Mogadishu Terrain Generation and Correlation for Crowd Modeling

Frederic D. McKenzie PhD
Hector M. Garcia
Quynh-Ahn H. Nguyen
Jen Seevinck
Mikel D. Petty
Virginia Modeling, Analysis and Simulation Center
Old Dominion University
Norfolk, VA. 23529
757.683.5590, 757.683.6367
fmckenzi@ece.odu.edu, hgarcia@odu.edu

Keywords:
Synthetic environment, visualization, game technology, terrain generation, correlation.

ABSTRACT: *A requisite component of crowd models is an environment in which to test, validate, and, in general, execute those models. The environment should necessarily provide a visualization of the behaviors produced by those models. We are developing a crowd federate capable of generating a wide variety of civilian individual and group behaviors at differing levels of fidelity. One question of interest is what effects do behaviors of differing fidelity have upon the outcomes of military missions in which civilian crowds are a factor. To answer this question, we have recreated to some extent a portion of the Blackhawk Down incident in Mogadishu, Somalia using JSAF and a prototype of our crowd federate. In the prototype of the crowd federate used, game technology was utilized to represent the urban terrain and accompanying moving models. As one might imagine, terrain generation and correlation of needed geographical regions was not a simple task. The objective of this paper is to discuss the process involved with the development and correlation of terrain databases used in military simulation with those employed in game simulations. In addition, the correlation of the moving models involved will also be discussed. This paper will also describe and illustrate the developed Mogadishu scenario. As a final point of discussion, the appropriateness and lessons learned in using these simulation models in the recreation of a historic event will be addressed.*

1. INTRODUCTION

Crowds of non-combatants play a large and increasingly recognized role in modern military operations, and often create substantial difficulties for the combatant forces involved. U. S. military actions in Mogadishu, Bosnia, Afghanistan, and Iraq exemplify the significant effects crowds may have on military operations. However, in spite of their potential significance, realistic models of crowds are essentially absent from current military simulations. For the scenarios considered likely in future conflicts the absence of crowds and of non-combatants in general would be a serious departure from realism.

VMASC is engaged in a two-phase research project aimed at developing a crowd modeling capability for military simulation. The first phase, now complete, consisted of three parts: a requirements analysis to identify military simulation crowd modeling requirements, a literature survey to examine psychological research relevant to crowd modeling, and a design study to explore design issues in the implementation of a crowd simulation [1]. The findings of the first phase included two that served to focus development efforts in the second phase. First, we found that the greatest requirements for a crowd simulation capability were in real-time tactical training applications. Second, we realized that there is an important distinction between crowds (hundreds to thousands of people) and populations (tens of thousand to millions of people) in terms of size, behaviors, duration, extent, and effects on military operations.

In the second phase, now well underway, we are developing a crowd simulation, implemented as a distributed simulation federate, that will be interoperable with existing military simulations and will have a credible psychological basis for the crowd behavior it generates. The second phase of the project has seven interrelated tasks. They are:

1. *Crowd federate implementation*; design and development of a simulation that generates and controls crowd members, is interoperable with existing military simulations via HLA, and has a reconfigurable architecture to allow later replacement of its component models.
2. *Cognitive model development*; acquisition of psychological information describing the behavior of crowds via both literature review and direct psychological research, the development of a computational model of crowd member behavior based on the psychological information, and the integration of that model into the crowd federate.
3. *Requirements analysis continuation*; continuation of the process of identifying requirements for crowd modeling in military simulation.
4. *Historical survey*; study and analysis of historical incidents where crowds had a significant effect on the course or outcome of military engagements.
5. *Reference scenarios*; development of documented, historically accurate scenarios in a military simulation of historical events involving crowds, for testing and validation of the crowd federate.
6. *Experiments*; conduct of two experiments planned to test the crowd federate, the first to examine the level of crowd behavior fidelity needed, and the second to test the architectural reconfigurability of the crowd federate.
7. *PMFserv evaluation*; independent evaluation of a psychological model based on performance moderator functions.

