May 2004

Enhancing the Behaviorial Fidelity of Synthetic Entities with Human Behavior Models

Michael van Lent
Institute for Creative Technologies

Ryan McAlinden
Institute for Creative Technologies

Paul Probst
University of Southern California

Barry G. Silverman
University of Pennsylvania, basil@seas.upenn.edu

Kevin O'Brien
University of Pennsylvania

See next page for additional authors

Postprint version. Presented at the *13th Conference on Behavior Representation in Modeling and Simulation (BRIMS), SISO*, May 2004, 9 pages. Published at: http://www.sisostds.org/

Enhancing the Behaviorial Fidelity of Synthetic Entities with Human Behavior Models

Abstract
Human-behavior models (HBMs) and artificial intelligence systems are called on to fill a wide variety of roles in military simulations. Each of the "off the shelf" human behavior models available today focuses on a specific area of human cognition and behavior. While this makes these HBMs very effective in specific roles, none are single-handedly capable of supporting the full range of roles necessary in an urban military scenario involving asymmetric opponents and potentially hostile civilians. The research presented here explores the integration of three separate human behavior models to support three different roles for synthetic participants in a single simulated scenario. The Soar architecture, focusing on knowledge-based, goal-directed behavior, supports a fire team of U.S. Army Rangers. PMFServ, focusing on a physiologically/stress constrained model of decision-making based on emotional utility, supports civilians that may become hostile. Finally, AI.Implant, focusing on individual and crowd navigation, supports a small group of opposing militia. Due to the autonomy and wide range of behavior supported by the three human behavior models, the scenario is more flexible and dynamic than many military simulations and commercial computer games.

Keywords
human behavior model integration, interchange standards, crowd simulation

Comments
Postprint version. Presented at the *13th Conference on Behavior Representation in Modeling and Simulation (BRIMS), SISO*, May 2004, 9 pages. Published at: http://www.sisostds.org/

Author(s)
Michael van Lent, Ryan McAlinden, Paul Probst, Barry G. Silverman, Kevin O'Brien, and Jason Cornwell

Enhancing the Behaviorial Fidelity of Synthetic Entities with Human Behavior Models

Michael van Lent, Ryan McAlinden, Paul Brobst
Institute for Creative Technologies
University of Southern California
13274 Fiji Way
Marina del Rey, CA 90292
310-574-5710
vanlent@ict.usc.edu; mcalinden@ict.usc.edu; brobst@ict.usc.edu

Barry G. Silverman, Kevin O'Brien, Jason Cornwell
Ackoff Center for Advancement of Systems Approaches (ACASA)
Electrical and Systems Engineering, University of Pennsylvania
Towne 229c, Philadephia, PA 19104-6315
215-573-8368
barryg@seas.upenn.edu

Keywords:
Human Behavior Model Integration, Interchange Standards, Crowd Simulation

ABSTRACT: *Human-behavior models (HBMs) and artificial intelligence systems are called on to fill a wide variety of roles in military simulations. Each of the "off the shelf" human behavior models available today focuses on a specific area of human cognition and behavior. While this makes these HBMs very effective in specific roles, none are single-handedly capable of supporting the full range of roles necessary in an urban military scenario involving asymmetric opponents and potentially hostile civilians. The research presented here explores the integration of three separate human behavior models to support three different roles for synthetic participants in a single simulated scenario. The Soar architecture, focusing on knowledge-based, goal-directed behavior, supports a fire team of U.S. Army Rangers. PMFServ, focusing on a physiologically/stress constrained model of decision-making based on emotional utility, supports civilians that may become hostile. Finally, AI.Implant, focusing on individual and crowd navigation, supports a small group of opposing militia. Due to the autonomy and wide range of behavior supported by the three human behavior models, the scenario is more flexible and dynamic than many military simulations and commercial computer games.*

1. Introduction

Human-behavior models (HBMs) and artificial intelligence systems are called on to fill a wide variety of roles in military simulations. These include allied teammates and subordinates supporting a human participant, enemy combatants working against the human participant, and civilians on the battlefield. Each of these roles requires an HBM that can support or emphasize different knowledge sets, cultural and personality factors, and even cognitive processes. For example, in an urban combat situation a U.S. Army Ranger's behavior will primarily be defined by mission and doctrine reflecting the soldier's extensive training. A civilian's behavior in the same situation will be determined more by emotions, such as fear, and goals, such as self-preservation. Ideally a complete Human-behavior model will include representations of all the factors that influence behavior and will therefore be capable of supporting the full range of roles. While a number of architectures are moving in this direction [1, 8], none of the currently available HBMs yet cover the full range of human cognitive function. Despite this, most simulations and research systems use only a single HBM to drives all their computer-generated forces (CGFs), no matter how broad the range of roles. The work presented here explores the use of multiple human behavior models in a single virtual environment to increase the realism of the simulated participants by tailoring the HBMs to each participant's role in the scenario.

