Agent-Based Simulation for Infrastructure Protection and Emergency Evacuation Training

Shri Rai
School of Information Technology
Murdoch University
Australia
61 8 9360 6090
s.rai@murdoch.edu.au

Kevin Wong
School of Information Technology
Murdoch University
Australia
61 8 9360 6100
k.wong@murdoch.edu.au

Abstract

Simulators have been used for some time to provide training to people in a number of occupations. Simulation systems enable what-if questions to be posed and the consequences of user actions to be studied in a cost-effective and safe manner. Simulations also enable detailed user behavior to be logged for later study to extract data that may be difficult to capture in real life. This paper describes the development of a multi-user simulation platform that enables certain emergency events to be simulated. All user behavior is logged for later analysis so that user behavior under certain stressful events can be studied. Software agents in the simulation system can be used to model crowds or agents with a particular intent. The system can be used to provide emergency evacuation training. The system can also be used to test the security of building designs to find out how the security of these buildings can be compromised.

Keywords

Agent-Based, Simulation, Emergency Virtual Evacuation Environment, Avatar.

1. Introduction

It is usual for building codes to incorporate fire safety regulations. As part of the duty of care, responsible building owners now have to take into account other threats to life and property. Such threats include terrorist acts. Evaluating the likelihood of such acts and how to prevent them is not easy. In fact, it is also not easy to accurately predict what would happen in case of a fire even when buildings are designed to satisfy fire safety regulations. This is because the fire safety regulations are updated because of lessons learnt from actual fires that took place. The regulations are not normally meant to predict but to avoid known problems. As an example, Australia's Commonwealth Scientific and Industrial Research Organization (CSIRO) had provided guidelines for predicting fire behavior in 2007 [1]. Unfortunately, the most recent bush fires in Eastern Australia demonstrated that fires can be quite unpredictable [2]. Compounding the unpredictability of fire is the fact that it is also not easy to model actual human behavior during fire. The study of such behavior even warrant their own symposiums [3].

If fire regulations catered for every possible scenario, whether real or imagined, it would result in significant compliance cost to building owners. This same argument would apply when trying to cater

for every imaginable terrorist related event. Building owners have to provide evacuation plans and conduct emergency drills presumably to meet the building insurance requirements. This means the placement of wall charts showing evacuation paths and the periodic conduct of drills in the hope that when an emergency event happens, no life will be lost. Neither of the above approaches necessarily implies that a building's occupants have constructed a cognitive map of what to do in the case of a real emergency where smoke and fallen debris may be obstructing the evacuation path. In animal studies, Poucet [4] pointed out that cognitive maps are used "*to indicate where in space potentially important objects are located.*" It was also pointed out [4] that animals do not exhibit behavior depending on their need (including hunger) until they become familiar with their environment. In his classic paper, Tolman [5] argued that when an individual is faced with a very difficult problem, the individual shows signs of "regression". During regression, the individual is not capable of adult like reasoning and goes to "childish ways of behaving". Tolman points out that regression is coupled with "fixation" where the individual goes back to the original path that they have learned even though the original path is in in-correct or is not a valid path anymore. What all this implies is that under situations of extreme stress, an individual's ability to think laterally suffers. So this means that even if the building occupants had participated in evacuation-drills, under situations of extreme stress they may resort to taking a learned path to escape even though the learned path may not be a viable path in that particular instance. It is often the case that during a drill, the occupants would not have used any alternative path as the evacuation wall chart usually shows one assembly area and a path to it.

The safety of people's life should not be seen as just insurance risk management as there are computer based technologies available that enable them to learn how to evacuate under various situations. Computer based multi-agent simulations can help as these simulations permit various scenarios to be examined quickly.

In this paper, we describe a multiplayer system that enables emergency events, including terrorist events, to be simulated. The system is designed using client-server architecture. The client sits on everyone's computer in their offices and the server resides on a designated server. The clients can be setup to load at boot time and can be used as chat tools with other building occupants when no simulation is being carried out. Chatting can be done remotely or virtual face to face chats where an avatar can walk to another person's room in the virtual world.

