Navigating Virtual Agents in Online Virtual Worlds

Russell Gayle Dinesh Manocha
Department of Computer Science
University of North Carolina at Chapel Hill
{rgayle,dm}@cs.unc.edu
http://gamma.cs.unc.edu/SecondLife

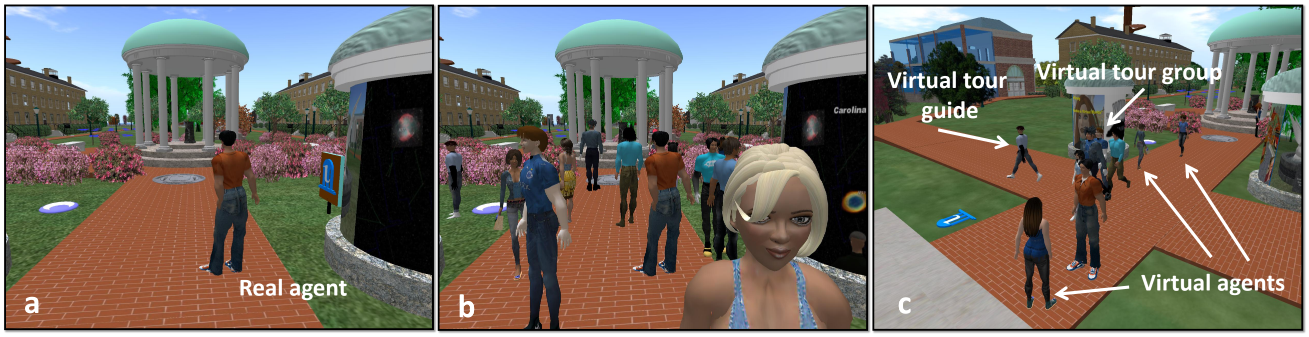

Figure 1: College campus*: (a) Many areas in online virtual worlds, such as this college campus in Second Life®, are sparsely inhabited. (b) We present techniques to add virtual agents and perform collision-free autonomous navigation. In this scene, the virtual agents navigate walkways, lead groups, or act as a member of a group. A snapshot from a simulation with* 18 *virtual agents (wearing blue-shaded shirts) that automatically navigate among human controlled agents (wearing orange shirts). (c) In a different scenario, a virtual tour guide leads virtual agents around the walkways among other virtual and a real agent.*

Abstract

We present an approach for navigating autonomous virtual agents in online virtual worlds that are based on a centralized server network topology. Each agent's motion is controlled through local and global navigation. Our local navigation model is based on artificial social forces that has been extended to account for inaccurate sensing from network latency. Global navigation for each virtual agent is based on cell decomposition and computes high level paths. The overall computation is balanced by performing local navigation on client machines and global navigation on the server. We have implemented our navigation algorithm into the Second Life virtual world and highlight our results by simulating up to 18 virtual agents over multiple different client computers.

Keywords: crowd simulation, virtual worlds, avatar behaviors

1 Introduction

Large, online 3D virtual worlds (VWs) have been growing rapidly in popularity due to rich environments, widespread interactivity and immersion, and large user bases. Millions of users are registered and are actively participating in worlds such as Second Life® (SL^{TM}), World of Warcraft^{TM} (WoW), or OLIVE^{TM} by Forterra Systems, Inc. In addition to these online worlds, Microsoft's Virtual Earth^{TM} or Google Earth^{TM} represent large and detailed navigable environments, although they do not currently support avatars. Many of these virtual worlds have been used for gaming applications and recently they have been shown useful for remote collaboration, economic planning, social simulations, and educational activities [Bainbridge 2007]. These online virtual worlds also provide a unifed environment to study complex global behaviors such as traffic flows or evacuations.

At any time, thousands of users are logged on, but they are typically widely distributed over the virtual world. As a result, many portions of these worlds appear sparsely inhabited. In simulations or experiences where interaction with other avatars is essential, this can pose a problem and can detract from the overall immersive experience. One way to overcome this problem is to add virtual avatars to populate these worlds and improve the overall immersive experience. In this context, a virtual agent or avatar is a member of the virtual world whose motion is controlled entirely by a simulation, whereas a human agent is controlled by a human user.