2. BACKGROUND

A requisite component of crowd models is an environment in which to test, validate, and, in general, execute those models. In order to accomplish this, a decision was made to use 3D studio Max, a common 3D modeling package for the gaming industry, as the primary tool for generating the synthetic environment database due to it's flexibility and powerful 3D modeling engine as well as to meet the requirements of producing multiple correlated data formats that were used in the Crowd Modeling project. These two formats were a CTDB file for JSAF and a Maya format file for the 3D stealth view, which was powered by a PC game engine called Renderware. After the environment was generated, we proceeded to convert the database into the several formats required. The tools used for such conversions are: Terravista with the DART plug-in tool, which allowed us to use the database with JSAF, and Multigen-Paradigm Creator, for generating the model files required by Terravista.. The process employed to generate this environment is described in further detail below.

3. GENERATING THE ENVIRONMENT

3.1 Generating the Mogadishu Terrain.

The first step we took in generating the environment was to locate an appropriate satellite picture of the area (Figure 1).

Figure 1: Mogadishu Satellite Photo (Credit: Space Imaging)

We then proceeded to import the image map into 3D studio MAX using the provided scale information on the image. A terrain block was created to scale, which covered the area shown in the picture, and textured with the satellite picture.

These steps provided a background for generating the 3D buildings used in the simulation. In parallel we used another satellite picture with a broader overview to correlate with a CTDB database which included the area of interest (Figure 2). The original CTDB database was used to geo reference all of the work that was done in 3D studio MAX. Terravista with the DART option was used to convert the CTDB into an Open Flight file, which then was imported into 3D studio MAX.

Once the terrains where imported into 3D studio MAX, the next step was to construct the buildings around the areas of interest.

3.2 Constructing the Buildings in Mogadishu

For the construction of the buildings in Mogadishu, we did not have a lot of source data, so we decided to take the data available from the satellite pictures, as a guide for tracing the footprints of the buildings, and then creating an 'extrusion' of such footprint to create the buildings. We also used Nova logic's Black Hawk Down game as reference material from which to create the buildings from.

Since each of the buildings was created from the satellite picture, they were already referenced to the coordinate system of the dataset. All of the buildings were given a 'generic' texture, and the 'target' building was given a more detailed look than the rest of the buildings (figure 3).

Once the buildings were created, then the task of preparing them to be translated into a CTDB file for use into JSAF began. We were able to use 3D studio MAX to quickly identify faces, which would provide the attributes 'footprint' and 'roofline' for the CTDB file. After the faces where identified and duplicated as part of each model, then a conversion was made within 3D Studio MAX to translate the files from the MAX file format to Maya format, for use in the stealth 3D viewer, and to the FLT file format for further work on attribute values necessary to reuse the dataset in a CTDB format.

