A Design Methodology for Domain Independent Computer Generated Forces

David A. Van Veldhuizen and Larry J Hutson
Department of Electrical and Computer Engineering
Air Force Institute of Technology
Wright-Patterson AFB, OH 45433-7765
{dvanveld, lhutson}@afit.af.mil

Abstract

The "synthetic battlespace", used by the military as a training arena, exhibits several deficiencies. These limit battlespace combatants from interacting in a realistic fashion. A promising method to correct these deficiencies is by using "intelligent" computer generated actors who display realistic behavior. This paper explores a generic architecture for these actors, and gives a design extending the architecture for use in our research effort. It also discusses a method for incorporating human behavior and skills into the computer actor's actions.

Introduction

To make better use of limited Department of Defense training funds, the military has turned to Distributed Interactive Simulation (DIS) technology, using computerized simulations as the training arena. This "synthetic battlespace", although successful in certain areas, exhibits several deficiencies resulting from participant's inconsistencies.

A major problem is that the threat environment (e.g., anti-aircraft artillery or aircraft) currently presented to all simulation participants is not consistent. This prevents battlespace combatants from interacting in a realistic fashion because they sense the environment at different fidelity levels. For example, simulators of combat force aircraft have native threat generation systems for stand-alone and networked testing. This heterogeneous capability often results in compatibility problems between simulators when adding new weapons or theaters of operation. A contributing problem is that many battlespace systems representing the same assets are implemented in a non-uniform manner. Many simulation developers created unique systems and implement modeling decisions supporting their users in specific scenarios. These differing systems are difficult to coordinate on a distributed network because they can't perform at the same fidelity level. This creates asset synchronization issues.

Lastly, a lack of actors, whether human or "intelligent" computer-generated, exists within many military simulations. Humans are able to rapidly identify and defeat current computer-generated entities using unrealistic tactics (i.e., "gaming the system"), reinforcing potentially fatal behaviors. State of the art still falls short of presenting believable agents in many simulations.

Taken together, these issues lead to great expense, uncoordinated threat development, and training inconsistencies between participants. A promising technology to fill this gap is Computer Generated Forces (CGFs), which are computer controlled actors displaying intelligent behavior. Unlimited numbers of simulation CGFs could be created on demand with minimal cost, and can be standardized in their presentation of and reaction to threats throughout the battlespace.

Our current research focuses on creating and implementing a design methodology for the Distributed Mission Training Integrated Threat Environment (DMTITE) project. The project's goals are identifying requirements for a distributed threat environment, evaluating similar efforts, and building a demonstration unit providing a distributed threat environment in the synthetic battlespace. We envision a "black box" where a generic design is used to construct varied CGF types, each displaying appropriate behaviors. This design allows users to specify any type of land-, air-, or sea-based CGFs for use in training simulations; however, initial development is limited to aircraft, surface-to-air missile units, radars, and anti-aircraft artillery.

This paper explores a generic CGF architecture, extends it for use in this application, discusses key issues, and suggests implementations for key architecture components. CGF characteristics, architecture design methodology, and human-like behavior are discussed in the following sections.

Domain Independent CGF Characteristics

Systems developed independently of one another make standardization difficult, if not impossible. Any standardized domain independent CGF approach must then consider certain requirements. This section reviews these requirements, discusses projects relevant

to the current work, and motivation driving the development effort.

CGF Requirements

Any CGF must satisfy a core of architecture, behavior, and knowledge-base requirements. Banks, et al. (Banks, Stytz, & Santos 1996), have suggested such a baseline set, identifying the following requirements as providing a well-engineered system development foundation: *modifiability*, *high fidelity*, *adaptable decision-making*, and *learning*. Briefly, their requirements are as follows. The ability to easily change CGF capabilities must exist. This includes knowledge-base expansion and flexibility of the underlying software architecture. Realistic CGF behavior is required; the CGF's physical and cognitive representations must display believable actions. CGF decision processes must deal with available information (or a lack thereof) and differing performance requirements. Finally, CGFs must employ a learning mechanism to improve their decision making processes to avoid the predictable actions currently allowing their identification as computer-generated actors.

Review of Relevant Efforts

Several efforts currently explore CGF creation and behavior. However, many do not address the entire set of core requirements above, but only develop a limited subset. Two efforts incorporate some aspect of the above and warrant mention here.

