A Distributed Framework for Scalable Large-Scale Crowd Simulation

Miguel Lozano[1], Pedro Morillo[1], Daniel Lewis[2], Dirk Reiners[2], and Carolina Cruz-Neira[2]

[1]Departamento de Informática
Universidad de Valencia
Avenida Vicente Andrés Estelles, s/n
46100 Burjassot, (Valencia), Spain
miguel.lozano@uv.es

[2]Louisiana Immersive Techonologies Entreprise (LITE)
University of Louisiana at Lafayette
537 Cajundome Boulevard
70506 Lafayette, (LA), USA
carolina@lite3d.com

Abstract. Emerging applications in the area of Emergency Response and Disaster Management are increasingly demanding interactive capabilities to allow for the quick understanding of a critical situation, in particular in urban environments. A key component of these interactive simulations is how to recreate the behavior of a crowd in real- time while supporting individual behaviors. Crowds can often be unpredictable and present mixed behaviors such as panic or aggression, that can very rapidly change based on unexpected new elements introduced into the environment. We present preliminary research specifically oriented towards the simulation of large crowds for emergency response and rescue planning situations. Our approach uses a highly scalable architecture integrated with an efficient rendering architecture and an immersive visualization environment for interaction. In this environment, users can specify complex scenarios, "plug-in" crowd behavior algorithms, and interactively steer the simulation to analyze and evaluate multiple "what if" situations.

1 Introduction

In less than a decade, the world has experienced a significant series of both man-made and natural disasters of unprecedented proportions, causing tremendous losses in terms of humans lives, as well as causing tremendous financial losses. The processes of responding, maintaining, and recovering from these disasters have made evident the strong needs to have better ways to train emergency responders, as well as to develop, analyze, and evaluate new effective approaches to incident management. Most of these incidents require well-coordinated and well-planned actions among all the different forces of emergency response within severe time constraints.

Furthermore, most of these incidents involve the population of the area in distress, requiring that the responders and government officials understand the impact of their actions on that population. Therefore, the ability of simulate the behavior of a crowd under different situations: stress, panic, danger, evacuation, as well as the ability to visualize it in the context it is being evaluated in is critical to develop a strong set of tools for emergency response. We propose a highly scalable distributed architecture integrated with an efficient rendering and immersive interactive space that can support large crowds with behaviors ranging from a single "mob mentality" to individualized behaviors for each member of the crowd. The key element of our architecture is the ability of handling the crowd control model and the realistic simulation all integrated in a real-time environment.

2 Background

Training emergency responders to effectively manage the kinds of large-scale disasters we face today needs to be approached through advanced computer simulations, visualizations and interactive environments. Traditional real-life training through mock-ups and actors [20] cannot provide a close reproduction of the complex interrelations among response forces, local, state, and federal officials, volunteers, and the affected population. The scale and extent of the situation lends itself to the application of virtual environments and simulation. A critical component of these environments is the ability of simulate large numbers of people within urban and transportation systems, with both crowd behavior as well as individual behaviors. However, there are several conflicting objectives involved in the real-time simulations of Emergency Reponses, especially when they are designed to study the behavior of citizens evacuating cities. On the one hand, this type of simulations must focus on rendering not only animated characters of humanoid appearance, but also the entire urban scenarios where they are located. On the other hand, these simulation systems should offer a wide rich variety of group and autonomous behaviors and actions associated with pedestrians in urban environments such as wandering, fleeing, panic, etc.

Crowd simulations have been used extensively in the entertainment industry to create realistic scenes containing large numbers of individuals. These techniques have been used to create crowd scenes in different movies, although the crowd models used in the entertainment industry are primarily concerned with creating visual realism, without regard to enabling robust behavior at the level of individuals. While these systems can create strikingly beautiful images, the level of behavioral realism is too low to be used for our purposes, and the simulations are far from interactive.

One traditional method of crowd simulation involves the use of a modified particle system, wherein each agent is represented by a single particle whose action is determined by a system of interaction rules [26]. This system is well suited to animating large crowds; however, the interaction rules are generally quite simple, limiting the complexity of behavior at the level of individuals.

