Game-Based Simulation for the Evaluation of Threat Detection in a Seaport Environment

Allen Christiansen, Damian Johnson, and Lawrence Holder

School of Electrical Engineering and Computer Science
Washington State University, Pullman, WA 99164
allen.christiansen@gmail.com, atagar1@gmail.com, holder@wsu.edu

Abstract. The ability to simulate a seaport environment, including illicit cargo and the sensors designed to detect such cargo, allows the evaluation of alternative detection methods in order to improve security at our nation's seaports. We describe our progress towards this goal. Specifically, we describe our modeling of threats at a particle emission level, modeling of sensors as particle detectors, modeling of the seaport dynamics (e.g, ships, cargo containers, cranes, trucks), and how the particles interact with the various structures and materials in the seaport environment as the cargo moves through the seaport. Ultimately, this simulation will serve as a testbed for the evaluation of sensor network data collection, fusion and decision making for threat detection in a seaport environment.

Keywords: Simulation, seaport, particle propagation, sensor modeling, security, threat detection.

1 Introduction

Advanced approaches to threat detection for homeland security will involve much more non-human intervention via networked sensors and autonomous platforms. Modeling and simulation of these integrated components is necessary for making decisions about deployment and evaluating the intelligence derived from their surveillance. Individual models and simulations of these components exist, but the modeling and simulation of the network of components, considering the physics of the sensors, the physics of the threats to be sensed, and the low-level interaction of the stimulus with the environment, has yet to be addressed.

For example, suppose we suspect a nuclear weapon of mass destruction (WMD) is hidden in cargo container destined for a U.S. seaport. What sensors should be deployed, and where, based on the likely signature of the threat (e.g., gamma emissions), the available sensors (e.g., gamma detectors) and sensor locations (e.g., indoor vs. outdoor), and details of the environment (e.g., intervening metal, stone, wood, soil or water)? Modeling the components of this scenario and simulating their interactions will help determine the proper course of action to maximize the ability to detect and locate the threat. Related approaches to simulating the seaport environment exist [5,7], but they focus on a one-shot computation of detection rates and wait times based on higher-level mathematical relationships among the relevant factors, and they

S.M. Stevens and S. Saldamarco (Eds.) ICEC 2008, LNCS 5309, pp. 221–224, 2008.

do not realistically visualize the real-time 3D aspects of the environment, which can be instructive in understanding the evaluation of detection methods.

The purpose of this simulation is to accurately model the information flow from the environment to enable the development and evaluation of large-scale, information fusion and data mining techniques to support decision making. Here we describe the modeling and simulation of the threats, sensors and seaport equipment in order to support the goal of evaluation and the automation of the decision-making process.

2 Threat and Sensor Modeling

We are developing a model of threat devices at the particle-emission level using particle tracking within the Torque Game Engine [8] and the Ageia PhysX processor [1] to track particles emitted from a point in all directions (as depicted in Fig. 1). We track particles and model their interactions with various materials in the environment at the particle level (e.g., refractions and absorptions, as well as reflections). Sensors are also modeled at a low-level, and the collision of particles with sensors simulates the level of activation of the sensor. These sensors readings (simulated or real) contribute to the flow of information from the seaport environment.

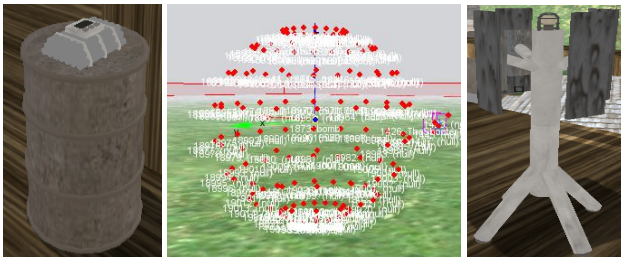

Fig. 1. Threats (left) are simulated as omni-directional particle emitters (middle), and sensors are simulated as particle detectors (right)

The main challenge with modeling these types of threats and sensors is for the particles in the environment to act in a realistic way, despite the computational challenges of tracking vast numbers of particles at relativistic speeds. One approach is to use Monte Carlo simulation tools (e.g., MCNP [3]) which exist to accurately handle radiation particle interactions with given materials. Using data generated by such tools will allow us to better represent realistic behavior.