There are several challenges in adding realistic virtual agents to online virtual worlds. First, virtual agents should automatically navigate the environments and act appropriately in a variety of situations, such as maintaining a separation from other agents or moving in a formation. Secondly, networking issues such as limited bandwidth and client-to-server latency add uncertainties into much of the navigation process, resulting in incomplete information about the environments. Finally, performance is a key consideration in evaluating online virtual worlds and adding virtual agents should not greatly degrade the performance of the online virtual world system.

Main results: We present an approach for adding autonomous virtual agents into online virtual worlds with a centralized server network topology. We combine techniques to control the motion of each agent based on local and global navigation with a general networking model. Briefly, local navigation determines how agents coordinate with nearby objects whereas global navigation allows agents to select goals in order to complete their current task or reach the goal position. To reduce the impact of network latency, we make assumptions about the linearity of motion in a short time interval and augment the model with a velocity-bias social force. Since global planning is agent-specific, our algorithm that runs on the server can be used to balance the computational load between client and server.

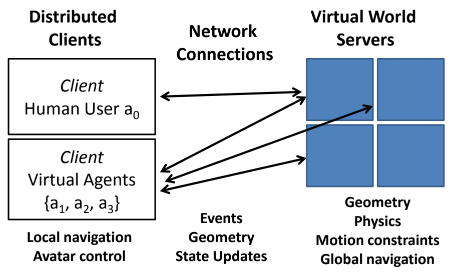

Figure 2: Online Virtual World Architecture: *The virtual world servers are responsible for maintaining and correcting the state of all avatars and objects in the world. We assume that all clients communicate with the individual servers in a centralized manner.*

We have implemented our approach into the Second Life virtual world using a motion controller built with Libsecondlife (LibSL). We highlight our results by simulating up to 18 agents over two different client computers of varying computation power, in different geographic locations and using different internet bandwidths.

Organization: This paper is organized as follows. Section 2 describes prior work in multi-agent simulation and navigation. Our virtual agent navigation model is described in Section 3. Section 4 concludes the paper with a description of the implementation and results.

2 Related Work

In this section, we give a brief overview on prior work on motion planning and navigation of multiple agents. For theory and applications of general motion planning for multi-robot systems, we refer the readers to [LaValle 2006]. And, for details on design and implementation of networked virtual worlds, we refer the readers to [Singhal and Zyda 1999].

There has been a great deal of work on modeling the motion of individual agents as well as those in small groups and large crowds [Ashida et al. 2001; Shao and Terzopoulos 2005; Thalmann et al. 2006; Treuille et al. 2006; van den Berg et al. 2008]. Several approaches have been proposed for generating motion of crowds; including agent-based methods, cellular automata, and methods based on discretized and continuous flows.

Agent-based methods typically include methods and rules for determining a heading based on information local to an individual agent. One of the earliest works by Reynolds [1987] described simple local rules for efficient and plausible flocking and herding behaviors. This model has been extended in many ways to include other factors including psychological [Pelechano et al. 2005] and sociological [Musse and Thalmann 1997] preferences.

Cellular automata methods model the evolution of agent locations by solving cellular automata. Different techniques for generating rules have been proposed, including methods based on static and dynamic fields [Hoogendoorn et al. 2000], and behavioral models [Tu and Terzopoulos 1994].

Flow-based approaches treat the motion of multiple agents like that of physical flows. In discretized approaches, agents are treated like particles in a 2D dynamic simulation [Helbing et al. 2003; Lokoba et al. 2005]. Our motion formulation is most closely related to that of Helbing et. al [2003].

Additionally, several approaches have been proposed to include global navigation for groups of human agents. Voronoi graphs have been efficiently used to compute and update navigation graphs [Sud et al. 2007a]. Other statically generated navigation graphs allow agents to navigate [Pettre et al. 2007] and dynamically updating graphs have been used to combine global navigation with local behavior [Sud et al. 2007b].