Section 2 provides and overview of the system. Section 3 explains how the simulation world is constructed and section 4 looks at the make up of avatars and agents in the simulation world. Section 5 has results and discussion.

2. System Overview

Various approaches were considered for systems development. One possibility was the use of Game Engines. Game Engines like the Unreal engine have been used to develop military simulations [6], [7]. We decided not to use an existing Game Engine for two reasons: we wanted to understand the

software development issues involved in building simulations and we wanted total control of everything that could be done. An in-house engine (Tornado engine) was created (written in C++) that permitted simulations to be built. The engine was capable of handling rendering, scene management, networking, Artificial Intelligence (AI) scripting using Lua [8] scripts, Path finding, Physics, Sound using IrrKlang [9] , Networking [10]. It had state logging capabilities which stored all activities and events in a relational database (MySQL). There was also a simple dialog system which enabled agents to communicate with users. The engine was then used to build the components of the simulation system called Emergency Virtual Evacuation Environment (EVE^2 or just EVE for short) prototype. EVE consisted of a central server and multiple clients which interacted with Tornado through Tornado's Application Programmers Interface (API).

The server internal operations include:

- Server creates scene manager without window visibility.
- Server creates world and passes the scene manager.
- Server initializes and validates database.
- Server loads scripted world parameters for scene manager population.
- Server begins constantly listening for command line input.
- Server begins constantly listening for client connections.
- Server accepts client connections one at a time, and server pauses to pass networked world and scenario data to the client. Server requests database to allocate space and to return a unique identifier for the new client. New clients can be added at any time during execution of server application.
- Once a Client has established a connection, a log is created for that client. Everything the client interacts with will also be logged.
- Server continues processing normally after client has received all world data and has its data structures synchronized with the server.
- Server updates internal world data structures continuously.
- Server submits and receives updated world data across the network continuously.
- Server initiates scenarios at specified intervals dependant on console input from server administration.
- Server shutdown must first issue a packet broadcast to clients, forcing them to shutdown their current world. Server then de-allocates its memory and shuts down normally.

The client's internal operations include:

- Client creates scene manager with window visibility.
- Client creates world and passes it the scene manager.
- Client creates initial factory (design pattern) populates it with menu world data (which is stored locally on the client, but is updatable).
- Client selects a server from the listing generated on the server selection window, inputs personal parameters, and accepts to send connection request to server.
- Client receives acceptance to server (or is denied in which case the client will need to select a different server).
- After client establishes a server connection, it will begin accepting world and scenario information and start filling its localized scene manager and data structures with the received data.
- While client is accepting world and scenario data, the main menu screen will be displayed with the 'start game' option greyed out (un-selectable) until all game data has been accepted over the network and the application has been synchronized to a current snapshot of the server world data. During this time the user can still access the other functionality of the menu (i.e. options, credits).
- Client has accepted all world and scenario data, and the 'start game' option will become available for the user to enter the world. At this time constant world updates are in effect.

- Client scene management will now be updating two factories (design patterns), one for the main menu (with graphical rendering) and one for the world data, which will be constantly updated in the background from data received over the network from the server.
- Client selects the 'start game' option and the main menu factory is disengaged and not updated until next accessed. At this point the world factory is engaged and its data representations begin to be rendered to the graphical window.
- Client continually requests and submits data over the network during the runtime of the world.
- Client exits the application by re-accessing the main menu and selecting the 'exit' option from the main menu (this re-engages the main menu factory to be con-currently updated alongside the main world). Client exit can also be triggered forcefully by server broadcasts that issue a shutdown.
- Client application stops receiving and submitting data across the network, de-allocates its localized data structures and safely shuts itself down.

Figure 1 shows the various clients connected to the server through a switch. It is not necessary for the clients to be on the same subnet. The clients can connect to the sever using an Internet connection. Both clients and server run on Microsoft Windows.