We are using several tactics to address the computational challenges. First, it is unrealistic to model millions of particles in the environment, so we are using a more modest number of larger objects to represent a group of particles. Rather than trying to keep track of lots of little particles traveling and colliding all over our scene we are instead launching much larger (anywhere from 20 to 100 times the original size) objects. A similar approach has been taken to real-time sound propagation in games [6]. This has the benefit of drastically decreasing the computational load, as well as allowing us to simulate a much larger number of particles active in our environment. Second we are trying several “delivery systems” (using built-in particle emitters,

projectiles, third party physics engines, etc.) to determine which will be the most system-resource efficient. One of these methods is the Torque Game Engine's (TGE) [8] built-in particle emitter. This required changes to the engine code to allow custom collisions between the particles and the objects in the environment (e.g., reflection, refraction, and absorption depending on particle energy and object material). However, not having enough control over how the particles were being emitted (one at a time vs. waves like we wanted) caused us to look into a different direction.

The second approach we looked at was just launching the particles with the engine's projectile system. This allowed us to control exactly how many objects we were launching and how they were launched. So we used this to launch our particles in a sphere from the desired launch point and set the velocity as high as we could while still being able to collide with everything we desired. In an attempt to decrease some of the computations required of the main processor, we looked at importing a 3rd party physics engine, Aegia's PhysX engine [1], into the TGE. Eventually we were able to have the PhysX card handle all of the collisions, and just update the TGE of their locations. Since we had a separate processor handing collisions, this allowed us to have many more objects active in the environment at once and use the same functionality as in TGE's projectile system, easily creating and launching particles in any direction. Custom collisions were handled using call backs that allow us to define pairs of objects or groups of objects as collision pairs, which prompts the PhysX engine to make these function calls when objects of pairs collide. This allowed us to perform custom collisions between the objects that we desired, but having these callbacks fired for all of the primary objects in our environment might be too much of an overhead to handle. We are still working to clear up some of these issues.

3 Seaport Modeling

For our simulation we have decided to focus on operations performed by ships, cranes, yard hustlers (trucks), pickers, and trains in their movement of cargo. Cargo and modes of transportation were modeled in the Blender open-source 3D modeling tool [2]. All models were based after photos of seaport machinery (mostly high resolution images from seaports and the Navy). This has allowed us to determine some semblance of scale and proportions between the models. Renderings, models, and sources for the various finished components can be found via the project's web site at http://www.eecs.wsu.edu/sgl (the WSU Simulation and Gaming Lab). Fig. 2 illustrates these models in Blender and as viewed in the Torque Game Engine.

After models are finished their basic operations within the game engine are implemented via Torque Script, a proprietary scripting language that takes advantage of hooks within the engine, such as collision events, timed triggers, and basic physical properties (such as mass and elasticity). Implemented functionality includes methods for lifting, transferring, and carrying cargo for all modes of transportation.

The real time aspect is not strictly necessary for experimental results. However, including it provides a blend of simulation and data visualization that opens additional uses of interest to a wider audience of users. For instance an accurate real time simulation could possibly be useful to seaport workers in determining security holes introduced after sensors are disabled in an accidental collision or during adverse weather conditions.

Another project with similar aims is the L3DGEWorld data visualization tool for network analysis [4]. This project utilizes OpenArena (a derivative of the Quake III Arena game engine) to reflect network activity, and players actions as network commands (such as blocking IP ranges). While the environments differ considerably, both have a similar goal of allowing users to sift through sizable amounts of real-world data via a player. One lesson we take from this project is to abandon realism at times in favor of usability (such as providing a HUD or aerial view upon request).

Fig. 2. Models of the ship, cargo containers and crane in Blender [2] (top three). Seaport models viewed in the Torque Game Engine [8] (bottom two).

References

1. Ageia Physx Processor, http://www.ageia.com/physx
2. Blender, http://www.blender.org
3. Brown, F., Barnett, N.: A Tutorial on Using MCNP for 1-Group Transport Calculations, Los Alamos National Laboratory (July 2007)
4. Harrop, W., Armitage, G.: Real-Time Collaborative Network Monitoring and Control Using 3D Game Engines for Representation and Interaction. In: VizSEC 2006 Workshop on Visualization for Computer Security, Virginia, USA (October/November 2006)
5. Koch, D.: PortSim: A Port Security Simulation and Visualization Tool. In: Proc. of the 41st Annual IEEE Intern. Conf. on Security Technology (October 2007)
6. Raghuvanshi, N., Lauterbach, C., Chandak, A., Manocha, D., Lin, M.: Real-Time Sound Synthesis and Propagation for Games. Communications of the ACM 50(7) (2007)
7. Sekine, J., Campos-Nannez, E., Harrald, J., Abeldo, H.: A Simulation-based Approach to Trade-off Analysis of Port Security. In: Proc. of the 38th Conf. on Simulation (Winter, 2006)
8. Torque Game Engine, http://www.garagegames.com