SYNTHETIC URBAN ENVIRONMENTS USING QUAKE II ENGINE

Benito Graniela, Jaime Cisneros, Dr. Douglas Reece

Science Applications International Corporation

Orlando, FL

Abstract

Game technology, and in particular first person shooter (FPS) games, such as Quake II, Quake III and Unreal Tournament, provide attractive capabilities for some of the traditional modeling, training and simulation (M&S) applications. A PC game-based simulation would be most useful if it operated with existing military simulations using their existing terrain databases (TDBs). However, PC games use different formats from the ones traditionally used by military simulations.

This paper will describe the necessary steps to import the traditional M&S TDBs into the format used by several common FPS games, in particular Quake II MAP format. A brief overview of the MAP format will be provided and compared to the traditional polygonal format used in the modeling and simulation community. Details of the conversion process of an OpenFlight Military Operations on Urban Terrain (MOUT) database to a Quake II level will be presented, as well as the conversion of a Quake level to a semi-automated forces (SAF) system's terrain database (TDB) format. A brief overview of a prototype's DIS engine added to Quake II will be provided, along with some lessons learned. Finally, comments will be presented as to the suitability of the Quake II game engine environment format and runtime engine for M&S applications.

BENITO GRANIELA is software engineer at SAIC, where he works on visual and computer generated forces representation for urban synthetic environments. Prior to SAIC he did research at the Institute for Simulation and Training in interoperability of visual distributed environments and as a result developed a deep understanding of the basic elements of networked distributed simulation. Mr. Graniela earned a B.S.E in Electrical Engineering from the University of Puerto Rico in 1995 and a M.S.E in Computer Engineering from the University of Central Florida in 1992.

JAIME CISNEROS is a Senior Software Engineer at SAIC, where he works on behavior modeling in DISAF, research in cognitive architectures and the application of game technology to Modeling and Simulation. Mr. Cisneros holds a Master's and Bachelor's degree in Computer Science from the University of Central Florida. His previous experience includes the development of DIS and HLA testing tools, the integration of heterogeneous simulations, research in reactive behavior, obstacle avoidance, and targeting algorithms. His research interests are in the areas of artificial intelligence, real-time simulation, and game technology.

DR. DOUGLAS REECE is a Senior Scientist at SAIC in Orlando. He has been developing physical and behavioral models for individual combatant CGFs for the past six years. He is currently the software architect for DISAF projects; previously, he was the principal investigator for Computer Controlled Hostiles for the Marine Corps' Team Target Engagement Simulator. He received his Ph.D. in Computer Science from Carnegie Mellon University in 1992

INTRODUCTION

Over the last several years, the entertainment industry has been driving advances in the development of 3D graphics on low cost general purpose processing platforms. The results of these advances are games such as Quake II, Quake III, and Unreal Tournament. These games use very realistic looking environments due to excellent levels of detail and features, such as lighting and smoke effects, as well as, particle and character animations. The realism of the game environments allows for player immersion in the 3D environment, which is one of the most important aspects that the M&S Community desires in order to enhance the training experience.

For traditional M&S applications to be able to interoperate with FPS games such as Quake II, it will be necessary to have consistent environment representations, similar simulation models and compatible communication protocols. Although having correlation in the environment and communications protocols does not guarantee interoperability between or among simulations, it is a prerequisite.

The purpose of this paper is:

- To highlight those game environment features in Quake II, which might be useful for M&S applications.
- To present a process for importing traditional M&S environments (terrain databases) into Quake II.
- To present a process for exporting Quake II environments.
- To describe a prototype interoperable with the Dismounted Semi-Automated Forces (DISAF) system that uses a known urban Quake II level, and one imported from a DISAF terrain database.
- To expose some of the limitations and advantages of the use of game technology in M&S.

QUAKE II GAME ENVIRONMENT FEATURES

The features of the Quake II game engine that are of interest for this paper are those which are directly tied to the game environment rendering at run time. Quake II obtains the best rendering performance possible by using optimized scene management techniques. Thus, in Quake II, levels are compiled into a runtime format, which divides the world into a Binary Space Partitioning (BSP) tree. The compilation process also generates two databases, which contain visibility and lighting information.

