Agent-Based Simulation of Crowds in Indoor Scenarios

Rafael Pax and Juan Pavón

Abstract Crowd simulation models usually focus on performance issues related with the management of very large numbers of agents. This work presents an agent-based architecture where both performance and flexibility in the behaviour of the entities are sought. Some algorithms are applied for the management of the crowd of agents in order to cope with the performance in the processing of their movements and their representation, but at the same time some alternative reasoning mechanisms are provided in order to allow rich behaviours. This facilitates the specification of different types of agents, which represent the people, sensors and actuators. This is illustrated with a case study of the evacuation of the building of the Faculty of Computer Science, where different types of human behaviours are modelled for these situations. The result is the simulation of more realistic scenarios.

1 Introduction

There are several models for crowd simulation but in general these focus on scalability issues derived from the management of a large number of agents in real time, specially when considering their visualization or the way the agents find their way while avoiding obstacles and other agents [1]. Different techniques have been proposed to cope with these issues, by relying on specific assumptions of the problem under study. This has, however, an effect on limiting of the flexibility of agents' behaviour, which is quite homogeneous in most of the cases.

In this work the scope of the problem is the simulation in indoor scenarios, where the number of agents may be large (thousands) but not very large as in a city (mil-

R. pax · J. Pavón (✉)
Universidad Computense Madrid (Spain), Madrid, Spain
e-mail: rpax@ucm.es
Url:http://grasia.fdi.ucm.es

J. Pavón
e-mail: jpavon@ucm.es

P. Novais et al. (eds.), *Intelligent Distributed Computing IX*,
Studies in Computational Intelligence 616,
DOI 10.1007/978-3-319-25017-5_12

lions). This gives more room on performance constraints, which opens the possibility to modelling of individual agents with heterogeneous behaviours.

Several tools for simulation and design of how people behave in indoor scenarios exist [8, 13–15, 17], but they are still considering the agents more like a crowd that can be characterized by simple behaviours with a fixed number of parameters, instead of considering them as individuals. Other works, like [7, 9–12] have addressed the specification of richer agent behaviours, but the methodological aspects for a design process when developing them are not sufficiently exposed.

Taking this into account, this work proposes an agent-based model for indoor scenarios where both performance and flexibility in the behaviour of the entities are sought. Agents are specified and managed individually, but the effects of the crowd are taken into account by several methods that take advantage of characteristics of the indoor domain in order to cope with the efficiency and scalability issues in the processing of their movements and their visualization. At the same time, some alternative reasoning mechanisms are provided for each agent in order to allow modelling of rich and heterogeneous behaviours.

Section 2 introduces the MASSIS (Multi-agent System Simulation of InDoor Scenarios) architecture and components. They allow for the specification of the elements for indoor scenarios simulation. Special attention is given to the definition of the behaviour of humans under different situations, which includes the process for decision making of the agents. Other relevant aspects to model are interactions among agents and with their environment, the events on the environment, and the precise representation of the building.

The strategies that facilitate an efficient management of crowd simulation aspects are presented in Sect. 3. This is illustrated with a case study on the evacuation of the building of the Faculty of Computer Science in Sect. 4. The model facilitates the specification of different types of agents, which representing the people, and the sensors and actuators of the environment. Different views can be generated and manipulated during the simulation. Finally, Sect. 5 presents the conclusions.

2 Overview of MASSIS Architecture

MASSIS has a component-based architecture, where some part of the infrastructure is well-proven open source software. An agent-based framework above of the core infrastructure allows the specification of flexible agent behaviour types, as well as interactions among agents and the elements of the environment.

The environment initially is defined using SweetHome3D, a well known package that is used to model all components involved in an indoor environment, such as walls, doors, stairs, people, etc. In order to facilitate more flexibility of the characterization of the physical elements, it is possible to define a set of plugins, to specify the characteristics of the elements of the building, which will be represented by agents. For instance, in the case of sensors and actuators, they will be reactive agents, with simple behaviour and attributes. In the case of people, some physical characteristics

can be defined such as weight, speed, but also some inherent attributes of the person (fear, courage, etc.) and a link to their behaviour. The types of behaviours are defined as components.

The simulation engine of MASSIS is MASON [5], a lightweight multi-purpose agent-based simulation library. Agents' behaviour is controlled with the Pogamut's POSH engine [4]. Some tests have been performed in order to check the performance of their integration in MASSIS. In the order of 10 thousand agents the experimentation has shown that execution times grow linearly with respect to the number of agents, so the results are satisfactory.