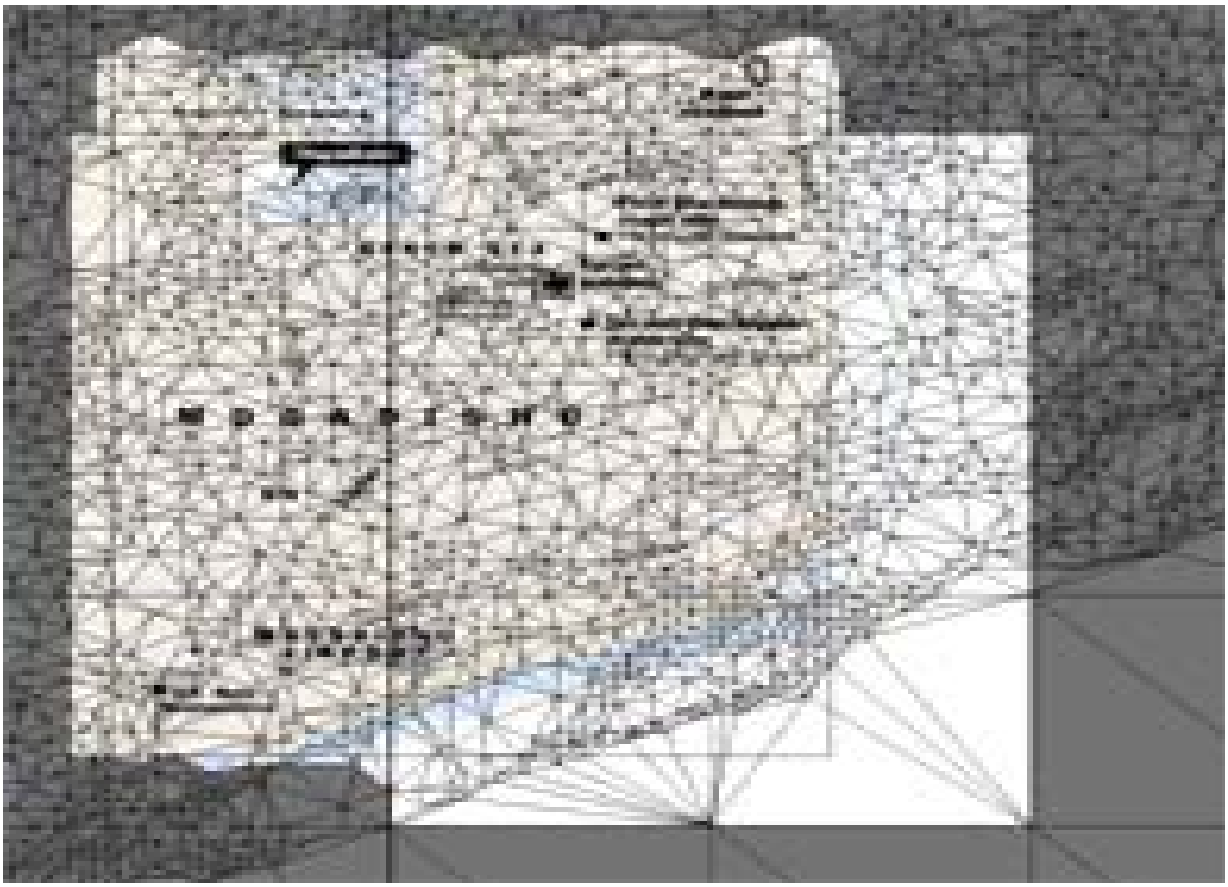

Figure 2: Correlated terrain with map and satellite picture

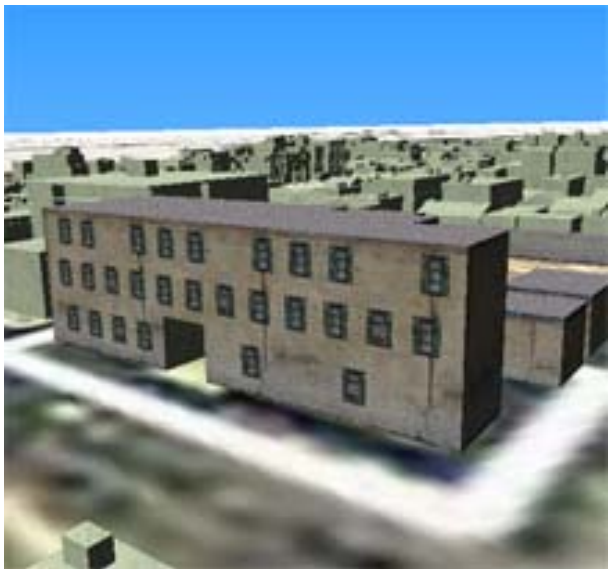

Figure 3:
3D view of 'extruded' target building

3.3 Preparing the MAX environment for CTDB compatibility.

After the translation of the Mogadishu 3D Studio MAX file into the Open Fight format, we continued to define attribute parameters of the database inside Multigen-Paradigm's Creator.

Creator was used to give the proper flag attributes to each of the faces previously identified in 3D Studio MAX as a 'roofline' and a 'footprint'. One of the reasons why the extra step was needed to define these attributes in Creator, is because 3D Studio MAX was not originally intended to model 3D environments for real-time simulations, although for the past 3 years this tool has been evolving to be the tool of choice for real-time game design.

Once these attributes were defined, and then the database was saved in the Open Flight format it was then taken into Terravista to prepare it for final conversion to a CTDB file.

3.4 Using Terravista and DART for generation of CTDB

We used a plug-in module in Terravista called DART (Database Automatic Re-Use Technology) to be able to reuse the original CTDB file and then add the new environment information created with 3D Studio MAX.

This procedure involved importing the Open Flight files of all the buildings into Terravista and adding them as culture data.