Using the events depicted in the popular book and movie Black Hawk Down [3] as motivation, we have defined a light infantry, urban combat scenario. The synthetic participants in this scenario include a fire team of U.S. Army Rangers, a small group of opposing militia, and a number of civilian non-combatants. The human participant acts as the leader of the U.S. Army fire team. The four synthetic entities in the human led fire team are controlled by four Soar agents [14]. Soar was selected for its strength in knowledge-based and goal-directed behavior. The opposing militia is

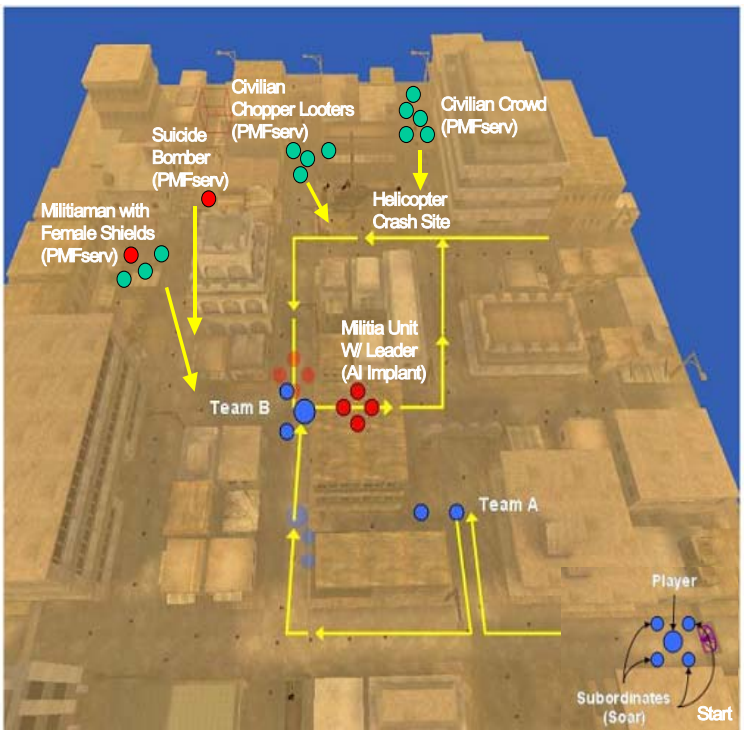

Figure 1: A top-down view of the terrain for the Mogadishu test bed with the actions of the Soar (blue), AI.Implant (red), and PMFServ (green) entities.

controlled by AI.Implant [2], a game industry AI middleware tool selected for its ability to control multiple entities navigating as a team. Civilians are controlled by the Performance Moderator Function-based PMFServ architecture [11]. This scenario takes advantage of PMFServ's models of emotion, stress and a range of coping styles.

The next section describes the Mogadishu testbed in more detail and the game-engine based simulation environment in which the testbed is implemented. The following section describes the common interface used by all three human behavior models. Developing a common interface, applicable to a wide range of models, can reduce the cost inherent in interfacing multiple HBMs into a single simulation environment. The three subsequent sections describe each HBM: Soar, PMFServ, and AI.Implant, in more detail and describe how each supports a role in the testbed. Finally, the final section presents some areas of future research.

2. Mogadishu Testbed Environment

The Mogadishu testbed consists of a partial recreation of a scenario from the popular book and movie Black Hawk Down. This testbed was developed using the Unreal Tournament game [5] in conjunction with the freely available Infiltration "mod" (or modification) of the game. Infiltration increases the realism of the game environment through more realistic character and weapon models, environments, and damage models.

1.1 Mogadishu Scenario

As stated above, the testbed is based on a *Black Hawk Down*-inspired, asymmetric, urban combat scenario. The testbed currently includes three distinct HBMs that are integrated and operating concurrently within the virtual environment. The HBMs control synthetic entities, or Non-Player Characters (NPCs), serving three different roles within the game (U.S. Army Rangers, civilian crowd, asymmetric opponents). All three HBMs share a common interface to the simulation environment. Using Unreal Tournament as the underlying simulation, a Mogadishu-based scenario demonstrates various HBM capabilities such as tactical maneuvers and behavior, coordinated group movement, and threat sequences. Custom art assets have been developed including terrain, buildings, and 3D models and textures for soldiers and weapons.