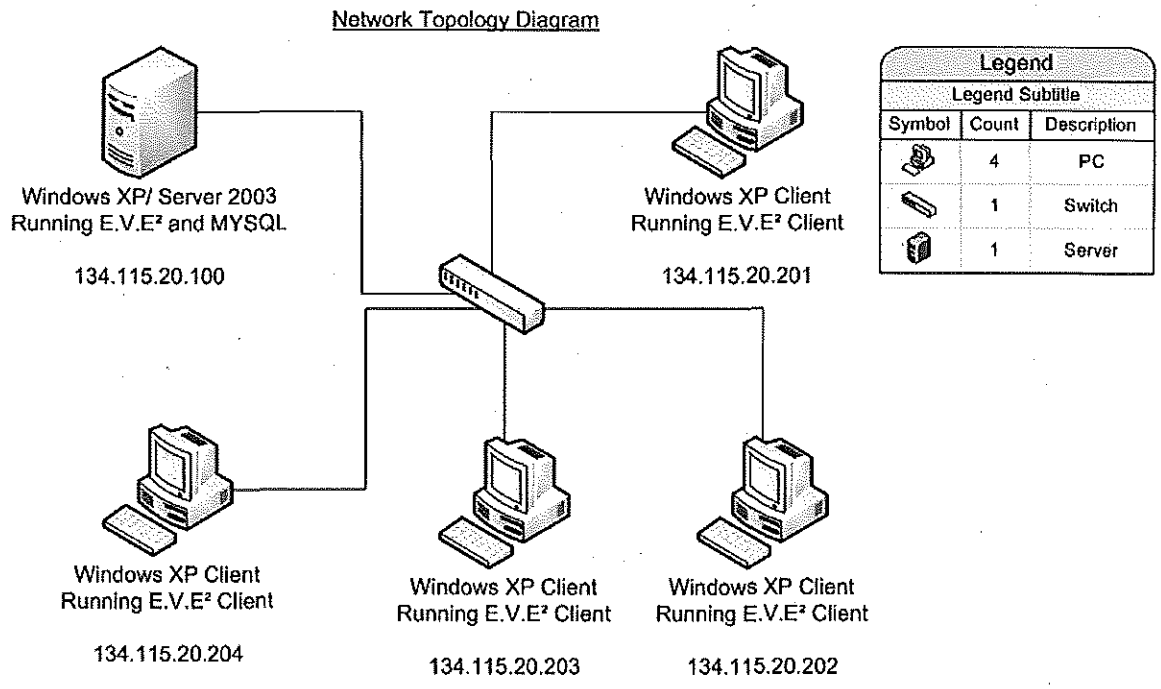

Figure 1

Figure 2 shows the user interactions on the system. The Level 0 user termed "Employer" has control of the world that Level 1 user termed "Employee" sees and experiences. In figure 2, the terms "employee" and "employer" are just terms used to differentiate the type of users. The employer can be the safety officer. The Employer can initiate emergency events on the system.