In order to do this, we imported an Open Flight vector file, originally created in 3D Studio MAX, by using the DART Terrain Converter importer. Once the file was imported we used the editor tool in Terravista to assign the buildings we previously imported to the vector file (Figure 4).

It is of importance to mention that within the scope of this project and the strict time schedule we only developed building structures that were not accessible on the inside, therefore we made no use of the MES (multi elevation structure) capabilities, but this is not to say that we could not easily expand on our existing model to add such features.

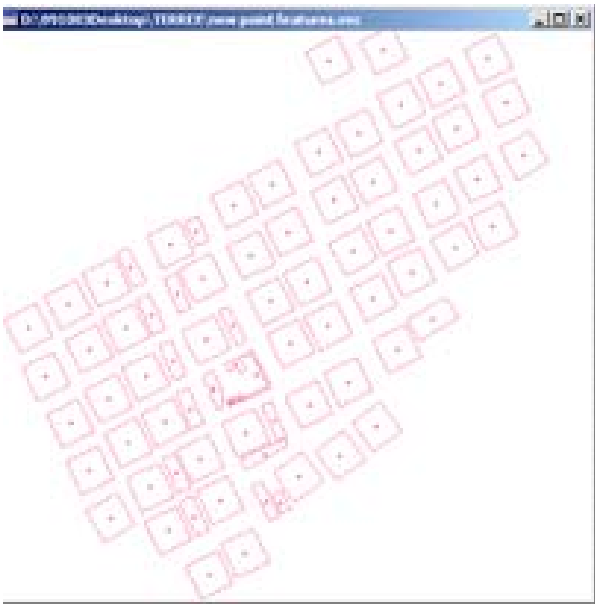

Figure 4:
Culture vector file with building footprints assigned

Finally the 'gaming area' was defined from the terrain tiles making sure that the gaming area followed CTDB rules for area selection, which in this case was to make sure an 8x8 tile area was used.

We then proceeded to build the 'gaming area' with the SAF_USA option enabled, which produces a CTDB file of the built project from Terravista.

The resulting CTDB file was used in JSAF as the terrain database on which to run the crowd federate.

4. CHARACTER MODELING, RIGGING, AND ANIMATION

The creation of the civilian and soldier models is constrained by the real time performance requirements of

the end application. Since many figures are being rendered at any one time (a crowd), these must be optimized for maximum realism and performance.

Reducing the resolution of textures and minimizing the number of polygons in a figure reduces the computational load on both the rendering engine and the crowd behavioral model.

A lower polygon count is synonymous to a lower resolution and the end application (e.g. the distance of the virtual eye to the crowd) must be considered when determining thresholds, else the realism of the end product may be compromised. A range of 500 - 800 polygons per figure was determined as optimum for our application.

Each figure model is rigged with an inverse kinematic skeleton. Once this hierarchical system of bones is built to fit the mesh, it is then bound (attached) to the mesh model of the figure and finally configured to enable animation:

1. Skeleton joints are appropriately constrained for animation purposes (eg knees don't bend backwards.)
2. The relative weighting of bones and their deformative effect on the bound mesh is adjusted for smooth deformations rather than folds in the mesh.
3. Pole vectors and animation handles are defined.

The project required animations of two types: cycling and blending (for between cycles). Cycles such as walking were keyframed in maya then rendered to clips that the AI-Implant behavioral engine can call. When a change in behavior was required, a blend animation clip would be called (eg walk to stand) followed by the new animation clip (stand cycle).

Keyframing is, however, a fairly laborious and time consuming process (10 seconds of animation typically takes an animator 40 hours to create). Therefore motion capture data is now being utilized where this is affordable and the movements are available (e.g. for standard/common motions, from the Kaydara library). Kaydara produces a tool called MotionBuilder that automates many aspects of character rigging and animation.

The motions are edited into cycles in Kaydara. In order to utilize the character rigging feature and since the export of animation cycles from Kaydara to Maya (.fbx file format) creates keyframes at every frame, all animation must now be done in Kaydara, whether motion capture or keyframed.

Previously all the animation was done in Maya. Modeling and the generation of animation clips that our crowd behaviors can call are still done within Maya. Texturing uses reference images from the web and is done in Adobe Photoshop image manipulation software.

