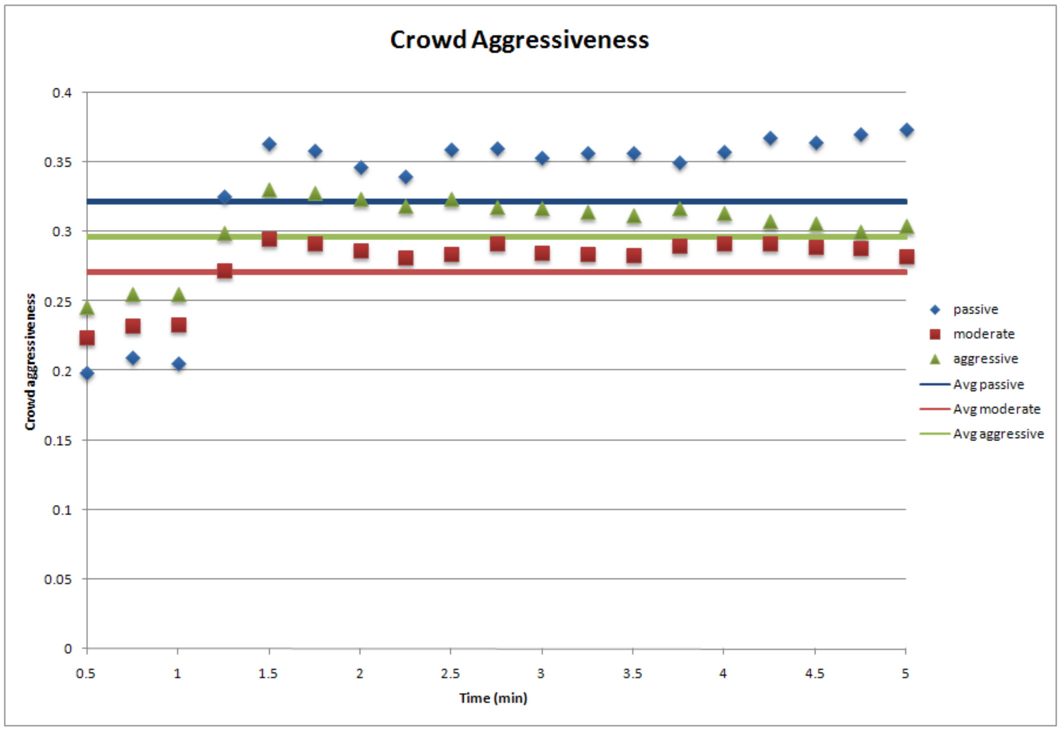


Figure 6.10: Trial Results #2 of Crowd Aggressiveness with Scenario 2

Table 6.14: Trial Settings #4 for the Moderate Crowd Distribution

Social Identity	**Number**
Bystanders	42 (17%)
Demonstrators	170 (68%)
Demonstrator Leaders	5 (2%)
Instigators	28 (11%)
Instigator Leaders	5 (2%)

Table 6.15: Trial Settings #4 for the Aggressive Crowd Distribution

Social Identity	**Number**
Bystanders	9 (4%)
Demonstrators	170 (68%)
Demonstrator Leaders	5 (2%)
Instigators	58 (23%)
Instigator Leaders	8 (3%)

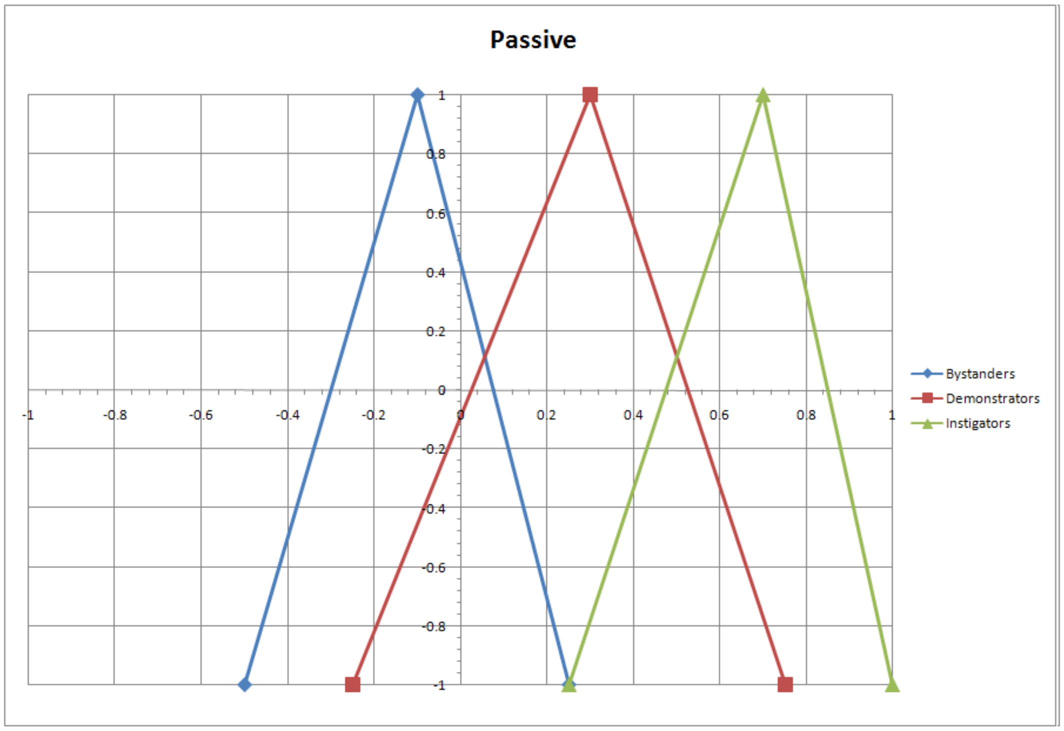


Figure 6.11: Trial Settings #3 for Passive Crowd Members

Table 6.16: Trial Results #4 of Crowd Aggressiveness with Scenario 2

Crowd Type	**Aggressiveness Level**
Passive	0.230
Moderate	0.272
Aggressive	0.277

gas or plastic bullets were used. In fact, in all three cases, only two or three crowd members were injured enough to leave. More tests would be required to ascertain the exact cause(s) of this situation.

The next trial will aim to correct the situation by adjusting the number of agents in each social identity for the aggressive crowd.

6.6.6 Trial #5

The crowd composition for the aggressive crowd was modified to solve the problems found in the previous sub-section and the new values are shown in Table 6.17.

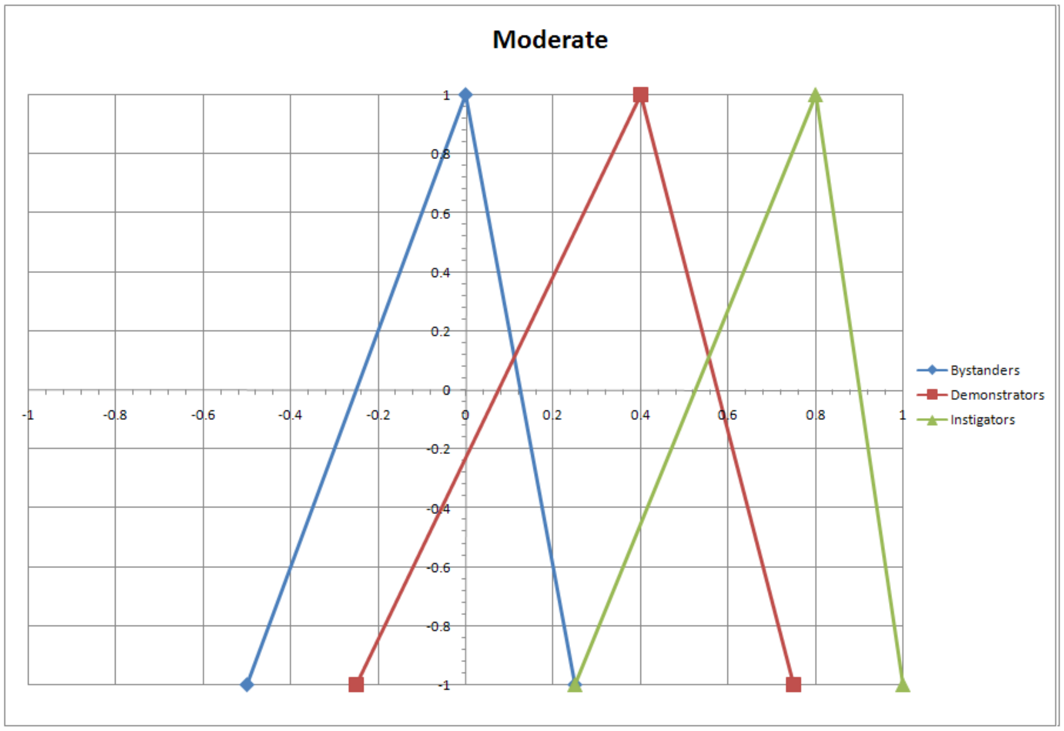


Figure 6.12: Trial Settings #3 for Moderate Crowd Members

Table 6.17: Trial Settings #5 for the Aggressive Crowd Distribution

Social Identity	**Number**
Bystanders	9 (4%)
Demonstrators	90 (36%)
Demonstrator Leaders	5 (2%)
Instigators	138 (55%)
Instigator Leaders	8 (3%)

Scenario 2 was run again and even though the number of instigators in the aggressive crowd is now much higher, the overall aggressiveness level has not increased much compared to settings #4. The suspected cause is that the demonstrators that used to become instigators now start as instigators, making the two scenarios highly similar. Thus, settings #5 will be reverted and another strategy will be tried. In addition to adjusting the number of agents for each social identity, the probability distribution of profiles will be calibrated.

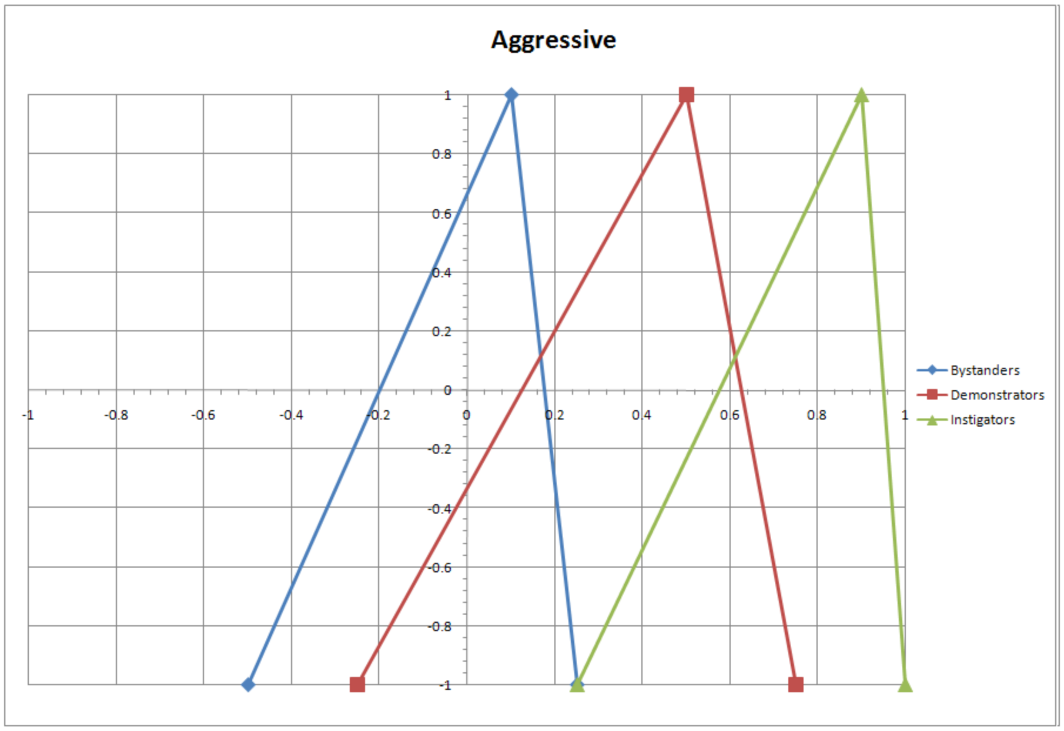


Figure 6.13: Trial Settings #3 for Aggressive Crowd Members

6.6.7 Trial #6

The crowd compositions were modified to solve the problems found in the previous sub-sections and they are shown in Tables 6.18, 6.19, and 6.20.

Table 6.18: Trial Settings #6 for the Passive Crowd Distribution

Social Identity	**Number**	**Passive(%)**	**Moderate(%)**	**Aggressive(%)**
Bystanders	83 (33%)	33	34	33
Demonstrators	150 (60%)	75	15	10
Demonstrator Leaders	5 (2%)	0	100	0
Instigators	9 (4%)	50	25	25
Instigator Leaders	3 (1%)	0	100	0

Tests with Scenario 2 returned the results shown in Table 6.21 and Figure 6.16.

These results show problems similar to those encountered in the first few trials, especially near the end of the scenario with the passive crowd becoming more aggressive and the aggressive becoming more passive. However, the average aggressiveness levels between the three crowds are now clearly different.

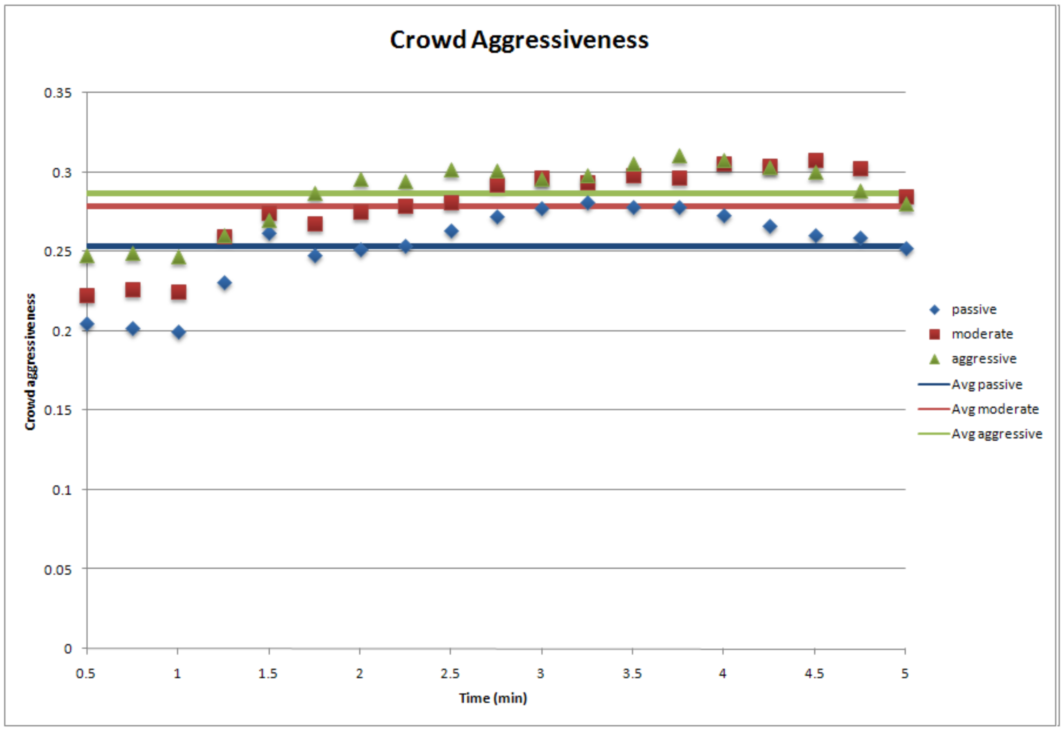


Figure 6.14: Trial Results #3 of Crowd Aggressiveness with Scenario 2

Table 6.19: Trial Settings #6 for the Moderate Crowd Distribution

Social Identity	**Number**	**Passive(%)**	**Moderate(%)**	**Aggressive(%)**
Bystanders	52 (21%)	33	34	33
Demonstrators	150 (60%)	50	25	25
Demonstrator Leaders	5 (2%)	0	100	0
Instigators	38 (15%)	25	50	25
Instigator Leaders	5 (2%)	0	100	0

Table 6.20: Trial Settings #6 for the Aggressive Crowd Distribution

Social Identity	**Number**	**Passive(%)**	**Moderate(%)**	**Aggressive(%)**
Bystanders	9 (4%)	33	34	33
Demonstrators	150 (60%)	15	20	65
Demonstrator Leaders	5 (2%)	0	100	0
Instigators	78 (31%)	25	25	50
Instigator Leaders	8 (3%)	0	100	0

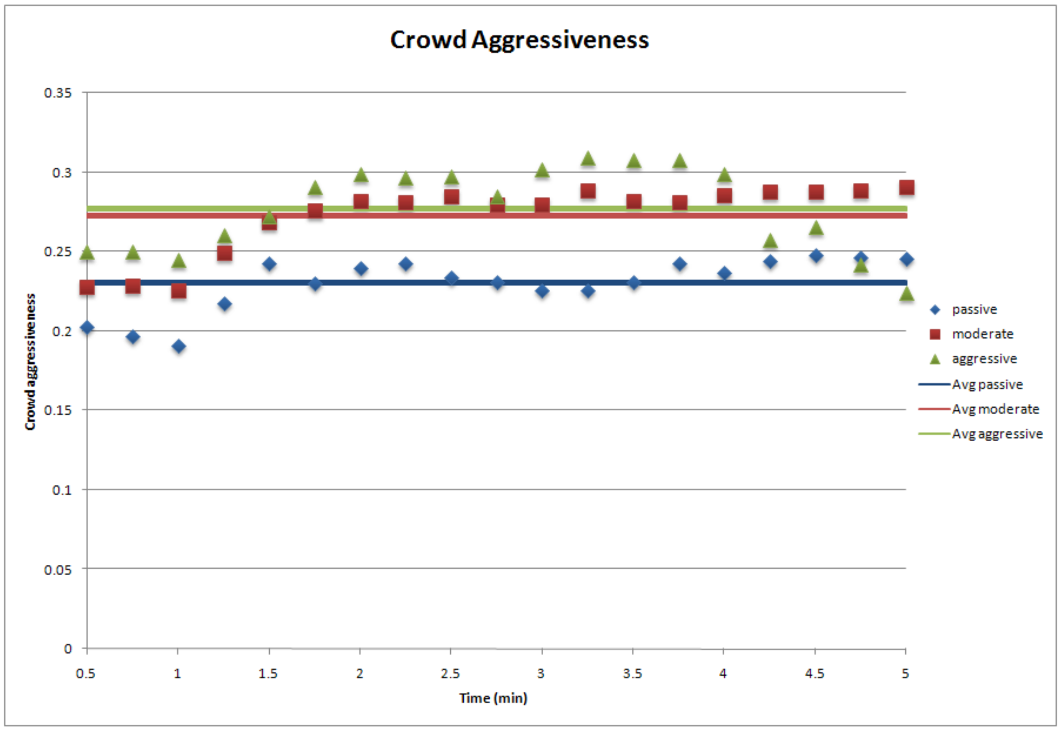


Figure 6.15: Trial Results #4 of Crowd Aggressiveness with Scenario 2

Table 6.21: Trial Results #6 of Crowd Aggressiveness with Scenario 2

Crowd Type	**Aggressiveness Level**
Passive	0.248
Moderate	0.275
Aggressive	0.301

At this point, all parameters that seem influential have been adjusted. It seems improbable that further tests with Scenario 2 would yield to a better calibration. Thus, the next subsection will continue the calibration process by using the current settings on more complex scenarios.

6.6.8 Final Settings

This section uses the best settings found yet (settings #6) and tries them on more complex scenarios to see if further calibration is required.

