System Issues in Multi-agent Simulation of Large Crowds

Vidit Bansal, Ramachandra Kota, and Kamalakar Karlapalem

Center for Data Engineering
International Institute of Information Technology
Hyderabad, India
vidit@research.iiit.ac.in, ramachandra@students.iiit.ac.in,
kamal@iiit.ac.in

Abstract. Crowd simulation is a complex and challenging domain. Crowds demonstrate many complex behaviours and are consequently difficult to model for realistic simulation systems. Analyzing crowd dynamics has been an active area of research and efforts have been made to develop models to explain crowd behaviour. In this paper we describe an agent based simulation of crowds, based on a continuous field force model. Our simulation can handle movement of crowds over complex terrains and we have been able to simulate scenarios like clogging of exits during emergency evacuation situations. The focus of this paper, however, is on the scalability issues for such a multi-agent based crowd simulation system. We believe that scalability is an important criterion for rescue simulation systems. To realistically model a disaster scenario for a large city, the system should ideally scale up to accommodate hundreds of thousands of agents. We discuss the attempts made so far to meet this challenge, and try to identify the architectural and system constraints that limit scalability. Thereafter we propose a novel technique which could be used to richly simulate huge crowds.

1 Introduction

Simulating large crowds in rescue simulation systems throws up many challenges. For one, crowd events and their associated phenomenon are difficult to model. Crowds demonstrate a variety of emergent behaviours based on the behaviour of individuals in the crowd. Crowd dynamics have been extensively studied in the past and various socio-psychological and physiological theories have been put forth to explain crowd behaviour. The complexity also stems from the fact that compared to a simulation with limited parameters, the level of detail that could be incorporated in the model for a realistic social simulation can be quite high. Analyzing crowd dynamics has been an active area of research. Different types of crowd simulation systems have been developed, ranging from those based on force-modelling approaches [1, 2] to cellular automata based simulations [7, 8, 9] and rule-based architectures [5, 6]. Recently, many agent-based architectures have been proposed [13, 18, 19]. The multi-agent paradigm is adequately suited

L. Antunes, M. Paolucci, and E. Norling (Eds.): MABS 2007, LNAI 5003, pp. 8–19, 2008.

to a crowd simulation application. Social factors can be better modelled as human characteristics can be objectively mapped to agent behaviour.

In this paper we present a multi-agent based crowd simulation system developed on a continuous field force model. Our model supports - heterogeneity in agents to model the demographics of a population, a complex navigational behaviour including obstacle avoidance and navigation in a terrain with partial information and flocking behaviour. As scalability is a vital criterion for the system to realistically model a disaster scenario for a large city with hundreds of thousands of people, we evaluate the issue of scalability for multi-agent based crowd simulation systems. The organization of the paper is as follows. We discuss the environment model that we use for our simulations in Section 2. We then move on to propose a novel technique for simulation of large crowds in Section 3. We present our results in Section 4.

2 Background and Related Work

Amongst the different approaches to model crowd behaviour, one which has been widely put to use is the force based model developed by Dirk Helbing et al.[1, 2]. This model tries to simulate the motion of each individual in a crowd (henceforth referred to as a civilian) under forces that are exerted by other civilians and inanimate objects. Each civilian feels, and exerts on others, two kinds of forces, "social" and physical. The social forces are not exactly physical forces such as a *push* or a *pull*; they reflect the intentions of a civilian to avoid collisions and to move in a chosen direction. The movement of the civilian can be tracked by equations defining his motion under the sum of all forces. Further refinements were suggested in [3]. One problem with Helbing's model is with its computational complexity. Every civilian must be tracked with respect to every other civilian to calculate the net force acting on it. Alternative approaches proposed[3] avoid computing every agent's effect on all the others. While the model has been successful in simulating a number of crowd behaviours demonstrated by real-life crowds, particularly arch formation at exits in egress situations, scalability issues were not tackled. The model by itself does not incorporate navigation models, or any form of social interaction.

Cellular Automata based models have also been proposed most notably in [7, 8, 9]. Cellular automata based simulations also model forces on a civilian, but are discrete in space and time. Efforts have been made to build multi-agent based systems for crowd simulation. Prominent among these are [11, 12, 13, 18, 19]. However, scalability issues in multi-agent based crowd simulation systems have not been rigorously analyzed.