In the Mogadishu scenario a group of U.S. Army Rangers traverse the streets of Mogadishu in an attempt to locate a downed Black Hawk helicopter. Along the way, they encounter a variety of asymmetric threats and civilian crowds, each of which must be dealt with appropriately. The terrain consists of

approximately 16 city blocks in a 4x4 street grid (see Figure 1). These blocks consist of interspersed multi-level buildings, obstacles, and a series of alleys. The general objective of the scenario is for the human player (and his Soar-controlled subordinates) to engage and neutralize the OPFOR militia (AI.Implant) while minimizing BLUFOR and civilian (PMFServ) casualties.

1.1 Unreal Tournament: Infiltration

Unreal Tournament (UT) is the underlying simulation environment used to develop the Mogadishu test bed. UT is a popular First Person Shooter (FPS) game, released in 1999, that includes one of the most widely used interfaces to allow hobbyists, researchers and developers to extend and adapt (or "mod") the game to meet particular needs. "Modding" the game occurs on a variety of different levels, from 3D model, texture, and terrain design and importation (using the Unreal Editor), to defining new behaviors and action sequences within the game itself.

The UT Game Engine (UTGE) is the driver behind any game or simulation scenario developed in UT. Through the free mod interface, many of the UTGE components have been "exposed" giving hobbyists and developers a consistent programming interface to make changes to many aspects of the existing game (rendering, physics, AI, networking). UT was selected as the simulation environment for this project due to this extensible and flexible interface that allows for relatively simple integration with external software modules (i.e. HBMs). One of the most powerful tools provided by the UT mod interface is UnrealScript [13]. UnrealScript is a high level programming language used to construct game mods, either within the confines of the game engine or through an interface to external programs. It is object-oriented and is very similar in syntax to Java. Moreover, it allows developers to extend baseline game engine functionality with native C++ code. These native functions allow for relatively seamless integration of external software components with the game engine. Typically, the flow of data between these components and the game engine is managed by a threaded Windows-based Dynamically Loaded Library (DLL).

The off-the-shelf version of Unreal Tournament is not a realistic simulation of urban combat. However, a mod called Infiltration [10], developed by Sentry Studios, modifies UT to include more realistic soldier and weapon models, base-level behaviors, and tactics. Infiltration replaces the default weapons in UT (laser guns, rail guns) with weapon models common to armed conflicts today (such as the M16, the M4, and the AK47). Moreover, the character models have been modified so that they resemble soldiers and civilians rather than futuristic robots. Infiltration provided the baseline character movement (walking, running) and weapon handling (firing, reloading, unjamming) actions. We enhanced the Infiltration mod with the custom urban terrain and custom character models representing Somali civilians.

3. Interface

The Human Behavior Model Interface Standard (HBM IS) is a control methodology and set of data specifications that allows disparate HBMs (such as Soar, AI.Implant and PMFServ) to manage and control synthetic entities in a common simulation environment. One of the advances demonstrated by this project is the use of a single interface specification to support these three models running concurrently within the simulation. Each HBM is an external software module that operates asynchronously through an UnrealScript native function interface with the game engine.

The most important aspect of the HBM IS is that it is independent of any specific HBM or simulation environment. While a layer of the interface software has been developed specifically for UT, the methodology used is quite broad. Simulation data is polled each iteration through the game cycle, distributed to the appropriate HBM, processed by that HBM, and an entity-control command is returned to the simulation for execution (move, attack, orient). The HBM IS is the codification of a great deal of previous experience with HBM design on the part of a number of members of the development team. As a starting point, the interface builds off a portion of the Soar General Input/Output (SGIO) mechanism developed at the University of Michigan. This mechanism, combined with UT-specific native function software, allows for the management of data between Soar and UT. However, in the standard SGIO interface, this data is quite Soar specific. Several modifications have to be made to the methodology to support the incorporation of other HBM components. High-level modifications include:

- HBM initialization—specify the world state representation for each HBM's data model.
- HBM setup and teardown—structure HBM processes as separate threads so that do not interfere with one another or the game engine.
- HBM group actions—Add a specialized set of actions to support HBM control of groups of synthetic entities.