The Automated Wingman is an experimental CGF assembled at the Air Force Institute of Technology's (AFIT's) Virtual Environments Laboratory. It was developed as a collaborative effort showing the implementation of a domain independent framework and architecture for knowledge based CGFs (Benslay 1996; Zurita 1996). Its strengths are the identification and partitioning of domain knowledge into separate knowledge bases; and development and implementation of an asynchronous game-tree driven by fuzzy logic. Its shortfalls lie mainly in the incomplete population and partitioning of its knowledge bases, and inference mechanisms to reason over them.

TacAir-Soar is an intelligent automated agent for tactical air simulation. The goal of this research is to build intelligent, automated agents for tactical air simulation, creating automated pilots whose behavior in simulated battlefields is nearly indistinguishable from human pilots (Tambe *et al.* 1995). The strengths of TacAir-Soar are its performance in beyond-visual-range air combat scenarios, the ability to retain "memory" of actions performed and their explanation, and communication and coordination among multiple agents. However, it emulates the human cognitive *process*, which can vary across problem domains. Its implementation as a specialized (fighter) CGF complicates design of other CGF types where the cognitive process and resultant actions may be different, such as an underground command post. Finally, it does not handle uncertainty in its decision making process.

Requirement for New Approach

The CGFs we generate are designed for use in Distributed Virtual Environments (DVEs). In a DVE, participants must move correctly, behave believably, and approximate the complexity of the real world. CGFs exhibiting human behaviors are critical; their perceived behavior must be accurate and real enough so that the other CGFs and human participants in the DVE respond to the CGF as if it were a human-controlled actor (Stytz 1996).

More research in this area is required. The large and expanding numbers of interconnected simulations in use by the Department of Defense, coupled with the increasing reliance on these simulations for training purposes, mandate the most realistic representation of CGF behavior possible to maximize the training experience. This translates to civilian applications as well, such as medicine or air-traffic-control training. Lessons learned from these efforts are crucial in expanding the performance envelope of the CGFs we propose.

CGF Architecture and Design

A generic CGF architecture is desirable because it provides a general template for instantiation of several different CGF types, and for the variation of abilities within a single type. Santos, et al. (Santos, Banks, & Stytz 1996), proposed a generic, adaptable CGF architecture which accounted for "variety" in a given type of CGF and supported construction of vastly different CGFs. We build on their efforts, extending the architecture to that shown in Figure 1.

The key to our model lies in the separation of physical and cognitive processes, which allows several advantages. First, it eliminates dependencies between the two sets of processes; changes to one do not necessarily impact the other. Second, it shifts the focus from what knowledge is available to how the knowledge decomposes. This allows us to quickly identify areas in which the CGF will be faced with uncertainty. Finally, it allows instantiated CGFs to display a wide variety of abilities and skills. These advantages are critical if we want to eliminate the unnatural advantage human actors currently enjoy in the virtual battlespace. Current CGF performance still falls short of these goals.

The DMTITE system-level architecture is shown in Figure 2. Each DMTITE CGF sends and receives world state information via a shared database that maintains the most recent world state information received from the DIS World State Manager (WSM). Instead of receiving a constant stream of updated information, the WSM approximates positions of other entities in the DVE using local references and dead reckoning algorithms. Information is updated only as necessary, such as when an entity changes velocity or