There is typically a trade-off between maximum crowd size and behavioral richness, as increasing either rapidly increases the computational complexity of the simulation. Many crowd models, which are designed to accommodate large crowds, do away with individual behavior and instead focus on collective behavior of a crowd.

Crowds have been divided at different levels in order to attempt to decrease the computational complexity of simulations. For example, the ViCrowd system [19] divides the simulation at the level of crowds, groups, and individuals. Modern variants of these crowd-oriented simulations use continuum dynamics to reach interactive simulation speeds for thousands of characters [32]. Although these approaches can display very populated and interactive scenes, their usability for emergency response plans is questionable as the higher-level behaviors are not based on individual behavior.

There have been efforts on developing smaller- scale systems that provide support for sufficiently advanced behavior. Consideration of crowd behavior can be incorporated into the design of buildings and public places [24], as well as be used to train emergency personnel in a variety of realistic environments, even the specific environment they will be working in. However, these models, although they provide a good crowd simulation for smaller areas, do not scale well when hundreds of thousands or even millions of characters are needed. For example, the United States military already uses a wide variety of simulation systems for training [8,24], however, these simulations lack a satisfactory crowd models [22] that reflect both mob behavior and individual behaviors.

In the area of virtual environments, we have not yet fully explored effective ways to produce crowd simulations. Several efforts are driven by extending the particle system approach with different level of details [3,30] in order to reduce the rendering and computational cost. Although these methods can handle crowd dynamics and display populated interactive scenes (10000 virtual humans), they are not able to produce complex autonomous behaviors for the individual virtual humans. Other approaches focus on providing efficient and autonomous behaviors to crowd simulations [4,9,21,23,27]. However, they are based on a centralized system architecture, and they can only control a few hundreds of autonomous agents with different skills (pedestrians with navigation and/or social behaviors for urban/evacuation contexts).

Although some scalable, complex multi-agent systems have been proposed [31], most efforts focus on the software architecture, forgetting the underlying computer architecture and graphics system. As a result, important features like inter-process communications, workload balancing, network latencies, or graphics optimizations are not taken into account. When we are considering Emergency Response simulations, we are looking at how to balance the cost of providing rich behaviors, rendering quality, and scalability in a fairly complex integrated system. This balance is strongly dependent on what we call system throughput, defined as the maximum number of characters that the integrated system can handle in real time without reaching the saturation point.

In order to provide a high system throughput, we propose a distributed framework based on a hybrid architecture model. This framework consists of a networked-server distributed environment to manage the inter-awareness among the crowd members, and among the crowd and the environment. In addition, in order to support consistency and autonomous behaviors for each character, we propose a centralized software architecture. The results presented in later sections of the paper that were obtained in the performance evaluation show that this architecture can efficiently manage thousands of autonomous agents.

3 Architectures for Distributed Environments and Crowds

From the area of distributed systems, we know three basic architectures are: centralized-server architectures [25,33], networked-server architectures [12,13] and peer-to-peer architectures [15]. Figure 1(top-left), shows an example of a centralized-server architecture. In the context of crowd simulation, we can consider the virtual world as a plane and the avatars as dots in the plane. In this architecture there is only a single server and all the client computers are connected to this server. The server has a complete image of the world and all the avatars, and the clients simply report the local changes to the server. In this way, everything is easily synchronized and controlled. However, the centralized server limits the scalability of the system.

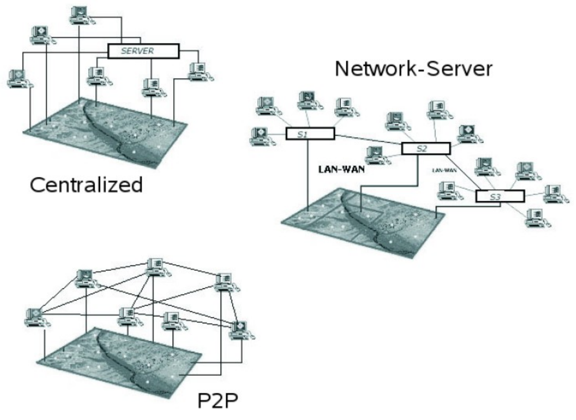

Fig. 1. Architectures for supporting DVE systems.