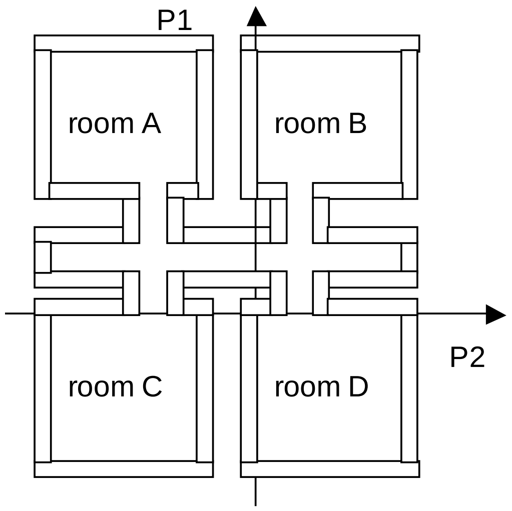

Figure 1 - The two arrows represent planes, P1 and P2, which separate the lower right room D from the other three rooms. The room D geometry and everything in it will be contained in a BSP leaf node. PVS for this mode will include the hallway and a reference to room B, which can be seen through the door of room D.

The BSP tree-building goal is to partition the environment into convex volumes by using structural planes, which are called brushes. For example, a simple room with four walls is a convex volume bounded by six planes. Convex volumes are important for visibility, as it is simple and fast to determine whether two convex volumes can see each other. In Quake II convex volumes are stored as leaf nodes in the BSP tree. The BSP tree is then primarily used to divide the map into regions, and to quickly determine which region the camera or player is in. The BSP is constructed through recursion on a volume starting with the entire level as a single volume. The volumes are split along the face of structural brushes, continuing with the first child volume, and repeating the operation until no more structural brushes are found. At that point, a leaf node is created for the volume (see Figure 1). Solid structural brushes block visibility between or among the content of leaf nodes. Windows and doors or portals allow visibility from one BSP leaf node to another. When the player viewpoint is anywhere in a leaf node which contains portals to other nodes, other leaf nodes need to be taken into consideration when rendering the volume. The more leaf nodes visible from a specific leaf node that contains the view point, the more objects that will need to be drawn. Leaf nodes are grouped together with neighboring leaves to form clusters. This technique allows for non-convex regions in a room to be grouped together as a cluster. For each cluster, a list of all of the other clusters, which are potentially visible, is stored. This is referred to as the potentially visible set (PVS).

To render a Quake II map, the BSP tree is traversed to determine which leaf the camera is located in. Once it is known which leaf the camera is in, the cluster that it's in is also known (remember that each leaf is contained in exactly one cluster). The PVS for the cluster is then decompressed giving a list of all the potentially visible clusters from the camera location. Leaves also store a bounding box, which is used to quickly cull out leaves that are not within the viewing frustum. Using this information, the rendering engine generates a display list. A recursive method that uses the PVS information, viewpoint and portals is used to generate a display list, which contains all the geometry that is visible from the current viewpoint. This technique improves rendering performance by reducing depth complexity.

The BSP Tree algorithm in Quake II has been optimized for indoor scenarios by taking advantage of its characteristics, which among other things include a small number of portals. These same optimizations do not work well in large open spaces, because outdoors scenes contain a large amount of visible leaf nodes. However BSP Tree and PVS work very well on highly occluded environments like those seen on urban areas, buildings and ship interiors. The M&S community is seeing an increased need for the use of highly occluded environments and will greatly benefit from using BSP like scene management techniques.

CONVERSION OF OPENFLIGHT TO QUAKE II LEVEL

As an exploratory investigation, SAIC converted a section of the McKenna MOUT terrain database into a Quake II level format. The task was divided into terrain skin geometry conversion, texture format conversion and culture conversion, which included one building and several trees. The entire McKenna MOUT terrain database was not converted because the primary purpose of the conversion process was to investigate the possibility and potential of game engine use.

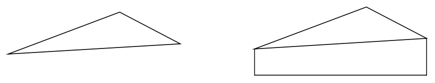

Figure 2 - Triangles are converted to brushes by extruding along the normal.

In order for the terrain skin to be converted to a Quake II level, the terrain triangles were converted into brushes. For this purpose, a computer program was developed, which could generate a five-plane brush for each terrain triangle. The end result was equivalent to extruding each terrain triangle along its normal (see), and adding texture-mapping information. Since each one of the terrain triangles in the terrain tin could potentially become a leaf node in the BSP tree, a large BSP could be generated. Large BSP tree are inefficient, as each one of the leaf nodes has a large PVS node list.