All twelve intervention strategies were tried with all types of crowds (when applicable) to cover the 26 scenarios listed in Table 6.1. The results showed that all output data seemed

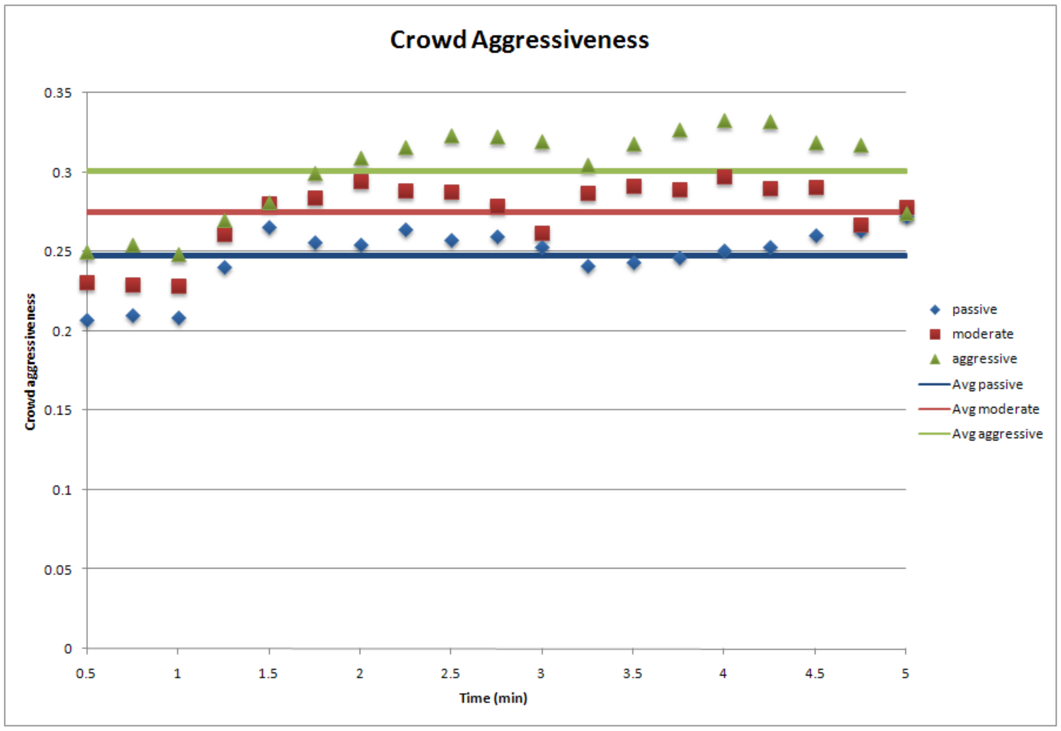


Figure 6.16: Trial Results #6 of Crowd Aggressiveness with Scenario 2

normal, as the aggressiveness level of the aggressive crowd was always the highest and the aggressiveness level of the passive crowd was always the lowest. In fact, the difference between all curves was always at least 0.02. This observation was desirable because it distinguishes the crowds. A difference of 0.05 would be desirable, but all attempts that succeeded in obtaining such a distinction between crowds were not well calibrated overall.

Consequently, the final settings are presented in Tables 6.22, 6.23, 6.24, 6.25 and Figures 6.17, 6.18, 6.19.

Table 6.22: Initial Settings for Collective Actions

Collective Action	**Aggressiveness Level**
Chanting	0.15
Showing banner	0.2
Yelling	0.4
Insulting	0.7
Fighting	0.9

Table 6.23: Final Crowd Distribution for a Passive Crowd

Social Identity	**Number**	**Passive(%)**	**Moderate(%)**	**Aggressive(%)**
Bystanders	83 (33%)	33	34	33
Demonstrators	150 (60%)	75	15	10
Demonstrator Leaders	5 (2%)	0	100	0
Instigators	9 (4%)	50	25	25
Instigator Leaders	3 (1%)	0	100	0

Table 6.24: Final Crowd Distribution for a Moderate Crowd

Social Identity	**Number**	**Passive(%)**	**Moderate(%)**	**Aggressive(%)**
Bystanders	52 (21%)	33	34	33
Demonstrators	150 (60%)	50	25	25
Demonstrator Leaders	5 (2%)	0	100	0
Instigators	38 (15%)	25	50	25
Instigator Leaders	5 (2%)	0	100	0

Table 6.25: Final Crowd Distribution for an Aggressive Crowd

Social Identity	**Number**	**Passive(%)**	**Moderate(%)**	**Aggressive(%)**
Bystanders	9 (4%)	33	34	33
Demonstrators	150 (60%)	15	20	65
Demonstrator Leaders	5 (2%)	0	100	0
Instigators	78 (31%)	25	25	50
Instigator Leaders	8 (3%)	0	100	0

6.6.9 Observations

This sub-section points out remarks that were noticed during calibration.

Choice of Variables for Calibration

It should be noted that even though calibration focused on aggressiveness levels, several other variables could have been used. In fact, although not presented in this thesis, the crowd sizes, number of injured agents, and other variables were kept in mind while calibrating the system. Because of the complexity of calibrating with multiple variables, it was decided to start by

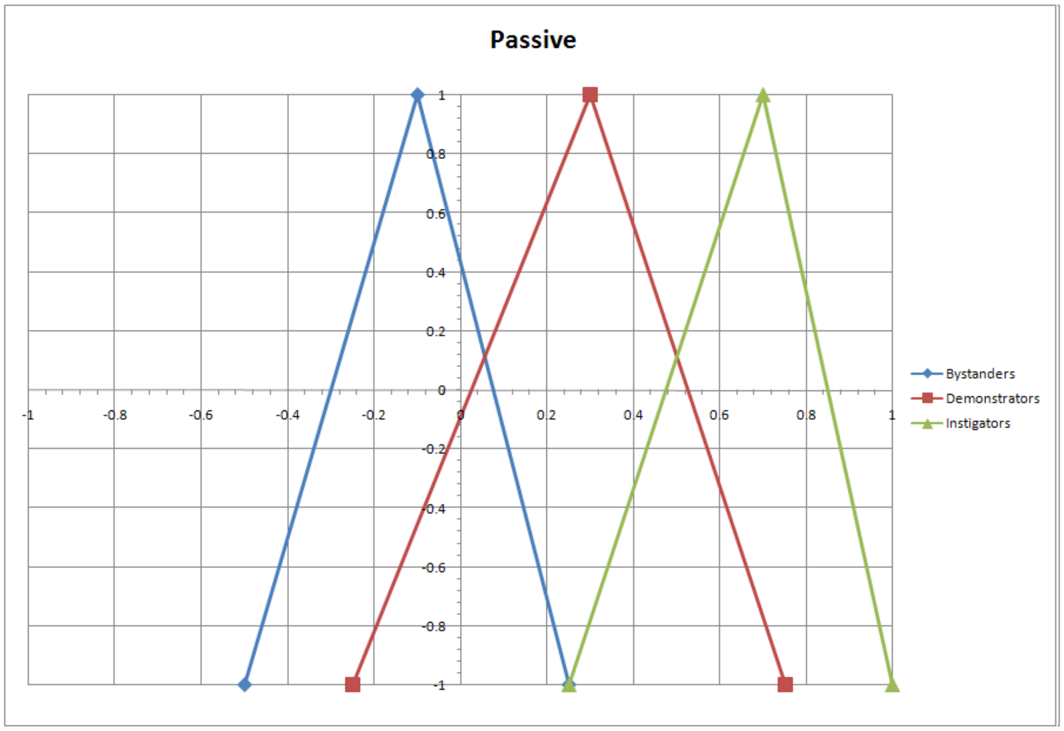


Figure 6.17: Final Settings for Passive Crowd Members

calibrating with aggressiveness levels, and to refine further with other variables afterwards. However, other variables may require empirical data. For example, it is impossible to claim that a passive crowd should diminish in size faster than an aggressive crowd, since many factors could come into play. For example, more prompt and aggressive action from control forces against an aggressive crowd could have led this one to disperse more quickly compared to a passive crowd. For this reason, aggressiveness level was selected as the key variable for calibration, since it is natural that an aggressive crowd exhibits a higher aggressiveness level than a passive crowd.

Final Size in Moderate vs. Aggressive Crowds

The number of agents remaining at the end of the scenario was logged for all tests. It was observed that in aggressive crowds, fewer agents remain than in moderate crowds — in some scenarios. This fact is due to the control forces, who use more tear gas and plastic bullets on aggressive crowds. This technique leads to more agents fleeing or getting harmed, and thus fewer agents remain at the end of the scenario.

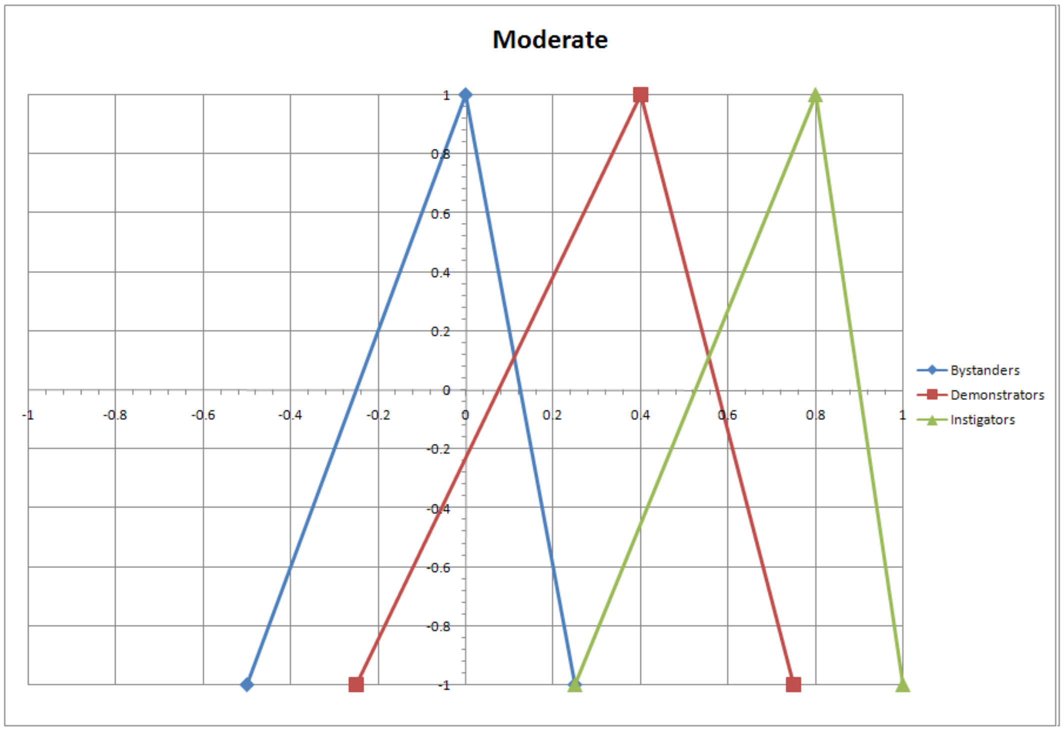


Figure 6.18: Final Settings for Moderate Crowd Members

Effectiveness of Tear Gas vs. Plastic Bullets

It was observed that the use of plastic bullets has a larger impact on the average crowd aggressiveness than tear gas does. Table 6.26 shows an example of two scenarios where the control forces are in *Physical Presence* and the only difference is the allowed non-lethal weapon. The values show the average crowd aggressiveness levels over the entire duration of the scenario.

Table 6.26: Effectiveness of Tear Gas vs. Plastic Bullets

	Average Crowd Aggressiveness	
	Moderate Crowd	Aggressive Crowd
Tear Gas	0.235	0.259
Plastic Bullets	0.175	0.217

6.7 Experimentation

Once the system is well calibrated, it is possible to perform experimentation in order to increase domain understanding and to analyse phenomena of interest. This section presents

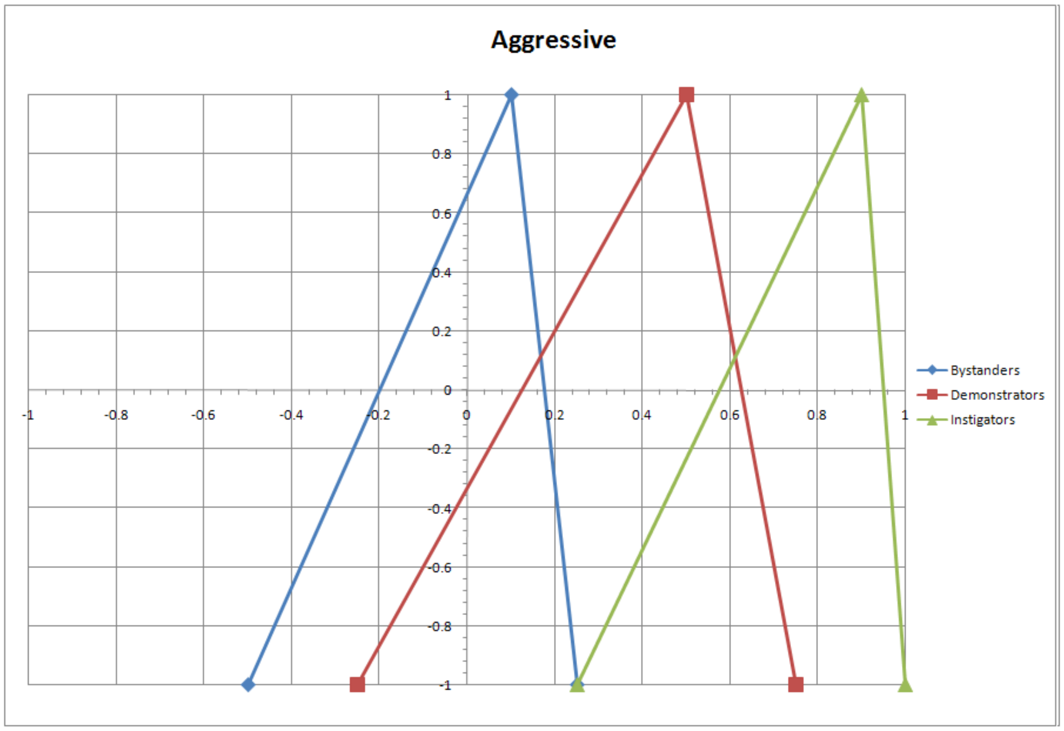


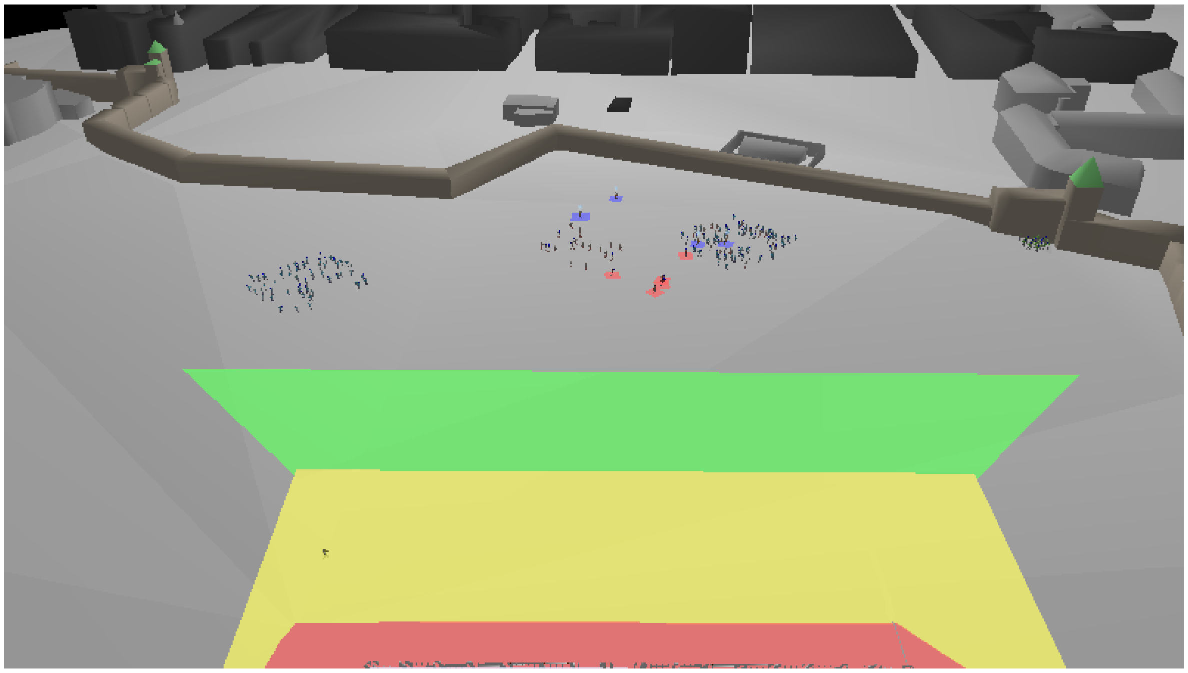
Figure 6.19: Final Settings for Aggressive Crowd Members

four experiments that were conducted in this purpose. More specifically, it was observed that the initial spatial distribution of agents in the environment, the number of agents, and the time at which agents arrive all have a significant impact on the crowd aggressiveness. Moreover, the number of leaders does not have an impact. This section is exploratory and thus these factors should be studied further before designing more complex scenarios.

6.7.1 Initial Spatial Distribution of Agents

This section investigates the effects of the initial spatial distribution of agents on the average crowd aggressiveness. More specifically, two agent generators used for the creation of demonstrators were moved from coordinates (-50, 280) and (25, 225) to (-120, 280) and (50, 165) while a generator for demonstrator leaders was moved from (30, 230) to (-120, 230). The change is better explained through Figure 6.20.

This new configuration was run on Scenario 2b (no tear gas or plastic bullets). The original average crowd aggressiveness was 0.275 after 5 minutes and the revised number is 0.259. Thus, there was a non-negligible impact of the spatial distribution. A qualitative difference between the two settings resulted in the crowd being more uniformly distributed along the fences after a few minutes of protesting. Figure 6.21 illustrates this fact.


a) Initial Settings

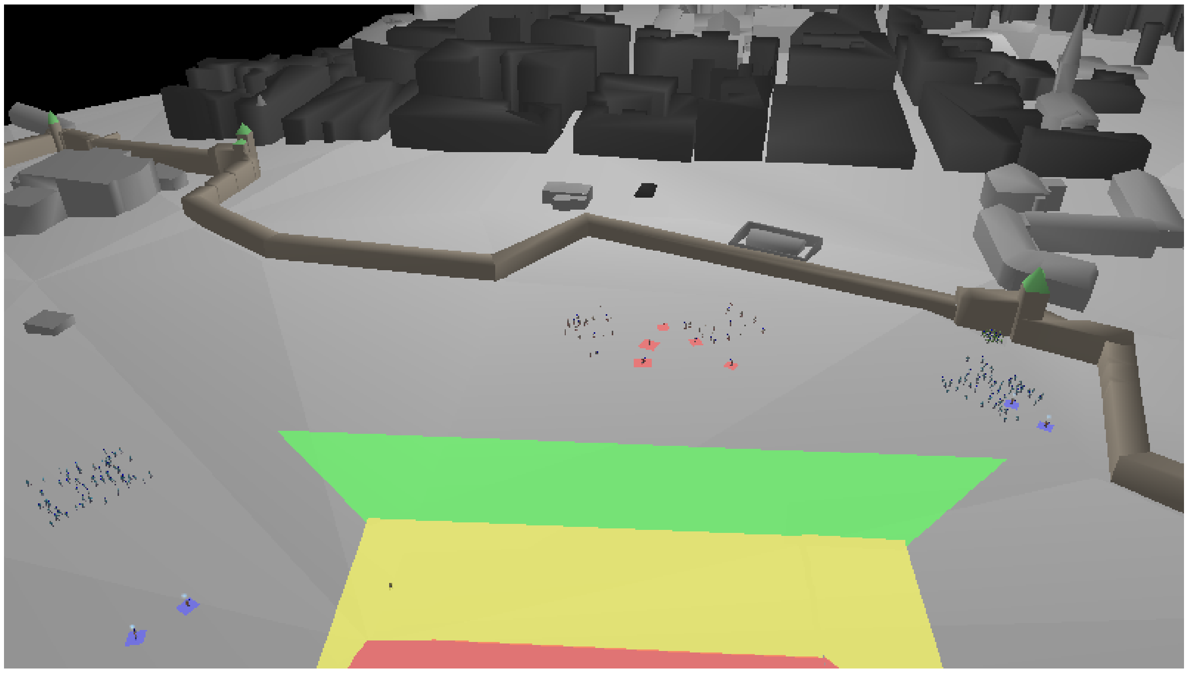
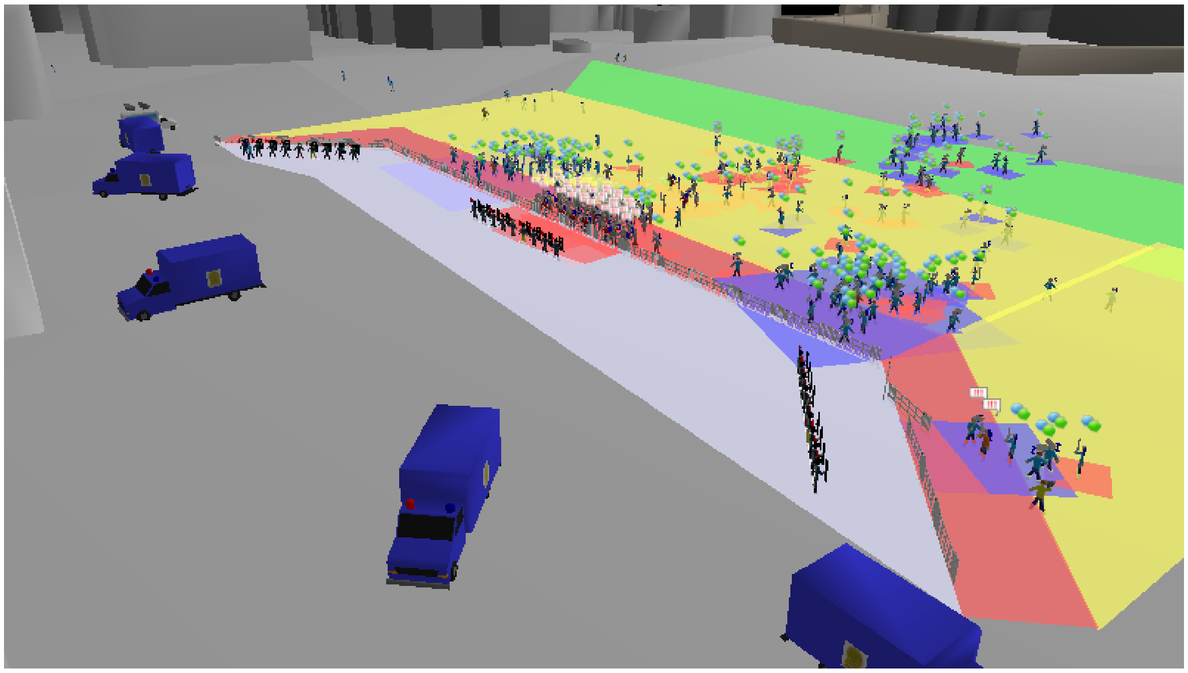