Data Control

The quantity of sensor data coming into each of the HBMs through the interface can be extremely large. Sensor data is updated from the simulation multiple

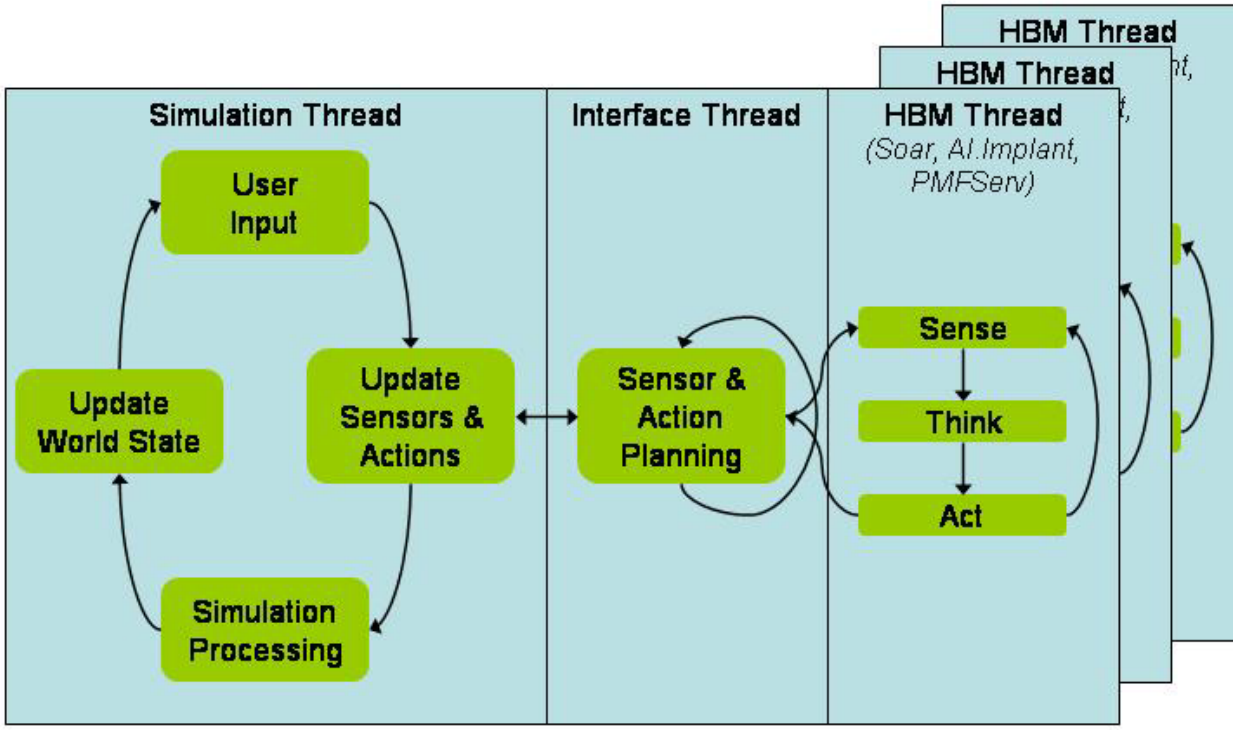

Figure 2: The Human Behavior Model Interface Standard's multithreaded approach allows the simulation and HBMs to operate asynchronously.

times per second (about 10 times per second), requiring the input mechanism for each HBM to be constantly monitoring for the most recent simulation updates. This occurs with the use of a custom Unreal Tournament DLL that is loaded into the game engine process during startup. This DLL interface has the advantage of providing a high throughput, low latency interface between the simulation and the HBM components. However, DLLs are a feature supported only in Microsoft Windows which somewhat decreases flexibility when considering non-Windows environments. Linux and MacOS both have equivalent but different features which are not supported by the HBM IS at this time.

One particularly important aspect of the Human Behavior Model Interface Standard is the use of multiple processing threads to decouple the game engine's processing cycle from each HBM's processing cycle (see Figure 2). The Unreal Engine runs within a single thread for maximum efficiency. The HBM interface, however, carries over SGIO's multithreaded approach to allowing the simulation and HBM components to operate asynchronously. A key feature of the UT engine is the real-time rendering of the environment. To support this each step in the internal game loop must be carefully controlled to execute and return quickly. If any step takes too long the frame rate of the game can drop below the acceptable 30 frames per second. HBMs, especially those from the academic research community, have not yet been engineered to fit into these strict processing limits. Soar, for example, generally takes well less than 50 msec per cycle but in degenerate cases may take seconds per cycle. Encapsulating each HBM in a separate processing thread allows the game engine and HBM component to operate independently.

Data Types

In order for the HBMs to accurately model any level of intelligent behavior, from path-planning to emotional effects, data on the environment must be gathered from the simulation to serve as sensor input to the particular behavior model. This sensor data can include a wide variety of information about world state:

- Entity information (name, position/orientation, team, equipment, health, fatigue)
- Static and dynamic objects and terrain (obstacles, path nodes, projectiles)
- Goal and mission information (map name, game type, time limit)
- Spatial representations (weighted path node graph)
- Communication between synthetic entities and with the human user (formation, engage requests)

This sensor data is organized as a multi-level hierarchy. The top level of the hierarchy groups the sensors into general classes; Agent, Feedback, Objects, Game, Map, Sound, Message.