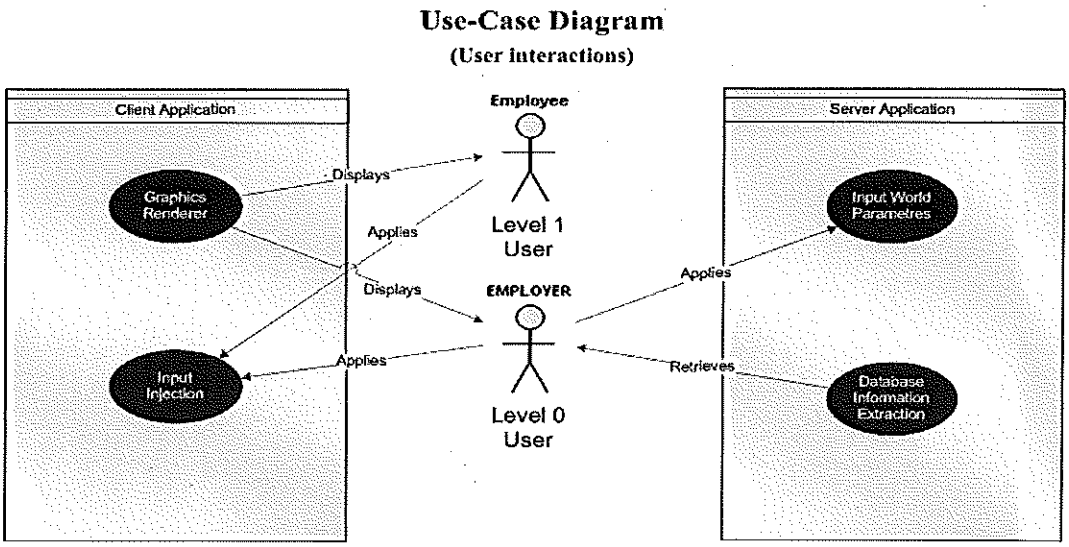

Figure 2

The timing and location of the emergency event can be decided by the Employer. When no emergency events are active, the system can be used as a chat system. Once an emergency event is activated, audio and visual alarms sound. Employees have to find their way out of the building to the evacuation area. All interactions are logged when emergency event is active. The employer can access the logs to find out what went on during the emergency event. This includes employee behaviour.

3. World Generation

The building model is created using tools like 3D Studio Max or Maya. The model is based on actual CAD plans for the building (Figure 3). Emergency events like fires and resulting smoke are modelled using particle systems. The spread of fires is also modelled based on passages and openings in the buildings and the locations of materials which are deemed to be combustible. Fires move from one area to the next if there are passages linking the two areas and there is material designated as combustible.

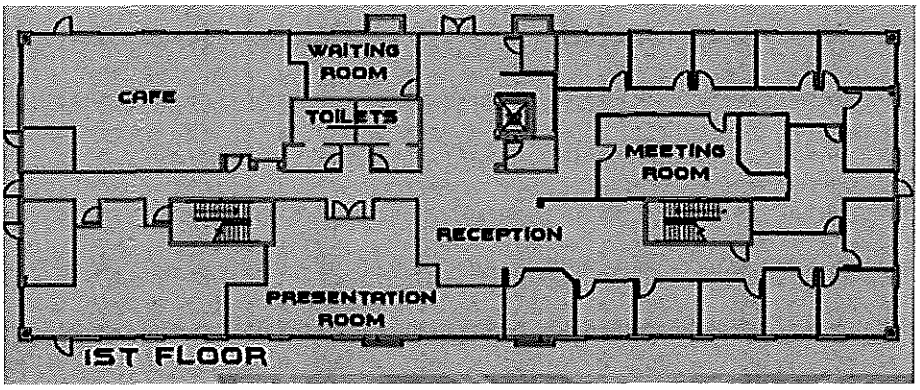

Figure 3

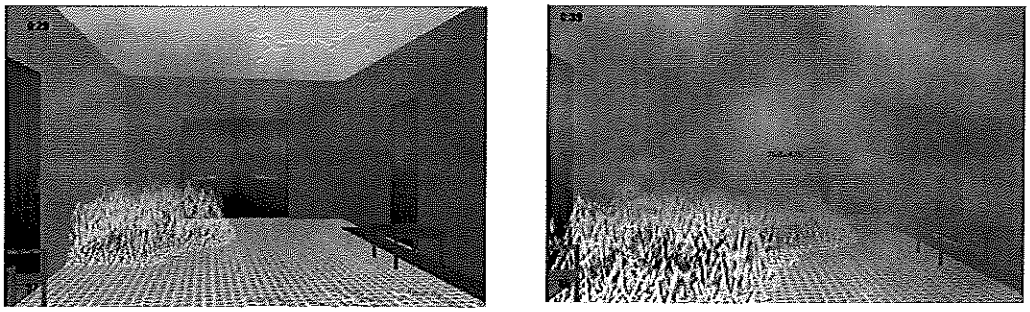

Figure 4

The left image in Figure 4 shows fire and smoke about 29 seconds into the simulation. A fire and smoke build up a few seconds later is shown in the right image of Figure 4.

