LA-UR-04-2720

Approved for public release; distribution is unlimited.

Title: VIRTUAL INTERACTIVE SIMULATION AND INSPECTION TOOL (VISIT): MODELING SENSOR NETWORKS IN A VIRTUAL CITY

Author(s): David Moore

Submitted to: ISRM Conference
Warrenton Virginia
April 26-29, 2004

Virtual Interactive Simulation and Inspection Tool (VISIT): Modeling Sensor Networks in a Virtual City

David M. Moore
Los Alamos National Laboratory, Los Alamos, New Mexico 87545 USA

Abstract

The U.S. government is currently investigating the deployment of radiation sensor systems to protect cities against nuclear and radiological threats. Due to the high cost of installing such systems, there is a need to analyze the effectiveness of a variety of sensor configurations in detecting such threats before installing such systems in the field. The Virtual Interactive Simulation and Inspection Tool (VISIT) is a computer program developed for various virtual-reality applications in national security programs, and is presently being adapted to test the efficacy of a variety of sensor configurations in a virtual urban environment. The value of a particular sensor configuration will be assessed by running virtual exercises in which a threat team will choose a radiological device and route to a target and a detection team will specify the locations and types of sensors to be placed in the city to attempt detection of the threat prior to it reaching its target. This paper will discuss the VISIT package, its proposed application, and lessons learned from modeling done to date.

VISIT Concept

The Virtual Interactive Simulation and Inspection Tool was created in response to a request for a detailed physical model of a large, remote nuclear facility[1]. The model was requested because there was a need to demonstrate the many features of this facility to several interested parties, such as security analysts, inspectors, workers, and congressional sponsors who would be unable to visit the site in person. The organization requesting the model wanted to demonstrate as many features of the facility as possible, requiring the construction of many intricate dynamic parts. The creation of such a physical model would be challenging, and once it was constructed the size and fragility of the model itself would make it unsuitable for being moved across the country for various people to see it.

The novel solution to this problem came from a team member who was an avid fan of computer games. There has been a widely popular genre of video games available since the mid 1990s, termed "first-person shooters," in which people control their "players" to navigate their way through 3D interactive worlds, generally battling bad guys or evil space aliens. He realized that competition in this popular entertainment

[1] Seitz, Sharon L., et al. Virtual Interactive Simulation and Inspection Tool (VISIT). In: *Proceedings of the 43rd Annual Meeting of the Institute of Nuclear Materials Management* (Orlando, Florida) LA-UR-02-3205, June 2002.

medium had pushed the limits of real-time 3D rendering to amazing levels and this technology could be adapted to allow people to walk through representations of real facilities and observe and interact with complex computer representations of real processes.

The use of commercial video game engines[2] provided several advantages over a traditional table-top model. For one, the virtual model is easily distributable by CD-ROM or FTP, whereas a table-top model would either be unmovable or expensive to ship at best. The computer model also gives the user a realistic first-person viewpoint as if they were standing at the actual site. This can be especially valuable when doing security assessments to analyze lines of sight from guard stations or defensive barriers.

A high level of interactivity can be programmed into the games: features such as doors, cranes, elevators, security alarms, and light switches can all be included in the virtual environment. Additionally, an extremely high level of visual fidelity can be achieved in the 3D model when desired. In one of our deliverables we were able to "paint" the walls of the facility using digital photographs we took on site, so that when walking in the virtual facility the user was able to see details such as construction markings written on the wall with a black permanent marker.

Unreal® Engine

Before we explain how we are adapting this technology to radiation sensor network modeling, we will explain a few details about how the engine works. The core of the VISIT package is the Unreal® Engine. Developed by Epic Games, it has been used in over 40 games to date[3], and is named after the series of games for which it was originally created, Unreal® Tournament. The genre of games for which it is best suited and most often used is "first-person shooters." The demands of the genre made three factors critical in the design of the game engine: fast rendering of 3D environments, efficient networking for fast multiplayer gaming, and a scripting environment suitable for the easy development of game modifications designed with game programming in mind. The details of how the rendering system works are not pertinent to this paper; it will be sufficient to note that there are a variety of methods available to optimize the rendering of 3D environments such that the engine can provide an immersive environment in which the player can move around in real time.