In addition to the sensor values, the HBM IS also specifies a set of action outputs that the HBM issues to control an entity in the environment. Each of these actions includes a number of parameters that inform the simulation as to the details of how to execute that action. Examples of action outputs include movement commands, firing commands, inter-agent communications, and special actions such as surrender.

Both the control mechanism and data types used in the interface are sufficiently independent that interfacing new human behavior models should be fairly simple. Some custom interface code will need to be written but, the HBM IS abstracts both sensor inputs and action outputs so that the integration process should be straightforward.

4. Soar

Initially developed at Carnegie Mellon University by Allen Newell, John Laird, and Paul Rosenbloom, Soar [8] is the human behavior model that controls entity movement, formation, and attack behaviors for the U.S. Army Rangers operating as subordinates of the human player.

The Soar-controlled characters in the Mogadishu scenario are designed to coordinate their behavior closely with the human player. The Soar characters initially form a box formation around the player and move with the player in that formation. The player can also order the Soar characters into a line formation. In addition, the player can order all four Soar-controlled Rangers or either pair of Rangers to hold position. Finally, the player can give the Rangers a weapons-tight command, forbidding them from firing, and a weapons-loose command, allowing them to fire at any hostile characters. Using combinations of these orders, the player's fire team can execute a number of basic tactics. For example, the player can set a base of fire with one pair of Rangers and move with the second pair to flank hostile forces.

One important distinction between the original Soar-UT integration, done at the University of Michigan and Soar Technology [14], and the one described here is use of player inputs to control mission-specific behaviors to be executed by the subordinate NPCs. In the Mogadishu scenario, the player can issue commands to the Soar entities directing their behavior. The player can issue formation commands, weapons tight and weapons loose commands, and hold position commands to all four Soar entities or individual pairs of Soar entities. There are two different approaches to modifying the behavior of the Soar-controlled entities based on these commands. The first, developed at the University of Michigan as part of this project, treats the player's commands as sensor inputs that modify Soar's internal operator and action selection. This approach encodes all aspects of player-directed behavior as Soar productions. The second approach, developed at ICT for this project, is encoded as a combination of high-level Soar productions and extensions to the sensors and actions available to the Soar entities. The player's commands are sent to the interface which then interprets the command and generates command-specific sensor inputs for the individual NPCs. For example, if the player issued a command for the Soar team to move in a line formation the HBM interface would interpret the command and provide each individual Soar entity with the specific coordinates of its position in the formation.

Each of these approaches has advantages and disadvantages. In the ICT approach the low-level formation holding behaviors are encoded in the interface which results in quicker reactions to the user's movements within the game. Also, since these low level behaviors exist in the interface they are not Soar-specific and may be used by other HBMs. However, while the reactions are faster, the resulting behavior seems significantly less natural (almost robotic) than the Michigan approach. The Michigan approach also has the advantage of making these low-level behaviors available to the Soar model where they can be affected by the higher-level goal-directed behaviors. For example, soldiers under fire might choose to move out of formation slightly to take better cover.

5. Performance Moderator Functions

The Performance Moderator Function Server, or PMFServ, is a flexible, composable approach to rapid generation of scenarios from reusable, previously validated components and agents. Over the past three years, PMFServ has been developed to construct a number of scenarios, including civilian and military crowd scenes, a car buying family and asymmetric leaders and followers. PMFServ was conceived as a software product that would expose a large library of well established and data-grounded Performance Moderator Functions (PMFs) for use by cognitive architectures deployed in a variety of simulation environments [11]. Its principal feature has been and continues to be a physiologically/stress constrained model of decision-making based on emotional utility [6] as follows:

1) Stress-Constrained Coping -- Physiological data across a range of measures (including arousal, fatigue, hunger, thirst, injury, etc) are combined to set the levels of a series of stress reservoirs. The stress reservoirs then determine the agent's coping style (a measure of the agent's current awareness and capacity for rational thought) for the current decision cycle. We follow Decision Conflict Theory's five stages of coping for a given agent: When bored or under-stimulated, such as in a prolonged surveillance mission, people tend to use defective coping, often blindly following procedures without thinking or double checking their execution -- slips and lapses are likely to arise. Under perfect conditions (moderate stress causing vigilant mode of thought), humans are presumed to be rational and often behave as Bayes Theorem and expected utility might predict, yet as conditions degrade (still more stress), they initially follow the dictums of subjective expected utility theory (Edwards, 1992) and, eventually, of recognition primed decision-making (Klein et al., 1993) if they are expert, or panic if they are inexperienced.