b) Revised Settings

Figure 6.20: Initial Spatial Distribution of Crowd Agents


a) Initial Settings

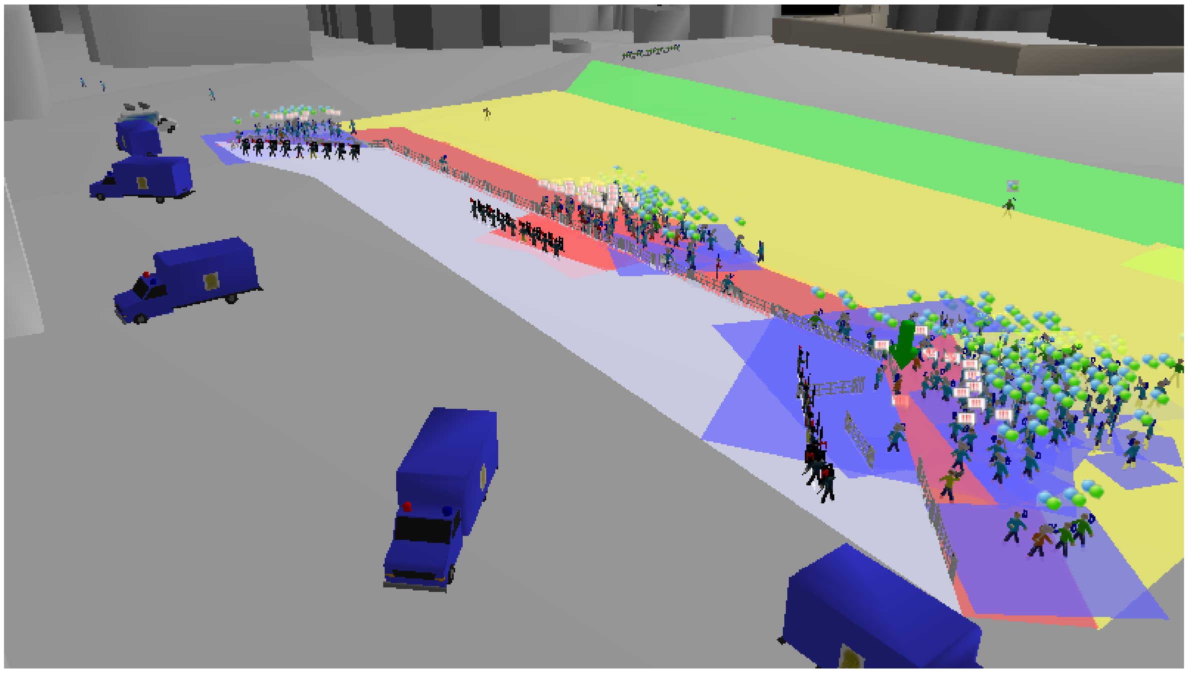

b) Revised Settings

Figure 6.21: Final Spatial Distribution of Crowd Agents

6.7.2 Initial Crowd Size

This section investigates the effects of the initial crowd size on the average crowd aggressiveness. Starting from the final settings of Sub-Section 6.6.8, larger crowds were created while keeping proportions. Crowds of 500 and 750 members were created and Scenarios 2a, 2b, and 2c were run. The results are presented in Table 6.27.

Table 6.27: Effect of Crowd Size on Average Crowd Aggressiveness

	Average Crowd Aggressiveness		
	250 Agents	500 Agents	750 Agents
Passive Crowd	0.248	0.241	0.239
Moderate Crowd	0.275	0.258	0.235
Aggressive Crowd	0.301	0.265	0.247

It is clear from these results that the difference in aggressiveness level between types of crowds gets smaller as crowds grow in size. This observation is difficult to explain and hence more exploration is required. One hypothesis is that among large crowds, a considerable number of agents cannot get close to the fence and possibly cannot see the control forces because other agents block their sight. In this case, these agents in the background tend to bring the aggressiveness level towards an average between the three crowds.

6.7.3 Agent Arrival Times

This section investigates the effects of the rate at which crowd members arrive at the protest (the time at which they appear in the simulation). So far, all scenarios used for calibration and experimentation populated the environment with all agents at once, so the crowds started with a large number of agents and decreased monotonically. In this section, agent generators are modified so that demonstrators arrive progressively. More specifically, 10 demonstrators will be created at each generator every 10 seconds, instead of being created at time 0. The total number of demonstrators that will be created will not change. The results are presented in Table 6.28 for Scenarios 2a, 2b, and 2c.

Table 6.28: Effect of Arrival Time on Average Crowd Aggressiveness

	Average Crowd Aggressiveness	
	Batch Arrival	Progressive Arrival
Passive Crowd	0.248	0.249
Moderate Crowd	0.275	0.248
Aggressive Crowd	0.301	0.267

These results show, like in the previous experimentation, that the changes reduced the difference in aggressiveness between the three types of crowd. This observation is difficult to explain and hence more exploration is required. One hypothesis is that the crowd leaders move towards the fence as soon as they can, and the other agents who arrive in the crowd later might not see any leader and thus might just stay in the back, reducing the overall crowd aggressiveness.

6.7.4 Number of Crowd Leaders

This section investigates the effects of the number of crowd leaders on the average crowd aggressiveness. Two situations have been tested: the first one where the control forces stay uninvolved and the second one where they get into the *Crowd Dispersion* mobilization level and use tear gas as well as plastic bullets. In each one of these situations, a scenario with a moderate crowd (using the calibrated settings) was run. In addition, a similar scenario with twice as many leaders (and fewer bystanders) was run. The results are shown in Figure 6.22.

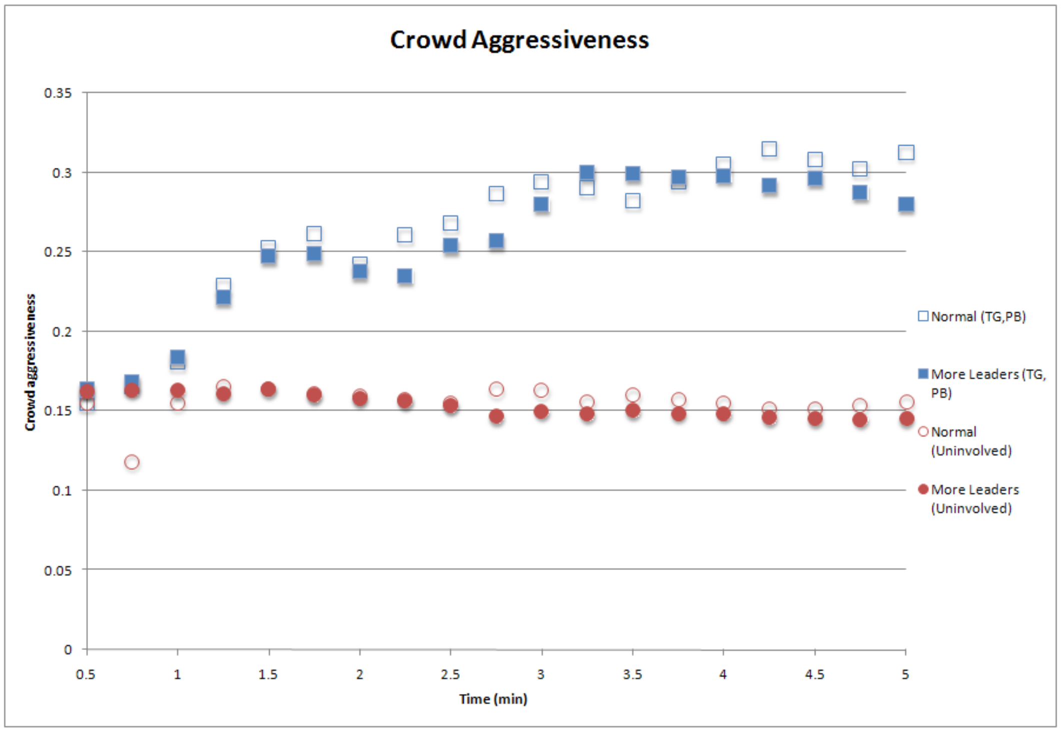


Figure 6.22: Crowd Aggressiveness with More Crowd Leaders

In contrast to what could be expected, the number of leaders in the crowd did not affect the overall crowd aggressiveness. A few reasons could explain the phenomenon. First, leaders perform the same collective actions as regular members, thus their aggressiveness level is not higher than other agents. Next, no behaviour has been developed for control forces to target specifically leaders; hence, they are treated in the same manner as the other agents. Also, it is possible that the extra leaders simply reduce the average number of agents per STG, without affecting the number of collective actions being performed.

6.7.5 Conclusions

This experimentation phase showed that the Crowd-MAGS system is flexible enough to enable an analyst to adjust the numerous parameters that influence a crowd simulation in order to carry out a large number of experiments by simply changing scenarios and collecting simulation results. Further work is needed, but all of the tools and techniques are now available and well-tested to speed up the experimentation process.

Some of the experimental results have been surprising and difficult to explain. Indeed, these experiments require a deep understanding of crowd dynamics, which is usually mastered by domain experts. Thus, it is recommended that further experimentation be done by these experts rather than by the system developers.

6.8 Validation with a System Dynamics Model

Validating crowd simulations is a difficult task (Min et al., 2007; Sargent, 2007). However, validating crowd control simulations is much more complex (Min et al., 2007). In effect, public safety is at stake in the situations to be analyzed so no 'real system' is available — unless a real crowd event arises. (Shi and Brooks, 2007) mentions that "in the absence of a specific real system, validation can only consist of a subjective assessment of the plausibility of the model structure and of the responses". In the light of such statements, one of the objectives of the Crowd-MAGS project was to integrate agent-based simulation and system dynamics simulation of similar scenarios in order to enhance the level of validation that was previously available. This section presents an overview of the validation activities that were performed and of the assessment of the results. A complete discussion is presented in (Larochelle et al., 2009).

As shown in Figure 3.1, the Crowd-MAGS models were calibrated partly using system dynamics models that generate macro-level output. In fact, this calibration is complementary to all calibration presented in Chapter 6 so far since it is based on a completely different modelling paradigm. More details about the SD model and the calibration process can be found in (Larochelle and Moulin, 2009) and (Larochelle et al., 2009).

It is not obvious why a macro-level system is used to calibrate a micro-level system. In addition, the level of complexity between the two models is quite different. For example, the Crowd-MAGS models include micro-behaviours and social theories that are not present in the SD model. In fact, only the macro-level data of the Crowd-MAGS system can be directly compared to the SD model. Thus, this technique of calibration may not be the most appropriate one, but it is certainly worth investigating.

Theoretical Validity

Table 6.29 presents the strengths and weaknesses of both modelling paradigms to help devise a useful strategy to combine them.

Table 6.29: Comparison of the agent-based and the system dynamics models

Paradigm	**Strengths**	**Weaknesses**
Agent-Based	- Spatial information - Visualization - Detailed behaviours	- Very slow - Limited number of agents - Complex analysis
System Dynamics	- Very fast - Simple analysis - Simple parameter tuning	- No spatial information - No visualization - Simple representation (actions, social aspects)

From the information presented in Table 6.29, it seems like the best way to combine both models is to take advantage of the SD model's speed and the Crowd-MAGS model's level of detail. Because the SD model is fast and is coupled with analysis tools such as Vensim, it can be used to quickly select the best intervention techniques from the complete set — in order to weed out the undesirable ones. From this subset, each technique can then be run on the Crowd-MAGS model to get more detailed results. Next, the user can decide from the micro-simulation results which technique is most appropriate, from both a qualitative and a quantitative point of view.

Another way in which the models can be combined is to increase the plausibility of the results. Indeed, the systems have been built on completely different modelling paradigms and have been calibrated on their own with different data sets. If both systems give similar results when running a similar scenario, then these results are more plausible than if the results came from only one of the two systems.

Practical Example

As listed in Section 6.5, twelve possible intervention techniques are modelled in the Crowd-MAGS system. Unfortunately, they cannot all be tested on each scenario due to the complexity of the Crowd-MAGS models. As a reference, running the twelve strategies on a 5-minute scenario requires one hour (and only 10 seconds on the SD model).

After trying on one of the more complex scenarios, it has been shown that the use of both models in synergy can reduce the time necessary for decision making by a significant amount. In this case, only 3 out of 12 intervention techniques were tested on the Crowd-MAGS model because the SD model considered that the others were clearly inferior. In addition to the time saved, the cognitive effort required from the user is significantly reduced since a user can focus on a few techniques rather than trying to compare all of them.

In order for this technique to be useful, both models must be able to evaluate the techniques in a relatively consistent manner. In other words, a technique that the SD model would classify as poor should not be classified as excellent by the Crowd-MAGS model. Unfortunately, the SD model that was used was calibrated using a very different data set than the one that was used for the Crowd-MAGS system. In fact, the scenarios normally run in the SD model last 8 hours, while they last 5 minutes in the Crowd-MAGS system. Consequently, the results showed that the rankings are close but that there are still significant differences, making it impossible to claim that both systems give similar results. Thus, strategies that seem best on the Crowd-MAGS model may be considered as mediocre by the SD model. More calibration is hence required to fully exploit the synergy between the two modelling paradigms.

6.9 Conclusions

This chapter showed that the Crowd-MAGS system is indeed useful. More specifically, the theoretical models have shown to recreate plausible situations and the system has proved to be easy and efficient enough to be used by domain experts. A generic scenario has been created to perform initial calibration. This scenario was later customized with different types of crowds, control force strategies, spatial distribution of agents, and more. Calibration and experimentation was performed using these scenarios.

Isolated calibration of parameters was initially performed by system developers, followed by qualitative calibration using simple scenarios. Next, parameters were fine-tuned with a large set of scenarios and the help of domain experts.

The system was coupled with a system dynamics model, which showed to be useful in speeding up decision-making when analyzing several scenarios with several intervention strategies. Nonetheless, experiments showed that both systems should be calibrated further to be more in tune.

Chapter 7 summarizes this thesis and presents contributions as well as avenues for future research in the field.

Chapter 7

Conclusions

Chapter 6 presented, through calibration, experimentation, and validation, a general idea of what is and what is not possible with the Crowd-MAGS system. This chapter follows well because it continues the discussion, as well as presenting future directions and the necessary research to fulfill further objectives. More precisely, Section 7.1 summarizes this thesis while Section 7.2 emphasizes theoretical and practical contributions. Finally, Section 7.3 discusses future works that could be undertaken to further enhance research in crowd simulation.

7.1 Summary

In short, this thesis proposed new agent models that explicitly take into account the social dimension that is used for the management of collective actions in groups of agents in order to assess the efficiency of non-lethal weapons and control force intervention strategies. More specifically, this thesis started by presenting a summary of existing agent models that incorporate or could incorporate a social dimension. Other models were discussed, in particular some related to crowds and control forces. Unfortunately, no model or platform that combined all elements required for this project was found. In consequence, new models were developed, with a strong influence from the social identity theory.

The Crowd-MAGS system incorporates social agent models, a 3D environment, the simulation of both crowd and control forces, as well as realistic non-lethal weapons. The environment is based on semantics that agents use to navigate and fulfill their objectives. The system is based on the PLAMAGS library, so non-lethal weapons are simulated by the PLAMAGS environment and agent behaviours are executed by the PLAMAGS behaviour engine. Concurrent agent objectives, such as the ones controlling agent navigation, are managed through the concept of resources. Agents have basic properties such as perception and memory but also possess features related to the social dimension or crowd control. Among them, a health level was implemented to simulate the impact of non-lethal weapons. Agents can also belong to social groups and emerging spatio-temporal groups. Agents behave according to their social identity, which may change as the simulation progresses. Finally, agents project an image that others can perceive to learn more about the agent. In particular, the projected image contains a history of recent actions performed by the agent.

A model for spatio-temporal groups also was proposed. STGs can be planned in scenarios but are normally emerging. They can be created and disbanded at any time, can possess a controller, and also project an image that agents use to get more information. No roles exist in STGs, but members are spatially organized in formation. Their actions determine the aggressiveness level of the group.

Generic behaviour models shared by all agents and detailed behaviour models of control force agents and crowd members have been developed. These models handle all levels of behaviour from basic navigation to comprehensive intervention strategies going through the management of the belonging to STGs and social identities. Several of these models are based on the concept of appreciation of aggressiveness.

This thesis proposed also an information model that is composed of the various data structures necessary to collect and organize the data obtained during simulation. The most common actions seen in protests from empirical videos were semantically defined in the system to be simulated by agents. The empirical data and the ones collected from the system were compared for analysis purposes.

Several mechanisms of the Crowd-MAGS application were discussed, such as the scheduler and the data collector that both use the information model to generate the data necessary for analysis. A scenario architecture in XML was proposed and it allows domain experts without programming knowledge to specify scenarios. The application's simple user interface allows the specification of scenarios in an even easier manner. This user interface allows in particular the customization of nearly all system parameters, providing complete freedom to scenario designers.

The user interface and scenario specification model proved to be useful because the system was tested by DRDC experts. Scenarios were created to calibrate the system while illustrating the role changes of control force agents, the interactions of five crowd social identities, a march organized by demonstrators, and the use of non-lethal weapons. This thesis proposed and followed a calibration strategy to adjust the large number of system parameters. The strategy consisted of characterizing three types of crowds (passive, moderate and aggressive) and a set of 26 scenarios that would be used to assess the impact of various intervention strategies based on the use of non-lethal weapons. Numerous simulation trials were executed to calibrate the most influential parameters. More advanced scenarios were devised to experiment with more complex parameters that could influence the choice of intervention strategies. A method of coupling the system with a system dynamics model was proposed and tested. It was shown that the Crowd-MAGS and the SD models should be used for different and complementary purposes. The SD model can be used to quickly and globally assess CF intervention strategies while the Crowd-MAGS model can be used to simulate these strategies at a micro-level.

7.2 Contributions

This section presents theoretical contributions brought forth by the agent and group models as well as practical contributions accomplished by the operational software and associated calibration procedure.

The following represent this thesis' main theoretical results.

- Agent models that take into account different social identities, group dynamics, and the appreciation of the collective actions
- Agent and group models that are both rich in information and computationally tractable
- Generic agent and group models that have been adjusted to the context of simulation of crowds and of control forces
- Generic behavioural models that apply to all agents and put in use the social aspects of agents
- Behavioural models that are specific to each one of the eight different social identities developed
- Models of control forces and non-lethal weapons that explicitly introduce a social dimension in the interaction with groups of agents

The following represent this thesis' main practical results.

- The implementation of a full set of behaviours using the PLAMAGS language
- A crowd simulator that is generic but that has been customized for crowd control simulations (including non-lethal weapons)
- A scenario specification model and tool that are easy enough to be used by domain experts
- A set of test and calibration scenarios that covers well the common crowd situations
- A calibration procedure for the numerous system parameters
- A crowd simulator that is well calibrated and can be used for the assessment of intervention strategies.
- A technique for coupling macro- and micro-simulations to enhance calibration and speed-up decision making
- Basic results on the impact of some crowd parameters on average crowd aggressiveness

The theoretical models show the feasibility of a large-scale agent-based simulation that considers social interactions between agents. The software application demonstrates the technical feasibility of these theoretical models. In fact, it is possible to simulate several hundreds of agents in crowd control situations using a set of social models (social identity change, explicit models of groups, interactions of individuals and groups). Even though complex analyses have not been performed yet on the simulation output data, the amount and type of information gathered by the models is significant and valuable for further developments.

This project has opened new avenues for the development of crowd simulations with agent models in which the social dimension is explicitly taken into account, not only at the individual level but also at the group level. The next section discusses some of these research directions.