All the changes made in the environment are reflected in real time by 3D (Fig. 5) and 2D (Figs. 2, 3 and 8) displays, and can be logged in JSON format, as a single zipped file or in a SQLite database for further analysis. Although 3D display is more realistic and nicer for demonstration purposes, the 2D view is useful for analysis and debugging. Also, the 2D visualization API allows the creation of user-defined layers in order to filter the different elements involved in the simulation.

Once a simulation is performed, the exported data can be used to playback all events that have occurred during the execution of the simulation, i.e., the agents will behave in the same way they did during the simulation. This is interesting to allow the users to review the simulation when analysing what has happened.

Modelling human behaviour with agents have to consider two main aspects: those related with their perception and the interaction with the environment, and those dealing with the reasoning on the context and the decision making. They are implemented as low-level and high-level behaviour components, respectively. When the high level component decides *what to do next*, the action is executed by the low-level component, which performs all the necessary operations. The relationship between these components is illustrated in Fig. 1.

The low-level components deal with the perception of the environment and a set of basic behaviours for interacting with it. These behaviours are mostly a combination of *steering behaviours* [2], which are explained in Sect. 3.

The high-level behaviour components deal with decision making, learning and communications with other agents. Decisions are taken on the knowledge of the environment, which is provided by the low-level components.

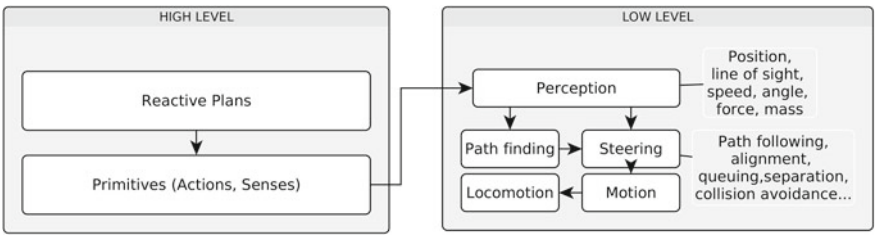

Fig. 1 MASSIS 's human behaviour agent model

Behaviour Oriented Design (BOD) [3] is applied to facilitate the design of agents. Developing a system of agents under BOD involves dividing their implementation into two different parts: A library of Behaviour modules and POSH Dynamic action selection scripts. Behaviour modules are a set of classes representing a set of modules for perception, action and learning (i.e. *primitives*). They can be called from the mechanism of action selection in order to determine *how* to do something. These senses and actions are created in the native language for the problem space (in the case of MASSIS, Java).

A POSH action selection script is a prioritized set of conditions and the related actions to be performed when certain conditions are satisfied. Figure 6 shows an example with the main elements: *drive collections*, *competences*, and *action patterns*. On the action selection step, the POSH engine executes the corresponding action pattern or competence from the drive collection with highest priority. Competences are similar to drive collections, but rules they do not interrupt each other. Finally, action patterns are simple, reusable sequences of actions.

The next sections present some of the components of the MASSIS framework that facilitate the efficient management of crowds of agents in a simulation.

3 Crowd Modelling and Simulation Issues

There are several aspects to take into account when modelling crowds of agents, with a trade-off between efficiency and flexibility of the specification of behaviours.

The building model, designed with SweetHome3D, is loaded and transformed into MASSIS internal representation to support the efficiency of algorithms implementation. This has an impact in the way agents interact with the environment, which is described below for different aspects: path finding, localization of elements and steering behaviours.

3.1 Path Finding

Pathfinding is one of the issues that has more impact when simulating crowd behaviours. Some models treat the crowd as a single entity or a group in order to simplify the number of calculations, such as in [6]. However, MASSIS, as it has been stated in the introduction, has as objective to support flexibility in agent behaviour, therefore the path finding model is implemented individually for each agent, but taking advantages of some assumptions from the problem domain in order to gain in efficiency.

When the action selection component (see Fig. 1) decides that the agent must go to a particular location, a path must be computed from the agent's location to the target. Its computation is done in MASSIS by an A* search over the polygons' visibility graph. The computational cost has been considerably improved by taking advantage of several characteristics of the environment:

1. The perimeter of rooms consists of walls or doors.
2. The path from a point A to a point B, being B in a different room than A, must pass necessarily through a door.
3. In an indoor scenario walls intersect each other *very often*.