2) Cultural & Affective Reasoning -- Each agent is guided by three value trees (with Bayesian importance weights) concerning (1) a goal hierarchy, (2) a standards tree which includes how people should behave (ethics, religion, laws, and doctrine), and (3) preferences for artifacts/situations an agent wants near or far away. Together these three trees are what we refer to as the Goal, Standards, Preference (GSP) trees. These three hierarchies (and relationship linkages) and the importance weights for the individuals being simulated determine the activation and decay of 11 pairs of emotions as agents interact in the simulated world. As a demonstration of this concept, we have implemented this model and produced several papers on how to generate the GSP trees for various individuals belonging to several security, civilian, and opponent groups: e.g., [11,12]. PMFServ also implements the Damasio/Lazarus [4,9] concept of emotion influencing action by summarizing the values of the 11 pairs of oppositely valenced emotions into a single somatic marker or subjective "expected utility" score for each of the next steps that each individual is weighing. This is used in the decision unit of the agent's cognition to prioritize next response choices.

With this architecture, decisions made by PMFServ agents are bounded by coping style and by culture/affect. PMFserv quickly grew to become a cognitive architecture in its own right – with the flexibility to either act as a meta-level emotional arbitrator for others' cognitive architectures or to provide a fully functional stand-alone system to simulate bounded rationality human decision making.

In any given implementation, PMFserv runs as a server that operates the mind and behavior of each bot it is hooked up to, while the game or simulation environment operates as a client displaying the scenes and body animations. In this instance, the client is the Unreal Tournament Engine, while we linked the PMFserv to it through the Microsoft COM interchange method. Through this interchange, each client-side body interacts with its server-side mind to find out its reactions to events and to determine its next action choices. For each agent, PMFserv would operate its perception and run its physiology and cognition to determine injuries and related stressors, grievances, tension buildup, impact of rumors and communicative acts, and various mobilizations. Then individual action decisions and instructions are passed back to the game platform to carry out the resulting and emergent behaviors. It is the interaction of these parameters from the first-person perspective of the agent, not a rule or schema coded by a knowledge engineer in the third-person, which allows the agent to decide its next course of action. The result is an agent attuned to characteristics of the environment that readily adapts and responds to different situations.

By knowledge engineering a large default "cast" of such agents, placing them in a reusable library, and by providing the analysts and trainers with easy to use editors and generators, we feel this capability could lead to the military being able to compose new agent types and rapidly compose them into scenarios of interest to their training and analytic goals. The cast for the current implementation consisted of Somali civilians (males and females), Somali militia members, and a terrorist bomber. These cast types were modeled in PMFserv by adjusting their GSP tree values (Bayesian weights derived from SMEs). On the Unreal side, we found two freeware "skins" that were a male and female Arab in robes and that had some of the animations needed, though we had to adjust some of these and add others.

Figure 3 – View of Some of the PMFserv Controlled Bots in the Unreal-Mogadishu Environment

In the Black Hawk Down scenario, as the player moves through the environment, the PMFserv controlled NPCs begin to be encountered. From here onward, a number of PMFserv controlled NPCs populate the world as the Somali civilians (males and females) and Somali militia members (see Figure 3). Also a terrorist bomber emerges.

As the player and his subordinates advance upon the crash site, they encounter two groups of PMFserv civilians, one gathered around the helicopter and the other looting inside it. The player must disperse the crowds of Somali civilians both inside and outside the helicopter. In general, these Somalis have grown up with violence and are not easily intimidated. Further, they recognize when Rangers are vulnerable to swarming behaviors such as when a Ranger is alone, or with weapon out of ammo.

If the player or Rangers kill a civilian, this will precipitate all males (and possibly a female) to feel so violated they will search for a way to revenge themselves on the Rangers. In many cases this will result in them appearing to flee, when in fact they are locating a weapon and intending to return fully armed and ready to engage. Also, the player and his Rangers may encounter a crowd of civilians with a Somali militia shooting from behind them. The female NPCs have to make a decision to act as shields or not for the militia. If they do act as shields, the militia's tactics are probably to try and get the Ranger to kill one of the civilians. If the player or Rangers kill a civilian, this will precipitate a second threat which is a suicide bomber who appears as any other civilian male and is undetectable except that he advances without halting.

None of these behaviors are programmed directly into our PMFserv NPCs. Rather, the Goal, Standard and Preference Trees and weights are such that the chain of events described might emerge. Whether these behaviors emerge or not depends on player's and Ranger's behavior as well as on what the individual PMFserv NPCs actually observe happening. The latter also extends to what objects and perceptual types they notice in the world and how those project what they afford to the NPC. More details exist at: http://www.acasa.upenn.edu/Final_Tech_Report.doc.