The efficient networking model was designed with the average user's unreliable and limited-bandwidth Internet connection in mind. The networking scheme is a modification of the basic client-server model; the server is the keeper of the authoritative game state and the clients both render the view of the world for each user and receive the human input into the game. The amount of network traffic is minimized by the server

---

[2] In this context by engine we mean software that will render a 3D representation of a physical environment in real time as a user moves around and also provides the capability for interaction between human-controlled entities and software controlled entities in the virtual environment.

[3] http://udn.epicgames.com/Powered/WebHome and http://udn.epicgames.com/Powered/FirstGenerationTitles

only sending information to a given client when it judges that data to be "relevant" to that particular client. Also clients are allowed to do some computation and rendering of effects not done on the server when that process or visual effect is only relevant to that client.

For example, say our 3D virtual world is a model of a typical person's home or apartment. We have one user Mike who is playing a networked game and controlling a "player," a virtual person, in the house, and another user Melissa who is controlling another player. Say Mike is in the bedroom in our virtual house with the door shut, while Melissa is in the kitchen and drops a plate. Melissa's client application will inform the server that a plate has been dropped by her player and sends the location of the plate to the server. The server will process the physics calculations of a plate falling, and will send the data back to Melissa's client to display the results on her computer.

Because Mike's player is in another room, the server does not send that information to his client application because it is not relevant for his player. However, if Mike leaves the bedroom and walks to the kitchen, when the kitchen enters his view the server will send him information on the location of the shattered pieces of the plate on the floor so his client can render them on his computer. The key point to understand is that the server is notified of actions taken by the client that modify the game state, and in turn sends information that is relevant back to each client so they can render a view of the virtual world for each person playing the game.

When our team decided to apply 3D gaming technology to the scientific problem domain we found there were several different 3D interactive game engines available from different commercial vendors. All of them provided amazing real-time rendering capabilities, but the factor which set the Unreal® Engine apart was the ability to make programming modifications to the engine to add new kinds of interactions never envisioned in the world of computer games. This capability is provided via UnrealScript, a compiled, object-oriented programming language that bears some similarity to Java, but that provides language constructs and keywords that make it easy to program in a networked, event and state-based environment. The details of this programming language are not as important to this paper as the fact that this language has enabled us to implement novel features such as database connectivity, interaction of real-world sensors with the virtual world (e.g., a magnetic door switch in our office triggered a door in a virtual world to open), and allowed us to model radiation transport and detection in this current application.

Model Sensor Network Application

With the increasingly viable threat of nuclear terrorism facing America, the U.S. government must consider options for defending its cities against such attacks. A network of radiation sensors is one such method of defense in consideration. We are currently investigating how we might apply the VISIT software to this problem domain.

There is a continuum of approaches one could take to evaluating the potential of

sensor networks to counter radiological threats. At one end of the spectrum are large-scale tests in the real world, such as the $16 million TopOff 2 terrorism drills involving over 8000 people that took place in Seattle and Chicago in May 2003. At the other end of the spectrum we could consider sophisticated mathematical simulations of radiation sources, detectors, and dispersion through complex materials.

The former case, the real-world exercises, may provide valuable insights into the response of large bureaucratic organizations to terrorist acts, but such drills are expensive, hard to coordinate, and limited by safety and cost concerns (e.g., real radiological sources may not be used in the exercises). The later case, intense computer simulations, can provide hard scientific data but these computational models are also very difficult to create, the simulations themselves could take anywhere from days to months to run, and generally the only human input into the simulation is in setting up the input parameters before a simulation is run. The VISIT approach is at the middle of the spectrum, using a simplistic model of radiation sources, detectors, and attenuation, while allowing a limited number of human participants to interactively participate in the simulation while it is being carried out.