5. INTEGRATING JSAF, RenderWare, & AI.implant

AI.implant™, developed by BioGraphic Technologies, is a game AI solution from the entertainment industry that offers a "real-time interactive artificial intelligence animation solution…to create incredibly rich character interactions." The AI.implant™ world is made up of stationary objects such as barriers, surfaces, terrains, and paths, as well as two types of dynamic objects, the autonomous and non-autonomous characters, which interact with this world. In the simplest terms, the behaviors of an autonomous character is controlled by AI.implant™, while a non-autonomous character is a character that is not controlled by AI.implant™ but may interact within the world. The behaviors provide the rules that determine how that character will interact with other objects within the world and generates the steering forces that change the character's position, orientation, and/or speed.

In our prototype, we were able to demonstrate two-way communication between our crowd federate and JSAF. We were able to create crowd members in our federate, publish it to the RTI, and have those characters be recognized and shown in the JSAF Panel View Display (PVD). Likewise JSAF entities were generated within the crowd federate.

AI.implant™ world is made up of stationary objects such as barriers, surfaces, terrains, and paths, as well as two types of dynamic objects, the autonomous and non-autonomous characters. The non-autonomous character could be used to model objects such as player-controlled characters, falling rocks, or any other dynamic object that does not have its behavior controlled by AI.implant™.

One of our first goals was to come up with an architecture that would allow us to integrate the HLA entities into our simulation world. Normally, we could have used AI.implant's non-autonomous player-controlled character to model our JSAF entities in our simulated world. However, our desire to also incorporate a 3D viewer made this choice impractical. Although we didn't want AI.implant to control any of the movement and behavior associated with the JSAF entities, we did need it to control the animation of the characters for the viewer. In order to accommodate this requirement, we had to modify

the implementation of the player-controlled character. Instead of using the non-autonomous character, we had to use an autonomous character that had no behavior associated with it. We developed a decision tree that was devoid of all behaviors, and had only commands that activated the appropriate animation clips for our 3D viewer.

As mentioned earlier, a 3D viewer was also incorporated into our crowd federate to allow us to visualize the entities in both federates. The 3D HLA viewer was developed using the RenderWare game engine by Criterion Software.

When we integrated the 3D viewer with HLA, we noticed some peculiar behavior exhibited by JSAF entities. We found that when we gave some JSAF entities the order to move in a direction opposite to the way they were facing, they would not change their orientation before moving in that direction. In fact, the soldiers that were used for our scenario were marching backwards! This is not so obvious when you view those same entities in the JSAF PVD, since it is a 2D display, but it became obvious in the 3D viewer.

6. SAF SCENARIO DEVELOPMENT

Developing the Black Hawk scenario in JSAF proved to be a more complicated enterprise than expected. The first step, was to examine the movie Black Hawk Down and the History Channel documentary on the Mogadishu incident along with various other sources. The goal was to determine the timeline and to develop the execution matrix needed for the military component of the scenario.

We were using the DVTE 1.0 version of JSAF, which was based off of the JNTC branch of the JSAF development tree. The types of entities that we needed were not V&V'd by the DVTE group, and were not part of the DVTE operation list. Therefore, we found that many of the behaviors that we needed from those entities did not function correctly. Working within these limitations, we had to make comprises and use workarounds to model the scenario.

An example of a workaround was in the modeling of the convoy of humvees and trucks staged behind the Olympia Hotel, waiting to pick up the prisoners and soldiers. Since there were no pre-defined aggregate convoy of humvees and trucks, we had to use individual humvee and truck entities, placing each one behind the other. We wanted these individual entities to act as a single convoy, so we wanted to utilize the "Follow leader" task order to have each vehicle follow the vehicle in front of it. However, we ran into some difficulties trying to get this to occur consistently. For one, the convoy had to make a left-hand turn to get onto Hawl Wadaag Rd, which is the road that the target building was located. The vehicles sometimes had problems navigating this turn, and would occasionally run into the buildings and get stuck. This would cause the vehicles behind them to become immobile, waiting for their lead vehicle to move. To overcome this problem, we had to stage the convoy as if they were already on Hawl Wadaag Rd., each with its own specified straight path route to follow. This workaround was needed to keep the vehicles from having to navigate through the narrow streets of Mogadishu and possibly becoming obstructed by any buildings.