6. AI Implant

AI.Implant is a commercial toolset, developed by BioGraphic Technologies [2], that allows developers to build and control in-game characters, focusing on group behaviors and basic navigation and path planning. It simplifies the need for the programmer to manually define particular behaviors and movements at the lowest level, allowing for enhanced game play and more intelligent characters. AI.Implant operates as a Maya or 3ds max plugin, both popular 3d modeling applications, and as a Software Development Kit (SDK) intended to be interfaced with other applications. The characters controlled by AI.Implant in the Mogadishu scenario used the SDK to connect to the HBM Interface built into Unreal Tournament.

AI.Implant contains its own internal representation of the terrain, separate from Unreal Tournament's representation. Classic game industry representations, such as Unreal Tournament's, involve embedding path-finding information directly into the 3D terrain model by creating a path node or waypoint network for an agent to follow. AI.Implant bases its terrain representation on the concept of barriers, walls and obstacles that an agent can collide with and see. These barriers form a somewhat simplified representation of the Unreal Tournament terrain indicating where the agent can and can't travel. Thus, agents can move freely around the environment, as long as they avoid the barriers instead of being forced to follow specific paths between nodes. AI.Implant can also use the barrier-based representation to automatically generate a path-finding network if one is needed. Unfortunately there is no simple way to extract the barrier geometry from UT so AI.Implant's barrier-based representation was created by hand.

In addition to the novel terrain representation, AI.Implant includes an extensible, agent-based system to generate individual and crowd navigation behaviors. Many default behaviors are included, such as flocking, avoiding barriers, following paths, and moving to a specified location. These behaviors are part of an inheritance hierarchy in order to facilitate easy extensions. Agents can have multiple behaviors active at one time and the system resolves the contribution each represents to the final actions performed. Simple decision trees are used to facilitate reactive behavior based on environmental sensors.

The AI.Implant-controlled militia in the Mogadishu scenario consists of two different behavior sets. The leader behavior set follows a pre-specified patrol path through the environment circling the crash site. The follower behavior set uses AI.Implant's built-in crowd flocking behavior to move with the leader around the patrol path. Only the leader agent has any knowledge of the patrol path so the path can be modified without any changes to the follower agents.

AI.Implant was an interesting choice for driving the opposing militia for several reasons. Using barriers instead of path-finding networks seems to be a promising representation to support agent navigation. This approach eliminates the need to embed navigation queues into the terrain representation which can be a time consuming process. In this case the barrier representation was created by hand, which was equally time consuming, however AI.Implant should eventually be tied closely enough to Unreal Tournament that it could automatically gather the barrier information. In addition, AI.Implant's multiple, parallel behavior system lends itself to simple, goal-based behavior by representing each goal as a behavior and activating each behavior via decision tree at the appropriate time.

5. Results and Future Work

The main purposes of this effort was to explore how to integrate pre-existing human behavior models into simulation environments and game engines in order to enhance the realism of the characters in different roles. This was accomplished by building a standard HBM interface to a commercial game engine and using three different "off the shelf" human behavior models to populate a military scenario. These three models, drawn from both academic and commercial developers, focused on three different areas of human behavior. The Soar model focuses on goal-directed behavior based on knowledge of tactics and doctrine. PMFserv is a physiologically/stress constrained model of decision-making based on emotional utility. AI.Implant takes a composite behavior-based approach to individual and crowd navigation.

The primary result of this effort is the Black Hawk Down scenario itself. Unlike the heavily scripted play of most commercial games, this scenario is very dynamic and can play out in a wide variety of different ways. This is primarily due to the autonomy and wide range of behavior supported by the three human behavior models. This scenario demonstrates the key contribution of this research; the integration of three HBMs into a single virtual environment through variations on a common interface architecture. Because pre-existing HBMs were employed, developing the behaviors for each character was fairly quick and efficient. As a result, most of the effort involved the development of the interface architecture and the extensions to this for each HBM.

Since the interface architecture was based on a previously developed Soar interface, the interface between Soar and Unreal Tournament was the easiest to accomplish. Soar communicated with Unreal Tournament through a combination of UnrealScript code and C++ code taking advantage of the SGIO system.

The AI.Implant interface was significantly more difficult. The AI.Implant SDK was still under development as the interface to Unreal Tournament was being developed. Fortunately, the SDK was sufficiently mature that the same combination of UnrealScript and C++ allowed us to interface AI.Implant with the game engine.

From the PMFserv perspective, we interfaced with Unreal Tournament via the Microsoft COM interchange method. This interchange protocol performed quite well in practice and did not lead to latency of note in the responses of the NPCs. What follows is a summary of the observed pros and cons of this approach.