VISIT Project Design

The goal of the VISIT Model Sensor Networks project is to create an interactive environment for testing a variety of sensor configurations and threat scenarios. The value of a particular sensor configuration will be assessed by running virtual exercises in which a threat team will choose a radiological device and route to a target and a detection team will specify the locations and types of sensors to be placed in the city to attempt detection of the threat prior to it reaching its target. This will be accomplished with the input of three teams of experts, termed the Red, White, and Blue teams, to contribute to the design of the software and running of the interactive simulations.

The primary job of the White team is to specify the characteristics of the urban environment in which the simulation is done, specify the target of interest, and specify the limits of interaction of the Red and Blue teams. The Red team is to define the radiological threat, including any shielding parameters, and will try to deliver the threat device to the target. The Blue team will set up a network of sensors in and around the virtual city, and can also respond to the sensor readings and attempt to intercept the threat. In other words, the Red, White, and Blue teams respectively correspond to the bad guys, the referees, and the good guys.

Instead of working like a traditional simulation in which the input parameters are defined and a process is started which calculates output values, with the VISIT software a virtual urban environment will be created in advance, the Blue team will place detectors in and around the city, and then a team of humans will interactively control the virtual terrorists who will try to deliver a device to the target. A group of first responders will actively monitor the sensor readings and will be allowed to try to respond to the threat and intercept it. The focus of the simulations will be to research the strategies the human-controlled actors use in the simulation to either deliver the threat or intercept it.

We utilize a very basic form of radiation modeling for this project, for reasons which will be explained below under the section "VISIT Limitations." Radiation transport is calculated as a straight line from the source to detector with attenuation calculated for all objects which intersect that line. Since we anticipate having many more detectors in the city than sources (perhaps one threat source and a few non-threat sources such as medical isotopes versus hundreds of detectors) and detectors will only be able to detect sources that are relatively close to them, each source will actually "look" for detectors within a certain radius, notify that detector of its presence, and the detector will run the necessary calculations to determine what radiation reading it would see, if anything. This will make our simulation much more efficient than having each detector search for sources at each unit of time in the game, which is very important as will be discussed in the section "VISIT Limitations" below.

VISIT Advantages

Using software such as the Unreal® Engine, designed for play by gamers across the world, may seem ill-suited for application to something as intricate as radiation modeling and detection. There are limitations of working within the gaming genre, but there are also some unique benefits that are granted by using this software. By recognizing the inherent strengths and weaknesses of the software, we can tailor our approach to the problem to make the best use of what we have and avoid some pitfalls.

The strengths of the software are the fast 3D rendering, the efficient networking model, and the support for interactivity. So rather than trying to create a simulation which once started is largely autonomous, we are designing a program in which the users will be active participants controlling the progress of the simulation. Human ingenuity will be the main factor which will make our approach to the problem domain unique.

The greatest strength of the Unreal® Engine is the 3D rendering software, as creating immersive 3D environments is a primary goal of the commercial games built using the engine. What are a few ways in which we can use the 3D perspective to our advantage? First, the sensors in our simulation will be represented visually in addition to their computational representation. So if the sensors being deployed are above ground and stand out visually, the person playing the role of the terrorist may spot the detector and change his route of delivery to avoid passing the detector.

By building a realistic and complex urban landscape, the deliverer of the threat device has a wide range of options available to try to circumvent detection. The driver may attempt to cross through parking lots, cut corners through gas stations, or drive through alleys to stay off main roads which they may fear are more subject to detector surveillance. While in-depth study of detector design and efficacy is absolutely critical and the subject of other research, the VISIT software will allow for study of the possibilities for avoidance of detectors which would render even the best detectors useless.

The multiplayer capacity of the engine adds further opportunities for novel strategies of both delivery and interception of the threat. The Red Team can attempt to elude detection by dividing the radiological material between several vehicles and approach the target from several directions at once. Assistants in other cars could help shield the device from detectors stationed at street level. The Blue Team could use multiplayer capabilities to have several participants driving around the city in vehicles with mobile detectors prepared to drive the scene where there have been potential detections. First responders could also evaluate possible responses to detection by setting up roadblocks at key points when detectors signal alarms.