As part of the terrain skin conversion, it was then necessary to limit the extent of the database and to convert the floating-point vertex information to integers. At the time of the conversion, it was believed that one integer unit in a Quake II level was equivalent to an inch in the real world. It turned out that a limiting factor in the conversion process was the editable extents that the level-editing tool could process. The tool used for visualization of the brush data limited level extents to 4096 x 4096 units. If each unit was equivalent to an inch this allowed for a 104.04 x 104.04 meter section of the database to be manipulated by the tool. Therefore, it was decided that a 100 x 100 meter section at the center of the McKenna MOUT terrain database was to be extracted and converted (see Figure 3).

At this point in the conversion process, the terrain skin geometry had already been converted from a triangle representation to a brush representation called a MAP. A MAP is an ASCII file, which contains brush and entity data for a Quake II level. Every Quake II level needs to be completely contained within a volume (defined by six brushes). Failure to do so results in BSP compilation errors and runtime visual anomalies. The other element needed by every Quake II level is what is known as an entity spawn point. The spawn point tells the Quake II engine where to initially start the player and

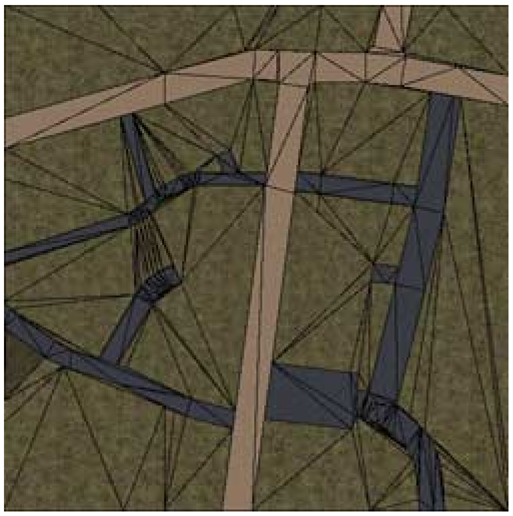

Figure 3 - Top view of the 100x100 M McKenna terrain skin

computer controlled entities. Levels may contain multiple spawn locations, but at least one is needed. This technique provides the game engine with location for starting entities, and eliminates the need for keeping extents and origin information in the terrain database.

To complete the terrain skin conversion process, it was necessary to convert the original terrain database textures to the Quake II WAL format. The original terrain database contained textures in OpenFlight RGB, RGBA and INT formats. Two methods were tested in the conversion process. The initial method used Adobe PhotoShop WAL and RGB plug-ins to perform the operation. The initial conversion process did not yield satisfactory results, so an alternate solution was investigated. Textures converted by this process were either too bright or had the incorrect color mappings. A special purpose shareware tool called WALLY was used for the texture conversion process in order to generate the correct texture. Since the tool could not directly process RGB textures, it was necessary to convert the textures to PCX format before the conversion process could take place. RGB textures were loaded into PhotoShop and converted from RGB to index mode and saved as PCX. The PCX images were then loaded into WALLY and converted to WAL format. This still provided textures that were too bright when rendered by the Quake II engine. After further investigation, it was discovered that the mapping of RGB colors into the Quake II texture palette was the cause of the color conversion anomalies. RGB texture colors that are too similar were mapped into the same color in the WAL textures. The best results were achieved by spreading the distribution of colors on the original RGB image before the conversion to index mode. This yielded a wider range of Quake II palette colors. Nevertheless, it still required further processing of the textures, which involved darkening and sometimes increasing the contrast on the image.

In addition to the terrain skin, several tree models were placed in the terrain, not as billboard trees, but instead as equivalent volumetric pine trees.

The conversion of buildings with interiors was also explored. The same program used to convert the terrain skin was used to generate a MAP file of one of the McKenna MOUT buildings. Although the basic building geometry was generated, extra geometry was also produced, which caused display anomalies. Again, the net effect of the conversion process was to extrude the building triangles to generate brushes, which yielded extra amounts of brushes inside walls, widows and doors. Some of them contained coplanar planes, while others contained planes which were not visible or were unnecessary. Since the development of a conversion tool was not the main purpose of the investigation, it was decided to edit the model, instead of enhancing the tool to handle buildings. Extensive model editing was necessary to remove the extra geometry before it could be incorporated into the rest of the database. When an acceptable model was obtained, it was pasted into the MAP file, which contained the terrain skin. This file containing the combination of terrain, buildings and trees was converted to the Quake II runtime format.