2.1 Background

We have attempted to develop a crowd simulation model which preserves the granularity of simulation at the individual level and at the same time is scalable and can richly simulate behaviour of huge crowds. The problem that we are trying to address is scaling up in terms of the number of civilians that can be

simulated on a single computer given a set of system constraints. Typically agents are implemented as threads. For any multi-agent based architecture, there are two key system constraints – the size of the RAM, which determines the number of agents that can be kept in the main memory and the number of threads that can be handled by the processor. Threads by themselves require both the main memory and the CPU usage, thereby imposing a dual constraint. We show the results of our experiments on our simulation model described below.

2.2 Description of the Experimental Model

Our simulation model is based on the continuous force model by Helbing [2]. The map of the environment is a two dimensional continuous grid. A complex terrain can be specified with obstacles and walls. There may be multiple safety exits towards the periphery of the map. Civilians are placed at different locations on the map at the beginning of the simulation. Each civilian c_i has body mass m_i and a maximum velocity v_i^{max}. The civilian occupies a circular area of size r_i which is proportional to the the square root of its body mass m_i . Each civilian also has a defined visibility range e which determines a sphere of influence. A civilian is affected by all objects i.e both by other civilians and walls within their visibility range. A civilian experiences a repulsive force from other civilians in its visibility. This force decreases as the distance increases and is null beyond the visibility. The civilian experiences a similar repulsive force from a wall or a blockade. The repulsive force along the x-direction on civilian i due to civilian j has the form:

$$f_{ij}^x = \begin{cases} C_x/(x_j - x_i) & \text{if } |x_j - x_i| \leq e \\ 0 & \text{otherwise} \end{cases}$$

Walls exert a force f_{iw}^x on civilian c_i :

$$f_{iw}^x = W_x/d$$

where d is the distance of the closest point of the wall from the civilian. Obstacles avoidance is achieved through the application of a repulsive force f_{io}^x on civilian c_i :

$$f_{io}^x = O_x/d_o$$

where d_o is the distance of the obstacle from the civilian. C_x , W_x and O_x are constants and are set to ensure appropriate ranges for the forces.

The civilians may have partial or complete knowledge of the terrain. Their knowledge is in the form of a sequence of landmarks or points they might remember to reach the safety exits. This sequence can also be updated by external factors such as a communication from another agents. The civilian agent can also follow another agent. In an emergency evacuation scenario there might rescue agents to coordinate rescue activity. Let F_{dir}^x be the force which determines the direction the agent wants to move in.

Therefore the total force acting on the civilian c_i along the x-direction is

$$F_i^x = \sum_j f_{ij}^x + \sum_w f_{iw}^x + \sum_o f_{io}^x + F_{dir}^x$$

Similarly

$$F_i^y = \sum_j f_{ij}^y + \sum_w f_{iw}^y + \sum_o f_{io}^y + F_{dir}^y$$

The model described above is the basic minimum model, and it can be enriched by specification of other parameters. For instance, we have also modelled injuries, simple communication models and specialized rescue agents. In this paper however we'd like to focus on the scalability aspects of our system.

For our experiments, we used MASON multi-agent simulation library [22]. MASON is coded in Java and provides a comprehensive functionality set for simulations. We conducted the experiments on a Linux server with the following configuration: 2100 MHz CPU, with a 512 MB RAM.

2.3 A Micro-agent Architecture

We start with a simple architecture. In this architecture, each civilian is modelled as an agent. Hence, each civilian has a thread dedicated to it. The CPU would run as many threads as there are civilians. In every simulation step, we compute the movement of each civilian agent under the influence of the forces on it. Also, each agent stores its state information and hence needs memory. If the memory required for one civilian is M_x, then the total memory required would be nM_x where n is the number of civilians. Henceforth we refer to this architecture as the Micro-Agent architecture in this paper. With the model described in the previous subsection, we are able to show common egress behaviour of crowds from an enclosed room. Arch formation at the exit is shown in figure 1(b).