7.3 Future Works

This section presents various avenues for future research that could be carried out to enhance the Crowd-MAGS system. The four main areas for improvement are: data collection, development of the social models, elaboration of scenarios, and the software system itself.

7.3.1 Data Collection

As Chapter 6 mentioned, the lack of experimental data prevented a complete calibration, especially since the Crowd-MAGS system and the system dynamics model could not use the same data set. One of the first recommendations is therefore to seek more detailed data from real crowd situations to feed the agent models and scenarios. For example, formal collaboration could be established with organizations involved in the control and monitoring of crowd situations, such as the Royal Canadian Mounted Police (RCMP) and the Sûreté du Québec (SQ). Confidentiality is a sensitive issue but hopefully the fact that the Crowd-MAGS system is operational may convince the authorities of the potential of such a project.

7.3.2 Development of the Models

The second avenue for future research concerns the refinement of the agent models. Since the simulation framework is stable and basic components, such as non-lethal weapons, have been developed, agents need to react in more intelligent and coordinated ways to various events. Indeed, more work is needed to develop new and more sophisticated behaviours for the different social identities, both in the crowd and in the control forces. A few examples of the behaviours that would be highly beneficial are: CF arresting crowd members, CF pursuing a crowd member, CF pushing the crowd back, crowd members attacking the environment, crowd members encouraging violent acts, agents reacting to the media, and agents helping other injured agents. More basic actions that were observed in surveillance and amateur videos are still unimplemented and could greatly enhance the behaviour models. While improving these models, more advanced control force strategies should be developed. Although the implementation of the behaviours should be performed by software developers, the conceptual design of the behaviours should be in the hands of social scientists.

7.3.3 Creation of Scenarios

In addition to improving the social models, it is recommended to develop scenarios reflecting more complex typical crowd situations. For example, some scenarios should involve the CF pushing back the crowd through narrow streets, like in the events of the Summit of the Americas in 2001. Another scenario could involve a small number of CF agents trying to control an overwhelmingly large but passive crowd. Another idea would be improving the plausibility of the existing scenarios by leaving resources lying around in the environment before the simulation starts. For example, a pile of rocks could be set-up so that protestors could grab them and throw them at the CF.

7.3.4 System

Finally, some improvements could be made on the software platform itself. For example, scenarios must be run from start to end without possibility of saving them in the middle — they can at least be paused. However, it would be very useful to stop a scenario at some point and to try different CF strategies starting from that point. Thus, it should be possible to save scenarios, replay them, and even change the course of the events. Such features would be helpful for analysis but also training purposes, if the system were used to train CF officers. In addition, tools for analyzing output data could be enhanced. These tools would allow for more complex or targeted data analyses.

Another recommendation is to use a VGE with a topologic representation of the environment, as discussed in Sub-Section 4.7.1. The integration of the notion of informed virtual geographic environment would allow the development of a semantically-enriched virtual environment with precise spatial subdivisions and possibilities of data abstraction. Such an environment representation is desirable for plausible path planning behaviours and a better perception of the agents' surroundings. Such features would greatly enhance the possibility of crowd simulations in complex urban environments.

Finally, the performance of the PLAMAGS behaviour engine should be improved. As mentioned in Section 5.4, no more than a few hundred agents can be simulated at once before the system cannot run in real-time anymore. From various tests presented in Appendix C, it seems like the PLAMAGS behaviour engine causes the main bottleneck in the system. Thus, improving this component could allow the simulation of many more agents, possibly thousands. This improvement would allow the creation of much more elaborated scenarios and crowd interactions.

No one supposes that a computer simulation of a storm will leave us all wet... Why on earth would anyone in his right mind suppose a computer simulation of mental processes actually had mental processes?

John Searle, 1980

Bibliography

Adams, P. C. (1998). Teaching and learning with simcity 2000. *Journal of Geography*, 97:47–55.

Aggarwal, R., Black, S. A., Hance, J. R., Darzi, A., and Cheshire, N. J. (2006). Virtual reality simulation training can improve inexperienced surgeons' endovascular skills. *European Journal of Vascular and Endovascular Surgery*, 31(6):588–593.

Ali, W. (2006). *Developping 2D and 3D Multiagent Geosimulation, a Method and its Application: the Case of Shopping Behavior Geosimulation in Square One (Toronto)*. PhD thesis, Université Laval — Département d'informatique et de génie logiciel.

Ali, W. (2008). *2D/3D MultiAgent GeoSimulation: The Case of Shopping Behavior of Square One Mall (Toronto)*. VDM Verlag, Saarbrücken, Germany.

Atkinson, R. C. and Shiffrin, R. M. (1968). Human memory: A proposed system and its control processes. *The Psychology of Learning and Motivation: Advances in Research and Theory*, 2:89–195.

Bakker, B. (2000). The adaptative behavior approach to psychology. In *Cognitive Processing*, volume 1, pages 39–70.

Bédard, Y., Rivest, S., and Proulx, M.-J. (2006). Spatial on-line analytical processing (solap): Concepts, architectures, and solutions from a geomatics engineering perspective. In *Data Warehouses and OLAP: Concepts, Architecture, and*, pages 298–319.

Benenson, I. and Torrens, P. M. (2003). Geographic automata systems: A new paradigm for integrating GIS and geographic simulation. In *Proceedings of the 7th International Conference on GeoComputation*.

Benenson, I. and Torrens, P. M. (2004). Geosimulation: object-based modeling of urban phenomena. *Computers, Environment and Urban Systems*, 28:1–8.

Bürki-Cohen, J., Go, T. H., Chung, W. W., Schroeder, J., Jacobs, S., and Longridge, T. (2003). Simulator fidelity requirements for airline pilot training and evaluation continued: An update on motion requirements research. In *Proceedings of the 12th International Symposium on Aviation Psychology*.

Champion, A., Éspié, S., and Auberlet, J.-M. (2001). Behavioral road traffic simulation with ARCHISIM. In *2001 Summer Computer Simulation Conference*.

Chevillona, G. and Hendry, D. F. (2005). Non-parametric direct multi-step estimation for forecasting economic processes. *International Journal of Forecasting*, 21(2):201–218.

Commonwealth Human Rights Initiative (2005). Standards and procedure for crowd control. Publicly available standards.

Conte, R., Gilbert, N., and Sichman, J. S. (1998). MAS and social simulation: A suitable commitment. *Lecture Notes in Computer Science*, 1534:1–9.

Cronin, P. and Reicher, S. D. (2006). A study of the factors that influence how senior officers police crowd events: On SIDE outside the laboratory. *British Journal of Social Psychology*, 45(1):175–196.

Daiger, S. P. (2005). Genetics: Was the human genome project worth the effort? *Science*, 308(5720):362–364.

Dauxois, T. and Ruffo, S. (2008). Fermi-pasta-ulam nonlinear lattice oscillations. World Wide Web Electronic Publication. `http://www.scholarpedia.org/article/Fermi-Pasta-Ulam_nonlinear_lattice_oscillations`, accessed 2009-04-16.

Davidsson, P. (2000). Multi agent based simulation: Beyond social simulation. In *MABS 2000: Proceedings of the second international workshop on Multi-agent based simulation*, pages 97–107, Secaucus, NJ, USA. Springer-Verlag New York, Inc.

Desgagné, M., McTaggart-Cowan, R., Ohfuchi, W., Brunet, G., Yau, P., Gyakum, J., Furukawa, Y., and Valin, M. (2006). Large atmospheric computation on the earth simulator: The LACES project. *Scientific Programming*, 14(1):13–25.

Drury, J. and Reicher, S. D. (2005). Explaining enduring empowerment: A comparative study of collective action and psychological outcomes. *European Journal of Social Psychology*, 35(1):35–58.

Ferber, J. (1995). *Les Systèmes multi-agents: Vers une intelligence collective.* Dunod, Paris, France.

Franses, P. H. (1998). *Time Series Models for Business and Economic Forecasting (Themes in Modern Econometrics).* Cambridge University Press, Cambridge, United Kingdom.

Garneau, T. (2007). *Spécifications Du Langage.* Université Laval — Groupe de recherche en informatique cognitive, Québec City, QC, Canada.

Garneau, T. and Moulin, B. (2008). PLAMAGS: A language and environment to specify intelligent agents in virtual geo-referenced worlds. In *Modelling and Simulation.*

Gleick, J. (1987). *Chaos Theory: Making a New Science.* Vintage.

Goh, C. K., Quek, H. Y., Tan, K. C., and Abbass, H. A. (2006). Modelling civil violence: An evolutionary multi-agent, game theoretic approach. In *2006 IEEE Congress on Evolutionary Computation*, pages 1624–1631.

Herath, S. (2001). Geographical information systems in disaster reduction. *Information Technology For Disaster Management*, 1:25–31.

International Association of Chiefs of Police (2001). Police use of force in america. Report.

Jager, W., Popping, R., and van de Sande, H. (2001). Clustering and fighting in two-party crowds: Simulating the approach-avoidance conflict. *Journal of Artificial Societies and Social Simulation*, 4(3).

JAMSEC (2009). Replacement of the earth simulation. World Wide Web Electronic Publication. `http://www.jamstec.go.jp/es/en/es2/index.html`, accessed 2009-04-17.

Jennings, N. R., Sycara, K., and Wooldridge, M. (1998). A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems*, 1(1):7–38.

Kaup, D. J., Clarke, T. L., Malone, L. C., Jentsch, F. G., and Oleson, R. (2008). Introducing age-based parameters into simulations of crowd dymanics. In *Proceedings of the 2008 Winter Simulation Conference*, pages 895–902. Winter Simulation Conference.

Keeney, R. L. and Gregory, R. S. (2005). Selecting attributes to measure the achievement of objectives. *Operational Research*, 53(1):1–11.

Kelton, W. D., Sadowski, R. P., Sturrock, D. T., Kelton, W., Sadowski, R., and Sturrock, D. (2003). *Simulation with Arena*. McGraw-Hill Professional, Dubuque, IA, USA.

Kenny, J. M., McPhail, C., Waddington, P., Heal, S., Ijames, S., Farrer, D. N., Taylor, J., and Odenthal, D. (2001). Crowd behavior, crowd control, and the use of non-lethal weapons. Human Effects Advisory Panel — Report of Findings.

Klügl, F. (2004). Multi-agent simulation. Lecture notes for The Sixth European Agent Systems Summer School 2004.

Lamarche, F. and Donikian, S. (2002). Automatic orchestration of behaviours through the management of resources and priority levels. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*.

Larochelle, B. (2008). Multi-criteria decision analysis of the efficiency of non-lethal weapons through multi-agent simulation. Technical report, Université Laval.

Larochelle, B., Hamdi, B., and Moulin, B. (2008). Analysis of data collection based on protests videos. Technical report, DRDC Valcartier.

Larochelle, B. and Moulin, B. (2009). The Crowd-MAGS system on the PLAMAGS platform: A scientific and technical view. Technical report, DRDC Valcartier. CR 2009-064.

Larochelle, B., Moulin, B., Marcotte, D., and Gagné, G. (2009). Experimentations with the Crowd-MAGS system. Technical report, DRDC Valcartier. CR 2009-065.

Lenhard, J., Küppers, G., and Shinn, T. (2006). *Simulation: Pragmatic Construction of Reality*. Springer, New York, NY, USA.

Lin, S.-J., Atlas, R., and Yeh, K.-S. (2004). Global weather prediction and high-end computing at NASA. World Wide Web Electronic Publication. `http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1255818`, accessed 2009-04-17.

Lord, R. G. and Levy, P. E. (1994). Moving from cognition to action: a control theory perspective. *Applied psychology*, 43(3):335–398.

Lorenz, F. M. (1996). Non-lethal force: The slippery slope to war? *PARAMETERS*, 36(3):52–62.

Lynch, S. and Rajendran, K. (2005). Multi-agent systems design for novices. *Computer Science Education*, 15(1):41–57.

Maier, F. H. and Größler, A. (2000). What are we talking about? — A taxonomy of computer simulations to support learning. *System Dynamics Review*, 16(2):135–148.

Manojlovich, J., Prasithsangaree, P., Hughes, S., Chen, J., and Lewis, M. (2003). UTSAF: a multi-agent-based framework for supporting military-based distributed interactive simulations in 3d virtual environments. In *Proceedings of the 2003 Winter Simulation Conference*, volume 1, pages 960–968.

Marcotte, D. (2007). Environnement 3D de PLAMAGS — documentation interne. Technical report, Université Laval — Groupe de recherche en informatique cognitive.

Masco, J. (2006). *The Nuclear Borderlands: the Manhattan Project in post-Cold War New Mexico*. Princeton University Press, Princeton, NJ, USA.

McKenzie, F. D., Nguyen, Q.-A. H., Petty, M. D., Weisel, E. W., Camp, J., Anthony, J., and Albright, R. (2005a). Crowd federate implementation for maneuver support simulation. In *Proceedings of the Fall 2005 Simulation Interoperability Workshop*, pages 574–587.

McKenzie, F. D., Xu, Q., Nguyen, Q.-A. H., and Petty, M. D. (2004). Crowd federate architecture and api design. In *Proceedings of the Fall 2004 Simulation Interoperability Workshop (SIW)*.

McKenzie, F. D., Xu, Q., Nguyen, Q.-A. H., and Petty, M. D. (2005b). Designing physical layer components in a reconfigurable crowd federate. In *Proceedings of the Spring 2005 Simulation Interoperability Workshop*.

Min, F., Ma, P., and Yang, M. (2007). A knowledge-based method for the validation of military simulation. In Henderson, S. G., Biller, B., Hsieh, M.-H., Shortle, J., Tew, J. D., and Barton, R. R., editors, *Proceedings of the 2007 Winter Simulation Conference*, pages 1395–1402, Piscataway, NJ, USA. IEEE Press.

Moss, S. and Edmonds, B. (2005). Sociology and simulation: Statistical and qualitative cross-validation. *American Journal of Sociology*, 110:1095–1131.

Moulin, B. (2009). Towards a multi-level approach of the social dimension of crowd simulations: Explicitly modeling groups' and agents' interactions. Research Report DIUL_RR 09_02. Université Laval — Département d'informatique et de génie logiciel.

Moulin, B., Bouden, M., L'heureux, G., Mekni, M., and Sahli, N. (2007). Literature review on the modeling and simulation of crowd behaviours, monitoring and non-lethal weapons. Technical report, DRDC Valcartier. CR 2007-047.

Moulin, B., Chaker, W., Perron, J., Pelletier, P., Hogan, J., and Gbei, E. (2003). MAGS project: Multi-agent geosimulation and crowd simulation. In *Proceedings of the COSIT'03 Conference*, pages 151–168.

Murakami, Y., Ishida, T., Kawasoe, T., and Hishiyama, R. (2003). Scenario description for multi-agent simulation. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 369–376, New York, NY, USA. ACM.

Murakami, Y., Minami, K., Kawasoe, T., and Ishida, T. (2002). Multi-agent simulation for crisis management. In *IEEE Workshop on Knowledge Media Networking, 2002. Proceedings.*, pages 135–139.

Musse, S. R. and Thalmann, D. (2001). A hierarchical model for real time simulation of virtual human crowds. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):152–164.

Newell, A. (1990). *Unified Theories of Cognition.* Harvard University Press.

Nguyen, Q.-A. H., McKenzie, F. D., and Petty, M. D. (2005). Crowd behavior cognitive model architecture design. In *2005 Brims Conference.*

Oakland Police Department (2004). OPD crowd management/ crowd control policy. Doctrine.

Orcutt, G. H. (1960). Simulation of economic systems. *The American Economic Review*, 50(5):893–907.

Osaragi, T. (2004). Modelling of pedestrian behaviour and its applications to spatial evaluation. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, volume 2, pages 836–843.

Pan, X. (2006). *Computational Modeling Of Human And Social Behaviors For Emergency Egress Analysis.* PhD thesis, Stanford University.

Paris, S. (2007). *Caractérisation des niveaux de services et modélisation des circulations de personnes dans les lieux d'échanges.* PhD thesis, Université de Rennes 1.

Paris, S., Donikian, S., and Bonvalet, N. (2005). Towards more realistic and efficient virtual environment description and usage. In *First International Workshop on Crowd Simulation (V-Crowds'05).* VRlab, EPFL.

Paris, S., Mekni, M., Moulin, B., Larochelle, B., Garneau, T., Marcotte, D., and Hamdi, B. (2008). The CrowdMAGS project: Agent models and behaviours specification. Technical report, DRDC Valcartier. CR 2008-311.

Paris, S., Mekni, M., Moulin, B., and Marcotte, D. (2009). CrowdMAGS project: Informed Virtual Geographic Environment — models and C++ implementation. Technical report, DRDC Valcartier. CR 2009-066.

Reeves, H. and Reeves, N. (2009). Quelques points sur une feuille de papier. Public Presentation. 2009/05/01 19h00. Église St-Roch, Québec, QC, Canada.

Roy, B. (1985). *Méthodologie Multicritère d'Aide à la Décision.* Production et Techniques Quantitatives Appliquées à la Gestion. Ed. Economica, Paris, France.

Russell, S. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach.* Prentice Hall, Upper Saddle River, NJ, USA.

Sahli, N. (2006). *La géosimulation orientée agent : un support pour la planification dans le monde réel.* PhD thesis, Université Laval — Département d'informatique et de génie logiciel.

Santos, G. and Aguirre, B. E. (2004). A critical review of emergency evacuation simulation models. In *DRC Preliminary Papers*. Disaster Research Center.

Sargent, R. G. (2007). A knowledge-based method for the validation of military simulation. In Henderson, S. G., Biller, B., Hsieh, M.-H., Shortle, J., Tew, J. D., and Barton, R. R., editors, *Proceedings of the 2007 Winter Simulation Conference*, pages 124–137, Piscataway, NJ, USA. IEEE Press.

Schweingruber, D. and McPhail, C. (1999). A method for systematically observing and recording collective action. *Sociological Methods & Research*, 27(4).

Shi, D. and Brooks, R. J. (2007). The range of predictions for calibrated agent-based simulation models. In Henderson, S. G., Biller, B., Hsieh, M.-H., Shortle, J., Tew, J. D., and Barton, R. R., editors, *Proceedings of the 2007 Winter Simulation Conference*, pages 1198–1206, Piscataway, NJ, USA. IEEE Press.

Silverman, B. G., Bharathy, G., and Nye, B. (2006a). Gaming and simulating ethnopolitical conflicts. Descartes Conference on Mathematical Models in Counterterrorism. Washington, DC, USA. 29 September 2006.

Silverman, B. G., Cornwell, J., and O'Brien, K. (2003). *Metrics and methods in human performance research toward individual and small unit simulation*, chapter 9. Human Systems Information Analysis Center, Washington, DC, USA.

Silverman, B. G., Johns, M., Cornwell, J., and O'Brien, K. (2006b). Human behavior models for agents in simulators and games: Part i: Enabling science with pmfserv. *Presence: Teleoperators and Virtual Environments*, 15(2).

Silverman, B. G., Johns, M., O'Brien, K., Weaver, R., and Cornwell, J. (2002). Constructing virtual asymmetric opponents from data and models in the literature: Case of crowd rioting. Postprint version of an article.

Stemate, L. (2006). Crowd control M&S capability. Powerpoint Presentation.

Sulistio, A., Yeo, C. S., and Buyya, R. (2004). A taxonomy of computer-based simulations and its mapping to parallel and distributed systems simulation tools. *Software - Practice & Experience*, 34(7):653–673.

Tajfel, H. (1978). *Differentiation Between Social Groups: Studies in the Social Psychology of Intergroup Relations.* Academic Press, London, UK.

Thalmann, D., Musse, S. R., and Kallmann, M. (2000). From individual human agents to crowds. *INFORMATIK/INFORMATIQUE.*

Varner, D., Royse, S. D., Micheletti, J., and Apicella, G. (2000). USMC Small Unit Leader Non-Lethals Trainer (SULNT). Southwest Research Institute (SwRI).

Vassileva, J., editor (2001). *International Conference on AI and Education*, San Antonio, TX, USA.

Vincenzi, D. A. and Wise, J. A. (2008). *Human Factors in Simulation and Training*. CRC Press, Boca Raton, FL, USA.

Wijermans, N., Jorna, R., Jager, W., and van Vliet, T. (2007). Modeling crowd dynamics: influence factors related of the probability of a riot. Paper presented at The Fourth European Social Simulation Association Conference.

Wijermans, N., Jorna, R., Jager, W., and van Vliet, T. (2008). Modeling crowd dynamics: influence factors related of the probability of a riot. In *Proceedings of the 2nd World Congress on Social Simulation*.

Woodaman, R. F. A. (2000). Agent-based simulation of military operations other than war small unit combat. Master's thesis, Naval Postgraduate School Monterey.