3 Autonomous Virtual Agents

In this section, we introduce the notation used throughout the paper and present our approach.

3.1 Notation and Definitions

We represent an agent as $a_i \in \mathcal{A} = \{a_0, \ldots, a_n\}$ and is represented by a cylinder of finite radius r_i. The state $\mathbf{q}_i$ of a_i at time t is denoted by $\mathbf{q}_i(t) = \{\mathbf{x}_i(t), \mathbf{v}_i(t)\}$ for position $\mathbf{x}_i$ and velocity $\mathbf{v}_i$. Note that we make no distinction between whether other agents are controlled by human users (i.e. real agents) or are virtual agents. This allows our virtual agent to treat all other agents in a uniform manner.

We assume that environment E is composed of polygonal or 3D polyhedral obstacles $\mathcal{O} = \{o_0, \ldots, o_m\}$. The number of obstacles in $\mathcal{O}$ can vary for each agent. Additionally, we assume there is a centralized network topology, such that each agent connects to a server, or group of servers with a shared database. These servers are responsible for sending the agent's state information to all other agents connected to that particular server (See Fig. 2).

3.2 Collision Free Navigation

Due to uncertainties and latencies caused by the network, the position of other agents and the obstacles may not be precisely known. As a result, it is difficult to compute absolutely collision-free motion.

Our approach breaks down agent's navigation into two portions; local and global navigation. Local navigation allows the agents to make adjustments to their current path based on nearby agents and obstacles. Global navigation provides a sequence of subgoals for an agent so that it can reach its final goal or destination.

3.2.1 Local Navigation

The local motion model for each virtual agents is based on the concept of *social forces*, i.e. non-physical forces that can mimic decisions and behavioral responses [Helbing et al. 2003]. These forces are used to guide an agent along a path toward its intermediate or final goal. The general idea is that each agent and obstacle generates a repulsive or attractive force field around itself. At each discrete time instance, the agent a_i samples a force field at its current location. The resulting force is applied to a_i, resulting in a motion trajectory.

Our formulation of social forces extends the models proposed by Helbing et al. [2003] and later extended by Lakoba et al. [2005]. In practice, this model is able to capture emergent behavior of crowds with varying number of agents per in different areas. For sake of space, we refer interested reader to these articles for additional details [2003; 2005]. Fig. 3 provides a brief illustration and description of the ideas.

Contact Handling: While the repulsive forces generally allow agents to avoid contacts, they do not give guarantee of a collision-free motion. Social forces with a large magnitude can cause the agents to move directly towards each other. Moreover, network communication latencies can result in incomplete information about the location of other agents, objects or dynamic obstacles in

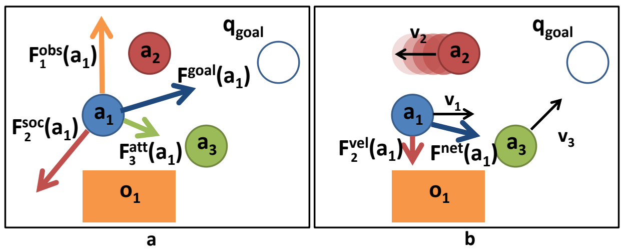

Figure 3: Local Navigation via social forces: *The colors of the arrows correspond to the specific object that generates the social force. (a) Agent a_1 is acted upon by social repulsive force $\mathbf{F}_2^{soc}(a_1)$ from a_2, attractive force $\mathbf{F}_3^{att}(a_1)$ from a_3, repulsive obstacle force $\mathbf{F}_1^{obs}(a_1)$ from o_1, and goal force $\mathbf{F}^{goal}(a_1)$. Repulsive and attractive forces encourage the agent to move toward or away from agents and obstacles, respectively. (b) An additional velocity bias force $\mathbf{F}_2^{vel}(a_1)$ is computed to account for agent a_2's velocity during inaccurate sensing. By assuming a linear trajectory over a short period of time, we can help to reduce the impact of network latency. No velocity bias force is computed for a_3 since it is heading in the same direction as a_1. The final net force $\mathbf{F}^{net}(a_i)$ reflects the sum of all the forces and serves as the agent's next heading.*

the scene. In these situations, contacts between the agents must be resolved. We assume that the central server is ultimately in charge of resolving intersections between both agents and obstacles. Since latency is a factor even in the absence of virtual agents, it enforces hard constraints on agent positions to prevent intersection and returns that information to the clients.