We conducted further experiments trying to scale up by increasing the number of civilians in the map. For this set of experiments our simulation environment map consisted of a huge hall with one exit and a few randomly placed obstacles. We observed that an increase in the number of agents (which is equivalent to the number of threads) corresponded to an almost linear increase in the time to the run the simulation for a fixed number of cycles. This was expected as a civilian agent has to consider only the agents within its sphere of influence as against every other civilian in Helbing's model. But the simulation hits a major block

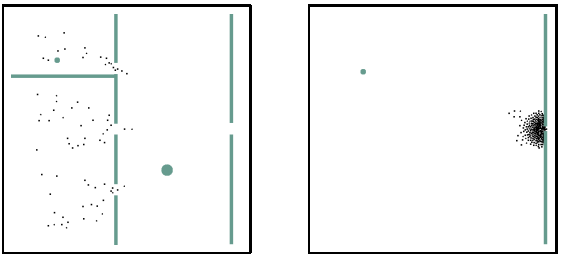

Fig. 1. Evacuation from a hall:: Fig (a): Civilians(black dots) rushing towards the exits in an evacuation scenario Fig (b): Arch formation at exit : A common egress behaviour exhibited by crowds

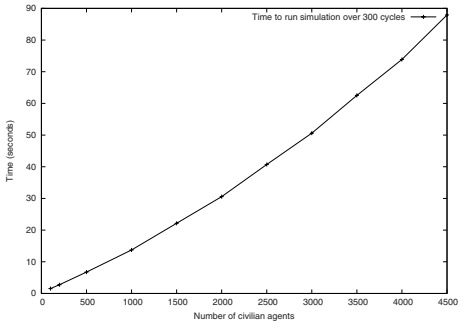

Fig. 2. Time taken to run simulation for 300 cycles

when it comes to the number of threads that can be run on the system. In our experiments we hit a top of approximately 5000 threads initially. The maximum number of threads that can be can be simulated depend on the operating system and the Java Virtual Machine(JVM). The available stack size on the system is one the factors that determine the maximum number of threads that can be run. The default stack size for a thread is 512Kb on Linux (kernel versions 2.6) but it can be set to a lower value. In our experiment we were able to set the minimum value of 105Kb. With this setting we were able touch close to 9600 threads. Note that these values were specific to our experiment's environment and these could vary under different settings. There are however two important aspects we would like to convey – *i*) that there is an upper limit on the number of threads that can be run on a system with given specifications, and *ii*) the more complex the application is, the fewer number of threads it would be able to support. This is because given a fixed stack size on a system, if the stack space occupied by a thread increases, fewer threads would be accommodated in the stack.

3 A Macro-micro Architecture

3.1 Using a Database

We see that in the Micro-Agent architecture, the limited resources put a constraint on the number of civilians that can be created and sustained in the environment. The limiting factor is the size of the main memory. The standard approach to counter a memory limitation would involve getting an external memory with a paging or swapping scheme. This approach was successfully demonstrated by Yamamoto and Nakamura [20] in an distributed electronic commerce scenario.

We could conceptualize a solution to this problem, overcoming the main memory limitations by using a database, with each civilian existing as a tuple in a *civilian* table in a database. In each time cycle, the simulation program would retrieve each civilian's details from the database, get its location, identify its neighbours, compute the net force on the civilian and calculate its next step and

update the corresponding tuple of the civilian table. It would then move onto the next civilian. After one complete scan it could update positions of all civilians. Since, we are using secondary memory storage in this method, the memory constraints would not limit the number of civilians that can be simulated. Also, since only one process is being executed, there couldn't be any constraints on the number of civilians due to the limited processing capability. However, as the secondary memory would have to be accessed for each agent for each time cycle and since there is no parallelism, each step in the simulation would take a large amount of time. Thus it would be too slow to be of any significance.

3.2 A Macro-micro Architecture

We propose an architecture, which we call the Macro-Micro architecture which uses the database to bypass the limitations of the memory, and at the same time uses the abstraction of a crowd to reduce the complexity. The multi-agent system architecture we propose in the following discussion essentially is *augmented* by a database to help it scale with respect to the number of agents that can be created and sustained in a given environment.

A crowd can be said to be a transient group of individuals, sharing some common space and environment and moving together, with all individuals having nearly the same velocity. Individuals who are deep within a crowd have a restricted freedom of movement. Their movement is decided by the movement of their neighbours. However the individuals who are at the periphery of the crowd are considerably more free to move. Thus the individuals on the periphery of the crowd shape its boundary and dictate its movement. The individuals within a crowd have an alignment towards the average direction of motion of outer individuals [21]. Further the crowd can be considered as a single entity moving in a certain direction with a certain velocity, with individuals inside the crowd sharing the same motion characteristics.