Wrembel, R. and Koncilia, C. (2006). *Data Warehouses And OLAP: Concepts, Architectures And Solutions*. IRM Press, Hersey, PA, USA.

Bibiographical Database

BibTeX is a reference management system used to organize and format lists of references. The BibTeX tool has significantly grown in popularity and is now one of the most commonly used formats for dealing with bibliographical information on the web. This format is endorsed by ACM, Amazon, Google Scholar, Science Magazine, Springer and many others. The following is the BibTeX bibliographical database used for this thesis; this database includes descriptions for each reference and when possible a URL that may help the reader in getting the resource more quickly.

The electronic version of this thesis contains two embedded attachments here:

The printed version of this thesis presents the contents of the files just below.

Appendix A

Extraneous Information

Some information could have been presented in the thesis' body but it was not deemed crucial to the understanding of the concepts. Therefore, it is presented in this chapter. First, Section A.1 presents collective actions and other observations gathered in data collection stage of the Crowd-MAGS project. Next, Section A.2 presents an example of an XML scenario file. Then, Section A.3 lists icons and pictures that show what agents looks like in Crowd-MAGS. Finally, Section A.4 lists the values selected before the calibration process was started.

A.1 Collective Actions from Observation

Chapter 2 showed that no exhaustive taxonomy of collective actions observed in crowd control situations has been found. Some papers have presented (incomplete) lists of collective actions performed by crowd members, but none of them gathered similar information for control forces. Since the literature review did not provide sufficient data for the Crowd-MAGS Project, it was decided to perform a data collection process to investigate a variety of crowd events in North America and Europe. The objective was to collect material that could be used to help the agent modelling effort and the calibration and validation process. More specifically, the goal was to establish and validate a corpus of observed behaviours of individuals, groups and control forces in typical crowd control situations. Each analyzed video has been logged into a 'fiche'; the corpus, presented in Table A.1, has been created from the aggregation of all fiches. The data collection and analysis stage was performed between November 2007 and March 2008 and it contains crowd events from the early 1990's up to 2007. Complete details of the process and results are presented (Larochelle et al., 2008).

Table A.1: Crowd-MAGS' Corpus of Collective Actions

Crowd Actions	**CF Actions**	**Common Objects**
Acting	Assaulting the crowd	Banners
Applauding	Charging towards a target group / individual	Batons
Attacking the group of police officers	Detaching from the group	Disguises
Blocking CF vehicles	Filming	Face scarves
Booing	Getting hurt	Fence
Burning objects	Giving orders to the crowd or an individual	Gas
Chanting	Helping colleagues hit by fire	Gas mask
Cheering	Holding the line	Megaphone
Following a leader or leading group	Ignoring protestors	Molotov cocktails
Getting hurt	Leaning down to avoid water	Pepper spray
Harassing police officers	Making an arrest	Photo camera
Helping a wounded colleague	Moving back slowly	Police shields
Holding banners	Moving back-and-forth	Projectiles
Insulting verbally	Person in civil helping a wounded officer	Speakers
Insulting with body language	Physically pushing the crowd	Tear gas cans
Kicking a tear gas can towards officers	Protecting a wounded colleague with shields	Vandalized property
Letting the way for the officers	Protecting colleagues	Video cameras
Physically assaulting an officer	Protecting self with shield	Water cannons
Playing music, dancing, or juggling	Pulling a protestor aside	
Raising hands to show compliance	Removing fire from suit	
Running away	Retreating quickly	
Shouting or yelling	Running through a crowd	
Showing off for the media	Shooting plastic bullets	
Shrilling	Shooting water towards specific protestors	
Sitting/Laying down blocking the way	Single officer running back	
Staying in a cloud of gas	Splitting a crowd	

Table A.1 – Continued on Next Page...

Table A.1 – Continued

Crowd Actions	**CF Actions**	**Common Objects**
Teasing or harassing officers	Spraying gas	
Throwing Molotov cocktails	Spraying water onto the crowd	
Throwing projectiles	Staying behind a fence	
Trying to stop other protestors from acting aggressively	Swinging a baton	
Vandalizing	Taking pictures	
Withstanding the water pressure without moving	Throwing or shooting gas cans	

A.2 Scenario File Example

The following is an example an XML scenario file. The file is longer than the simplistic example presented in Section 4.6 but it is still incomplete.

```
<Scenario>

 <Parameters>

  <Cameras>
   <Camera>
    <PositionX>-125</PositionX>
    <PositionY>156</PositionY>
    <PositionZ>108</PositionZ>
    <RotationX>27</RotationX>
    <RotationZ>-277</RotationZ>
   </Camera>
  </Cameras>

  <Agent>
   <Radius>0.3</Radius>
   <Height>1.8</Height>
   <Mass>65.0</Mass>
   <WalkingSpeed>1.4</WalkingSpeed>
   <FleeingSpeedFactor>2.5</FleeingSpeedFactor>
   <Perception>
    <DistanceRegular>20.0</DistanceRegular>
    <DistancePointObject>80.0</DistancePointObject>
    <DistanceGas>100.0</DistanceGas>
    <AngleRegular>120.0</AngleRegular>
```

```
      <AnglePointObject>120.0</AnglePointObject>
      <AngleGas>160.0</AngleGas>
    </Perception>
  </Agent>

  <Profiles>
    <Profile>
      <Name>CF Squad Leader Neutral</Name>
      <SocialIdentity>SquadLeader</SocialIdentity>
      <PreferredSTGType>CF Squad</PreferredSTGType>
      <AppreciationMin>-1.0</AppreciationMin>
      <AppreciationOptimal>-1.0</AppreciationOptimal>
      <AppreciationMax>0.25</AppreciationMax>
      <ImportanceOfPastEvents>0.75</ImportanceOfPastEvents>
      <AdhesionSTGMin>0.0</AdhesionSTGMin>
      <AdhesionSTGMax>0.6</AdhesionSTGMax>
      <EnthusiamGap>NaN</EnthusiamGap>
    </Profile>
  </Profiles>

  <ProfileDistribution>
    <Bystander>
      <Passive>0.34</Passive>
      <Moderate>0.33</Moderate>
      <Aggressive>0.33</Aggressive>
    </Bystander>
  </ProfileDistribution>
</Parameters>

<Components>
  <Fences>
    <Fence>
      <Name>Fence #13</Name>
      <AppearanceTime>0</AppearanceTime>
      <PositionX>-74.32499694824219</PositionX>
      <PositionY>163.3249969482422</PositionY>
      <Orientation>315.0000190109573</Orientation>
      <MaxAnchorForce>200.0</MaxAnchorForce>
      <MaxAnchorTorque>50.0</MaxAnchorTorque>
    </Fence>
  </Fences>
  <Journalists>
      <Journalist>
          <Name>Journalist #1</Name>
          <AppearanceTime>0</AppearanceTime>
          <PositionX>-90</PositionX>
          <PositionY>205</PositionY>
          <Orientation>180</Orientation>
      </Journalist>
      <Journalist>
         <Name>Journalist #2</Name>
         <AppearanceTime>0</AppearanceTime>
         <PositionX>-45</PositionX>
         <PositionY>125</PositionY>
         <Orientation>45</Orientation>
      </Journalist>
   </Journalists>

  </Components>

</Scenario>
```

A.3 Agent's Physical Appearance

This table, first introduced in Sub-Section 5.1.5, presents an exhaustive list of all 3D models and icons used in the Crowd-MAGS application. These icons could be changed easily with no impact on the system. The icons marked [1] come from `http://www.famfamfam.com/lab/icons/silk/` and those marked [2] come from `http://www.everaldo.com`.

Table A.2: List of All Used Icons

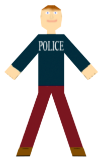
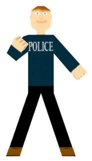
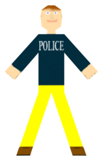
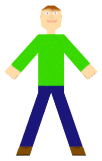
Concept	**Visual Representation**
Fundamental Social Identity: Squad Leader	
Fundamental Social Identity: Squad Member	
Fundamental Social Identity: Squad Deputy Leader	
Fundamental Social Identity: Bystander	

Table A.2 – Continued on Next Page. . .

Table A.2 – Continued

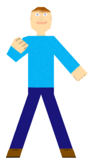
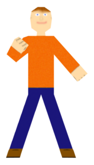
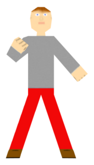
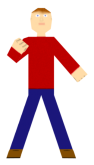
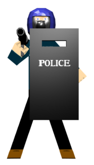
Concept	**Visual Representation**
Fundamental Social Identity: Demonstrator	
Fundamental Social Identity: Demonstrator Leader	
Fundamental Social Identity: Instigator	
Fundamental Social Identity: Instigator Leader	
CF Officer with armor, helmet, baton, and plastic bullet riffle	
Adopted Social Identity: Squad Leader	L
Adopted Social Identity: Squad Member	M
Adopted Social Identity: Bystander	B
Adopted Social Identity: Demonstrator	D

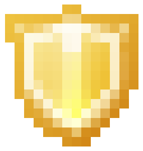
Table A.2 – Continued on Next Page. . .

Table A.2 – Continued

Concept	**Visual Representation**
Adopted Social Identity: Demonstrator Leader	DL
Adopted Social Identity: Instigator	I
Adopted Social Identity: Instigator Leader	IL
Collective Action: Chanting[2]	
Collective Action: Yelling[2]	
Collective Action: Insulting	
Looking for Resource: Megaphone[2]	
Looking for Resource: Body Armor[1]	
Looking for Resource: Helmet[2]	

Table A.2 – Continued on Next Page...

Table A.2 – Continued

Concept	**Visual Representation**
Looking for Resource: Gas Mask	
Looking for Resource: Shield[1]	
Looking for Resource: Baton	
Looking for Resource: Tear Gas Can	
Looking for Resource: Plastic Bullet Rifle	
Looking for Resource: Plastic Bullet	

A.4 Initial System Calibration

This list of parameters, introduced in Section 5.3, shows the initial configuration of the system before calibration and experimentation started. Specific details about individual profiles and profile distributions are not listed.

Physical Properties of Agents

- Mass of the agent: **65 kg**
- Height of the cylinder containing the agent's 3D representation: **1.8 m**
- Radius of the cylinder containing the agent's 3D representation: **0.3 m**

- Base walking speed for a healthy agent that moves normally: **1.4 m/s**
- Field of view for agents and components: **20 m opened at 120°**
- Field of view for interest shape: **80 m opened at 120°**
- Field of view for gas clouds: **100 m opened at 160°**
- Factor of the base speed when the agent flees something: **2.5**

Squad Characteristics

- Number of gas cans to get from the police truck when refilling: **3**
- Number of plastic bullets to get from the police truck when refilling: **10**
- Minimum aggressiveness necessary from the target to use tear gas: **0.15**
- Minimum aggressiveness necessary from the target to use plastic bullets: **0.15**
- List of required protective equipment: **armour, shield, baton, helmet, gas mask**
- Loose formation's radius for the *Uninvolved* mobilization level: **4 m**
- Tight formation's radius for the *Physical Presence* mobilization level: **3 m**
- Distance between agents when in line formation for the *Force Demonstration* mobilization level: **1.5 m**
- Distance between agents when in wedge formation for the *Crowd Dispersion* mobilization level: **1 m**

Health Parameters

- Amount of impact of tear gas: **5.0E-5 / particle**
- Amount of impact of Molotov cocktails: **0.5**
- Factor of increased tolerance to tear gas from control forces with respect to the crowd: **10**
- Amount of protection for punches and kicks from bodily armour: **10**
- Amount of protection for Molotov cocktails from bodily armour: **2**
- Amount of protection for punches and kicks from helmet: **5**
- Amount of protection for Molotov cocktails from helmet: **2**
- Amount of protection for punches and kicks from shield: **2**
- Amount of protection for Molotov cocktails from shield: **5**
- Critical health level of control forces: **0.5**

- Critical health level of bystanders: **0.8**
- Critical health level of demonstrators: **0.7**
- Critical health level of demonstrator leaders: **0.6**
- Critical health level of instigators: **0.5**
- Critical health level of instigator leaders: **0.4**

Aggressive Actions Parameters

- Minimum distance between a crowd member and control forces to assault them: **10 m**
- Minimum distance between a crowd member and control forces to attack them: **5 m**
- Safe distance for throwing Molotov cocktails (minimum and maximum): **12 m - 20 m**
- Impact of punches and kicks on health: **0.05**

Tear Gas Parameters

- Mass of a can: **0.5 kg**
- Safe distance for throwing a can (minimum and maximum): **10 m - 40 m**
- Duration of emission for a can: **30 s**
- Amount of time that a can stays in the air when thrown: **2 s**
- Inaccuracy when throwing a can: **5 %**

Plastic Bullet Parameters

- Velocity of a bullet: **60 m/s**
- Safe distance for shooting a bullet (minimum and maximum): **10 m - 100 m**
- Inaccuracy when shooting a bullet: **1 %**

Miscellaneous Parameters

- Scenario start time: **17h00**
- Time before demonstrator leaders consider the protest stagnant: **120 s**
- Minimum number of agents before the demonstrator leader starts the peace march: **10**
- Duration of the speech after the peace march: **30 s**
- Number of regular members (excluding deputy, chemical and rifle) in each squad: **1**

Appendix B

Crowd Control-Specific Behaviour Models

Beyond all behaviours already presented, Crowd-MAGS agents can be associated with other more specific behaviours. This chapter presents the agent behaviours that are assigned to given social identities and thus that cannot apply to all agents. These behaviours are all situated at the rational and social levels of Newell's pyramid (shown in Figure 4.22) and are specific to crowd control. Thus, these behaviours complement the generic ones presented in Section 4.5. Without the rational behaviours presented here, the Crowd-MAGS agents would be plausible, but they would just wander around doing nothing. Section B.1 starts by introducing the formalism that has been used to document the behaviours. Next, Section B.2 presents all behaviours related to the control forces and Section B.3 presents all of those related to the crowd.

B.1 Formalism for the Specification of Behaviours

Modelling human behaviour is a very difficult task that is too tedious to be performed completely by hand. Fortunately, PLAMAGS (introduced in Sub-Section 3.2.2) removes an important part of the complexity of specifying agent behaviours by allowing the specification of hierarchies of objectives. The structure of the behavioural graphs allows defining behaviours at different levels of abstraction and dividing behaviours into sub-behaviours. In addition, agents may possess several behaviours (graphs), allowing designers to create small re-usable behaviours that present low or no coupling between one another. For example, a behaviour for low-level functions such as navigation and perception could be implemented and used for all agents while rational behaviours could be specialized depending on agent types.

In the Crowd-MAGS project, behavioural graphs were used to specify the behaviours of certain types of agents at certain levels of Newell's pyramid. For example, a behavioural graph has been created for the social-level behaviour of bystanders. Since no visual tool is yet available with PLAMAGS, the entire behavioural graphs were typed by hand. Fortunately,

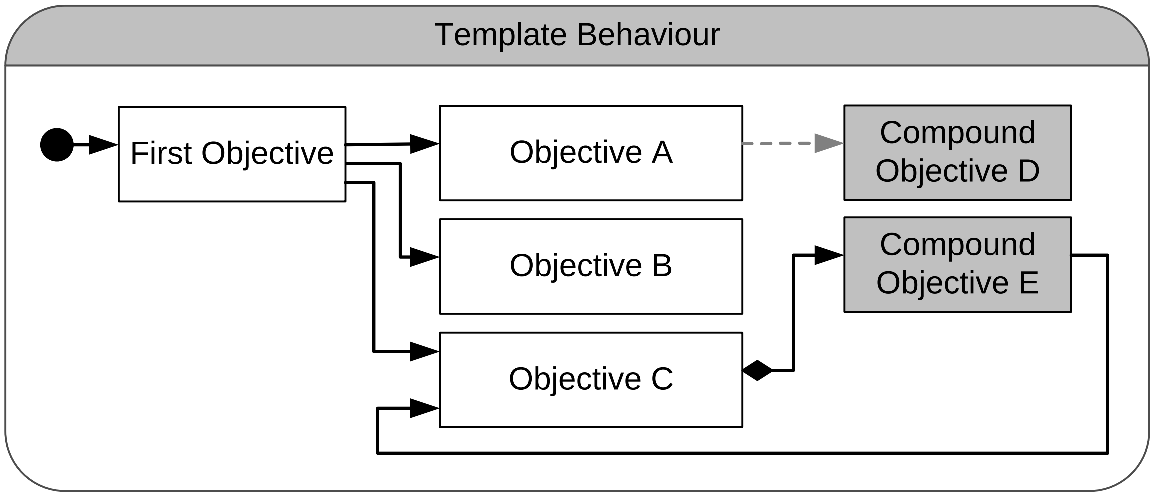


Figure B.1: Behaviour Specifications: Template

the syntax is relatively simple. Nonetheless, a visual aid of some sort was necessary when developing 'complete' behaviours because the complexity of these behavioural graphs quickly became difficult to handle. Thus, a formalism was adopted for the specification of behaviours. Table B.1 shows the symbols used to visually represent the objectives, which have been drawn by hand in Microsoft Visio. The visual representation of each objective will be shown in a graph such as Figure B.1 and its details will be listed in a table such as Table B.2. This formalism is not part of PLAMAGS and a different one could have been elaborated.

It is important to note that completion rules SUCCESS mean that the objective will terminate after the first time that it is executed. This often represents instantaneous actions, such as throwing a projectile. Completion rules N/A mean that the objective will never terminate on its own. However, other objectives could change its state terminate it. Another important point is the constant agent.SOCIAL_PRIORITY, whose value was set to 1. It is meant to represent the 'normal' priority of an objective at the social level but it has not been calibrated.

Table B.1: Formalism for the Specification of Objectives

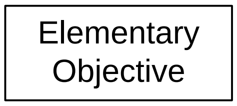
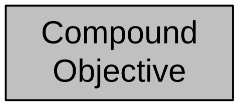
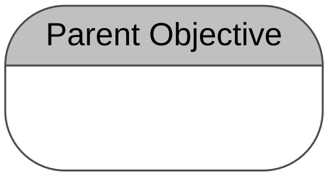
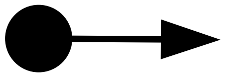
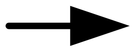
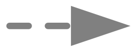
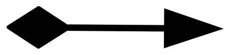
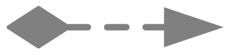
Symbol	Signification
	**Elementary objective:** performs the same actions every time that it is executed. These actions are often simple and usually last one iteration only (although they can be repeated to last longer).
	**Compound objective:** usually executes no action but rather contains sub-objectives that perform a complex task.
	**Parent objective:** graphically shows the sub-objectives composing a complex objective.
	**Entry Point:** identifies the objectives that are executed first.
	**SUCCESS transition:** indicates that on successful termination, the objective unconditionally transits to its successors.
	**FAILURE transition:** indicates that on failure termination, the objective unconditionally transits to its successors.
	**SUCCESS CONDITIONAL transition:** indicates that on successful termination, the objective transits to its successors if a condition is validated.
	**FAILURE CONDITIONAL transition:** indicates that on failure termination, the objective transits to its successors if a condition is validated.

Table B.2: Behaviour Specifications: Template

Objective Name	
Type:	Behaviour / Elementary / Compound
Resource Priority:	Numerical constant or variable
Resources:	List of resources
Execution Rules:	Description of the rules
Action:	Description of what the objective does
Completion Rules:	Description of the rules
Comments:	Comments, implementation details, things to keep in mind, etc.

B.2 Control Forces Models

This section presents in a detailed manner all of the CF agents' behaviours that have been implemented in the system.

B.2.1 Squad Leader Behaviour Specifications

This section presents the detailed behaviour specifications of the *squad leader* social identity.