Although the functional Quake II level was generated successfully, it was not as attractive as typical Quake II levels. The reason is that the textures did not translate well into the WAL format, and the geometry for the terrain skin resulted in a lot of BSP leaf nodes.

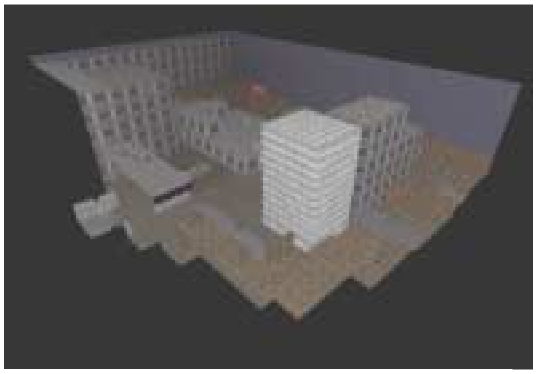

Figure 4- Top view of the Urban Quake II level

CONVERSION OF QUAKE II LEVEL TO CTDB

The Human Simulation (HSIM) team at SAIC has done extensive work with the Modular Semi-Automated Forces (ModSAF) Compact Terrain Database (CTDB), which among other things, includes the addition of Multi-Elevation Structures (MES) [1]. As part of an Internal Research and Development (IRAD) project, the HSIM team investigated the possibility of generating a CTDB for an existing Quake II level. After all, some training applications could benefit from using geo-typical databases, in particular those applications that require building interiors.

The Urban level (see Figure 4) was selected due to its popularity among the Quake II level community and for its urban characteristics, which included streets, parking garage, buildings with interiors, alleys, and rooftops. Since the brush file for the urban level was not available, it was necessary to manually extract vertex information using a level-modeling tool.

The urban BSP level was loaded into another 3D editing tool, called QuArk5 [2] * MERGEFORMAT , and selective building vertex information was extracted. Since it was the initial development process, rough measurements were made at the time, with hopes to perform more refined measurements later on. Using the extracted 3D coordinates, a 3D model was developed. The first step was to use the recorded building outline coordinates to develop a 2D outline for each building volume. After this step, each building was extruded to its corresponding elevation. This process is analogous to the way buildings are represented in CTDB.

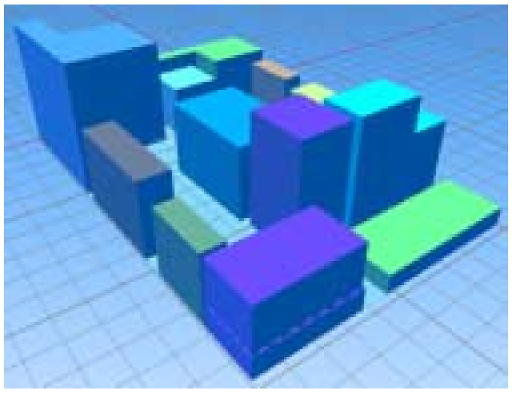

Figure 5 - 3D polygonal model of the Quake II Urban level

The 3D model of the Urban Level (see Figure 5) was then converted to meters by scaling it to 1/39.37 of its original size. The building outline information was then converted into ModSAF recompile modification files. These volumes were finally added to a CTDB using the ModSAF recompile tool.

Although a CTDB was generated (see Figure 6), its correlation to the original urban database was poor. Thus, a prototype tool that could convert brushes or plane data to triangles was developed. The prototype takes the Quake II brushes, and converts them to triangles, by clipping each one of the planes by the other n-1 planes and triangulating the result. Unfortunately, the algorithm was not flexible enough to handle brushes composed of more than six planes. Furthermore, separation of the level geometry into terrain skin, features, fixed models, such as building, trees and bridges, and linears (roads and rivers) will be needed so that the correct components of the CTDB can generated.

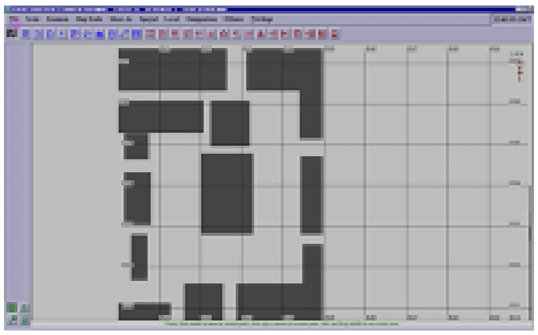

Figure 6 - Urban CTDB loaded in DISAF

QUAKE II DIS MOD

In M&S applications, the terrain database is only one of the components, which allows for interoperable systems. As part of our investigation, a DIS interface was added to Quake II to allow it to interact with DISAF. The goal of adding the DISAF interface was to test the interoperability of the environments described earlier.