In our simulation, we maintain a database table of all civilians. Consider a moving crowd, physically separated from other crowds where the distance between two individuals in the same crowd is less than between individuals of different crowds, i.e. typically a cluster of civilians. This crowd can be considered to be a single entity and we represent it by a *Crowd* agent. Since the individuals at the boundary play an important role in determining the shape and movement of the crowd, we consider it necessary to accord a special status to them - distinct from their respective Crowd agent. We represent them as *Boundary* agents *belonging* to the that particular Crowd agent. We designate the set of Boundary agents by the cover E.

In our approach, we essentially differentiate between those civilians who are within a crowd and share the motion of the crowd, and those which are at the periphery and may have significant motion characteristics of their own. The definition of a crowd in our model is based on physical proximity of individuals in a limited space. Models explaining crowd formation or how individuals organize themselves into groups are not yet available but any such model would be consistent with our definitions. The architecture proposed has two basic components,

the database and a set of agents running on the system. The database stores the state of all the civilians, who at any point of time might be activated as a boundary agent if they come to lie at the periphery of a crowd. Apart from the database we have two distinct sets of agents - Crowd agents which model the behaviour of the crowds and a set of Boundary agents who are essentially the activated civilians from the database. If at any point of time, a boundary agent is displaced inwards such that it comes to lie deep within the crowd, we deactivate the agent and it continues its existence as just a tuple in the database table. In effect, our approach is centered at activating and deactivating agents as the simulation proceeds. As the simulation proceeds, any of the following may happen:

- The crowd may disintegrate.
- A crowd might merge with another crowd.
- Individuals might leave or join a crowd.

In all these scenarios, the action takes place at the boundary as an agent can join or leave the crowd only at the periphery. In section 3.4 we present a set of incremental algorithms which trace the changes to the outer cover E for each crowd.

We are able to achieve scalability as the agents running on the system are either the civilians at the periphery or the crowd agents. Typically we were able to create and sustain up to twenty times more agents than we could by using the Micro-Agent architecture. The more important thing, however, is that we are still able to retain the granularity at the individual level since each individual's characteristics are stored in the database. We only update a small fraction of the tuples, that too only when a civilian leaves a crowd or joins a new one.

3.3 Components of the Architecture

Let us now examine each of the components of the architecture in detail.

1. **Database Table:** The table stores the details of all the civilians in the simulation. The schema of the table is –
   $\{CIV_{id}, X_{init}, Y_{init}, X_{rel}, Y_{rel}, CROWD_{id}, FLAG_b\}$.
   Each civilian is assigned an ID (CIV_{id}). It's initial position is stored as $\{X_{init}, Y_{init}\}$. $\{X_{rel}, Y_{rel}\}$ store its relative position with respect to the crowd

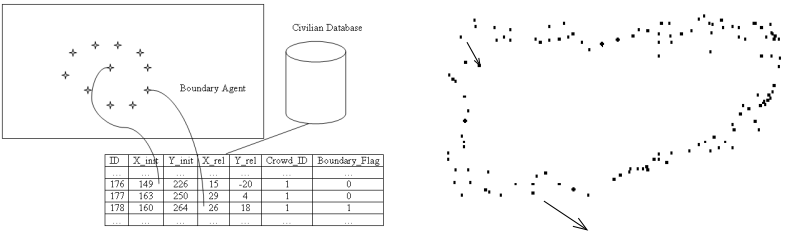

Fig. 3. (a) The system architecture. (b) An illustrative diagram showing a moving crowd. The civilian at the bottom moves away from the crowd. Another one at the top is moving in to join the crowd.

center, $CROWD_{id}$ marks the ID of the crowd to which the civilian currently belongs. The $FLAG_b$ (boundary flag) is set whenever the civilian is activated as a boundary agent and is unset when it loses that status.

2. **Boundary Agent:** Each boundary agent behaves as a single free civilian. Each boundary agent is a civilian and hence has a tuple in the database to represent it. This tuple is updated whenever the agent changes the crowd that it belongs to or loses its boundary agent status. It interacts with other agents in its visibility and moves as a result of the forces acting on it.
3. **Crowd Agent:** The crowd agent is an agent which represents all the civilians that are part of it. The movement of the crowd agent is the vector mean of the movements of the boundary agents. Each crowd agent can know all the civilians belonging to it by querying the database. It also maintains a list of all its boundary agents E_c.