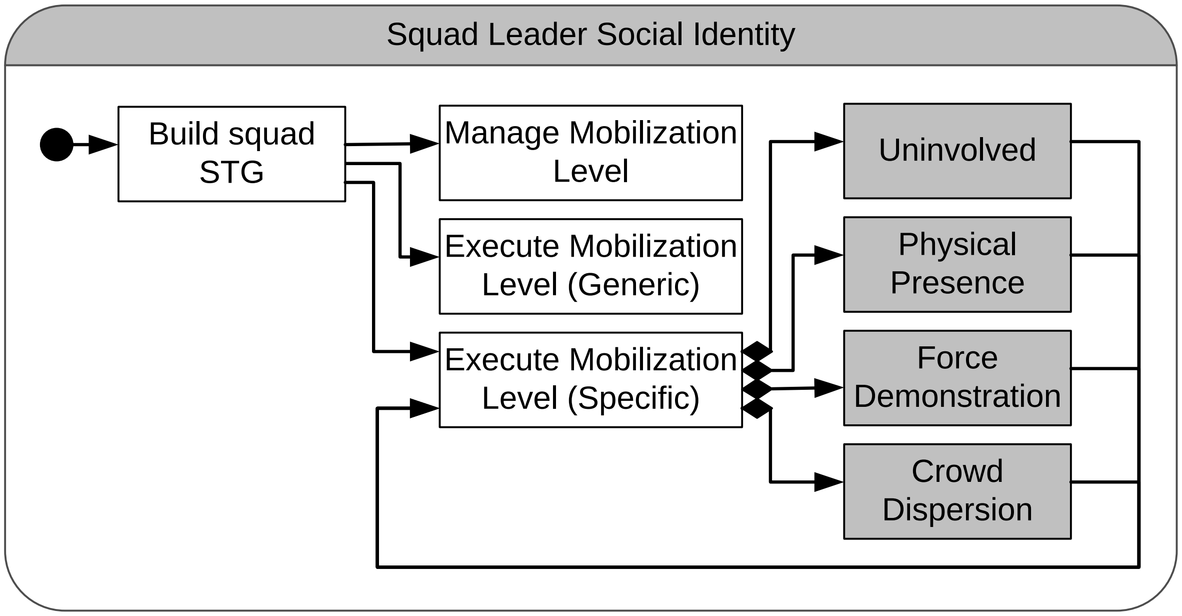

Squad Leader Social Identity
Build squad STG
Manage Mobilization Level
Execute Mobilization Level (Generic)
Execute Mobilization Level (Specific)
Uninvolved
Physical Presence
Force Demonstration
Crowd Dispersion

Figure B.2: Behaviour Specifications: Squad Leader Social Identity

Table B.3: Behaviour Specifications: Squad Leader Social Identity

Squad Leader Social Identity	
Type:	Behaviour
Resource Priority:	N/A
Resources:	N/A
Execution Rules:	N/A
Action:	Manages all aspects necessary to changing mobilization levels.
Completion Rules:	N/A
Comments:	

Table B.4: Behaviour Specifications: Build Squad STG

Build Squad STG	
Type:	Elementary
Resource Priority:	N/A
Resources:	N/A
Execution Rules:	If the agent does have a STG, then ACTIVE. Otherwise, SUCCESS.
Action:	Creates the squad STG.
Completion Rules:	SUCCESS
Comments:	

Table B.5: Behaviour Specifications: Manage Mobilization Level

Manage Mobilization Level	
Type:	Elementary
Resource Priority:	N/A
Resources:	N/A
Execution Rules:	N/A
Action:	Sets the current mobilization level of the squad to the one requested by the commander. Sets the appropriate formation.
Completion Rules:	N/A
Comments:	

Table B.6: Behaviour Specifications: Execute Mobilization Level (Generic)

Execute Mobilization Level (Generic)	
Type:	Elementary
Resource Priority:	agent.SOCIAL_PRIORITY
Resources:	Navigation
Execution Rules:	N/A
Action:	The agent moves to the position appropriate for the current mobilization level.
Completion Rules:	N/A
Comments:	

Table B.7: Behaviour Specifications: Execute Mobilization Level (Specific)

Execute Mobilization Level (Specific)	
Type:	Elementary
Resource Priority:	N/A
Resources:	N/A
Execution Rules:	N/A
Action:	None.
Completion Rules:	SUCCESS
Comments:	

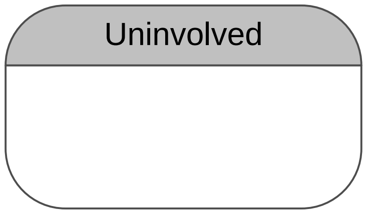


Figure B.3: Behaviour Specifications: Uninvolved

Table B.8: Behaviour Specifications: Uninvolved

Uninvolved	
Type:	Elementary
Resource Priority:	N/A
Resources:	N/A
Execution Rules:	N/A
Action:	None.
Completion Rules:	SUCCESS when the requested mobilization level != *Uninvolved*
Comments:	

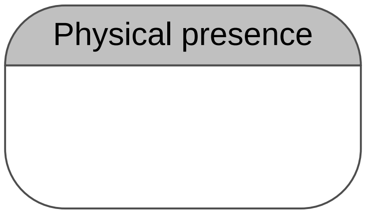


Figure B.4: Behaviour Specifications: Physical Presence

Table B.9: Behaviour Specifications: Physical Presence

Physical Presence	
Type:	Elementary
Resource Priority:	N/A
Resources:	N/A
Execution Rules:	N/A
Action:	None.
Completion Rules:	SUCCESS when the requested mobilization level != *Physical Presence*
Comments:	

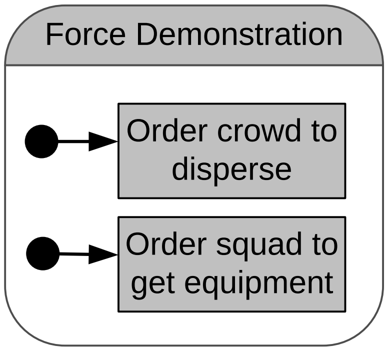


Figure B.5: Behaviour Specifications: Force Demonstration

Table B.10: Behaviour Specifications: Force Demonstration

Force Demonstration	
Type:	Compound
Resource Priority:	N/A
Resources:	N/A
Execution Rules:	N/A
Action:	Orders the squad members to get their complete equipment and periodically asks the crowd to leave.
Completion Rules:	SUCCESS when the requested mobilization level != *Force Demonstration*
Comments:	

Table B.11: Behaviour Specifications: Order Squad to Get Equipment

Order Squad to Get Equipment	
Type:	Elementary
Resource Priority:	N/A
Resources:	N/A
Execution Rules:	N/A
Action:	The squad leader periodically sends an order to its members to go get their equipment.
Completion Rules:	N/A
Comments:	

Table B.12: Behaviour Specifications: Order Crowd to Disperse

Order Crowd to Disperse	
Type:	Elementary
Resource Priority:	N/A
Resources:	N/A
Execution Rules:	N/A
Action:	The agent sends an order to the crowd to disperse every 10 seconds
Completion Rules:	N/A
Comments:	The value 10 has not yet been calibrated.

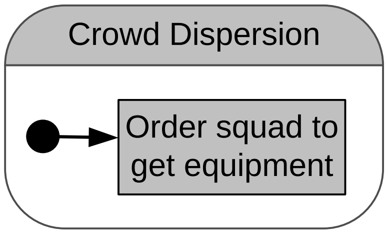


Figure B.6: Behaviour Specifications: Crowd Dispersion

The specifications of *Order Squad To Get Equipment* have been presented in Table B.11 and thus they are not repeated here.

B.2.2 Squad Deputy Leader Behaviour Specifications

The role of the deputy leader is only to start as a member and switch to leader if the formal leader leaves. The deputy leader comes back to a member role when the formal leader comes back. Thus, no specifications are necessary for this social identity.

B.2.3 Squad Member Behaviour Specifications

This section presents the detailed behaviour specifications of the *Squad Member* social identity.

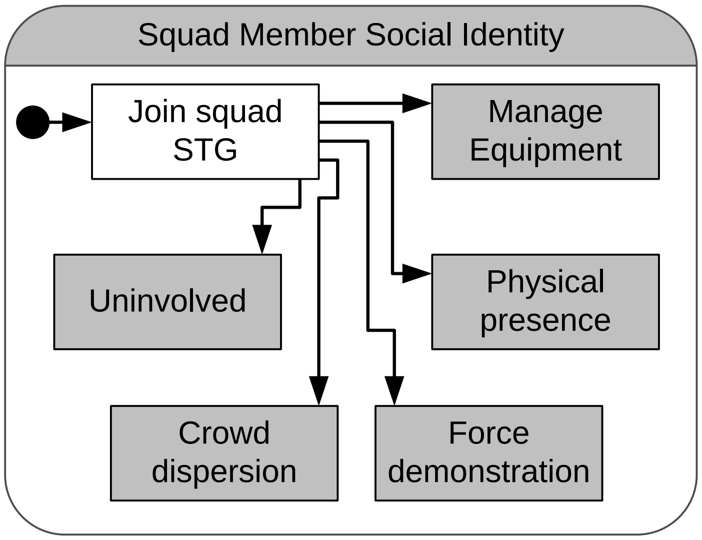


Figure B.7: Behaviour Specifications: Squad Member Social Identity

Table B.13: Behaviour Specifications: Squad Member Social Identity

Squad Member Social Identity	
Type:	Behaviour
Resource Priority:	N/A
Resources:	N/A
Execution Rules:	N/A
Action:	Joins the squad's STG and obeys the squad's orders.
Completion Rules:	N/A
Comments:	The four objectives that represent the four mobilization levels are coded but commented out because they are empty.

It must be noted that the objectives *Uninvolved*, *Physical Presence*, *Force Demonstration*, and *Crowd Dispersion* will not be detailed since they currently perform nothing. In fact, they are commented out in the code. In order to respect the squad's orders, the agent has to wear the proper equipment (ensured by *Manage Equipment* below) and to maintain formation (ensured by the behaviour presented in Sub-Section 4.5.2). No other behaviour is necessary.

Table B.14: Behaviour Specifications: Join Squad STG

Join Squad STG	
Type:	Elementary
Resource Priority:	N/A
Resources:	N/A
Execution Rules:	If the member is already in a STG, then the objective is set to SUCCESS. If the squad does not have a STG yet (the squad leader has not created it), then the objective is set to IDLE.
Action:	Joins the squad STG that manages the formations.
Completion Rules:	SUCCESS (the first time that the action is executed, the member joined the STG).
Comments:	The four successors that represent the four mobilization levels are coded but commented out because they are empty.

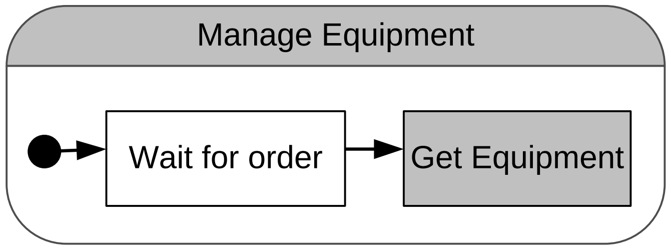


Figure B.8: Behaviour Specifications: Manage Equipment

Table B.15: Behaviour Specifications: Manage Equipment

Manage Equipment	
Type:	Compound
Resource Priority:	N/A
Resources:	N/A
Execution Rules:	Becomes ACTIVE when the mobilization level is *Force Demonstration* or greater
Action:	Handles orders to go get the crowd control equipment.
Completion Rules:	Becomes SUCCESS when the agent has his complete equipment
Comments:	

Table B.16: Behaviour Specifications: Wait for Order

Wait for Order	
Type:	Elementary
Resource Priority:	N/A
Resources:	N/A
Execution Rules:	N/A
Action:	Does nothing.
Completion Rules:	Becomes SUCCESS when the agent has received the order
Comments:	Waits for the first order from the squad leader about getting the equipment, although subsequent orders may be received.

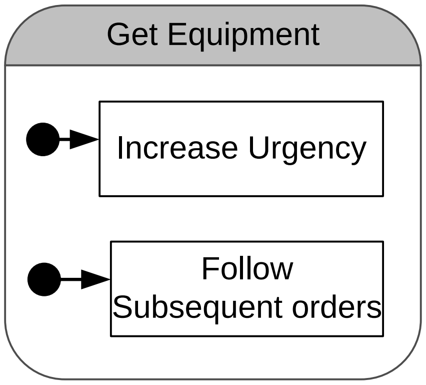


Figure B.9: Behaviour Specifications: Get Equipment

Table B.17: Behaviour Specifications: Get Equipment

Get Equipment	
Type:	Compound
Resource Priority:	N/A
Resources:	N/A
Execution Rules:	N/A
Action:	Handles orders to go get the crowd control equipment.
Completion Rules:	N/A
Comments:	

Table B.18: Behaviour Specifications: Increase Equipment Urgency

Increase Equipment Urgency	
Type:	Elementary
Resource Priority:	N/A
Resources:	N/A
Execution Rules:	N/A
Action:	Increases the priority of getting the equipment.
Completion Rules:	N/A
Comments:	Increases priority by 1%. This value has not been calibrated.

Table B.19: Behaviour Specifications: Follow Subsequent Orders

Follow Subsequent Orders	
Type:	Elementary
Resource Priority:	N/A
Resources:	N/A
Execution Rules:	ACTIVE only when the agent just received another order to get the equipment. IDLE otherwise.
Action:	Increases the priority of getting the equipment .
Completion Rules:	N/A
Comments:	Doubles priority. This value has not been calibrated.

B.2.4 Chemical Officer Behaviour Specifications

The chemical officer is not a social identity like the squad deputy leader. Rather, a simplified behaviour was implemented for this agent. Specifically, a chemical officer is a squad member that has an extra Java objective. This objective merely monitors every ten seconds for a STG with an excessively aggressive behaviour and throws a tear gas can at that STG if allowed. A STG is deemed excessively aggressive when its aggressiveness level (based on its history of actions) is above a certain level, customized by the user. When the use of gas cans is not allowed, this behaviour is not executed.

B.2.5 Rifle Officer Behaviour Specifications

The rifle officer is not a social identity like the squad deputy leader. Rather, a simplified behaviour was implemented for this agent. Specifically, a rifle officer is a squad member that has an extra Java objective. This objective merely monitors every ten seconds for an agent with an excessively aggressive behaviour and shoots at that agent if allowed. An agent is deemed excessively aggressive when his aggressiveness level (based on his history of actions) is above a certain level, customized by the user. When the use of plastic bullets is not allowed, this behaviour is not executed.

B.3 Crowd Models

This section presents in a detailed manner all of the crowd agents' behaviours that have been implemented in the system.

B.3.1 Common Crowd Objectives

This section presents specifications of simple objectives that are the building blocks of more complex behaviours executed by crowd members.

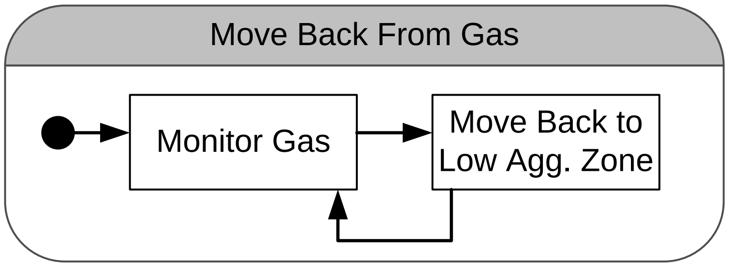


Figure B.10: Behaviour Specifications: Move Back from Gas

Table B.20: Behaviour Specifications: Move Back from Gas

Move Back from Gas	
Type:	Compound
Resource Priority:	N/A
Resources:	N/A
Execution Rules:	N/A
Action:	Does nothing.
Completion Rules:	N/A
Comments:	

Table B.21: Behaviour Specifications: Monitor Gas

Monitor Gas	
Type:	Elementary
Resource Priority:	N/A
Resources:	N/A
Execution Rules:	N/A
Action:	Does nothing.
Completion Rules:	SUCCESS when the agent is in contact with gas
Comments:	

Table B.22: Behaviour Specifications: Move Back to Low Aggressiveness Zone

Move Back to Low Aggressiveness Zone	
Type:	Elementary
Resource Priority:	99
Resources:	Navigation
Execution Rules:	N/A
Action:	Makes the agent go at a randomly selected point in a low aggressiveness zone.
Completion Rules:	SUCCESS when the agent is at its destination
Comments:	The value 99 has not been calibrated.

Table B.23: Behaviour Specifications: Fight with Squad Members

Fight with Squad Members	
Type:	Elementary
Resource Priority:	2 / distance to closest squad
Resources:	Navigation, arms
Execution Rules:	ACTIVE when the agent is close enough. IDLE otherwise.
Action:	Fights with a squad leader.
Completion Rules:	N/A
Comments:	Currently fights only the squad leader (and not the members). The value 2 has not yet been calibrated.

Table B.24: Behaviour Specifications: Flee and Exit

Flee and Exit	
Type:	Elementary
Resource Priority:	100
Resources:	Navigation
Execution Rules:	N/A
Action:	Leaves the STG if the agent is in one and runs to the nearest exit point.
Completion Rules:	When close to the exit point, the agent exits the simulation.
Comments:	The value 100 has not yet been calibrated.

Table B.25: Behaviour Specifications: Insult CF

Insult CF	
Type:	Elementary
Resource Priority:	agent.SOCIAL_PRIORITY + 1
Resources:	Mouth
Execution Rules:	ACTIVE when the agent is close enough to CF. IDLE otherwise.
Action:	Insults a squad.
Completion Rules:	N/A
Comments:	The priority is at agent.SOCIAL_PRIORITY + 1 just so it is higher than yell.

Table B.26: Behaviour Specifications: Show Banner

Show Banner	
Type:	Elementary
Resource Priority:	agent.SOCIAL_PRIORITY
Resources:	Arms, banner
Execution Rules:	N/A
Action:	Shows a banner (can be seen by the user).
Completion Rules:	N/A
Comments:	

Table B.27: Behaviour Specifications: Step Out of Gas

Step Out of Gas	
Type:	Elementary
Resource Priority:	99
Resources:	Navigation
Execution Rules:	ACTIVE when the agent is in contact with gas. IDLE otherwise.
Action:	Runs away from the gas cloud (just far enough not to be in the cloud anymore)
Completion Rules:	N/A
Comments:	The value 99 has not yet been calibrated.

Table B.28: Behaviour Specifications: Throw Molotov Cocktail

Throw Molotov Cocktail	
Type:	Elementary
Resource Priority:	agent.SOCIAL_PRIORITY + 5
Resources:	Arms, Molotov Cocktail
Execution Rules:	ACTIVE when the agent is at a good distance (not too close, not too far), IDLE otherwise
Action:	Throws a cocktail to the nearest squad STG.
Completion Rules:	N/A
Comments:	The value 5 has not been calibrated.

Table B.29: Behaviour Specifications: Walk to Ultimate Destination

Walk to Ultimate Destination	
Type:	Elementary
Resource Priority:	agent.SOCIAL_PRIORITY
Resources:	Navigation
Execution Rules:	N/A
Action:	Walks to the location where the agent would ultimately like to go.
Completion Rules:	SUCCESS when the agent is at destination
Comments:	

Table B.30: Behaviour Specifications: Yell

Yell	
Type:	Elementary
Resource Priority:	agent.SOCIAL_PRIORITY
Resources:	Mouth
Execution Rules:	N/A
Action:	Yells (at no one in particular).
Completion Rules:	N/A
Comments:	

B.3.2 Bystander Behaviour Specifications

This section presents the detailed behaviour specifications of the *Bystander* social identity.

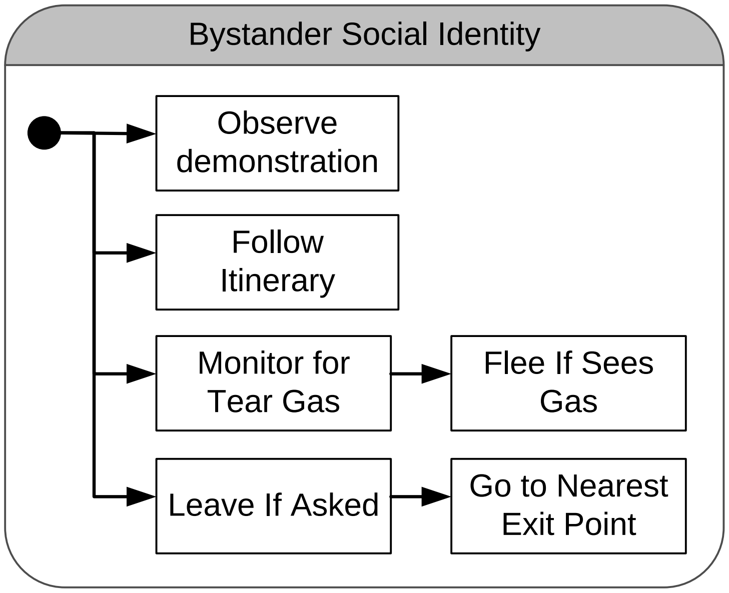


Figure B.11: Behaviour Specifications: Bystander Social Identity

Table B.31: Behaviour Specifications: Bystander Social Identity

Bystander Social Identity	
Type:	Behaviour
Resource Priority:	N/A
Resources:	N/A
Execution Rules:	N/A
Action:	Walks around in the city possibly observing a protest.
Completion Rules:	N/A
Comments:	

Table B.32: Behaviour Specifications: Observe Demonstration

Observe Demonstration	
Type:	Elementary
Resource Priority:	agent.SOCIAL_PRIORITY + 1
Resources:	Navigation
Execution Rules:	ACTIVE when there is something interesting to observe. IDLE otherwise.
Action:	The bystander stops moving but turns toward the most aggressive and closest STG.
Completion Rules:	N/A
Comments:	

Table B.33: Behaviour Specifications: Follow Itinerary

Follow Itinerary	
Type:	Elementary
Resource Priority:	agent.SOCIAL_PRIORITY
Resources:	Navigation
Execution Rules:	N/A
Action:	The agent follows an itinerary around the touristic locations near the parliament.
Completion Rules:	When completed, removes the agent from the simulation.
Comments:	

Table B.34: Behaviour Specifications: Monitor for Tear Gas

Monitor for Tear Gas	
Type:	Elementary
Resource Priority:	N/A
Resources:	N/A
Execution Rules:	N/A
Action:	Does nothing.
Completion Rules:	SUCCESS when the agent sees a gas cloud
Comments:	

Table B.35: Behaviour Specifications: Flee If Sees Gas

Flee If Sees Gas	
Type:	Elementary
Resource Priority:	maximum of (10) and (5 / distance to closest gas)
Resources:	Navigation
Execution Rules:	N/A
Action:	The agent runs towards the closest exit point.
Completion Rules:	When arrived, removes the agent from the simulation
Comments:	The values 10 and 5 have not been calibrated

Table B.36: Behaviour Specifications: Leave If Asked

Leave If Asked	
Type:	Elementary
Resource Priority:	agent.SOCIAL_PRIORITY + 1
Resources:	Navigation
Execution Rules:	N/A
Action:	Does nothing.
Completion Rules:	SUCCESS
Comments:	

Table B.37: Behaviour Specifications: Go to Nearest Exit Point

Go to Nearest Exit Point	
Type:	Elementary
Resource Priority:	agent.SOCIAL_PRIORITY + 2
Resources:	Navigation
Execution Rules:	N/A
Action:	Goes to the nearest exit point. The agent is not allowed to create or join STGs.
Completion Rules:	When arrived, removes the agent from the simulation
Comments:	

B.3.3 Demonstrator Behaviour Specifications

This section presents the detailed behaviour specifications of the *Demonstrator* social identity.