Figure 1(right) shows an example of a networked-server architecture. In this scheme, there are several servers and each client is exclusively connected to one of these servers. Again, in the context of crowd simulation, this can be thought of as the large virtual space being partitioned across several servers and the clients are distributed according to the avatars that are populating each one of the areas. This scheme is more distributed than the client-server scheme, and since there are several servers, it considerably improves the scalability compared to the client-server scheme. Figure 1 (bottom-left) shows an example of a peer-to-peer architecture. In this scheme, each client computer is also a server. This scheme provides the highest level of load distribution. Although the earlier distributed architectures were based on centralized architectures, during the last few years architectures based on networked servers have been the main standard for distributed systems. However, in problems related to crowd simulation, each new avatar introduced in a distributed system represents an increase not only in the computational requirements of the application but also in the amount of network traffic [16,17]. Due to this increase, networked-server architectures struggle to scale with the number of clients, particularly for the case of MMOGs [1], due to the high degree of interactivity shown by these applications. As a result, Peer-to-Peer (P2P) architectures have been proposed for massively multi-player online games [7,15]. One of the challenges of P2P architectures for crowd simulation is making avatars aware of other avatars in their surroundings [28]. Providing awareness to all the avatars is a necessary condition to provide time-space consistency. Awareness is crucial for our framework to maintain a coherent and consistent crowd model in the virtual space.

In a networked-server architectures, the awareness problem is easily solved by the existing servers, since they periodically synchronize their state and therefore they know the location of all avatars at all times. Each avatar reports its changes (by sending a message) to the server that it is assigned to, and the server can easily decide which avatars should be the destinations of that message (by using a criterion of distance). There is no need for a method to determine the neighborhood of avatars, since servers know that neighborhood every instant.

4 Architecture for Crowd Simulation

From the discussion above it seems that the more physical servers the DVE relies on, the more scalable it is. On the contrary, features like the awareness and/or consistency are more difficult to be provided as the underlying architecture is more distributed (peer-to-peer architecture). Therefore, we propose a networked-server scheme as the computer system architecture for crowd simulation. On top of this networked-server architecture, a software architecture must be designed to manage a crowd of autonomous agents. In order to easily maintain the coherence of the virtual world, a centralized semantic information system is needed. In this sense, it seems very difficult to maintain the coherence of the semantic information system if it follows a peer-to-peer scheme, where hundred or even thousand of computers support each one a small number of actors and a copy of the semantic database. Therefore, on top of the networked-server computer system architecture, we propose the software architecture as shown in Figure 2. This architecture has been designed to distribute the agents of the crowd in different server computers (the networked-servers). This centralized software architecture is composed of two elements: the *Action Server (AS)* and the *Client Processes (CP)*.

The Action Server. The Action Server corresponds to the *Action Engine* [10,11], and it can be viewed as the world manager, since it controls and properly modifies all the information the crowd can perceive. The Action Server is fully dedicated to verify and execute the actions required by the agents, since they are the main source of changes in the virtual environment. Additionally, another important parameter for interactive crowd simulations is the *server main frequency*. This parameter represents how fast the world can change. Ideally, in a fully reactive system all the agents send their action requests to the server, which processes them in a single cycle. In order to provide realistic effects, the server cycle must not be greater than the threshold used to provide quality of service to users in DVEs [6,18]. Therefore, we have set the maximum server cycle to 250 ms. Basically, the AS consists of two modules: the Semantic Data Base and the Action Execution Module. The SDB represents the global knowledge about the interactive world that the agents should be able to manage, and it contains the necessary functionalities to handle interactions between agents and objects. The semantic information managed can be symbolic (eg: object$_i$ free true, object$_i$ on object$_k$, ...) and numeric (eg: object$_i$ position, object$_i$ bounding volume, ..), since it has been designed to be useful for different types of agents.