3.4 Simulating the Behaviour of Boundary and Crowd Agents

1. Boundary Agent Parting away:
   In every iteration, the crowd agent checks for each boundary agent, whether it still belongs to the crowd. Those boundary agents whose distance from all the neighboring boundary agents is more than twice the visibility are considered no longer belonging to the crowd. They are considered as a new crowd with just a single agent. Such agents are removed from the boundary agent list of the crowd agent and their corresponding tuple in the database is updated. It also creates a new crowd agent which contains only one civilian corresponding to the run-away boundary agent. Thus the new crowd agent only has one boundary agent. The crowd agent also checks whether any boundary agent has been taken away by another crowd (it may happen during the merger of two crowds). It would come to know this through the $crowdID$ stored in each boundary agent. The crowd agent also removes such snatched-away agents from its boundary agent list.
2. Boundary Agent joining in:
   At every time step, the crowd agent checks in the vicinity (visibility) of its boundary. If it finds a boundary agent not belonging to it and if such an agent belongs to a single boundary agent crowd or a crowd agent of smaller size (size = total civilians i.e total tuples of the crowd), then the crowd agent grabs the boundary agent as one of its own. It updates the $crowdID$ of the boundary agent and also the corresponding tuple in the database. It also inserts the new boundary agent in its boundary agent list. Thus, the crowd agent has effectively snatched away a boundary agent from another relatively weaker crowd agent.
3. Change of Shape of Crowd:
   There are two ways in which the shape of the crowd agent can change. Either two boundary agents move apart and there is need for a new boundary agent between them or a boundary agent goes in and is no longer on the boundary and its boundary status has to be unset.

   At each step, the crowd agent traverses in order through all its boundary agents to check whether the distance between any two consecutive agents is

more than the visibility. If so, then the crowd queries the database for the civilians whose relative positions in the crowd lie between these two boundary agents, visible to both. It picks one amongst them, sets its boundary flag in the database and creates a new boundary agent to represent this civilian. This boundary agent belongs to the crowd and is inserted into the boundary agent list accordingly. At each step, the crowd agent also checks whether any boundary agent has moved inwards into the crowd. A boundary agent is considered to have moved into the crowd, if another boundary agent lies on the line which extends radially outwards from the center towards the boundary agent. In such a case, the boundary agent that lies closer to the center is no longer on the boundary and must be dealt with accordingly. For the civilian represented by this boundary agent, the boundary flag is unset in the database and its relative position with respect to the crowd center is also updated in the database. Finally, the boundary agent is deactivated as it serves no purpose since it is no longer on the boundary and has to move as the crowd does.

4. Splitting and Merging of Crowds:
   Splitting of a Crowd can be considered as a continuous process of boundary agents moving away from the crowd agent as in (1) above. Merging of two crowds can be considered as a continuous process of boundary agents joining one crowd from another. The stronger crowd agent would snatch away the boundary agents of the weaker crowd. At the end of the merger, the weaker agent would be left without any boundary agent or any civilian belonging to it and hence would terminate itself.

4 Results and Analysis

We ran multiple simulations in scenarios involving evacuation from a hall with varying number of civilians. The results are tabulated in Table 1.

In our experiments we were able to achieve a scale up in the number of civilians simulated by a factor of five to twenty. The results tabulated are different random runs. It is difficult to provide a quantitative analysis of the results based on a standard metric as it is not possible to recreate an experiment run exactly. Depending on the structure of the crowd clusters which are formed and how individuals decide to move in each step, the number of instructions executed to maintain the boundary could vary widely over experiments.

An important aspect is that the complexity of the system depends upon the number of boundary agents rather than the number of civilians. The number of crowd agents is typically quite less compared to the number of boundary agents. It is difficult to keep a tab on the number of boundary agents. Their number could vary with time in a particular scenario, and more generally could vary across different scenarios. In our experiments we create some well defined crowd clusters at the beginning of the simulation and mark the civilians at the periphery as the boundary agents. Thereafter the boundary is updated as per the algorithms specified above. To illustrate the effect of the number of boundary

Table 1. 1a: Time taken to run simulation for 300 cycles. 1b: Time taken to run simulation for 300 cycles for 30000 civilians.