We also had to make sure that we authorized each vehicle entity in the correct order, with delays between 5-7 seconds to allow them enough time to accept and execute their orders. During a demonstration, this requirement for the manual human-in-the-loop task order authorization makes JSAF difficult to use. In addition, the workarounds and the need for human intervention make it difficult to recreate a historic event. Nevertheless, we believe that there exists a level of fidelity of crowd behavior that will most closely match the outcome of the historic event and that level of behavior fidelity is not the level in which the crowd is excluded from the scenario.

7. CONCLUSIONS

Using 3D Studio MAX as a common platform from which to produce correlated source data that was valid across multiple applications proved to be very helpful for the production pipeline of the Crowd Modeling project, by allowing us to make quick modifications that could then be then propagated to the other tools.

As part of this pipeline, visuals were generated that we required for the 3D stealth viewer using the Renderware rendering engine while at the same time providing correlated information to JSAF.

This project explored he convergence of military simulation technology with gaming simulation technology as a front end. By doing this we are able to produce not only a highly realistic simulation, but also a capability of displaying the simulation in a highly detailed manner, by taking advantage of gaming technology rendering engines, which are optimized for visual quality and speed performance.

8. REFERENCES

[1] M. D. Petty, F. D. McKenzie, and R. C. Gaskins, "Requirements, Psychological Models, and Design Issues in Crowd Modeling for Military Simulation", *Proceedings of the Huntsville Simulation Conference 2003*, Huntsville AL, October 29-31 2003.

.

9. ACKNOWLEDGEMENT

This research described in this paper is sponsored by the Defense Modeling and Simulation Office and managed by the Air Force Research Laboratory. That support is gratefully acknowledged.

10. AUTHOR'S BIOGRAPHIES

Frederic (Rick) D. McKenzie is an Assistant Professor of Electrical and Computer Engineering at Old Dominion University. He received a Ph.D. in Computer Engineering from the University of Central Florida in 1994. Dr. McKenzie previously held a senior scientist position at Science Applications International Corporation. He has had several years of research and development experience in the software and artificial intelligence fields. Both his M.S. and Ph.D. work were in artificial intelligence.

Hector Garcia is responsible for the management of the VMASC's Virtual Environments research area. He also develops richly interactive and visually compelling immersive simulations. Mr. Garcia received an M.Arch from University of Houston in 1997, and a B.Arch from Universidad Regiomontana in Monterrey, N.L. Mexico. Mr. Garcia has worked in Immersive Visualization Environments since 1995. He previously worked for the Virtual Environments Technology Laboratory at the University of Houston.

Quynh-Ahn (Mimi) H. Nguyen is a Ph.D. student in the Modeling and Simulation (M&S) program at Old Dominion University and a Research Assistant at the Virginia Modeling, Analysis and Simulation Center (VMASC). She received a M.S. degree in M&S from Old Dominion University in 2003 and a B.S. degree in Electrical Engineering from George Mason University.

Jen Seevinck is a Research Scientist at VMASC East's virtual reality facility. Here she has worked on medical and engineering visualizations as well as the virtual environments for training. Her work at ODU includes object modeling/texturing/animation and project managing. Prior to joining VMASC in 2001, Ms Seevinck established and lectured in computer animation and multimedia at Australia's Deakin University. She has held similar faculty positions at the Australian National University and worked as a freelance designer on interactive museum exhibits, web, architecture, theatre and film. Ms Seevinck has an undergraduate degree in architectural design and a master in electronic arts with a thesis in VR interface design from the Australian National University.

Mikel D. Petty is Chief Scientist of the Virginia Modeling, Analysis and Simulation Center at Old Dominion University. He received a Ph.D. in Computer Science from the University of Central Florida in 1997. Dr. Petty has worked in modeling and simulation research and development since 1990 in areas that include simulation interoperability, computer generated forces, multi-resolution simulation, and applications of theory to simulation. He has published over 90 research papers and has been awarded over 30 research contracts. He has served on a National Research Council committee on modeling and simulation and is currently an editor of the journals SIMULATION: Transactions of the Society for Modeling and Simulation International and Journal of Defense Modeling and Simulation.