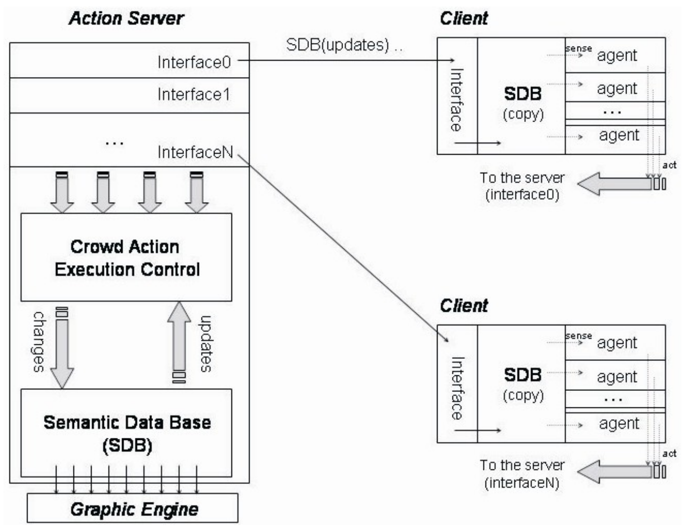

Fig. 2. The proposed software architecture

In order to manage the high number of changes produced, the AEM puts all the action effects in a vector (*vUpdates*) which reflects the local changes produced by each actuation (e.g.: an agent changes its position). Finally, when the server cycle has finished, this vector is sent to both, the clients and the SDB, which will update their correspondent environmental states (Figure 2).

The Clients. Each process in a client computer manages an independent group of autonomous agents (a subset of the crowd), and it is executed in a single computer as a single process. This process has an *interface* for receiving and updating the information from the server, and a finite number of threads (each thread for an agent). Using this interface, a client initially connects to the Action Server and downloads a complete copy of the SDB. From that instant, agents can think locally and in parallel with the server, so they can asynchronously send their actions to the server, which will process them as efficiently as possible (since each agent is a process thread, it can separately access to the socket connected to the server). When a server cycle finishes (every 250 ms.), the accepted changes are submitted to all the clients interfaces, that will update their SDB copies. The proposed multi-threading approach is independent of the agent architecture (the AI formalism driving the agent behavior), that is out of the scope of this paper. However, the proposed action scheme guarantees the awareness for all agents [28], since all the environmental changes are checked in a central server and then broadcasted to the agents. Although time-space inconsistencies can appear due to agent asynchronies and network latencies, all these inconsistencies are kept below the limit of the server period.

A classical complex behavior required by many crowd systems is pathfinding (eg: evacuations). In our system, a Cellular Automata (CA) [2] is included as a part of the SDB, and each cell has precomputed the *k-best* paths of length *l* to achieve the exit cells. To calculate all the paths (k paths per cell; cell = 1m side square), we are using a variation of the A* algorithm. Our algorithm starts from each goal cell, and by inundation, we can select the k best paths that arrive to each cell. This let us to manage large environments and to reduce the correspondent memory problems. Furthermore, the calculation of complete paths is not interesting generally, as agents can only evaluate the k-first steps before deciding its next cell.

During the simulation time, each agent can access to its cell and decide the next one according to this information, and the set of heuristics defined (eg: waiting time, path congestion). As an example, Figure 4 shows 7 snapshots captured from 2 different simulations of the system running an evacuation with 8000 agents in a (200mx200m) area. Both simulations start from the same intial state (left), which corresponds to a normal random distribution of the crowd. In the first case (Figure 4a), we have placed 4 exits in each side of the simulated area, so the crowd try to escape through them. In the other case (figure 4b) there are 20 exit cells located on the top of the maze, and we can easily recognize them by following the different crowd flows. The figure 4 also shows both crowd situations at cycle 50,150 and 300, where several congestions has already produced due to the design of the environment (a maze with narrow doors).