Network Modeling: Our virtual agent model is computed based on the information on a client database. This introduces certain amount of uncertainty in the position and velocity of other agents and also in other moving obstacles in two ways. First, some agents or obstacles may not have yet been sensed, i.e. their base information has not yet been received by the client from the server. Second, due to low network bandwidth or high network latency, the updated positions and velocities are not received in time by both the server and client.

To account for these issues, we augment the social force model with a velocity bias force, $\mathbf{F}_j^{vel}(a_i)$, to help reduce the impact of latency. Let $V_j = SSV(a_j, \mathbf{v}_j, t_{bias})$ be the spherical swept volume as agent a_j travels along heading $\mathbf{v}_j/||\mathbf{v}_j||$ for time t_{bias}. Then,

$$\mathbf{F}_j^{vel}(a_i) = \gamma e^{(r_i - d(V_j, \mathbf{x}_i)/\epsilon)} \mathbf{n}_{(i,V_j)},$$

where γ is a velocity bias scaling factor, ϵ is a bias dropoff distance, and $d(.,.)$ and $n(.,.)$ are the distance and normal direction between the a_i and volume V_j. Intuitively, the force naively assumes that agent a_j will proceed in its current direction for a fixed period of time and generates a force field around the volume swept of a_j along that heading (See Fig. 3(b)). As a result, other agents will tend to move away from their current direction of motion. The net effect of this force is that it gives virtual agents a way to estimate where other agents will go based on the information is available, and thus reduces the impact of latency or bandwidth limitations.

Goal Selection: The last portion of our model for local motion computation involves selecting intermediate goals. The selection of goals can result in a variety of behaviors. In our formulation, we use two different methods to select the goals. First, the goals are randomly selected within some radius of the agent. This gives the appearance of wandering or exploring within that radius. Second, scripted goals can be provided for other effects. For instance, in a panic situation the agents would be given goals away from the cause or location of the panic.

3.2.2 Global Navigation

In many situations, local navigation may be insufficient to navigate an agent through a complex environment. This may be due to the lack of a straight-line route to the goal or due to a local minima in the social force field which prevents progress from being made.

In these situations, a global planner is necessary. The global planner provides agents with a sequence of subgoals which will eventually lead to the final goal. There are a wide variety of options available for route planning or roadmap computation [LaValle 2006]. For simplicity, we use a cell decomposition based approach, extended for use with unknown and changing environments.

Our approach initially samples all navigable surfaces, such as floors in the buildings or walkways. It uniformly samples the surfaces with a sampling resolution based on the size and scale of the environment. Each sample is classified as free, in collision, or in the mixed region of the cell decomposition. Next, we connect neighboring free samples in order to generate a connectivity graph. When an agent requests a path from the global planner, an A* search is performed on this graph at runtime, and a sequence of subgoals is computed. The free or colliding state of each cell is updated as changes occur in the environment. These changes in turn notify the virtual agents when they need to recompute a path.

Server Planning: Within a localized region, there is little need to duplicate the global navigation system since it will be essentially the same for each virtual agent. One simple optimization would be to make a single module which would perform planning computation for each agent. By placing this module on the server, the cost of planning and replanning is reduced. Furthermore, it acts as a means to balance some of the overall computation by placing some work on the server while the client does the rest.

4 Discussion and Results

In this section we describe our implementation, show results of our virtual agent model, and discuss possible issues with the approach.

