1-1-1996

Terrain Representation And Reasoning In Computer Generated Forces : A Survey Of Computer Generated Forces Systems And How They Represent And Reason About Terrain

Mikel D. Petty

INSTITUTE FOR SIMULATION & TRAINING

Terrain Representation and Reasoning in Computer Generated Forces Version 2

December 11, 1996

Mikel D. Petty

Institute for Simulation and Training
University of Central Florida

University of Central Florida

IST-TR-96-50

T0 $9

Terrain Representation and Reasoning in Computer Generated Forces

A Survey Of Computer Generated Forces Systems
and how they Represent and Reason about Terrain

Mikel D. Petty
Institute for Simulation and Training
3280 Progress Drive, Orlando FL 32826-0544
407-658-5022 mpetty@ist.ucf.edu

Technical Report IST-TR-96-50
Version 2, December 11 1996

Terrain Representation and Reasoning in Computer Generated Forces

Technical Report IST-TR-96-50

Table of Contents (Part 1 of 2)

Table of Contents (Part 2 of 2)

1. Introduction

This section begins with an abstract for the overall document, then presents an informal motivation of Computer Generated Forces, states the purpose and structure of the document, and explains some essential background concepts.

1.1 Abstract

Distributed Interactive Simulation is an architecture for building large-scale simulation systems from a set of independent simulator nodes communicating via a common network protocol. DIS is most often used to create a simulated battlefield for military training. DIS simulations are real-time.

Computer Generated Forces (CGF) systems control multiple autonomous battlefield entities in a DIS simulation using computer equipment and software rather than humans in simulators. CGF entities serve as both enemy forces and supplemental friendly forces in a DIS simulation run. A number of different CGF systems have been implemented. They have been used for both research and training. Three important CGF systems are the IST CGF Testbed, the Loral ADS ModSAF system, and the CCTT SAF.

Specifying and generating realistic tactical behavior in CGF systems is an ongoing research topic. A number of different approaches, including many from artificial intelligence, have been applied to CGF behavior generation. One commonly used technique is based on finite state machines.

Effective use of terrain is a crucial element of military tactical decision making. In order to realistically simulate military forces, CGF systems must represent the battlefield terrain and reason about it. A variety of different terrain representation formats have been used in CGF systems. Those formats include gridded, polygonal, quadtrees, graphs, and others. Each of the formats has strengths and weaknesses in the context of CGF systems. Some CGF systems use multiple representations of the same terrain in different formats and perform each terrain task on the representation format that best suits that task.

Low-level tactical behavior, in real-world military units and in CGF systems, is highly dependent on the terrain. Therefore terrain reasoning is an essential part of CGF behavior generation. CGF algorithms exist for a number of terrain reasoning tasks, including intervisibility, route planning, and finding cover and concealment. Given the number of different terrain reasoning tasks, different terrain representations, and different algorithmic paradigms, it should be no surprise that there are a wide variety of terrain reasoning algorithms with varying degrees of robustness, performance, and tactical realism. Intervisibility algorithms based on polygon traversal, route planning algorithms based on A* search, and cover-and-concealment-finding algorithms based on geometric analysis of terrain elevation have all been implemented.

1.2 An informal motivation

This survey is intended to be a serious treatise. However, before beginning the scholarly business of defining terms and explaining system capabilities, it may be useful to describe in an informal way two examples of autonomous entities in simulation that will help serve to motivate the survey.

Consider the following scene: a young man sits in front of a personal computer, staring intently at the monitor before him. He grips a joystick, which he moves constantly, sometimes gently and sometimes with abrupt suddenness. He is playing *Wing Commander*, a best-selling space combat simulation game produced by Origin. The program places him in the cockpit of a simulated space fighter ship, equipped with rocket engines, afterburners, navigation systems, shields, energy and projectile blasters, and guided missiles. He must maneuver his fighter around interstellar terrain such as asteroid belts and space stations. His goal is to use his fighter's weapons to destroy enemy fighters and to avoid being destroyed by them.