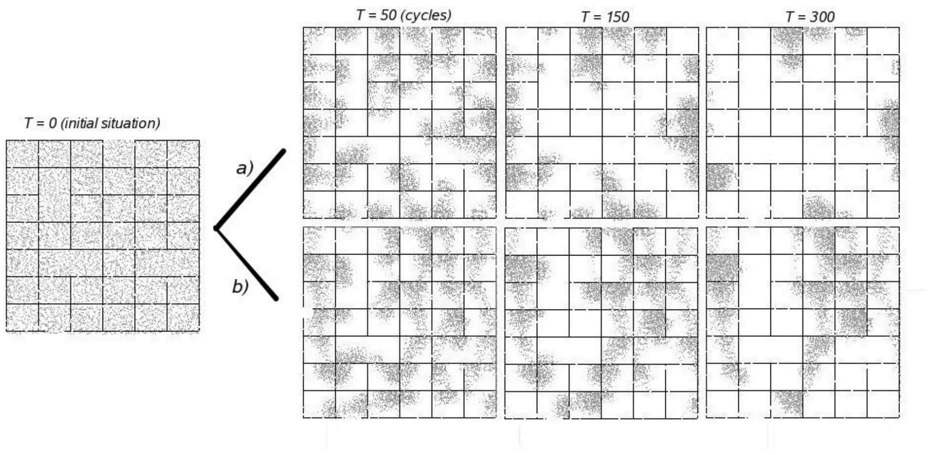

Fig. 4. Evacuation test with 8000 agents.

On other hand, it is important to ensure that the system can render good images of the virtual world in real time. Our architecture allows for the rendering processes to become one of the clients. In this way, all the information about the position and changes in each individual avatar is available to the rendering system in the same timeframe as it is available to the computational clients. This allows us for the synchronized rendering of the crowd as well as to apply rendering optimization approaches based on the viewpoint.

5 Performance Test

In order to evaluated actual performance, we have performed measurements on a real system with this architecture. Performance evaluation is based on wandering agents, since this type of agents is the one that generate the highest workload to the AS (they simply move). As cited previously, the AS cycle has been set to 250ms. First, we have focused on system throughput (the maximum number of agents that the system can efficiently support), which in our architecture is limited by the AS throughput. Concretely, we have measured the AS throughput when it is fully dedicated to collision detection tasks. The rationale of this test is to evaluate the number of actions that the server is able to carry out in a single cycle, since this could be a plausible bottleneck. When an action is requested by an agent, the server basically must access its cell and then it must compute a set of simple distance checking against the agents which are sharing the same cell, as figure 5-right shows. If no collisions are produced, then this process continues until the 8 neighbor cells pass the same test.

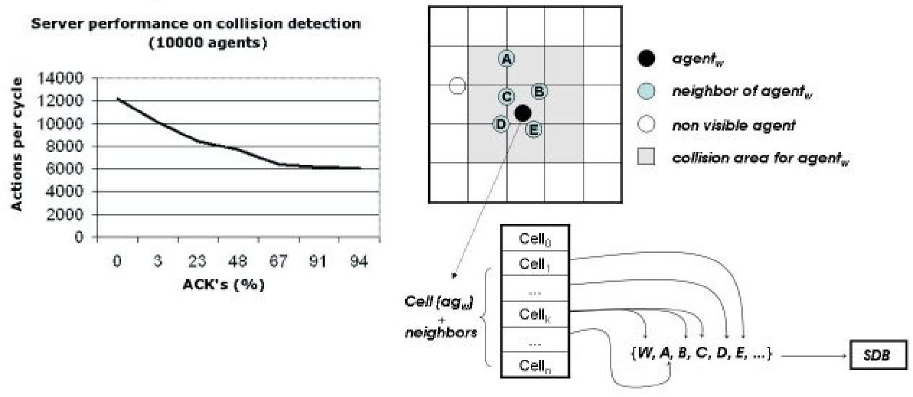

Fig. 5. Server Performance on Collision Test.