Number of civilians	Time taken (for 300 cycles)	Initial number of boundary agents	Final number of boundary agents
1000	0m03.560s	90	85
3000	0m10.970s	300	267
9000	0m34.483s	900	880
15000	0m45.283s	1500	1372
18000	1m17.640s	1800	1631
30000	2m8.647s	3000	2754
50000	12m28.001s	4500	4282
75000	18m17.522s	5400	5177
100000	23m51.707s	6000	5652

Number of civilians	Time taken (for 300 cycles)	Initial number of boundary agents	Final number of boundary agents
30000	0m18.136s	12	128
30000	0m16.688s	30	143
30000	0m19.179s	90	199
30000	0m20.393s	300	24
30000	0m29.537s	600	530
30000	1m59.602s	900	880
30000	2m8.647s	3000	2769
30000	26m40.784s	6000	5904
30000	29m57.944s	7500	7235
30000	49m27.345s	9000	8877

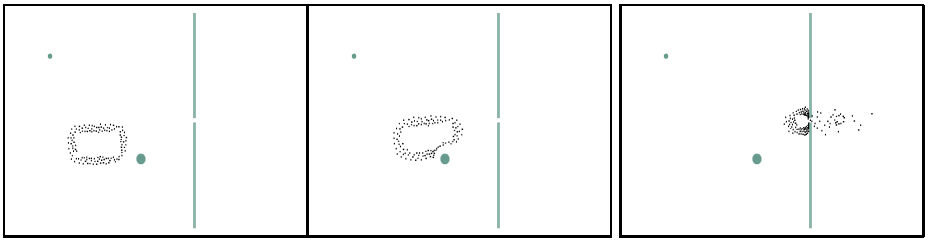

Fig. 4. Evacuation from a hall:: Fig (a): A rectangular crowd → Fig (b): Crowd expands, avoids obstacle → Fig (c): At the exit: Crowd broken into a number of small crowds(including unit crowds)

agents on the simulation time we tried varying the number of initial boundary agents while keeping the number of civilians constant. The results are shown in Table 1(b). The worst case scenario for the macro-micro architecture would be when the civilians are distributed and far apart to form any large clusters. In that event, the number of boundary agents would roughly equal the number of civilians. On the other hand, the best case scenarios would be when the civilians are distributed into thick and distinguishable clusters.

4.1 Keeping Track of Individual Civilians

Each civilian exists as a tuple in the database. Thus, its identity is preserved. Its relative position with respect to the crowd is stored and so is its crowd ID stored in the database. Thus at any point of time, its actual position on the map can be approximately known since the coordinates of the crowd center are available and the relative position with respect to the crowd center is also available through the database. The basic assumption here is that the relative position of

a civilian with respect to the center of the crowd will remain unchanged. This assumption is justifiable because a civilian inside a crowd is constrained by its neighbours. When a civilian becomes a boundary agent, then too its movement can be tracked since it exists as an agent on the map. When it joins another crowd, then the crowd ID, relative positions etc are updated accordingly and therefore its position is still known. Thus at any point of time, the location of any civilian on the map can be obtained. Thus all civilians can be tracked.

5 Conclusion and Future Work

System issues limit the scope of applying multi-agent systems for massive crowd simulation. In this paper we proposed a macro-micro simulation system based on a database to richly simulate massive crowds. We have presented our results demonstrating how our technique can be used to overcome the system constraints. More significantly we believe that this idea can be extended to a generic architecture where it is possible to richly simulate the behaviour of a large body of agents by focusing on a relatively smaller number of significantly active agents. Further research can focus on how database systems can be effectively used for augmenting multi-agent system architectures to attain scalability. It is possible to consider boundary agents as the leaders in the crowd. We plan to simulate emergency evacuation scenarios analyzing the role of these leaders in such cases.