Through his cockpit window, he can see the enemy fighters. The actions and maneuvers of the enemy fighters are generated by the game software. They turn towards him and fire, attempt to get onto his tail, use afterburners and evasive maneuvers to escape his fire, lock on and fire their guided missiles, and lure him away from the friendly ships the player is protecting. The actions for each enemy fighter are generated based upon the fighter's location relative to the player's fighter, the class or type of the enemy fighter, its damage state, and the combat style and aggressiveness of the particular fictional enemy pilot who ostensibly controls the fighter [Harrison,1992].

The simulation program is presented as entertainment, and it has succeeded. The player is completely engaged in the simulation. His entire attention is focused on the simulated control panel of his fighter and the violently maneuvering enemy fighters visible through the cockpit windows.

Now consider a second scene: four U.S. Army soldiers sit at the controls of a training simulator. The simulator is about the size of a garden shed; from the outside, it appears to be a connected set of computers, monitors, and large green fiberglass enclosures. From the inside, the simulator is a simplified but easily recognizable re-creation of the interior of a M1A1 Abrams tank.

The four soldiers are the M1A1's crew. They manipulate the simulator's controls as they would in an actual tank, driving the simulated tank through a simulated battlefield which they can view through the vision blocks of their tank. A computer image generator and monitor for each of the vision blocks shows a view of the battlefield as it would be seen from that location. The battlefield terrain is comprised of the terrain surface as well as features such as treelines, roads, bridges, buildings, and canopies; it is constructed from polygons.

A second crew is at the controls of another M1A1 simulator. That simulator may be adjacent to the first, or it may be hundreds of miles away. However, the two are connected by a computer

network, and in the simulated battlefield the second tank is following the first, about 30 meters behind.

The commander of the lead M1A1 is warily surveying the terrain from his vantage point in the cupola, atop the turret, searching for the enemy tanks that are likely to be nearby. Suddenly, as the tank crests a ridge, he spots two enemy tanks emerging from behind a treeline about 1500 meters away. The enemy tanks are generated in the battlefield by another simulator node, attached to the M1A1 simulator via the network. However, they are not controlled by human crews; rather, computer software is generating their behavior, and that of many other vehicles in the simulated battlefield.

The tank commander radios the commander of the second M1A1, who cannot yet see the enemy tanks, and warns him of the threat. Then, over the simulator's intercom the tank commander orders the driver to turn the simulated M1A1 to face its frontal armor towards the enemy tanks and to stop so as to provide the gunner an easier firing problem. The commander's feeling of urgency is easily heard in his voice as he tells the gunner where the enemy tanks are, which one to engage first, and what ammunition to use. As quickly as his skills allow, the gunner rotates the M1A1's turret and elevates the main gun to align the aiming reticle with the first target. In quick succession he thumbs the laser rangefinder button and squeezes the main gun trigger; the target bursts into flames.

The commander urges him to immediately engage the second tank, but it is too late. The driver of the second M1A1, in his haste to reach a location from which to fire, has crested the ridge right behind the stopped lead tank and collides with it. Both of the simulated tanks are abruptly jostled by the collision, and some damage is suffered by both. Before either crew can reorient themselves, the enemy tank sights the lead M1A1, turns towards it, and stops. Its turret swings around and the enemy tank fires. The sound system of the M1A1 simulator produces an unpleasantly loud crashing sound, and the screens of the simulator turn black; the lead M1A1 has been destroyed by the enemy tank. The tank commander pounds his controls in frustration.

The first scene described a simulation program used for amusement, the second a program used for the deadly serious job of training U.S. Army soldiers, yet the two scenes have at least two crucial elements in common. First, they both succeed in creating an environment with enough intensity and urgency to draw their users entirely into the simulated world. Second, they both include autonomous entities that oppose the simulation users, attempting to thwart and even destroy (in simulation!) those users. In a real sense, the first characteristic, i.e. the simulation intensity, and the resulting usefulness of the simulation system, is produced by the second characteristic, the autonomous opposition entities.

In both cases, the behavior of the automated opposing entities is produced algorithmically, with a computer hardware and software system; such a system is referred to as a Computer Generated Forces (CGF) system. In both cases the CGF system that generates the behavior must reason about the simulated terrain of the battlefield in order to generate realistic behavior.

1.3 Purpose and structure of this document

This document is a survey of computer systems used to produce realistic or intelligent behavior by autonomous entities in simulation systems, i.e. of CGF systems. In particular, it will be concerned with the data structures used by CGF systems to represent terrain and the algorithmic approaches used by those systems to reason about terrain.