With these assumptions, several aspects have been improved to gain in computational efficiency:

- **Faster visible edges computation.** In order to generate more realistic paths, the obstacle polygons are *inflated*, so the points of the path are slightly detached from the polygon edges. The advantage of this separation in the case of pathfinding, is that the edges of the obstacle polygons are now inside the rooms boundaries, and the number of possible reachable nodes from the agent's location decreases (only the nodes inside the room's boundaries are considered).
- **Obstacle polygons reduction.** The obstacles polygons that intersect can be merged, forming a bigger polygon. If most of the walls can be merged (as stated in assumption 3), the number of obstacles is drastically reduced (in the case of the Faculty of Computer Science, the number of wall obstacles are reduced from 2351 to 171, less than 8 % of the original) and consequently also the number of intersection tests.
- **The complete path is not needed at once.** The assumption 2 implies that it is not necessary to compute always the whole path between two points A and B. It is frequent, due to events in the environment, that the agent decides to change its current targets before it has reached the end of the path. To avoid unnecessary calculations, at the beginning of the simulation, a navigation graph based on door-room connections is created, and the doors are converted into waypoints. When the agent requests a path, the A* algorithm runs only in the current room.

3.2 Elements Localization

An intuitive way for storing the locations of the elements present during the simulation is using uniform grids. However, uniform grids are useful when the spatial data is distributed in an homogeneous way, which is rarely seen during simulation. The use of such grids can easily degenerate into situations where there are many redundant sparse cells. Furthermore, depending on the required accuracy, they can consume too many resources.

People, sensors and actuators need real time information about the elements that surround them. This implies that, during simulation, lots of query ranges must be performed. In order to minimize the impact of this calculation, MASSIS uses a QuadTree [18], a variable resolution data structure for retrieving agents' neighbours within a radius in an efficient manner (see Fig. 3). Although some CPU time is required in order to update the structure with the change of each agent, the gain obtained is worth it.

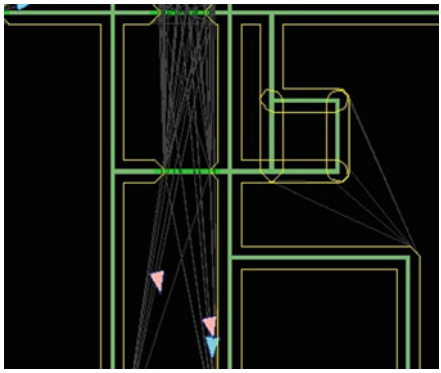

Fig. 2 Visibility graph and merged walls layer

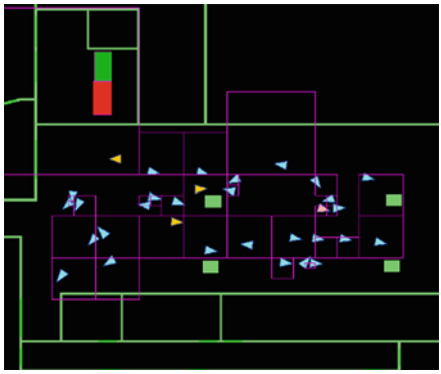

Fig. 3 Agents' Quad Tree layer, showing the space partitioning

3.3 Steering Behaviours

When the goals of the agent have been determined and the path to the target is known, the agent can start moving in the environment, avoiding collisions with obstacles (e.g., other agents, walls, etc.). These basic skills of the agent can be described with *steering behaviours* [2], which need as parameters the agent's mass m, the agent's location $\mathbf{L}$, a maximum force f_{max} and a maximum speed s_{max}.

Every step n, the computed steering forces are applied to the agent's location (limited by f_{max}), producing an acceleration whose magnitude is inversely proportional to the vehicle's mass.

$$\mathbf{A}_n = \left(\frac{trunc(\mathbf{F}_n, f_{max})}{m} \right) \tag{1}$$

The velocity of the agent in every step n is approximated by the Euler integration. Adding the velocity at the previous step ($\mathbf{V}_{n-1}$) to the current acceleration($\mathbf{A}_n$), produces a new velocity:

$$\mathbf{V}_n = trunc(\mathbf{V}_{n-1} + \mathbf{A}_n, s_{max}) \tag{2}$$

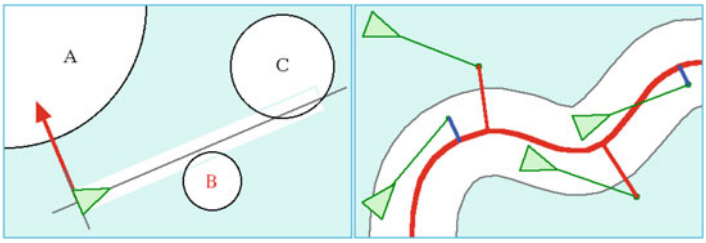

Fig. 4 Some of the steering behaviours implemented in MASSIS framework: seek and flee, obstacle avoidance and path following

Finally, the velocity is added to the agent's location.

$$\mathbf{L}_n = (\mathbf{L}_{n-1} + \mathbf{V}_n) \tag{3}$$

MASSIS provides a flexible implementation of several behaviours of this type (e.g., seek, arrival, separation, collision avoidance, wall containment, and path following, see Fig. 4), that can be grouped into more complex behaviours, like flocking or queuing.