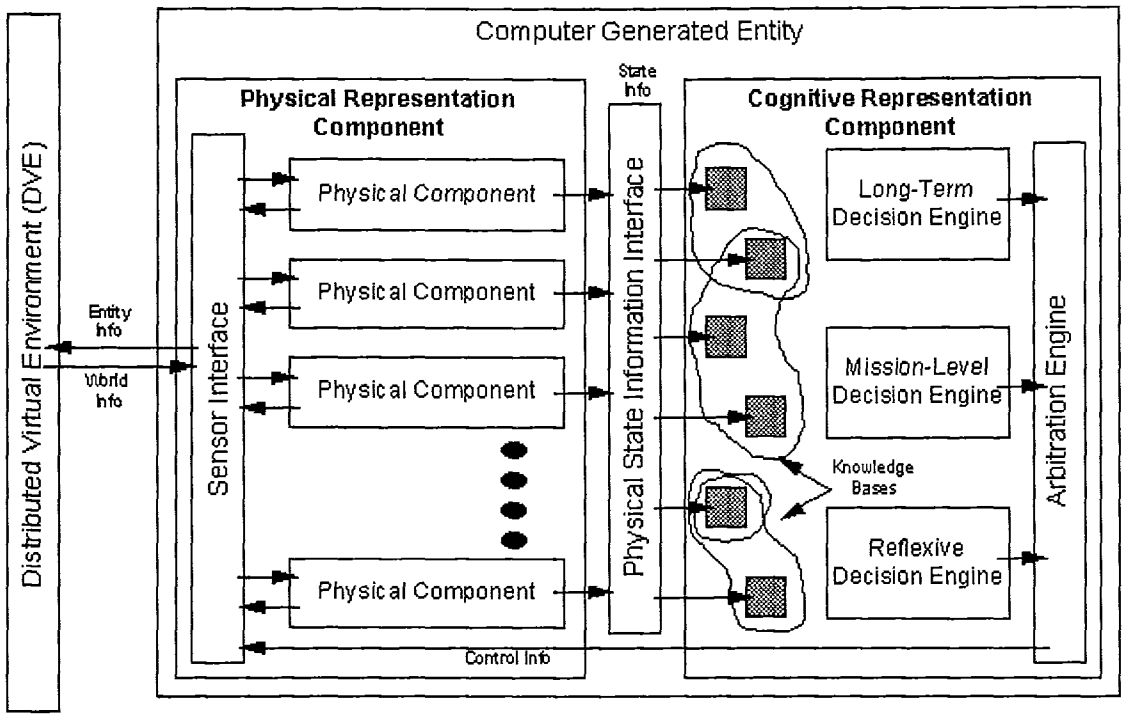

Figure 1: **Design Methodology for Generic CGF Architecture**

heading. Each local WSM is the sole provider of information to other WSMs in the DVE.

Our extended CGF architecture supports a domain-independent approach for implementing CGFs. The Physical Representation Component (PRC) is comprised strictly of the physical attributes and properties of the CGF; those characteristics that are subject to physical laws. This representation supports the decomposition of physical characteristics into objects and operators. The modularity of physical components allows for easy addition, deletion, and modification of these components without requiring corresponding changes to the cognitive representation.

The PRC also acts as the CGF's only gateway to the outside world. Since the WSM receives state information for *all* participants in the DVE, the PRC uses a sensor interface to filter out all information not applicable or desired by the CGF it supports. Data for a particular CGF is passed to the corresponding physical component(s); data not applicable to that CGF is ignored. This eliminates the need for each physical component to determine whether received data is to be processed.

The Cognitive Representation Component (CRC) includes CGF characteristics that are less physical and more cognitive in nature. It is responsible for emulating the *outcomes* of the human cognitive process (decisions), but does not require a model of the cognitive process be implemented. However, the CRC is responsible for far more than just decision-making. In addition to decision engines, it also contains the CGF's goals, entity profile, and knowledge bases. The entity profile specifies the CGF's skill level and other combat performance factors, which are discussed in more detail in the next section.

The knowledge bases are separate from the decision engines and are sub-divided into smaller, more atomic knowledge bases. For example, an "evasive flight" knowledge base might decompose into "known maneuvers", "opposing aircraft capabilities", and "terrain information". This results in tightly focused knowledge bases visible to *any* decision engine making use of the knowledge. These knowledge bases require state information from physical components, thus, an interface is constructed to act as an information gateway between the physical and cognitive representations. While this supports indirect access to physical state information, changes to the physical representation impact *only* the interface, not the knowledge bases. The interface also informs the knowledges bases as to the type of information it provides. Finally, each knowledge base is designed to handle "good", "corrupted", and missing information. This approach yields modular knowledge bases containing unique information.

Our decision engines have specific responsibilities and "areas of concern". The Long-Term Decision Engine reasons over strategic plans and goals of concern to the CGF. For example, it allows the CGF to iden-