Models in the world are constructed of more elementary objects. These objects exhibit affordances [11]. For example a chair affords sitting. This is how virtual agents know where to sit when they are tired. The particular version implemented is described as Perceived Affordance by the Cognitive psychologist Donald Norman [12].

4. Avatars and Agents

Avatars are the employer and employee users in the system. Each employee or employer is enrolled in the system. In the current system, they select a visual appearance from a pre-defined set. They provide a user name and mobility information. In the prototype, only the maximum walking speed for each individual was used. This determined the maximum speed their virtual personas could move in the simulation. It is possible to have other detailed mobility characteristics to be used including the use of wheel chairs. Wheel chair bound users were not used in the prototype but such users are an important category with special requirements during evacuation as they are unable to use stairs.

Emergency events are created by employers. The employer's avatar can walk through the simulation system and initiate events in the simulation system.

Software agents are made to populate the world to give the illusion that the world is occupied. They loosely follow the normative approach as described by Vicente [13]. These agents have simulated emotional systems which range, for example, from joy to sadness and from fear to acceptance. The emotional states get affected by the environment they are in [14]. The agents exhibit goal based behaviours which are specific to each agent and depends on their current state and the state of the environment. For example, if they are hungry, they start looking for food. They find food, amongst other things by querying the affordances of the objects in their environment. Agent-world object interaction is also determined by the affordance system. They move around in the world by executing the A* search algorithm [15] as their path finding algorithm.

The Tornado engine allows avatars and agents to affect the world as they are able to move world-objects around and this includes remodelling the interior of the building during run-time, if necessary. In EVE, restrictions can be placed on the avatars and agents if it is not deemed desirable that a user would enter another user's office and remodel the office. These are just artificial restrictions and need not be placed on agents (and avatars) whose purpose is mischief. Terrorists can be modelled this way. Terrorist software agents can query the affordance system to determine flammability of the objects in their vicinity and can start fires. The AI scripting part of the engine permits the construction of agents for particular purposes. So, for example, penetration agents can be constructed. Their purpose is to penetrate regions which afford "security".

5. Results and Discussion

During the simulation runs, it was found that some participants had problems navigating. The keyboard interface using the "W", "A", "S" and "D" keys which would be natural to PC gamers is not as intuitive to non-gamers. The non-gamers would collide into anything and everything and their evacuation times were much higher than gamers. Some of them even tended to get lost as they were not used to building a mental model of a virtual world. Fortunately, these users got better with practice. This has important implications if simulations are to be used for evacuation training. The user interface needs to be friendlier. The keyboard arrow keys should also substitute for the actual letter keys. Building occupants should also be encouraged to walk around in the simulated world in their own time if they do not wish to repeatedly explore the real world on foot. In any case, as has been argued by Takeuchi and colleagues [16], just showing the simulation runs serves an important educational purpose to the stake holders.

The avatars simulate the behaviour of the current building occupants and this includes getting in each other's way, if for example, they all rush to the same flight of stairs to escape. The 3D buildings are constructed to scale from CAD drawings and all movement is appropriate to the scale. Many buildings have automatic opening doors which are movement triggered and they take some time to open. In one of the simulation runs, it was found that when many of the building occupants somehow made it to such a door almost at the same time, egress from the building was actually delayed because of the way the door operated. Avatars and agents started to collide with each other hampering the exit. This demonstrated a design flaw in the door opening mechanism which might cause people to be injured in a real life emergency event.