4 Simulation of the Evacuation of a Building

It is common practice in public buildings to define some emergency protocols, which may involve, for instance, evacuation of the building. Planning and testing these protocols might be costly, but making simulations about these situations can help to this task (at least, as a first approach). This case study addresses this kind of situation for the building of the Faculty of Computer Science at UCM, which is represented in Fig. 5.

Three kinds of roles have been modelled based on the behaviours described by Proulx [16]:

- **Students** have some knowledge about the building. Their priority is the evacuation, but if they see someone needing help, they will try to assist. They also pay attention to the evacuation signals and indications displayed in the Faculty's CCTV. Their behaviour can be modelled as a POSH plan (see Fig. 6).
- **Well-trained staff members** are persons who work in the faculty (like a professor or administrative staff). They give instructions to the non-trained people, and try to assure that the evacuation is being done properly, following the established protocol.
- **Visitors** represents persons who have never been in the faculty, so the building is unknown to them. They interpret the fire alarm as something that is happening, so they will start searching for any person, expecting someone to tell them what

Fig. 5 MASSIS 3D representation versus a real photo

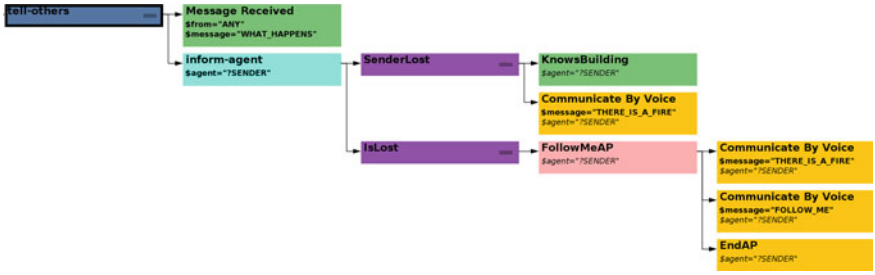

Fig. 6 Partial view of a student's POSH plan, having a drive element (*blue*), a competence(*cyan*), competence elements (*purple*), actions (*yellow*) and an action pattern (*pink*), which models the way in which the agent communicates with other people during the evacuation, reacting differently depending on the characteristics of the other agent. *Note* some elements were omitted for clarity

to do, if something serious is really happening. TV messages can help them to understand that they must evacuate the building, and also other people (an *Student* or a *Well-trained staff member*).

Elements of the environment (sensors and actuators) can be also modelled as agents, with their respective plans, which are usually simpler than those of agents representing people. For instance, Fig. 7 shows a fire detector's plan: the existence of a fire triggers its only action, which is the activation of the fire alarm.

These behaviour definitions suggest that the agents in this context must be capable of using their visual perception of the environment, what they hear and the ability of interacting with other agents, in order to accomplish higher-level goals. These abilities are modeled using low-level behaviours, managed by POSH primitives, which are the leaves of any reactive plan tree. Figure 6 illustrates parts of the reactive plans used in this case study for the people.

Fig. 7 Simpliest reactive plan: a fire detector

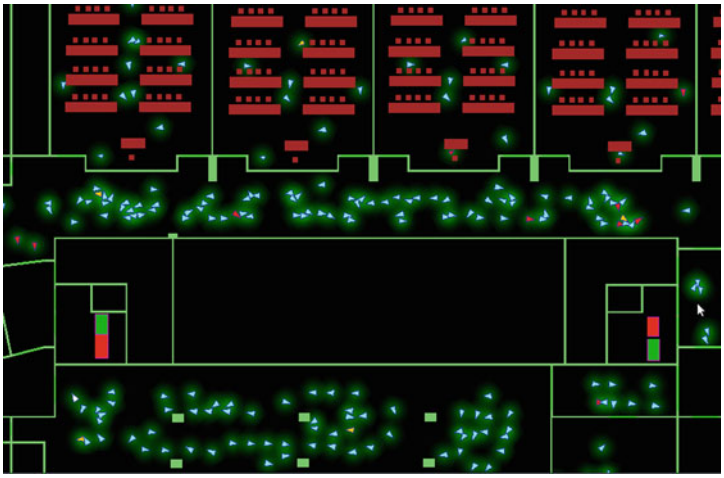

Fig. 8 Simulation 2D view, representing the different state of the agents by color