The reader of this document is assumed to possess a general familiarity with computer science, including data structures, algorithm design, simulation, and artificial intelligence. A detailed knowledge of simulation or artificial intelligence is not assumed, as the particular terms and ideas in those areas will be explained as needed.

The next subsection defines and explains a number of background terms and concepts that are conceptual prerequisites to understanding the issues of behavior generation and terrain reasoning for CGF systems. They include real-time simulation, networked virtual simulation, and DIS. Following that, the three main sections of this document survey CGF systems, terrain representation in CGF, and terrain reasoning in CGF. In particular, section 2 first presents a structural overview of a typical generic CGF system and discusses CGF system architecture, and a number of existing CGF systems. A few of the most important CGF systems (in the author's opinion) are explained in some detail, and many others are identified and discussed more briefly.

Section 3 moves from CGF systems in general to terrain representation in CGF systems. First, background concepts of terrain representation are introduced. Then terrain representation formats used by existing CGF systems and other related systems are surveyed and explained. The explanations discuss the data structures for each of the representational formats and compare their strengths and weaknesses in the CGF context.

Finally, in section 4 CGF terrain reasoning algorithms are examined. In order to organize the exposition, special attention will be given to terrain reasoning algorithms for three crucial CGF terrain reasoning tasks: route planning, intervisibility, and finding cover and concealment. In each case the task is defined and then the existing CGF terrain reasoning algorithms for the task are presented.

Figure 1.1 is a graphical representation of the subject matter domain of this document.

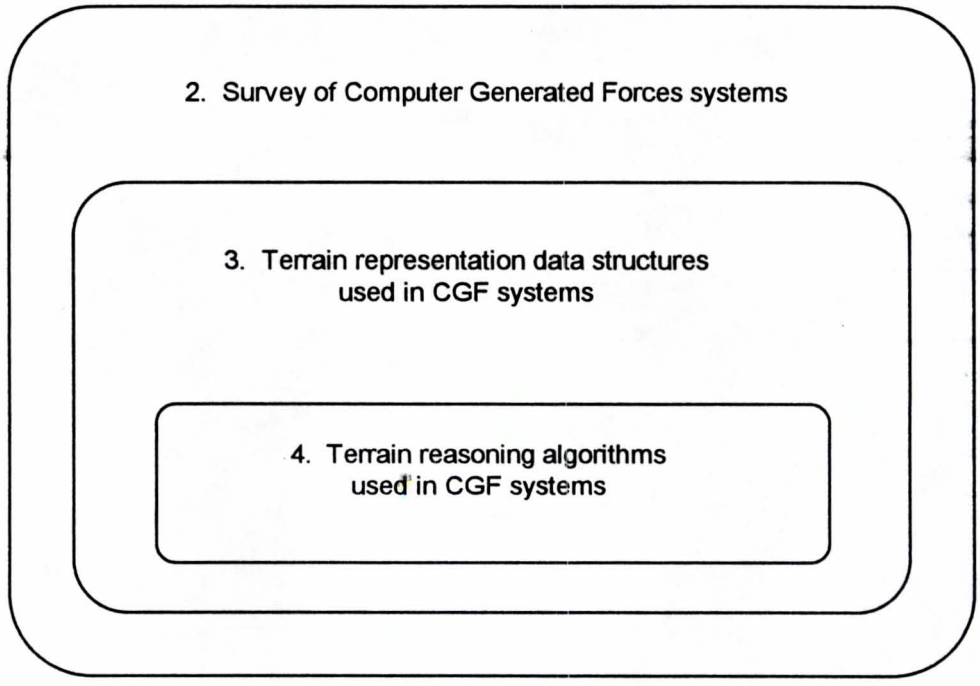


Figure 1.1 Subject domain of the sections of this document.

1.4 Background

This subsection defines background terms and concepts that will be needed in order to discuss CGF systems and terrain reasoning algorithms.