Figure 5-left shows the results obtained in a collision detection test performed in a single server where 10.000 agents demand a random position as soon as they can, in order to saturate the server. The purpose of this experiment is to know how fast the server can run, in terms of the average of actions that it is able to process in a single cycle. As this value highly depends on the density of the crowd, we have represented this parameter as the percentage of finally executed actions (ACKs), since it is more informative for our purposes. Thus, an ACK percentage of 0% occurs when no motion is allowed because the crowd is completely full and no one can move. On the other hand, when all the agents pass a full collision test, all the actions are allowed (100% of ACKs) and all the agents finally move. In these experiments, this case (94% of ACKs) represents the worst case because the server needs to access to 8 + 1 cells and compute a variable number of distance checks for each action.

Figure 5-left shows that in the worst case (94% of ACKs) the server is able to process around 6000 actions in a single cycle (250 ms). However, when the density of the crowd increases the percentage of ACKs decreases because the probability of collision increases in very dense worlds. This will allow the server to finish the cell checking without visiting all the neighbors cells. As a consequence, the server can process a higher number of actions requests per cycle (12000 actions for a percentage of 0% ACKS). It is also worth mention that for a medium case (48% ACKs), the system can manage around 8000 agents.

5 Conclusions and Future Work

In this paper, we review the computer architectures used in the literature to support distributed environments and analyze how they can support the simulation of large crowds. Based on this analysis, we present a scalable hybrid distributed framework for large-scale crowd simulations. At the lowest level, the framework consists of a computer system based on a networked-server architecture, in order to achieve scalability while providing awareness and time-space consistency. At a higher level, our framework integrates a software architecture based on a centralized semantic information system that can easily maintain the coherence of the virtual world through a single copy of the semantic database. Performance evaluation results show that this architecture can efficiently manage thousands of individual, autonomous agents at interactive rates.

This preliminary work has assisted us to define the next steps. We plan to distribute the action server in multiple machines by using distributed databases techniques in order to improve the scalability. In addition, we plan to characterize the requirements of different kinds of autonomous agents. The idea is to use each client for supporting one (or more) kind of agents, according to the computational power of the client and the requirements of the agents. Thus, by properly balancing the existing load among the clients we expect to improve the system throughput. Furthermore, in the rendering side, we are looking into the extension of our scene-graph to make use of largely distributed graphics systems to avoid the rendering bottleneck. By using distributed rendering methods like sort-last composition we can keep the data transfer overhead minimal and use local graphics hardware in the servers to scale to interactively displayed scenes with hundreds of thousands or millions of avatars.

References:

1. T. Alexander. Massively Multiplayer Game Development II. Charles River Media, 2005.
2. S. Chenney. Flow Tiles. In Proceedings of the 2004 ACM SIGGRAPH-EuroGraphics Symposium on Computer animation. ACM Press, 2004.
3. S. Dobbyn, J. Hamill, K. O'Conor, and C. O'Sullivan. Geopostors: a real-time geometry/impostor crowd.
4. S. Donikian. Informed virtual environments. In Proceedings of the 2004 ACM SIGGRAPH/Eurographics.
5. R. Fikes and N. Nilsson. Strips: a new approach to the application of theorem proving to problem solving. Artificial Intelligence, 5(2):189-208,1971.
6. Henderson and S. Bhatti. Networked games: a qos-sensitive application for qos-insensitive users? In Proceedings of the ACM SIGCOMM 2003,ACM Press / ACM SIGCOMM, pp. 141-147 2003.
7. L. Gautier and C. Diot. Design and evaluation of Mimaze, a multiplayer game on the internet. In Proceedings of IEEE Multimedia Systems Conference, 1998.
8. C.R. Karr, D. Reece and R. Franceschini. Synthetic soldiers - military training simulators IEEE Spectrum. Vol. 34, no. 3, pp. 39-45. March. 1997.
9. A. Iglesias and F. Luengo. New goal selection scheme for behavioral animation of intelligent virtual agents. IEICE Transactions on Information and Systems.
10. A. Iglesias and F. Luengo. Behavioral Animation of Virtual Agents (2003). Sixth International Conference on Computer Graphics and Artificial Intelligence, Limoges, France, May 2003, pp. 99-114.