An interesting behavioural result emerged during the operation of the system. This result was originally unintended. It was noted that during the actual emergency drills conducted on campus, staff and students would walk to the evacuation area with no sense of urgency. However, in the simulation, the avatars go to the designated evacuation area with some sense of urgency. As one participant put it "I don't want everyone to know that I got killed for the fourth time". The system has logging capabilities and enables replay of the simulation to see what happened. What was observed during the simulation was that even when evacuation paths were obstructed, participants found alternative

paths quickly. This may mean the participants were learning the escape paths as well as behaving as they normally would in a real emergency. The knowledge that the system will let everyone know who lived, died or were injured gave the participants some impetus to behave as if it was a real emergency.

Conclusion

This paper has described the development of an emergency evacuation simulation system. The system was custom built to give total control of all system parameters. The system permits the logging of all interactions when an emergency event is simulated. Although not described in this paper, the analysis of the logs gives insight as to how people might behave in an actual emergency event. The current version of the system has only basic customization for avatar's mobility although the underlying engine permits a more detailed matching of user and avatar's mobility. This should be investigated in the future. It should also be possible for customize the visual appearance of the avatars to include mobility depictions.

References

[1] CSIRO, New guidelines for predicting fire behaviour, Web site: http://www.csiro.au/news/ProjectVesta.html, Accessed: March 2, 2009

[2] A. News, Australians 'unprepared' for bushfires, Web site: http://www.abc.net.au/news/stories/2009/02/10/2487088.htm, Accessed: March 2, 2009

[3] Web site, 4th International Symposium Human Behaviour in Fire 2009, Web site: http://www.intercomm.dial.pipex.com/html/events/hb09a1.htm, Accessed: March 2, 2009

[4] B. Poucet, Spatial Cognitive Maps in Animals: New Hypotheses on Their Structure and Neural Mechanisms, *Psychological Review*, **100** (1993), 163-182.

[5] E.C. Tolman, Cognitive maps in rats and men, *Psychological Review*, **55** (1948), 189–208.

[6] M.H. Zyda, J.; Mayberry, A.; Wardynski, C.; Capps, M.; Osborn, B.; Shilling, R.; Robaszewski, M.; Davis, M., Entertainment R&D for defense, *IEEE Computer Graphics and Applications*, **23** (2003), 28-36.

[7] R.E. Wray, Synthetic adversaries for urban combat training in: *AI Magazine*, AAAI Press, 2004, pp. 82--92.

[8] R. Lerusalimschy, Celes, W., de Figueiredo, L. H., Lua Scripting Language, Web site: http://www.lua.org/about.html, Accessed: November 17, 2008

[9] Ambiera, irrKlang, Web site: http://www.ambiera.com/irrklang/index.html, Accessed: August 20, 2007

[10] J. Software, RakNet, Web site: http://www.jenkinssoftware.com/, Accessed: August 20, 2007

[11] J.J. Gibson, The theory of affordances, in: *Perceiving, Acting and Knowing*, R. Shaw, Bransford, J., ed, Erlbaum, Hillsdale, 1977.

[12] D. Norman, Affordance, Conventions and Design, Web site: http://jnd.org/dn.mss/affordance_conventions_and_design_part_2.html, Accessed: August 23, 2006

[13] K.J. Vicente, *Cognitive Work Analysis: Towards Safe, Productive, and Healthy Computer-Based Work*, L. Erlbaum Associates Inc., Hillsdale, 1999, 408 pp.

[14] D. Norman, Emotion and design: Attractive things work better, in: *Interactions Magazine*Vol. ix, 2002, pp. 36-42.

[15] P.E. Hart, Nilsson, N.J., Raphael, B., A Formal Basis for the Heuristic Determination of Minimum Cost Paths, *IEEE Transactions on Systems Science and Cybernetics*, **4** (1968), 100 - 107

[16] I. Takeuchi, Kakumoto, S., Goto, Y., Towards an Integrated Earthquake Disaster Simulation System, in: *Proceedings of the First International Workshop on Synthetic Simulation and Robotics to Mitigate Earthquake Disasters*, 2003.