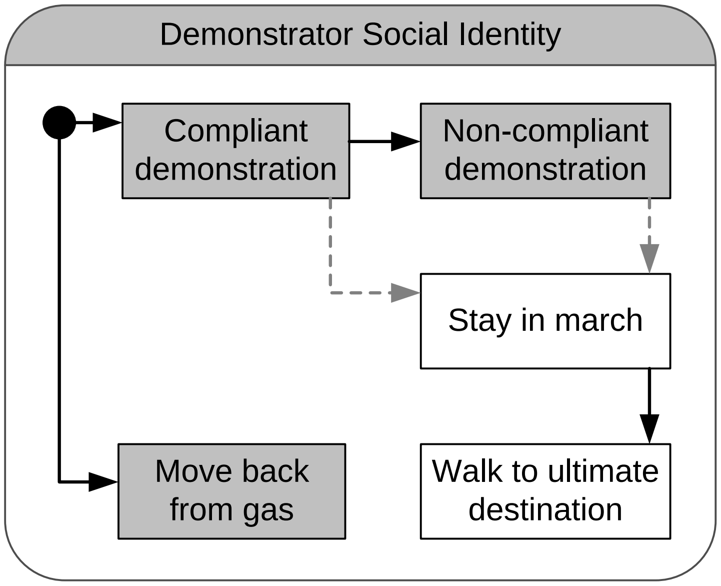

Demonstrator Social Identity
Compliant demonstration
Non-compliant demonstration
Stay in march
Move back from gas
Walk to ultimate destination

Figure B.12: Behaviour Specifications: Demonstrator Social Identity

Table B.38: Behaviour Specifications: Demonstrator Social Identity

Demonstrator Social Identity	
Type:	Behaviour
Resource Priority:	N/A
Resources:	N/A
Execution Rules:	N/A
Action:	Shows support for a cause in a mildly aggressive manner. May join a march.
Completion Rules:	N/A
Comments:	

The specifications of *Move Back from Gas* have been presented in Table B.20 and thus they are not repeated here.

The specifications of *Walk to Ultimate Destination* have been presented in Table B.29 and thus they are not repeated here.

Table B.39: Behaviour Specifications: Stay in March

Stay in March	
Type:	Elementary
Resource Priority:	N/A
Resources:	N/A
Execution Rules:	N/A
Action:	Does nothing (the agent stays in formation but does not show a banner or chant).
Completion Rules:	SUCCESS when the march is over
Comments:	

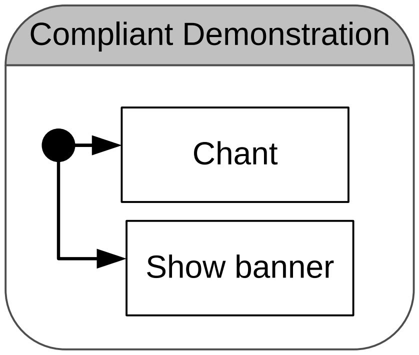


Figure B.13: Behaviour Specifications: Compliant Demonstration

Table B.40: Behaviour Specifications: Compliant Demonstration

Compliant Demonstration	
Type:	Compound
Resource Priority:	N/A
Resources:	N/A
Execution Rules:	ACTIVE if the agent is in a STG, IDLE otherwise
Action:	Protests and complies with orders from CF.
Completion Rules:	SUCCESS if the control forces are perceived as too aggressive. FAILURE when the agent has protested long enough.
Comments:	

Table B.41: Behaviour Specifications: Chant

Chant	
Type:	Elementary
Resource Priority:	agent.SOCIAL_PRIORITY
Resources:	Mouth
Execution Rules:	N/A
Action:	The agent chants a slogan.
Completion Rules:	N/A
Comments:	

The specifications of *Show Banner* have been presented in Table B.26 and thus they are not repeated here.

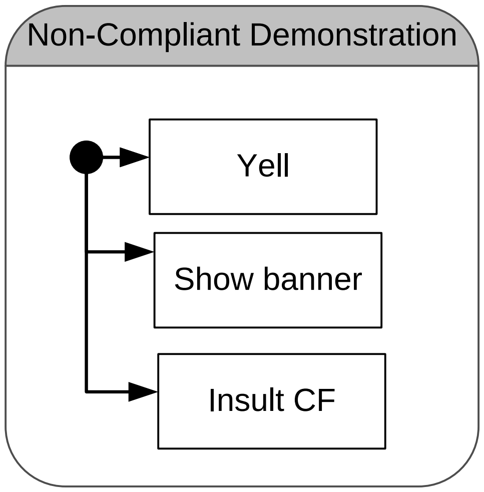


Figure B.14: Behaviour Specifications: Non-Compliant Demonstration

Table B.42: Behaviour Specifications: Non-Compliant Demonstration

Non-Compliant Demonstration	
Type:	Compound
Resource Priority:	N/A
Resources:	N/A
Execution Rules:	N/A
Action:	Protests but does not comply with orders from CF.
Completion Rules:	FAILURE when the agent has protested long enough
Comments:	

The specifications of *Yell* have been presented in Table B.30 and thus they are not repeated here.

The specifications of *Show Banner* have been presented in Table B.26 and thus they are not repeated here.

The specifications of *Insult CF* have been presented in Table B.25 and thus they are not repeated here.

B.3.4 Demonstrator Leader Behaviour Specifications

This section presents the detailed behaviour specifications of the *Demonstrator Leader* social identity.

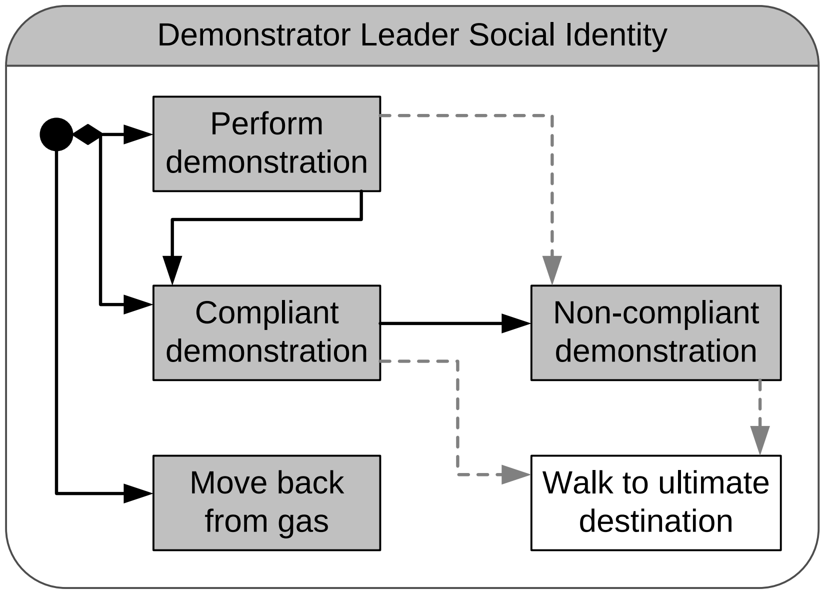

Demonstrator Leader Social Identity
Perform demonstration
Compliant demonstration
Non-compliant demonstration
Move back from gas
Walk to ultimate destination

Figure B.15: Behaviour Specifications: Demonstrator Leader Social Identity

Table B.43: Behaviour Specifications: Demonstrator Leader Social Identity

Demonstrator Leader Social Identity	
Type:	Behaviour
Resource Priority:	N/A
Resources:	N/A
Execution Rules:	N/A
Action:	Shows support for a cause in a mildly aggressive manner. May lead a march.
Completion Rules:	N/A
Comments:	

The specifications of *Move Back from Gas* have been presented in Table B.20 and thus they are not repeated here.

The specifications of *Walk to Ultimate Destination* have been presented in Table B.29 and thus they are not repeated here.

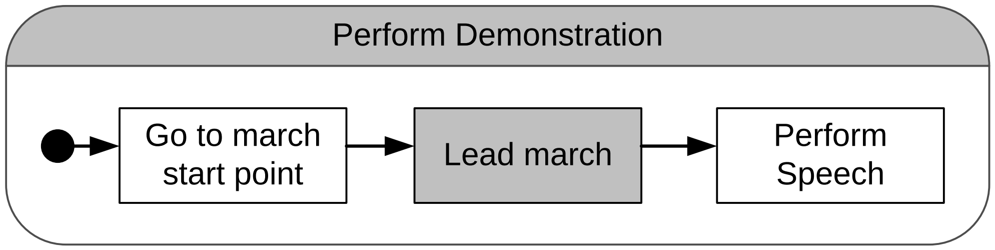


Figure B.16: Behaviour Specifications: Perform Demonstration

Table B.44: Behaviour Specifications: Perform Demonstration

Perform Demonstration	
Type:	Compound
Resource Priority:	N/A
Resources:	N/A
Execution Rules:	IDLE if not in STG, FAILURE if the CF are too aggressive, ACTIVE otherwise.
Action:	Handles the entire process of a leading a march.
Completion Rules:	SUCCESS when *Perform Speech* is completed
Comments:	

Table B.45: Behaviour Specifications: Go to March Start Point

Go to March Start Point	
Type:	Elementary
Resource Priority:	agent.SOCIAL_PRIORITY
Resources:	Navigation
Execution Rules:	N/A
Action:	The agent goes to where the itinerary of the march starts.
Completion Rules:	SUCCESS when at destination
Comments:	

Table B.46: Behaviour Specifications: Perform Speech

Perform Speech	
Type:	Elementary
Resource Priority:	agent.SOCIAL_PRIORITY
Resources:	Navigation, mouth
Execution Rules:	N/A
Action:	Delivers a verbal address.
Completion Rules:	FINISH when the speech is over
Comments:	

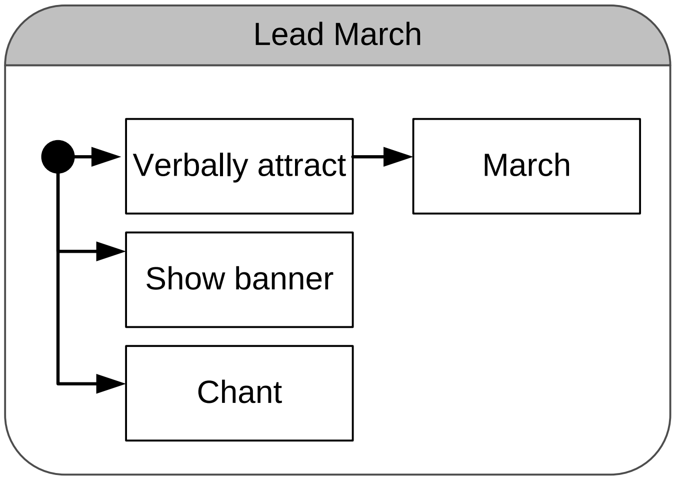


Figure B.17: Behaviour Specifications: Lead March

Table B.47: Behaviour Specifications: Lead March

Lead March	
Type:	Compound
Resource Priority:	N/A
Resources:	N/A
Execution Rules:	ACTIVE if the agent is in a STG, IDLE otherwise
Action:	Makes the agent navigate and entertain his STG while he follows the itinerary.
Completion Rules:	N/A
Comments:	

The specifications of *Show Banner* have been presented in Table B.26 and thus they are not repeated here.

The specifications of *Chant* have been presented in Table B.41 and thus they are not repeated here.

Table B.48: Behaviour Specifications: Verbally Attract Crowd

Verbally Attract Crowd	
Type:	Elementary
Resource Priority:	agent.SOCIAL_PRIORITY + 1
Resources:	Mouth
Execution Rules:	N/A
Action:	Attracts other agents to join.
Completion Rules:	SUCCESS when the march can begin.
Comments:	

Table B.49: Behaviour Specifications: March

March	
Type:	Elementary
Resource Priority:	agent.SOCIAL_PRIORITY
Resources:	Navigation
Execution Rules:	N/A
Action:	Follows the march's itinerary.
Completion Rules:	SUCCESS when the itinerary is complete.
Comments:	

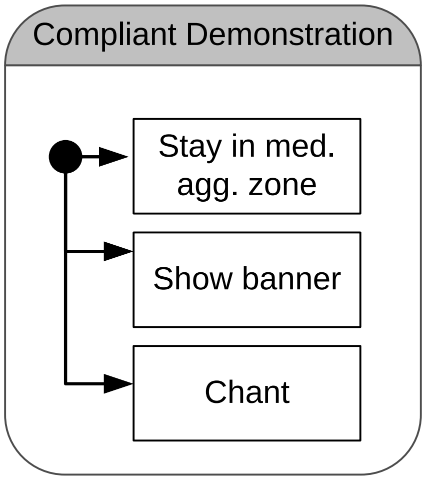


Figure B.18: Behaviour Specifications: Compliant Demonstration

Table B.50: Behaviour Specifications: Compliant Demonstration

Compliant Demonstration	
Type:	Compound
Resource Priority:	N/A
Resources:	N/A
Execution Rules:	ACTIVE if the agent is in a STG, IDLE otherwise
Action:	Protests and complies with orders from CF.
Completion Rules:	SUCCESS if the CF becomes too aggressive, FAILURE if no new protestor has joined recently.
Comments:	

The specifications of *Show Banner* have been presented in Table B.26 and thus they are not repeated here.

The specifications of *Chant* have been presented in Table B.41 and thus they are not repeated here.

Table B.51: Behaviour Specifications: Stay in Medium Aggressiveness Zone

Stay in Medium Aggressiveness Zone	
Type:	Elementary
Resource Priority:	agent.SOCIAL_PRIORITY
Resources:	Navigation
Execution Rules:	N/A
Action:	Stays in a medium aggressiveness zone.
Completion Rules:	N/A
Comments:	

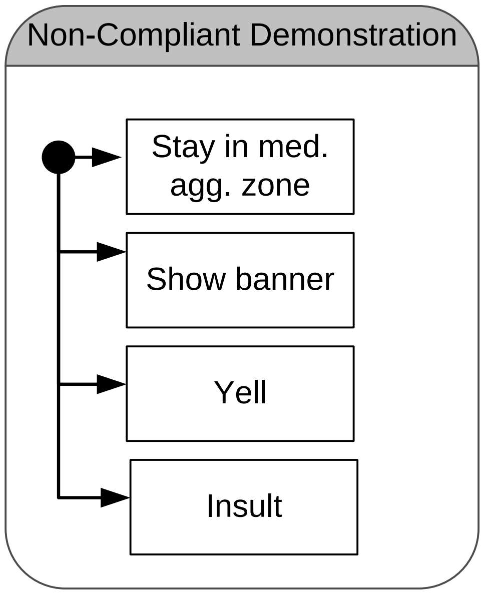


Figure B.19: Behaviour Specifications: Non-Compliant Demonstration

Table B.52: Behaviour Specifications: Non-Compliant Demonstration

Non-Compliant Demonstration	
Type:	Compound
Resource Priority:	N/A
Resources:	N/A
Execution Rules:	ACTIVE if the agent is in a STG, IDLE otherwise
Action:	Protests but does not comply with orders from CF.
Completion Rules:	FAILURE if no new protestor has joined recently
Comments:	

The specifications of *Show Banner* have been presented in Table B.26 and thus they are not repeated here.

The specifications of *Yell* have been presented in Table B.30 and thus they are not repeated here.

The specifications of *Insult CF* have been presented in Table B.25 and thus they are not repeated here.

Table B.53: Behaviour Specifications: Stay in High Aggressiveness Zone

Stay in High Aggressiveness Zone	
Type:	Elementary
Resource Priority:	agent.SOCIAL_PRIORITY
Resources:	Navigation
Execution Rules:	N/A
Action:	Stays in a high aggressiveness zone.
Completion Rules:	N/A
Comments:	

B.3.5 Instigator Behaviour Specifications

This section presents the detailed behaviour specifications of the *Instigator* social identity.

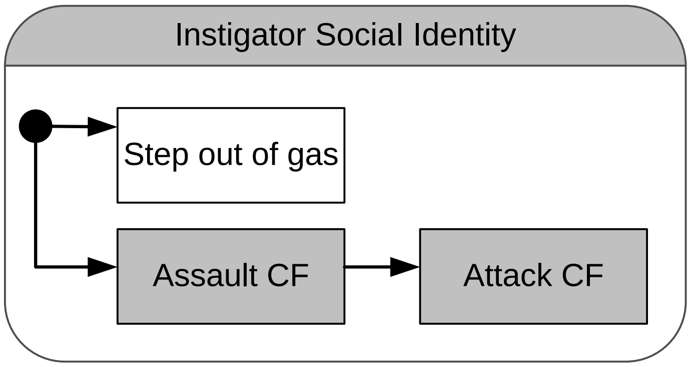


Figure B.20: Behaviour Specifications: Instigator Social Identity

Table B.54: Behaviour Specifications: Instigator Social Identity

Instigator Social Identity	
Type:	Behaviour
Resource Priority:	N/A
Resources:	N/A
Execution Rules:	N/A
Action:	Protests in an aggressive manner.
Completion Rules:	N/A
Comments:	

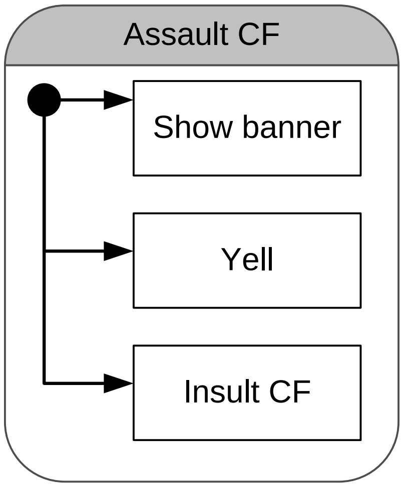


Figure B.21: Behaviour Specifications: Assault CF

Table B.55: Behaviour Specifications: Assault CF

Assault CF	
Type:	Compound
Resource Priority:	N/A
Resources:	N/A
Execution Rules:	N/A
Action:	Confronts the CF in an aggressive but non-violent manner.
Completion Rules:	SUCCESS if a squad is in *Crowd Dispersion* mobilization level or tear gas has been perceived.
Comments:	

The specifications of *Show Banner* have been presented in Table B.26 and thus they are not repeated here.

The specifications of *Yell* have been presented in Table B.30 and thus they are not repeated here.

The specifications of *Insult CF* have been presented in Table B.25 and thus they are not repeated here.

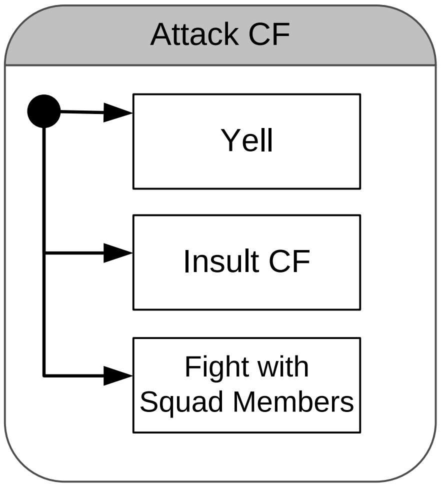


Figure B.22: Behaviour Specifications: Attack CF

Table B.56: Behaviour Specifications: Attack CF

Attack CF	
Type:	Compound
Resource Priority:	N/A
Resources:	N/A
Execution Rules:	N/A
Action:	Confronts the CF in an aggressive and violent manner.
Completion Rules:	N/A
Comments:	Behaviours to throw projectiles and cheer at observed aggressive actions are ready to be implemented

The specifications of *Yell* have been presented in Table B.30 and thus they are not repeated here.