1.4.1 Analytical and descriptive models

For the purposes of this survey, a *model* is a description, generally mathematical, of some existing or potential real-world object, process, or system. A *simulation* is a realization or implementation of a model (or more than one), generally done using computer software and hardware. An execution of a simulation is an *exercise*, *run*, or *trial*, depending on the context. While a simulation is executing, the events, activities, and state transitions that occur within the model have a correspondence, defined by the model, with events and activities that may occur in the modeled system. A set of connected or linked simulations operating cooperatively is a *simulation system*; an individual simulation within a simulation system is a *simulation (or simulator) node*. Models implemented as simulations are often symbolic, in that symbols within the model are used to represent objects, relationships, actions, and processes. Within the category of symbolic simulations, [Kreutzer,1986] identifies two types of models. *Analytical* models are based on some strong mathematical theory and are deductive. They allow the use of mathematical methods to find a desired state of the modeled system and provide general solutions to classes of systems. However, analytical models often depend on extensive simplifying assumptions to make them mathematically tractable.

In contrast, *descriptive* models symbolically represent the possible states of the modeled system (i.e. the problem space), without providing any analytical methods for finding particular states of interest. Because they do not need to be amenable to mathematical analysis, the system being simulated can be modeled in much more detail. Using a descriptive simulation (i.e. an implementation of a descriptive model) is an inductive experiment.

1.4.2 Real-time simulation

The simulations to be considered in this survey all fall into the descriptive category. However, this categorization is not narrow enough. *Time-stepped* simulations model time by advancing the simulation's internal clock a fixed interval and determining what events, if any, have occurred during that interval. *Event-driven* simulations instead determine when the next event will occur and advance the simulation clock to that time. This survey will focus on time-stepped simulations where the time interval is set small enough to produce a quasi-continuous simulation (as defined in [Kreutzer,1986]). Furthermore, the topic of interest will primarily be simulations where the time-slice advances of the simulation clock occur at a rate that causes the simulation clock to match the real-world clock of the simulation's human user; that is, events in the simulation occur at the same speed as the modeled events do in the system being modeled. Such a simulation is usually called *real-time*.

The intended use of a simulation determines whether it will run in real-time, faster than real-time, or slower than real-time. Faster than real-time simulations compress time, so that a future state of

a system can be found in less time than the modeled system would take to reach that state. This permits many different system futures to be produced and analyzed. Slower than real-time simulations stretch time, so that experimenters may examine in detail simulated events that may occur too quickly to study in real-time.

The primary reason for a simulation to run in real-time is to realistically model the passage of time for the user of such a simulation. This survey is considering simulations used primarily for training, where the user(s) learns or improves some skill set as a result of interacting with the simulation; such a change is called *training effect*. If the user is interacting with or participating in the execution of the simulation, it is generally assumed that the maximum value of that participation, i.e. the maximum training effect, will be derived from the user experiencing the simulated events at the same rate he or she would experience the events in the system being modeled.

It is possible, of course, that training effect may in fact be maximized by having simulated time pass faster or slower than real-time. [Guckenberger,1992] describes an experiment in which trainees who were trained in a tank gunnery simulation with time progressing at an accelerated or "above real-time" rate showed a greater training effect than those trained at real-time. Nevertheless, the generally accepted practice is that the best training environment is real-time until proven otherwise for a specific application. The CGF systems to be surveyed later all are intended to operate at real-time.

1.4.3 Networked virtual simulation and DIS

The combination of computer simulation technology and computer networking technology creates a wide range of new simulation architectures. Networked computers running simulations may share processing or data access workload among nodes, allow specialized hardware architectures to perform specific simulation computations for which they are suited, or facilitate simultaneous use of a simulation by multiple users at remote sites. [Goldiez,1995] reviews the history of networked simulation.

One particular networked simulation architecture provides the context for many of the simulations being considered in this survey. Distributed Interactive Simulation (DIS) is an architecture for building large-scale simulation systems from a set of independent simulator nodes communicating via a common network protocol [DIS,1994] [Loper,1995b]. The simulator nodes each independently simulate the activities of one (or more) entities in the simulated environment, and report their attributes and actions of interest via the network to other simulator nodes. In a typical DIS simulation system, the simulated entities coexist in a common environment, (e.g. a terrain database) and can interact in real-time by exchanging network packets [Loper,1991]. An important characteristic of DIS simulation systems is that they are real-time; events in the simulation systems occur at the same rate as their real-world counterparts.