4.1 Implementation

A preliminary implementation of his approach has been developed for Second Life, based on the LibSecondlife (LibSL) framework. LibSL is used to reverse engineer the SL network protocol. With this, it is possible to create a SL client session without using the official SL viewer resulting in avatars that appear as full clients to the server. The avatar motion control model along with the local agent dynamics has been implemented on top of this framework.

For testing, one PC ran only virtual agents while a second PC ran both virtual agents and a viewer. These computers were placed on different networks, a university network connection and an at home cable modem respectively.

4.2 Results

We have tested our approach in a two scenarios:

- **Populating a city block**: This city block scenario includes 2 streets, a fountain and a patio of a building. Virtual agents are added to the city block to improve realism (Fig. 4). Virtual agents randomly select points of interest while interacting with each other through repulsive forces. The server managed 18 virtual agents and 3 real agents. The 17 virtual agents were distributed between two PCs.

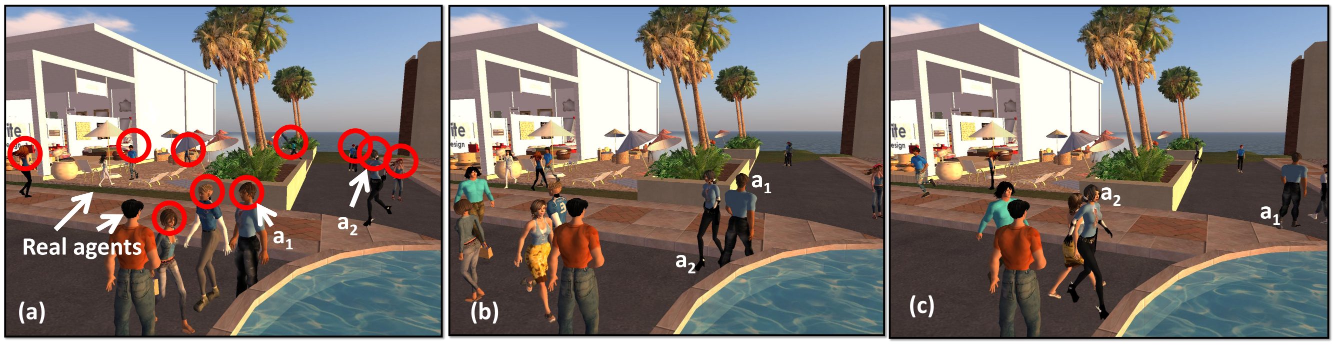

Figure 4: City block: *A sequence of still images following the evolution of virtual agents in a small city block. Several virtual citizens (circled in red and wearing shirts in various shades of blue) move around the streets and occasionally stop by the fountain. Agents a_1 and a_2 are identified to demonstrate their avoidance of each other as they cross near the fountain. A human controlled agent (orange) acts as an obstacle and must be avoided. The scene has* 18 *virtual agents distributed over* 2 *PCs.*

- **Campus tour guide**: To show a wider variety of behaviors, we added agents to a university campus. This scene included three classes of virtual agents; wandering students, a tour group, and a tour guide. Grouping and following along with the effects of both attractive and repulsive behaviors and following are observable in this scene (See Fig. 1(c)). There are a total of 20 agents, 2 real and 18 virtual agents, in the scene. As before the virtual agents were simulated on two client PCs.

The overall run-time performance in both scenarios was about the same. Virtual agents were distributed over 2 client computers. After tuning the computers, we were able to achieve interactive performance. So, it is likely that this approach will scale with additional client computers.

While the simulation computation was relatively cheap, the biggest limiting factor was the available bandwidth. Each agent, even if they're hosted on the same client computer, sends and receives a extensive amount of information to the server. In our experiments, each PC could support about 8 to 15 agents without noticeable degradation in performance. It should be noted that our limit of 18 agents is unrelated to the performance. Instead, it was the number of accounts on Second Life we were able to use. In the future, we expect to use many more agents.