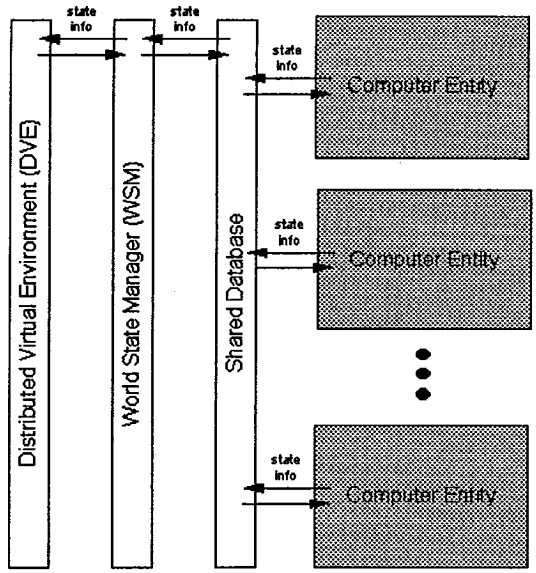

Figure 2: **DMTITE System-Level Architecture**

tify "targets of opportunity" that advance the ultimate mission goals simulation, but were not part of the CGF's original tasking. The Mission-Level Decision Engine reasons over moment-to-moment and short-term actions of the CGF. Such actions include initiating a bomb drop or attacking enemy aircraft. The Reflexive Decision Engine directs survival (fight or flight) and other reflex actions performed by the CGF. An example of a reflexive action would be a soldier's instinct to drop to the ground when fired upon. The main advantage of this approach is that multiple decision engines scoped to different levels and tasks allow us to fine-tune one set of behaviors with minimal impact on other, unrelated behaviors. A major disadvantage is that each engine will most likely render a different decision for the same situation.

To overcome this situation, our architecture includes an Arbitration Engine, a specialized decision engine determining which decision is ultimately selected for action. However, it considers not only the decisions rendered by the other engines, but each decision's merit (relative to the others) and the current state of the entity profile. For example, the flight (in "fight or flight") decision is unlikely to be acted upon by a CGF representing a highly-trained, highly-disciplined combat entitity; such a decision has relatively low merit to the CGF in most cases. However, a CGF representing a poorly-trained, poorly-disciplined entity is *more* likely to act upon such a decision, perceiving it to have relatively high merit. When multiple decisions have relatively similar merit and are similar in nature, the Arbitration Engine combines them into a new, composite decision. For example, in response to a heat-seeking missile, the Mission-Level Decision Engine decides to "deploy flares and execute a *slight* turn to the left". However, the Reflexive Decision Engine decides to "deploy flares and execute a *sharp* turn to the left". The Arbitration Engine combines these two decisions and tells the CGF to "deploy flares and execute a *moderate* turn to the left". The result is an action that, to an outside observer, emulates momentary indecisiveness on the part of the CGF–a very human-like behavior.

When the Arbitration Engine receives multiple decisions of relatively similar merit but dissimilar nature, it relies on the entity profile to determine which action to select. For example, if the entity's profile reflects low confidence and an actual inability to cope with the current situation, the Arbitration Engine may choose to act upon the reflexive decision (possibly to the point of abandoning the mission). On the other hand, if the profile reflects a high degree of confidence but an actual inability to cope with the current situation, the Arbitration Engine may choose to act upon the other decision. This deliberate action possibly places the CGF in significant, but otherwise avoidable, danger. Other entity profile states may cause the Arbitration Engine to ignore all of the decisions offered (emulating an overwhelmed or "shell-shocked" entity), or demand new decsions from the other decision engines. In short, the Arbitration Engine ultimately decides the course of action a CGF takes while attempting to incorporate human-like behaviors in those actions.

Incorporating Human-Like Behavior: The Entity Profile

Human opponents rapidly identify and defeat CGFs acting upon random or scripted decisions, which are not decisions based on combinations of world state, mission success, or self-preservation. To prevent this unnatural advantage, CGF behavior must be modeled upon actual human behavior. This involves identifying and incorporating concepts such as doctrine, tactics, training, and information into the behavior of the CGF. This raises two key concerns: *unpredictability* and *certifiability* (Banks, Stytz, & Santos 1996). Unpredictability does not imply random behavior; instead, it is concerned with the lack of exploitable patterns in CGF behaviors. Certifiability is the concern that CGF behaviors be measurable against, and comparable with, those displayed by humans in similar situations. The latter issue is addressed primarily through the use of multiple CGF skill levels.