References

1. Helbing, D., Farkas, I.J., Molnr, P., Vicsek, T.: Simulation of pedestrian crowds in normal and evacuation situations. In: Schreckenberg, M., Sharma, S.D. (eds.) Pedestrian and evacuation dynamics, pp. 21–58. Springer, New York (2002)
2. Helbing, D., Farkas, I., Vicsek, T.: Simulating dynamical features of escape panic. Nature v. 407, 487–490 (2000)
3. Lakoba, T.I., Kaup, D.J., Finkelstein, N.M.: Modifications of the Helbing-Molnar-Farkas-Vicsek Social Force Model for Pedestrian Evolution In Simulation, vol. 81, pp. 339–352 (2005)
4. Henein, C.M., White, T.: Agent-Based Modelling of Forces in Crowds. In: Proceedings of the Workshop on Multiagent and Multiagent-based Simulation (MAMABS) at AAMAS 2004, July 19-23 (2004)
5. Schweiss, E., Musse, S.R., Garat, F., Thalmann, D.: An Architecture to Guide Crowds Using a Rule-Based Behavior System. In: Proceedings of Autonomous Agents (1999)
6. Farenc, N., Musse, S.R., Schweiss, E., Kallmann, M., Aune, O., Boulic, R., Thalmann, D.: A Paradigm for Controlling Virtual Humans in Urban Environment Simulations. Applied Artificial Intelligence 14(1) January 1 (2000)
7. Kirchner, A., Schadschneider, A.: Simulation of evacuation processes using a bionics-inspired cellular automaton model for pedestrian dynamics. Physica A 312, 260–276 (2002)

8. Dijkstra, J., Jessurun, A.J., Timmermans, H.J.P.: A Multi-Agent Cellular Automata System for Visualising Simulated Pedestrian Activity. In: Proceedings of the Fourth International Conference on Cellular Automata for Research and Industry, pp. 29–36. Springer, Berlin (2000)
9. Hamagami, T., Hirata, H.: Method of crowd simulation by using multiagent on cellular automata. In: Proceedings of the IEEE/WIC International Conference on Intelligent Agent Technology (IAT 2003) (2003)
10. Pan, X., Han, C.S., Law, K.: A multi-agent based simulation framework for the study of human and social behavior in egress analysis. In: The Proceedings of The International Conference on Computing in Civil Engineering, Cancun, Mexico, July 12-15 (2005)
11. Murakami, Y., Minami, K., Kawasoe, T., Ishida, T.: Multi-Agent Simulation for Crisis Management. In: Proceedings of the IEEE Workshop on Knowledge Media Networking (KMN 2002) (2002)
12. Ishida, T.: Social Agents in Digital Cities
13. Ohta, M., Koto, T., Takeuchi, I., Takahashi, T., Kitano, H.: Design and implementation of the kernel and agents for the RoboCup-Rescue. In: Proceedings of IEEE, Fourth International Conference on MultiAgent Systems (2000)
14. Koto, T., Takeuchi, I.: The Distributed Simulation of RoboCupRescue Version 1. In: Proceedings of SICE 2002 (2002)
15. Koto, T., Takeuchi, I.: A Distributed Disaster Simulation System That Integrates Sub-simulators
16. Tadokoto, S., et al.: The RoboCup-Rescue Project: A Robotic Approach to the Disaster Mitigation Problem. In: Proc. IEEE International Conference on Robotics and Automation, pp. 87–96 (2000)
17. Brenner, M., Wijermans, N., Nussle, T., de Boer, B.: Simulating and Controlling Civilian Crowds in Robocup Rescue. In: RoboCupRescue Simulation: Infrastructure competition (2005)
18. Toyama, M., Bazzan, A., da Silva, R.: An Agent-Based Simulation of Pedestrian Dynamics: from Lane Formation to Auditorium Evacuation. In: Proceedings of International Joint Conference on Autonomous Agents and Multi-Agent Systems (2006)
19. Osaragi, T.: Modelling of pedestrian behaviour and its applications to spatial evaluations. In: Jennings, N., Sierra, C., Sonenberg, L., Tambe, M. (eds.) International Joint Conference on Autonomous Agents and Multi-Agent Systems, pp. 836–843 (2004)
20. Yamamoto, G., Nakamura, Y.: Architecture and Performance Evaluation of a Massive Multi-Agent System. In: Proceedings Autonomous Agents 1999 (1999)
21. Reynolds, C.W.: Flocks, Herds, and Schools: A Distributed Behavioral Model. In: Computer Graphics (SIGGRAPH 1987 Conference Proceedings), vol. 21(4), pp. 25–34 (1987)
22. MASON: Java based multiagent simulation library. MASON has been developed by George Mason University's ECLab Evolutionary Computation Laboratory and the GMU Center for Social Complexity,
`http://cs.gmu.edu/~eclab/projects/mason/`