PROS of the Interchange Architecture

- Uses a standardized software approach that's widely available on all PCs
- Microsoft's COM layer is straightforward, well documented, and rapid to implement
- Runtime performance was excellent – no noticeable latency between events and responses

CONS of the Interchange Architecture

- COM is a Microsoft artifact, and not a universal standard
- Limits portability to platforms using Windows
- COM approach doesn't solve many interchange issues, but pushes most of the interchange responsibility onto other layers
- Since there are no naming conventions or translation standards in general for human behavior models, the resulting Custom Unreal Script was difficult to create and grew to about 1,000 lines of code, code that is not itself very reusable.

Due to time constraints, most of the custom UnrealScript had to be dedicated to nuances of this interchange environment and more specifically to this exact scenario. Given a few more such interchanges one might observe some useful patterns and

conventions might emerge that would further help the field of human behavior model interchange. Certainly that is a worthy goal and a trend that should be encouraged in the field as more M&S environments attempt to benefit from existing and complementary types of human behavior models.

Acknowledgements: The authors would like to thank Joe Toth, Michael Young and John Tyler for their guidance. The authors would like to thank John Laird and Robert Marinier at the University of Michigan for their assistance with the Soar/Unreal Tournament interface and the Soar architecture. The authors would like to thank Dr. Paul Kruszewski at BioGraphic Technologies for his assistance and the generous donation of an AI.Implant license for this project. This paper was developed with funds of the Defense Modeling and Simulation Organization under contract number 53-0820-0139 and funds of the Department of the Army under contract number DAAD 19-99-D-0046. Any opinions, findings and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the Defense Modeling and Simulation Organization or the Department of the Army.

3. References

[1] Anderson, J. R. (1993). *Rules of the Mind*. Erlbaum, Hillsdale, NJ.

[2] BioGraphic Technologies (2000). AI.Implant. Retrieved February 13, 2004, from http://www.ai-implant.com/.

[3] Bowden, M. (1999). *Black Hawk Down*. Atlantic Monthly Press, New York, NY.

[4] Damasio, A.R. (1994): *Descartes' Error – Emotion, Reason, and the Human Brain*, Avon, New York 1994.

[5] Epic Games (2003). Unreal Tournament. Retrieved February 13, 2004, from http://unrealtournament.com/.

[6] Janis, I.L. and Mann, L. (1997): Decision Making: A Psychological Analysis of Conflict, Choice and Commitment, The Free Press, New York 1977.

[7] Klein, G.A., Orasanu, J., Calderwood, R., and Zsambok, C.E. (1993): Decision Making in Action: Models and Methods, Ablex, Norwood, NJ 1993.

[8] Laird, J.E., Newell, A. and Rosenbloom, P.S. (1987). Soar: An architecture for general intelligence. Artificial Intelligence 33:1-64.

[9] Lazarus, R. (1991): *Emotion and Adaptation, Oxford* University Press, Oxford 1991.

[10] Sentry Studios (2003). Infiltration: This is as real as it gets. Retrieved February 13, 2004, from http://infiltration.sentrystudios.net/.

[11] Silverman, B.G., Johns, M., et al. (2002), "Constructing Virtual Asymmetric Opponents from Data and Models in the Literature: Case of Crowd Rioting ",11th Conf. On Computer Generated Forces and Behavioral Representation, SISO, May. 2002.

[12] Silverman, B.G. (2003), "Human Performance Simulation" to appear in Ness, J.W., Ritzer, D.R., & Tepe, V. (Eds.) (2003) *Metrics and methods in human performance research toward individual and small unit simulation.*

[13] Sweeney, T. (1998). UnrealScript Language Reference. Retrieved February 13, 2004, from http://unreal.epicgames.com/UnrealScript.htm.

[14] Wray, R.E., Laird, J.E., Nuxoll, A. and Jones, R.M. (2002). Intelligent opponents for virtual reality trainers. In *Proceedings of the Interservice/Industry Training, Simulation and Education Conference (I/ITSEC) 2002*, Orlando, FL, Dec 2002.

Author Biographies

MICHAEL VAN LENT is a Project Leader at the USC Institute for Creative Technologies. Dr. van Lent received his Ph.D. from the University of Michigan in 2000.

BARRY G. SILVERMAN is a Professor in Engineering, Medicine, and Wharton, Director of ACASA, and a core faculty member of several other research centers at the University of Pennsylvania.

RYAN MCALINDEN is a software developer at the USC Institute for Creative Technologies, where he investigates commercial game engines and their applicability to various research environments.

KEVIN O'BRIEN is a graduate student at the University of Pennsylvania.

JASON CORNWELL is a graduate student at the University of Pennsylvania.

PAUL BROBST is a graduate student in Computer Science at the University of Southern California.