DIS systems are being used for [DIS,1994]:

1. Training in a realistic and large scale synthetic environment
2. Planning and rehearsal of actual military missions
3. Development of tactical and operational doctrine
4. Testing of new weapons systems early in their development cycle.

Because it is distributed and interactive, DIS exercises allow "testing of group-level operations or procedures that require cooperation" [Papelis,1994].

The ARPA/US Army SIMNET system is the prototypical example of a DIS-style simulation. SIMNET is used to train tank and vehicle crews in cooperative and team tactics. (The SIMNET literature is extensive; [Thorpe,1987], [Nelms,1988], [Pope,1989], and [Cosby,1995] are good examples). In SIMNET, the simulator nodes typically represent single vehicles, such as tanks or armored personnel carriers. SIMNET simulator nodes are substantial devices consisting of a simulation computer, a computer image generator, and a physical re-creation of the vehicle interior; they are operated by three or four human trainees. During the execution of a scenario, each simulator node's simulation computer continuously tracks the location of the vehicle relative to a terrain database common to all vehicles in the scenario. The trainees maneuver their vehicle over the terrain and interact with (e.g. fire their weapons at) other vehicles. All simulator nodes in the simulation are linked by the SIMNET network, which carries the packets needed to mediate inter-vehicle interaction.

By exchanging these packets, actions taken by one simulator node are made known to other simulator nodes in real time. Each vehicle broadcasts location report packets, which are used by other vehicles to generate visual images, and fire and impact packets, which are used to signal and adjudicate fire combat.

A DIS-type system depends on two areas of agreement between the networked simulators. The first is the shared "playing field", or simulated environment. All entities in the simulation must have identical or isomorphic simulated environments in which to operate, or events and actions that are valid for one simulator node may be invalid for another. For example, if two tank simulators share terrain databases which are identical except that a bridge present in the first is omitted in the second, when the first tank traverses the bridge it will appear to be floating on air or water to the second.

The second required area of agreement is the network protocol. A DIS protocol specifies the various types and formats of network packets which the simulator nodes will exchange to support the simulation system. Additionally, the protocol defines the precise circumstances under which each packet type should be sent by a simulator node, and the interpretation that should be performed when each packet type is received.

Currently, the DIS protocol standard defines a specific set of fixed format packets, or protocol data units (PDUs). The standard precisely defines the content of each PDU at a bit-by-bit level of detail as well as specifying the circumstances under which each PDU type should be sent and what action should be taken upon its receipt. Recently however, DIS researchers have begun to consider a more flexible DIS protocol composed of a set of optional PDU components that are

assembled as needed for a particular simulation application or exercise [Bouwens,1995]. The different combinations of PDU components are referred to as *profiles*.

DIS exercises (an exercise is a simulation session, or a single run of a scenario) are often recorded for later replay and analysis. The recording is done by reading and storing as a data file the network packets (PDUs) that are sent on the network during the exercise. The PDUs are timestamped as they are recorded so that the playback can be properly timed. A data file of recorded DIS PDUs is often called a *log*, and a utility program to record and playback the PDUs is a *logger*.

An important large-scale use of the DIS concept and standards is the U.S. Army's Close Combat Tactical Training (CCTT) program. CCTT will provide a synthetic environment for training in armor and infantry combat; see [Pope,1995b] for an overview of the CCTT program. [Pope,1995a] discusses how CCTT development has helped drive the evolution of the DIS standard and architecture.

For the most part, DIS-type systems have been used primarily for training, as opposed to more analytical uses such as weapons systems testing and tactics development. The training may be general tactical and team training, where the terrain database used is generic or representative and the trainer sets the scenario [Byers,1988], or it may be mission rehearsal, where a specific military operation is practiced on a terrain database created for that purpose [Branch,1989] [Donovan,1990].

Some non-training military uses of DIS exist; for examples see [Nelms,1988], [Karr,1993b], or [Courtemanche,1994] for descriptions of some analytic uses of SIMNET and DIS. Additionally, attention has been given to extending DIS to non-military applications [Loper,1994] [Loper,1995a]. However, unless otherwise stated this document will focus on military training applications.

1.4.4 Definitions

Before proceeding, several key terms and concepts should be defined. First, the terms *entity* and *unit* will be used throughout this document.