Finally, the scripting language provided by the Unreal® Engine provides the capability to modify the simulation rules and properties. This gives both the software developers and even the end-users the ability to customize the game play and environment. For example, we can easily change properties of objects in the simulation, such as the maximum speed of a certain type of car, without having to recompile the source code. Other changes, such as adding a new type of weapon or car, could be accomplished with only a partial recompilation of source code.

VISIT Limitations

The graphically-focused real-time interactivity of the engine comes at a cost. To understand part of the problem it helps to know a little bit more about how the engine works. The most important concept to understand is that of a "tick."

The tick is the smallest discrete unit of time over which the state of the virtual world changes. At each tick, every object, be it a car, human-controlled player, script-controlled player, or radiation detector, in the virtual environment is given a chance to update its state via a function in its class called "tick." When all the objects are finished updating their state, the view of the world is rendered onto each client's computer monitor. Because the length of time it will take for each object to update its state may vary, the length of a tick is variable. The engine accounts for this by informing each object of the length of time since the last tick occurred, the "delta time," via a parameter in the "tick" function. For example, when a moving car is "ticked," it multiplies its velocity by the delta time and adds this to its current position to get its new position for this moment in time, which is then rendered on each client's computer which has the car in its field of view.

For each client, at the end of each tick their screen is updated to show the current state of the world. Therefore, the number of ticks per second is equal to the number of screen updates per second, called the "frame rate." Like watching TV or a movie, a frame rate of about 30 images per second is sufficient to give the appearance of smooth, continuous motion. When the frame rate falls below 20 frames per second not only does the rendering of movement get more choppy, but user input gets processed less frequently, so it becomes harder to control your character or vehicle. If the frame rate goes below 10 frames a second, it becomes very hard to control a character because the keyboard input is only processed a couple times per second. The frame rate is inversely

proportional to the time it takes to update the state of each object between the ticks, so optimizing the code called by the tick function of each actor is critical to the production of a run-able interactive simulation.

The need to optimize this tick function excludes the possibility of doing computationally expensive operations when calculating the radiation dispersion and attenuation so modeling methods such as Monte Carlo are out of the question. Instead, our radiation model is based on calculating the attenuation of the radiation strength along a straight line from the source to the detector. While this is admittedly a very simplistic model of radiation transport, the focus of the work is not so much to do a rigorous assessment of the ability of individual detectors to detect different threats, but to run many scenarios to investigate how terrorists may attempt to elude detection and what could be done to counter the threat.

While the Unreal® Engine is designed for multiple client interaction, there is a limit of about 32 simultaneous clients before the network communication and amount of processing that must be done by the server to determine what is relevant for each client makes the game run too slow to be playable. Therefore we must recognize this limitation and only consider scenarios with a relatively small group of interactive clients. For the problem domain, this should not present a problem: terrorists tend to work in small groups and we will only consider the response of a small interdiction team that might be the first ones to respond to a potential attack.

Finally, while the engine has a very high fidelity physics model which gives us the capability to make detailed models of the physics of cars, it is also very computationally expensive, and thus it is impossible to populate a city with a realistic number of cars using the special car physics model. Instead we are working on some alternative methods of simulating the effects of traffic, such as using other tools like TRANSIMS[4] to simulate general traffic patterns and then import that data into our game in the form of speed limits for certain roads. In addition, we may make simple graphical representations of cars so there is still a visual element that the user will see, but which move along pre-scripted paths instead of using complicated and computationally expensive artificial intelligence and realistic physics calculations.

Conclusion

We hope to demonstrate the value and feasibility of using interactive 3D simulations in studying ways to counter nuclear and radiological threats. Our approach has a valuable position among the methods of testing sensors systems when a live exercise is unsafe or too expensive and when detailed simulations are overkill and too complex to produce results in a short time frame. The VISIT program can provide the capability to quickly evaluate possible terrorist methods of radiological attack and potential ways of countering such threats.

---

[4] TRANSIMS is a regional transportation system simulator developed by the Los Alamos National Laboratory (http://transims.tsasa.lanl.gov/).