Quake II is divided into two components: an executable and a game Dynamic Link Library (DLL). The game DLL is responsible for the behaviors of the computer-controlled entities, weapons and the physical models (see Figure 7).

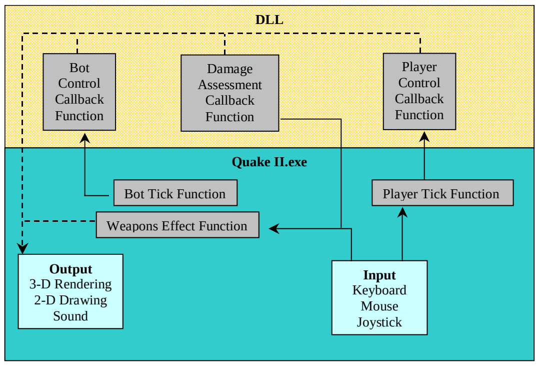
DLL
Bot Control Callback Function
Damage Assessment Callback Function
Player Control Callback Function
Quake II.exe
Bot Tick Function
Player Tick Function
Weapons Effect Function
Output 3-D Rendering 2-D Drawing Sound
Input Keyboard Mouse Joystick

Figure 7 - Quake II Software Architecture

The server and its networking code reside in the game executable portion of Quake II. The server is the authority over the game state, and by being the authority, it determines how actions take place in the game world. In the public version of Quake II, the game executable is not available in source code form, which prevented the modification of the server and its networking code. Hence, the DIS interface was added to the game DLL to circumvent this issue. This solution is not the most adequate because the server is the owner of the state of the world, which the DIS interface cannot have access to. This will become apparent, as results of using the prototype will show.

The DIS interface added to the game DLL, then, turned Quake II into a distributed application on the DIS network

By adding the DIS interface, Quake II was able to:

1. Receive Entity State PDUs to represent external entities. External entities are treated as computer-controlled entities.

2. Receive Fire and Detonation PDUs to show the firing and detonation action, as well as, the possible damage to the player if it is being fired upon.

3. Send Entity State PDUs to describe the player's location, orientation, velocity, appearance, etc.

4. Send Fire and Detonation PDUs in response to firing events originated from the player.

Figure 8 shows the new Quake II game DLL and executable software architecture. Input interactions are based on data received from the DIS network and player inputs. The Quake II state is transmitted to network as DIS Entity, Fire and Detonate PDUs.

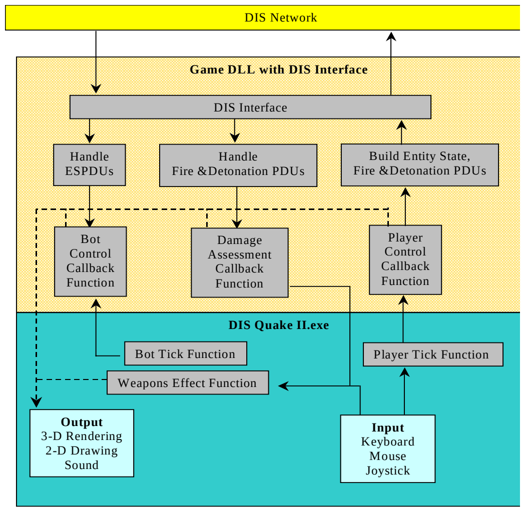
DIS Network
Game DLL with DIS Interface
DIS Interface
Handle ESPDUs
Handle Fire &Detonation PDUs
Build Entity State, Fire &Detonation PDUs
Bot Control Callback Function
Damage Assessment Callback Function
Player Control Callback Function
DIS Quake II.exe
Bot Tick Function
Player Tick Function
Weapons Effect Function
Output 3-D Rendering 2-D Drawing Sound
Input Keyboard Mouse Joystick