11. A. Iglesias and F. Luengo. Framework for simulating the human behavior for intelligent virtual agents. part I: Framework architecture. Lectures Notes in Computer Science, 3039:239–236, 2004.
12. C. Greenhalgh, A. Bullock, E. Frion, D. Llyod, and A. Steed. Making networked virtual environments work. Presence: Teleoperators and Virtual Environments, 10(2):142–159, 2001.
13. J.C. Lui and M. Chan. An efficient partitioning algorithm for distributed virtual environment systems. IEEE Trans. Parallel and Distributed Systems, 13, 2002.
14. D.C. Miller and J.A. Thorpe. SIMNET: the advent of simulator networking. In Proceedings of the IEEE, volume 8 of 83, pages 1114--1123, Aug. 1995
15. S. Mooney and B. Games. Battlezone: Official Strategy Guide. BradyGame Publisher, 1998.
16. P. Morillo, J. M. Orduña, M. Fernandez, and J. Duato. On the characterization of distributed virtual environment systems. In Euro-Par' 2003 - Lecture Notes in Computer Science 2790, pages 1190–1198. Springer-Verlag, 2003.
17. P. Morillo, J. M. Orduña, M. Fernandez, and J. Duato. Improving the performance of distributed virtual environment systems. IEEE Transactions on Parallel and Distributed Systems, 16(7):637–649, 2005.
18. P. Morillo, J. M. Orduna, M. Fernandez, and J. Duato. A method for providing qos in distributed virtual environments. In 13th EuroMicro Conference on Parallel, Distributed and Network-based Processing (PDP'05). IEEE Computer Society, 2005.
19 . S.R. Musse and D. Thalmann. Hierarchical model for real time simulation of virtual human crowds. IEEE Transactions on Visualization and Computer Graphics, 7(2):152– 164, 2001.
20. National Emergency Response and Rescue Training Center (NERRTC),
21. H. Nakanishi and T. Ishida. Freewalk/ social interaction platform in virtual space. In VRST '04: Proceedings of the ACM symposium on Virtual reality software and technology, pages 97–104, New York, NY, USA, 2004. ACM Press.
22. Q.H. Nguyen, F.D. McKenzie, and M.D. Petty, "Crowd Behavior Architecture Model Cognitive Design", Proceedings of the 2005 Conference on Behavior Representation in Modeling and Simulation (BRIMS), Universal City CA, May 16-19,2005, pp. 55-64.
23. S. Raupp and D. Thalmann. Hierarchical model for real time simulation of virtual human crowds. IEEE Transactions on Visualization and Computer Graphics, 7(2):152–164, 2001.
24. A. Penn and A. Turner, Space syntax based agent simulation. 1st International Conference on Pedestrian and Evacuation Dynamics, 2001, University of Duisburg, Germany
25. Quake: http://www.idsoftware.com/games/quake/quake/
26. C.W. Reynolds. Flocks, Herds, and Schools: A Distributed Behavioral Model. Computer Graphics, 21(4), July 1987, pp. 25-34.
27. W. Shao and D. Terzopoulos. Autonomous pedestrians. In SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation, pages 19–28, New York, NY, USA, 2005. ACM Press.
28. R.B. Smith, R. Hixon, and B. Horan. Collaborative Virtual Environments, chapter Supporting Flexible Roles in a Shared Space. Springer-Verlag, 2001.
29. S. Singhal and M. Zyda. Networked Virtual Environments. ACM Press, 1999.
30. F. Tecchia, C. Loscos, and Y. Chrysathou. Visualizing crowds in real time. Computer Graphics Forum, 21, 2002.
31. H. Tianfield, J. Tian, and X. Yao. On the architectures of complex multi-agent systems. In Proc. of the Workshop on "Knowledge Grid and Grid Intelligence", IEEE/WIC International Conference on Web Intelligence / Intelligent Agent Technology, pages 195–206. IEEE Press, 2003.
32. A. Treuille, S. Cooper, Z. Popovic. Continuum Crowds. In ACM Transactions on Graphics 25(3) , (SIGGRAPH) 2006
33. Unreal Tournament: http://www.unrealtournament.com/