5 Conclusion and Future Work

We have presented a virtual agent navigation algorithm for online worlds. In our implementation, each virtual agent navigates using an established pedestrian model augmented to take sensing latencies into account. Our preliminary results are promising and the algorithm can control and simulate several virtual agents from a single computer.

While the approach works well, there are some limitations. The approach generally requires more bandwidth, which can become a bottleneck. Moreover, server-based solutions probably require modifications on the server if not already supported and thus may be less practical. As with many "potential field" planners, there is no guarantee of success in reaching a goal. Finally, we hope to scale our work to include much larger crowds in virtual worlds.

Acknowledgements

This research was supported in part by ARO Contracts DAAD19-02-1-0390 and W911NF-04-1-0088, NSF awards 0400134, 0429583 and 0404088, DARPA/RDECOM Contract N61339-04-C-0043 and Intel. The first author is also supported by a Department of Energy High Performance Computer Science Fellowship administered by the Krell Institute.

References

Ashida, K., Lee, S. J., Allbeck, J., Sun, H., Badler, N., and Metaxas, D. 2001. Pedestrians: Creating agent behaviors through statistical analysis of observation data. *Proc. Computer Animation*.

Bainbridge, W. S. 2007. The scientific research potential of virtual worlds. *Science*.

Helbing, D., Buzna, L., and Werner, T. 2003. Self-organized pedestrian crowd dynamics and design solutions. *Traffic Forum 12*.

Hoogendoorn, S. P., Luding, S., Bovy, P., Schreckenberg, M., and Wolf, D. 2000. *Traffic and Granular Flow*. Springer.

LaValle, S. M. 2006. *Planning Algorithms*. Cambridge University Press.

Lokoba, T. I., Kaup, D. J., and Finkelstein, N. M. 2005. Modifications of the helbing-molnr-farkas-vicsek social force model for pedestrian evolution. *Simulation*.

Musse, S. R., and Thalmann, D. 1997. A model of human crowd behavior: Group inter-relationship and collision detection analysis. *Computer Animation and Simulation*, 39–51.

Pelechano, N., O'Brien, K., Silverman, B., and Badler, N. 2005. Crowd simulation incorporating agent psychological models, roles and communication. *First International Workshop on Crowd Simulation*.

Pettre, J., Grillon, H., and Thalmann, D. 2007. Crowds of moving objects: Navigation planning and simulation. In *ICRA*, 3062–3067.

Reynolds, C. W. 1987. Flocks, herds, and schools: A distributed behavioral model. *Comput. Graph. 21*, 4, 25–34. Proc. SIGGRAPH '87.

Shao, W., and Terzopoulos, D. 2005. Autonomous pedestrians. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM Press, New York, NY, USA, 19–28.

Singhal, S., and Zyda, M. 1999. *Networked Virtual Environments: Design and Implementation*. Addison-Wesley Professional.

Sud, A., Andersen, E., Curtis, S., Lin, M., and Manocha, D. 2007. Real-time path planning for virtual agents in dynamic environments. *Proc. of IEEE VR*, 91–98.

Sud, A., Gayle, R., Andersen, E., Guy, S., Lin, M., and Manocha, D. 2007. Real-time navigation of independent agents using adaptive roadmaps. *Proc. of ACM VRST*. to appear.

Thalmann, D., O'Sullivan, C., Ciechomski, P., and Dobbyn, S. 2006. *Populating Virtual Environments with Crowds*. Eurographics 2006 Tutorial Notes.

Treuille, A., Cooper, S., and Popovic, Z. 2006. Continuum crowds. *Proc. of ACM SIGGRAPH*, 1160 – 1168.

Tu, X., and Terzopoulos, D. 1994. Artificial fishes: Physics, locomotion, perception, behavior. In *Proceedings of SIGGRAPH '94*, A. Glassner, Ed., 43–50.

van den Berg, J., Patil, S., Sewall, J., Manocha, D., and Lin, M. C. 2008. Interactive navigation of individual agents in crowded environments. *Proc. of ACM Symposium on Interactive 3D Graphics and Games*.