The specifications of *Insult CF* have been presented in Table B.25 and thus they are not repeated here.

The specifications of *Fight with Squad Members* have been presented in Table B.23 and thus they are not repeated here.

B.3.6 Instigator Leader Behaviour Specifications

This section presents the detailed behaviour specifications of the *Instigator Leader* social identity.

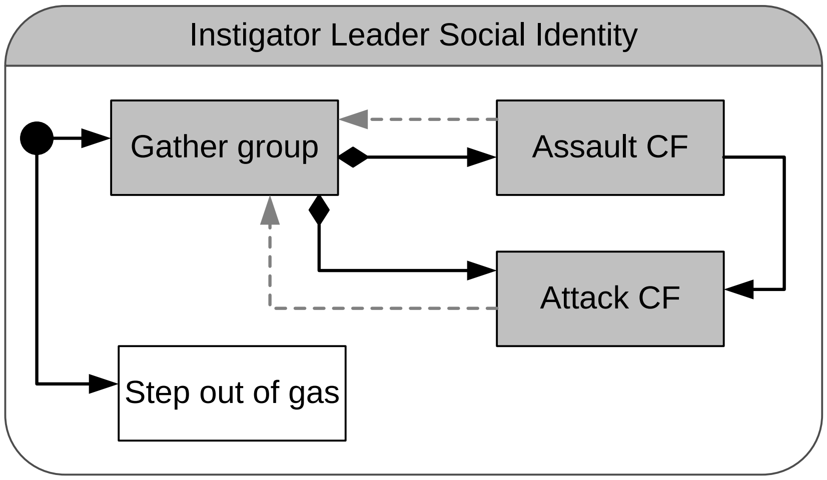


Figure B.23: Behaviour Specifications: Instigator Leader Social Identity

Table B.57: Behaviour Specifications: Instigator Leader Social Identity

Instigator Leader Social Identity	
Type:	Behaviour
Resource Priority:	N/A
Resources:	N/A
Execution Rules:	N/A
Action:	Gathers a group to protest in an aggressive manner.
Completion Rules:	N/A
Comments:	

The specifications of *Step Out of Gas* have been presented in Table B.27 and thus they are not repeated here.

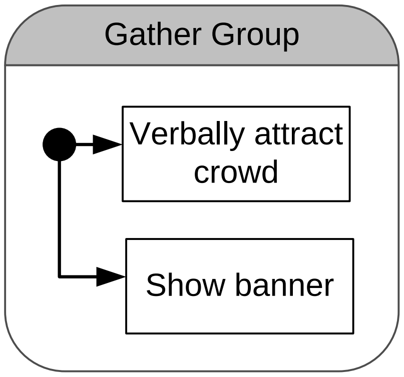


Figure B.24: Behaviour Specifications: Gather Group

Table B.58: Behaviour Specifications: Gather Group

Gather Group	
Type:	Compound
Resource Priority:	N/A
Resources:	N/A
Execution Rules:	N/A
Action:	Invite other crowd members to join the intigators STG.
Completion Rules:	SUCCESS when the STG is large enough
Comments:	

Table B.59: Behaviour Specifications: Verbally Attract Crowd

Verbally Attract Crowd	
Type:	Elementary
Resource Priority:	agent.SOCIAL_PRIORITY
Resources:	Mouth
Execution Rules:	N/A
Action:	Asks other agents to join the STG.
Completion Rules:	N/A
Comments:	

The specifications of *Show Banner* have been presented in Table B.26 and thus they are not repeated here.

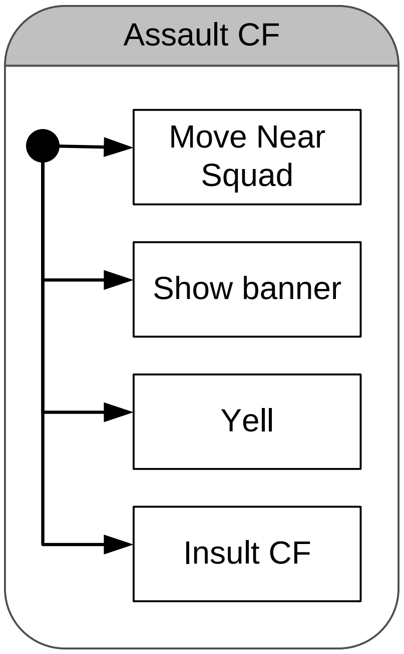


Figure B.25: Behaviour Specifications: Assault CF

Table B.60: Behaviour Specifications: Assault CF

Assault CF	
Type:	Compound
Resource Priority:	N/A
Resources:	N/A
Execution Rules:	N/A
Action:	Confronts the CF in an aggressive but non-violent manner.
Completion Rules:	SUCCESS if a squad in *Crowd Dispersion* mobilization level is perceived or tear gas is perceived. FAILURE if the STG becomes too small.
Comments:	

The specifications of *Show Banner* have been presented in Table B.26 and thus they are not repeated here.

The specifications of *Yell* have been presented in Table B.30 and thus they are not repeated here.

The specifications of *Insult CF* have been presented in Table B.25 and thus they are not repeated here.

Table B.61: Behaviour Specifications: Move Near Squad

Move Near Squad	
Type:	Elementary
Resource Priority:	agent.SOCIAL_PRIORITY
Resources:	Navigation
Execution Rules:	N/A
Action:	Brings the group near a CF squad.
Completion Rules:	N/A
Comments:	

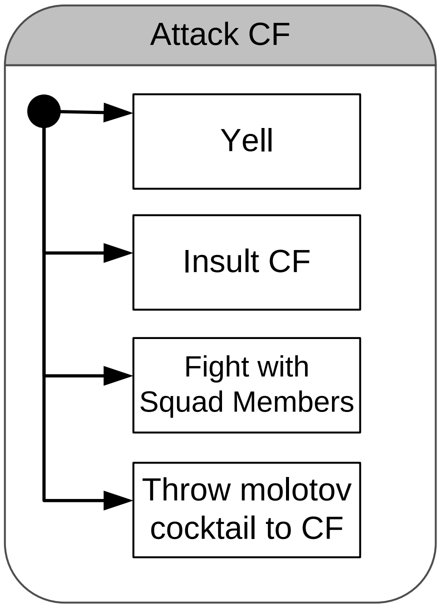


Figure B.26: Behaviour Specifications: Attack CF

Table B.62: Behaviour Specifications: Attack CF

Attack CF	
Type:	Compound
Resource Priority:	N/A
Resources:	N/A
Execution Rules:	N/A
Action:	Confronts the CF in an aggressive and violent manner.
Completion Rules:	FAILURE if the STG becomes too small.
Comments:	Behaviours to throw back tear gas cans are ready to be implemented

The specifications of *Yell* have been presented in Table B.30 and thus they are not repeated here.

The specifications of *Insult CF* have been presented in Table B.25 and thus they are not repeated here.

The specifications of *Fight with Squad Members* have been presented in Table B.23 and thus they are not repeated here.

The specifications of *Throw Molotov Cocktail* have been presented in Table B.28 and thus they are not repeated here.

Appendix C

Performance Tests

Performance is often a critical issue in multi-agent simulators. Effectively, multi-agent simulations are some of the most computationally-demanding types of software applications. Because the Crowd-MAGS project is a simulation project and not an animation project, it is not necessary that the application runs in real-time. However, a certain level of performance is expected in order for the application to be convenient for users.

As explained in Sub-Section 5.5.2, performance is measured as a ratio of simulated time to real time. Simulated time is an absolute temporal base that is not influenced by external factors like the computer's power and that is known by agents. Real time is another temporal base that represents the time spent in reality while the simulation runs. The performance is measured as follows: performance = simulated time / real time. Three situations are possible concerning performance:

- Performance > 1: The simulation runs faster than reality. Agents seem to be moving at high speeds and visual analysis may be difficult.
- Performance $= 1$: The simulation runs exactly at the same speed as reality (i.e. it runs in real-time).
- Performance < 1: The simulation runs slower than reality. Everything is moving slowly and analysis takes more time.

For convenience's sake, it is recommended to keep performance at a ratio of at least 0.5 X. Also, speeds above 5 X are generally too fast to follow and are recommended only if the user does not need to watch the simulation unfold.

In addition of speed performance, it is important to keep in mind the amount of memory used by the simulation. Since the application was written in Java, this chapter refers to the amount of memory used by the Java Virtual Machine (VM). For example, tests involving 1000 agents failed due to lack of memory because more than 1600 MB of available RAM would have been necessary for the VM to run such heavy scenarios. However, tests were carried out with 900 agents in console mode (no GUI) and despite the very slow performance (< 0.1 X), the simulation ran.

All tests were run on the same computer, which is described in Table C.1. Section C.1 presents early tests that were executed with the initial version of Crowd-MAGS. These tests were run to find out what were the bottlenecks in the system so that efforts could be focused on improving these components. Next, Section C.2 dicusses results of tests run on an improved version of the application. Finally, tests on the final version with the scenarios introduced in Section 6.5 are presented in Section C.3.

Table C.1: Test Computer

Component	**Specification**
CPU	Intel Core 2 Quad Pro Q6600 2.4 GHz
RAM	2GB 1066 MHz DDR2 non-ECC
GPU	ASUS EAH3850 PCI-E 512 MB
Physics	PhysX (software mode)

C.1 Tests with Crowd-MAGS' Initial Version

This section presents tests that were run with the first version of the Crowd-MAGS application. They were executed to find the system limits and guide the development of the application and scenarios. Sub-Section C.1.1 presents the tests, Sub-Section C.1.2 presents the analysis performed to identify the components that slow down the system most, and finally Sub-Section C.1.3 presents conclusions about the initial system performance.

C.1.1 Tests

This section presents tests that are based on a scenario similar to those devised in Section 6.5. The number of squads is always three (5 agents in each) and the total number of agents is much lower because the system was not very efficient. There are two main parts during this scenario, hereafter called 'Before fences' and 'After fences'. In the scenario, crowd members gather and go protest in front of the fences. Once the control forces go get their equipment, the instigators try to break through the fences. At this moment, the second part begins and the crowd walks in the prohibited zone. From this point, the agents are packed more closely and thus the performance drops (due to the high density and the obstacle avoidance algorithm).

The timing results given below indicate how much time was necessary to simulate one second of real time in each part. Obviously, if the time taken is over 1000 ms, then the performance is below 1 X. It is important to note that in the initial version of Crowd-MAGS, PLAMAGS was configured to prevent the simulation from going faster than real time. Hence, the optimal result to reach is 1000 ms of real time to simulate 1000 ms of simulation time.

C.1.1.1 Test #1

Table C.2 shows a test that was executed with a total of **51** agents and no tear gas.

Table C.2: Settings for Test #1

Agent Type	**Quantity**
Control Forces	15
Bystanders	6
Demonstrators	16
Demonstrator Leaders	1
Instigators	10
Instigator Leaders	3
TOTAL	51

The following indicate the real time necessary to simulate one second before and after the protestors break through the fences.

- Before fences: 1100 ms (0.91 X)
- After fences: 1150 ms (0.87 X)

C.1.1.2 Test #2

Table C.2 shows a test that was executed with a total of **101** agents and no tear gas.

The following indicate the real time necessary to simulate one second before and after the protestors break through the fences.

- Before fences: 1525 ms (0.66 X)
- After fences: 1725 ms (0.58 X)

The test was run once more without physics to see how the system would react. Due to the lack of physics, agents could not break through the fences.

- Before fences: 1600 ms (0.63 X)
- After fences: N/A

Table C.3: Settings for Test #2

Agent Type	**Quantity**
Control Forces	15
Bystanders	6
Demonstrators	66
Demonstrator Leaders	1
Instigators	10
Instigator Leaders	3
TOTAL	101

C.1.1.3 Test #3

Table C.2 shows a test that was executed with a total of **42** agents and the use of tear gas.

Table C.4: Settings for Test #3

Agent Type	**Quantity**
Control Forces	15
Bystanders	6
Demonstrators	7
Demonstrator Leaders	1
Instigators	10
Instigator Leaders	3
TOTAL	42

The following indicate the real time necessary to simulate one second before and after the protestors break through the fences. The test was run once without tear gas and once with heavy use of tear gas. In the latter case, crowd members fled quickly so the timing results for the second part were not meaningful. Thus, they are not presented.

- Before fences (without tear gas clouds): 1060 ms (0.94 X)
- Before fences: (with 10 tear gas clouds): 1215 ms (0.82 X)

C.1.1.4 Test #4

Here, a set of tests with an increasing number of demonstrators was run. The results are shown in Figure C.1. Due to the time necessary for each run, a few tests were skipped.

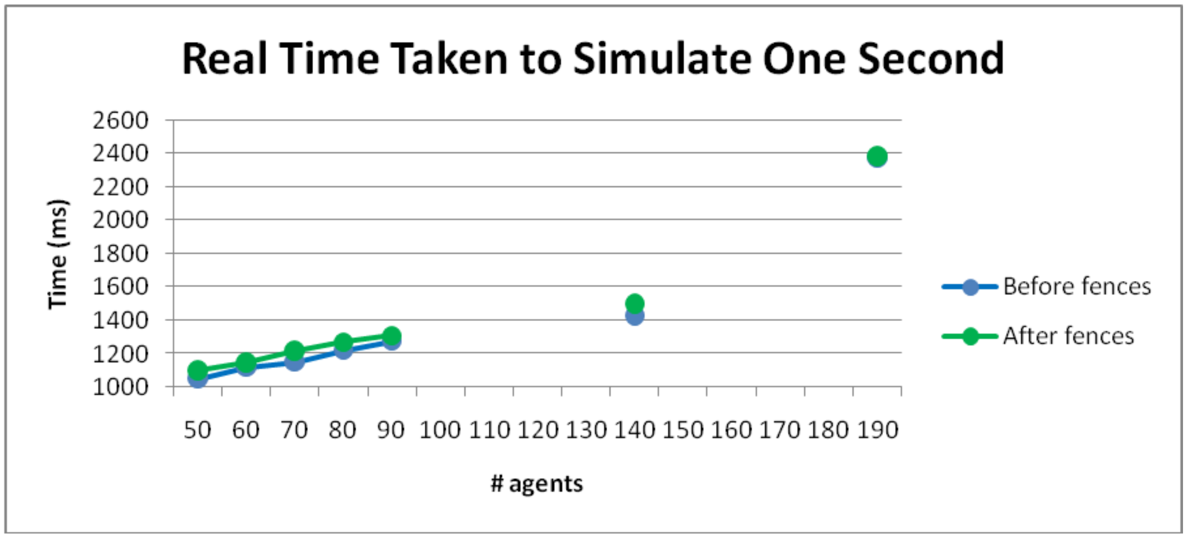


Figure C.1: Results of Initial Tests

C.1.2 Factors Influencing Performance

This sub-section presents the analysis performed to discover what caused the slowdowns observed in the system. The results were interesting, because most of the suspicions turned out to bear no measurable influence on the performance of the system.

C.1.2.1 Assertions

Many Java assertions are used to help debugging and ensure correctness of certain parts of the code. Theoretically, these assertions, when validated, affect the system only by slowing it down (i.e. they have no side effects). Fortunately, it has been verified that the assertions in place do not have any measurable impact on the performance of the system. Tests were run with and without assertions and the results were similar in all cases.

C.1.2.2 Gas

Simulation of tear gas was expected to be a large performance killer in our system. However, it has been shown that three tear gas cans do not show any measurable impact on the performance of the system. However, ten cans of gas slowed down the system by 150 ms per simulated second. These numbers are quite acceptable and thus the simulation of tear gas should not pose any problem under reasonable conditions.

C.1.2.3 Interest Zones

The display of interest zones was not tested formally, but through informal trials, all indications point to a non-measurable impact on performance.

C.1.2.4 Screen Display

It was suspected that showing a lot of agents on screen would slow down the simulation compared to turning the camera towards a wall, for example. However, no matter what the camera angle is and how many agents are on the screen, there is no measurable impact on the system performance. Tests were run with an angle that showed all agents and others with an angle that showed a wall and the results were similar in all cases.

C.1.2.5 Physics

The calculation of physics is necessary to ensure that the movement of all agents and objects in the simulation is plausible. For example, agents apply a certain force on a fence but this one moves only when enough agents are pushing. All calculations can be done in software mode, or using a dedicated card in the computer. All tests were run using software calculations, which are much slower.

Surprisingly, the performance stayed constant or improved when the physics calculations were turned on (compared to a simulation with no physics). Thus, the physics engine does not have a negative on system performance.

C.1.2.6 Perception

The perception of agents and objects was suspected to be a large bottleneck in the system. Tests were run with low, medium, and high densities of agents to see if the perception algorithm needed to be improved. Table C.5 shows that perception takes up a negligible time in each simulation iteration.

Table C.5: Time Taken by the Perception Algorithm

Density	**Real Time for 1 Second**	**Time for Perception**	**Proportion**
Low	1130 ms	15 ms	1.3 %
Medium	1675 ms	43 ms	2.5 %
High	2225 ms	61 ms	2.75 %

C.1.3 Conclusions

From the findings in this section, all plausible sources of slowdowns except for the core elements of the PLAMAGS language have been eliminated. At this point, only the behaviour engine, and especially the management of resources, can be suspected of being the bottlenecks in the system. It is believed that the interpretation of the language (rather than its compilation) probably has an impact, but far more mitigated than the behaviour engine algorithm. Thus, it is recommended to spend development time on the improvement of this algorithm.

C.2 Tests with Crowd-MAGS' Intermediate Version

Crowd-MAGS' intermediate version is very similar to the initial one. The main difference is that the code that stores into memory what was perceived by an agent was rewritten in Java — it was originally in PLAMAGS. It was found that although this code was very short, it caused a huge performance degradation. Many of the tests presented in Section C.1 were re-run. When the code was migrated to Java, the performance as a whole increased by 100%. Thus, it is possible to simulate 200 agents where only 100 were possible before.

C.3 Tests with Crowd-MAGS' Final Version

This section presents tests that were run on calibration scenarios (discussed in Section 6.5). These tests were run to help analysts in preparing scenarios that will run in reasonable time. These tests were run on the final version of the Crowd-MAGS system. This version allowed the simulation to run in console mode. In this mode, the application is started from the command line and there is absolutely no user interface except for simulation output. This mode has been tested separately because it yields much better performances.

C.3.1 Tests Without Tear Gas

Tests were executed with a scenario similar to the calibration scenario 3b, except that the control forces were denied the use of tear gas. The number of agents is approximate because bystanders come and go and other agents may get hurt and disappear from the simulation. The scenarios were run for five simulated minutes. The results are shown in Tables C.6 and C.7.

C.3.2 Tests With Tear Gas

Tests were executed with a scenario similar to the calibration scenario 3b. The number of agents is approximate because bystanders come and go and other agents may get hurt and

Table C.6: Test Results With No Tear Gas in GUI mode

Number of agents	**Performance Ratio**	**VM memory usage (MB)**
100	0.9 X	850
200	0.5 X	1066
300	0.3 X	1119
400	0.3 X	1463
500	0.2 X	1620

Table C.7: Test Results With No Tear Gas in Console mode

Number of agents	**Performance Ratio**	**VM memory usage (MB)**
100	1.9 X	569
200	1.0 X	802
300	0.7 X	1022
400	0.4 X	1046
500	0.3 X	1444
750	$<$ 0.1 X	$>$ 1551

disappear from the simulation. The scenarios were run for five simulated minutes. The results are shown in Tables C.8 and C.9.

Table C.8: Test Results With Tear Gas in GUI mode

Number of agents	**Performance Ratio**	**VM memory usage (MB)**
100	0.6 X	1040
200	0.4 X	1125
300	0.3 X	1400
400	0.2 X	1626
500	0.2 X	1632

C.3.3 Conclusions

From these new tests, a few elements that have significant on the system performance have been identified. First, running the system in console mode is much faster and uses much less memory, especially in scenarios with few agents. In consequence, it is possible to run scenarios with 750 agents in console mode but not in GUI mode. Next, tear gas does have a significant impact on performance, as opposed to the conclusion drawn from early tests. Although the impact is larger when there are fewer agents, it is still present in larger scenarios.

Table C.9: Test Results With Tear Gas in Console mode

Number of agents	**Performance Ratio**	**VM memory usage (MB)**
100	1.3 X	652
200	0.7 X	850
300	0.5 X	1095
400	0.3 X	1376
500	0.2 X	1558
750	$<$ 0.1 X	$>$ 1558