The simulation offers the option to work with 3D and 2D views. Usually 2D views are more practical for analysing what is happening in the simulation. For instance, Fig. 8 represents the state of the agents as colors. Other possibilities are 2D representations of crowd density, perception area, agent's IDs, paths, states, steering forces, etc. Other customized views can be easily integrated.

5 Conclusions

As it has been shown, MASSIS allows for the efficient simulation of indoor scenarios without losing the ability to specify rich and heterogeneous agent behaviours. This is achieved by simulating each agent individually, but with the support of several methods that take advantage of particularities of the indoor domain.

The extensibility of the MASSIS platform is well supported through its component-based architecture. For instance, different visualizations can be managed during the simulation, new algorithms and agent attributes can be supported and

monitored. Also, simulation is logged in order to be able to replicate it and facilitate further data processing to analyse models in more detail.

Acknowledgments This work has been been supported by the Government of the Region of Madrid through the research programme MOSI-AGIL-CM (grant P2013/ICE-3019, co-funded by EU Structural Funds FSE and FEDER), and by the Spanish Ministry for Economy and Competitiveness, with the project Social Ambient Assisting Living—Methods (SociAAL) (grant TIN2011-28335-C02-01).

References

1. Schuerman, M., et al.: Situation agents: agent-based externalized steering logic. J. Vis. Comput. Anim. **21**(3–4), 267–276 (2010)
2. Reynolds, C.W.: Steering behaviours for autonomous characters. In: Proceeding of Game Developers Conference 1999, San Jose, California, pp. 763–782 (1999)
3. Bryson, J.: Intelligence by design. Ph.D. thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology (2001)
4. Gemrot, J., et al.: Pogamut 3 can assist developers in building AI (Not Only) for their videogame agents. Agents for Games and Simulations. LNCS, pp. 1–15. Springer, Heidelberg (2009)
5. Luke, S., et al.: Mason: a multiagent simulation environment. Simulation **81**(7), 517–527 (2005)
6. Treuille, A.C., et al.: Continuum crowds. In: ACM Transactions on Graphics Proceedings of SIGGRAPH vol. 25(3), pp. 1160–1168 (2006)
7. Massive Software Simulating Life.: http://www.massivesoftware.com/ (2002). Accessed March 2015
8. Serrano, E., Botia, J.: Validating ambient intelligence based ubiquitous computing systems by means of artificial societies. Inf. Sci. **222**, 3–24 (2013)
9. Algeria l.saifi et al.: Approaches to modeling the emotional aspects of a crowd. In: EUROSIM'13: Proceedings of the 2013 8th EUROSIM Congress on Modelling and Simulation, pp. 151–143 (2013)
10. Wu, S., Sun, Q.: Computer simulation of leadership, consensus decision making and collective behaviour in humans. PLoS ONE **9**(1), e80680 (2014). doi:10.1371/journal.pone.0080680
11. Tibor Bosse et al.: Modelling Collective Decision Making in Groups and Crowds: Integrating Social Contagion and Interacting Emotions, Beliefs and Intentions, vol. 6443. Springer, Berlin (2010)
12. Bicharra, A.C., et al.: Multi-agent simulations for emergency situations in an airport scenario. Adv. Distrib. Comput. Artif. Intell. J. **1**(3), 69–73 (2013)
13. Legion | Science in Motion.: http://www.legion.com (2015). Accessed March 2015
14. Galea, E., et al.: The EXODUS evacuation model applied to building evacuation scenarios. J. Fire Prot. Eng. **8**(2), 65–84 (1996)
15. PedGo—TraffGo HT.: http://www.traffgo-ht.com/ (2006). Accessed March 2015
16. Proulx, G.: Occupant behaviour and evacuation. In: Proceeding of 9th International Fire Protection Symposium. pp. 219–232 (2001)
17. Pathfinder—Thunderhead Engineering. http://www.thunderheadeng.com/pathfinder/ (2006). Accessed March 2015
18. Finkel, R.A., Bentley, J.L.: Quad trees : a data structure for retrieval on composite keys. Acta Informatica **4**, 1–9 (1974)