Multiple skill levels allow CGFs to exhibit behaviors of "combat-hardened" individuals, those being exposed to fire for the first time, and the range of experience in between. Novices are more likely to display limited maneuvering tactics or miss their targets, whereas veterans display opposite behaviors. In a general sense, we are concerned with two types of skill: weapons and maneuver. Weapons skill is the effectiveness with which the weapon systems available are used. Maneuver skill encompasses how effectively the entity can maneuver

itself; for example, how well it flies an aircraft. These attributes reflect not only training, but also the inherent skill possessed by the entity. The values of these attributes will often remain constant over the course of a simulation, but are subject to change. By separating the representation of these skills, we allow a greater number of skill levels and resultant behavior than if they were combined.

However, human behaviors in combat aren't strictly a measure of skill level; they also reflect other, less tangible concepts. In fact, psychologists have identified four areas directly affecting combat performance: *psychological*, *social*, *instrumental*, and *physical* (Shalit 1988). These areas are too semantically vague to be used effectively by simulation planners. We use the following domain-independent attributes to represent these areas instead:

1. **Motivation**. Eagerness to engage an enemy. This attribute reflects concepts such as "sense of duty", "commitment", and "belief in a cause". The value this attribute takes on may change during the course of a simulation.
2. **Social Identification**. The degree to which the entity "belongs" to the society for which it fights. This attribute reflects concepts such as "patriotism". The value of this attribute is assigned at the start of a simulation and remains constant throughout the simulation.
3. **Perception of Enemy**. How the entity perceives itself in comparison to its opponent. Initially, this attribute reflects the effects of propaganda, but as a simulation progresses, will reflect the actual *qualitative* differences between the entity and its opponent.
4. **Means**. The quantity and quality of resources available to the entity to complete its mission. This attribute is strictly internal and its original value must be derived; users may not define an initial value. Obviously, this attribute will change over the course of a simulation.
5. **Physical Condition**. The relative position of the entity to an ideal, mission-ready condition. This attribute reflects concepts such as "fatigue", "injuries", and "damage". The value assigned to this attribute is dynamic over the course of a simulation.

These attributes, in conjunction with the skill attributes discussed previously, comprise the *entity profile*. Since these attributes are qualitative in nature, they can be implemented as *term sets* comprised of one or more *linguistic variables* (Banks, Stytz, & Santos 1996). While the Arbitration Engine is most likely the primary user of the entity profile, other components can incorporate this information as well. The entity profile can be used by the knowledge bases to "corrupt" world state information resulting in a desired behavior. For example, a CGF with poor weapons skill may "misread" its targeting information, selecting a short-range missile to target an out of range enemy . Decision engines can use the entity profile while determining what course of action to recommend; an entity with low motivation would be less likely to aggressively close in on an enemy, and the decision engine should take this into consideration.

Conclusion

The soundness of this design will become apparent as we begin implementation of varied CGF types for the DMTITE project. We will evaluate and incorporate lessons learned from construction of the first CGFs built into the later efforts. We plan on implementing computer-generated fighter, anti-aircraft artillery, and radar posts first, as many of the physical models and domain knowledge necessary exists and is well-documented. This allows rapid exploration and evaluation of our methodology.

We've shown an architecture and design methodology that is truly domain-independent, freeing the designer from a particular computer simulation or host platform. It gives a framework to build CGFs of any desired type and to exhibit varying levels of skilled behavior. These factors allow CGFs to smoothly integrate into simulations adhering to DIS standards. This reduces the expense of adding simulation entities, helps coordinate threat development, and removes inconsistencies between simulation participants.

References

Banks, S. B.; Stytz, M. R.; and Santos, E. 1996. Requirements for intelligent aircraft entities in distributed environments. In *18th Interservice/Industry Training Systems and Education Conference.*

Benslay, Jr., J. 1996. A domain independent framework for developing knowledge based computer generated forces. Master's thesis, Air Force Institute of Technology.

Santos, Jr., E.; Banks, S. B.; and Stytz, M. R. 1996. Engineering intelligent computer generated forces. Technical report, Air Force Institute of Technology.

Shalit, B. 1988. *Psychology of Conflict and Combat.* Praeger.

Stytz, M. R. 1996. Distributed virtual environments. *IEEE Computer Graphics and Applications* 19–31.

Tambe, M.; Johnson, W. L.; Jones, R. M.; Koss, F.; Laird, J. E.; Rosenbloom, P. S.; and Schwamb, K. 1995. Intelligent agents for interactive simulation environments. *AI Magazine* 15–37.

Zurita, V. B. 1996. A software architecture for computer generated forces in complex distributed virtual environments. Master's thesis, Air Force Institute of Technology.