Figure 8 - Quake II Game DLL and Quake Executable Interactions

The new Quake II communicated effectively with DISAF over the DIS network. However, some issues were found after interactions took place between the Quake II player and DISAF entities. The first issue had to do with controlling the external entities in the game. It was found that accurately controlling a computer-controlled entity's position was not possible because entities in Quake II are controlled by providing a velocity and a direction of movement for a period of time. This is the result of the Quake II server control over the entire game state, and the inability to modify the server. The location provided in the Entity State PDUs needed to be converted to velocity and movement direction over a period of frames. This means that the Quake II server code was in charge of moving the computer-controlled entities while maintaining a reasonable frame rate and a synchronized world state. This resulted on location differences between the DISAF entity location and the Quake II visual representation. The most significant issue found, nevertheless, had to do with correlation between DISAF's and Quake II's elevation data. The height of the terrain at a given place in Quake II never coincided with the height at the same spot in DISAF. This caused anomalies in weapon aiming and munition impact points. Therefore, interoperability was not fully achieved. Solutions have been analyzed and provided, but have not been tried in time to provide results in this paper.

LESSONS LEARNED / PRO AND CONS OF QUAKE II ENGINE UTILIZATION

One of the main advantages of using the Quake II engine is that it provides a complete environment over which a simulation could be built. Other advantages include the availability of a wide variety of shareware tools, which allow for the quick development of geo-typical environments. Quake II levels include dynamic and special effects, such as breaking glass, explosions, and the ability to open doors, move elevators and trigger actions by the use of buttons or volumes. Shareware tools are also available for the development of 3D models or computer-controlled entities. These tools allow for the development of new animations, new skins, and textures for Quake II entities.

In addition to game level modification, game logic modifications are also possible with a small amount of work. Prototype code for a computer-controlled entity can be incorporated into the game, leveraging the nice and fast visuals provided. The Quake II environment provides a complete environment with computer-controlled entity behaviors, which a developer can use to test new entity controls and sensing algorithms. Nevertheless, one of the negative aspects is that licensing fees for the Quake II Engine are high, making it prohibitive to use this game engine for the traditional commercial development of visual M&S applications.

Indoor representations on Quake II employ BSP tree techniques in conjunction with techniques that determine portal visibility. This optimized representation allows for fast rendering of indoor environments, which include a high degree of occlusion. Although polygonal terrain skins can be converted with no loss in correlation to the game format, the encoding of this data into BSP trees does not benefit from the same spatial benefits that indoor environment provide. The Unreal game engine seems to provide an alternative to this limitation. In fact, Epic Games is using a hybrid model, which combines the use of BSP for indoor scenes and other techniques for outdoors scenes in their new Unreal Tournament Game Engine.

Terrain representation of vertex information in Quake II is in integers vs. the traditional floating-point representations. This requires transformation of floating point numbers to integers using some sort of fixed-point basis. Scale changes and runtime compilation tools limit the extents to around 100 x 100 meters play box. Most likely the size of the play box have been limited to obtain the best rendering performance on typical game machines.

Tree representation in the Quake II engine needs to be volumetric. The Quake II format does not provide for billboard or interpenetrated polygon representations of trees.

At this time it does not look as if BSP trees allow for dynamic geometry modifications, other than the one provided through breakable brushes. However, considering that not too many simulations today can dynamically modify the environment, this does not seem like a big limitation.

CONCLUSIONS

The very same characteristic, which makes BSP practical for indoor representation, makes it impractical for outdoor use. Moreover, although binary partition techniques have been used in the M&S community before for outdoor scene representations, it is most likely that today's scene graph representations will still be used for outdoor environments. It is worth mentioning, that indoor environments are becoming more and more important as we build trainers and simulations for individual humans.

Another characteristic that will greatly benefit the indoor environments for M&S applications is the use of a combined visual and SAF database. Feature metadata, as well as visual data, should all be combined into the same environment because this feature information comes with a standard set of functions, which can be attached to structural elements. Environment modelers can use the set of standard functions to add or include built-in dynamic behaviors into the indoor environments. Possible functions include those used to open doors and windows, as well as those used to trigger buttons to turn lights on and to set alarm systems off by motions detectors. As the complexity of M&S environment increases due to higher detail in outdoor, indoor, and urban environments, the need to start building this functionality into environments becomes a necessity. This type of functionality will not only help interoperability, but it will also bring about richer environments.

REFERENCES

[1] Pigora M.A., & Graniela B., & Reece D. (2001) "Urban Human Simulation Environments in CTDB", *Proceedings of the Spring 2001 Simulation Interoperability Workshop*

[2] Planet Quake (2001), The Official QuArK Homepage (online), http://www.planetquake.com/quark/ (2001)