Entity. An entity is most often a single battlefield object; a tank, a helicopter, an airplane, a truck, and a infantry fighting vehicle are all entities. Sometimes a small aggregation of real-world objects that are simulated and controlled as a single simulation object can be an entity; the most common example of this is a squad or fireteam of soldiers. Note that entity has a specific meaning in the context of DIS; an entity is the item or object for which Entity State PDUs are sent, and each entity is assigned a unique identifying number. An entity is the "atom" of simulation in DIS.

Unit. A unit is a military organization, such as a platoon, company, or battalion. Units are collections of entities, usually organized hierarchically into subunits and units.

Simulation systems are often categorized as virtual, constructive, or live. The definitions given here for those terms are adapted from [Franceschini,1995b].

Virtual simulation. Virtual simulations represent each vehicle or infantryman as a distinct entity. A virtual simulation entity may be controlled by a simulator with a human crew or a CGF system. Humans in a simulator perceive the simulation as a type of virtual reality, and virtual simulation is almost always real-time. All necessary state information for each simulated entity is maintained, each entity is capable of independent action, and combat results are computed at the entity level. SIMNET, BDS-D, and CCTT are examples of virtual simulation systems. DIS is an architecture for constructing virtual simulation systems. In this document, we are concerned almost exclusively with virtual simulations and, in particular, DIS.

Constructive simulation. Constructive simulations represent a military unit (e.g. a tank company) as an aggregate without simulating each individual entity (e.g. tank) within the unit. The position, movement speed and direction, status, and composition of an aggregate unit are maintained for the unit as a whole, and are often computed as the result of statistical analysis of the unit's actions. BBS, CBS, and Eagle are examples of constructive simulations. Constructive simulations are sometimes referred to as "wargames". They are usually not real-time.

Live simulation. In a live simulation, human trainees participate in a simulated battle using actual military equipment. For example, the National Training Center, where Army units fight battles using real tanks and other vehicles (though laser senders and sensors are used instead of live ammunition) is a live simulation.

Three terms that are similar in meaning in common usage will be used in very specific ways in this document with important differences in meaning. They are point, location, and position.

Point. A mathematical point, identified by coordinates. Most often in this document points will be located in 3-space relative to a terrain database.

Location. A point in a terrain database where an entity is or might be located. An entity's location or potential locations are assumed to be appropriate for that entity, e.g. ground vehicle locations are assumed to be on the surface of the terrain rather than in mid-air.

Position. A geographical region of known extent that is the area in which a military unit (e.g. a company) is to deploy. A position will often be a simple polygon when projected onto the x,y plane. The individual entities that make up the unit will be at locations that are within the unit's position.

2. Computer Generated Forces

This section provides a tutorial on CGF in general and surveys existing CGF systems.

2.1 CGF tutorial

This subsection contains a tutorial on CGF systems. It explains the role of CGF systems in a battlefield simulation and the software architecture of a CGF system. It also surveys how behavior is generated and specified within a CGF system, how artificial intelligence is used in a CGF system, how CGF systems are validated, and what research directions are crucial to continued progress in CGF system development. [Petty,1992c], [Petty,1995b], and [Pickett,1995] are also CGF tutorials; [Brooks,1989], [Bailey,1989], and [Booker,1993] are CGF surveys.

2.1.1 Role of Computer Generated Forces

In the case of DIS (and its SIMNET predecessor), the system is intended to provide a simulated battlefield which is used for training military personnel. In such a battlefield, the trainees need an opposing force against which to train. There are at least three ways to provide the simulated opposing forces (see Figure 2.1).

In the first method, two groups of trainees in simulators may oppose each other. For example, it is possible to configure SIMNET simulators during startup so that the computer image generators in each force's simulators display their own force's vehicles as US vehicles (M1 Abrams and M2 Bradleys) and the opposing force's vehicles as enemy vehicles (T-72s and BMPs). Thus both sides see themselves as US forces and their opponents as the enemy. This method is often used, and the soldiers enjoy the competitive aspects of the arrangement, but it has several disadvantages. First, it increases the number of expensive simulators needed at a training site. Second, it requires that to train any given military unit a second unit be available to provide the opposition. Finally, the trainees are faced with opponents who, despite the appearance of their vehicles in the video screens, use U.S. Army tactical doctrine because U.S. Army soldiers are controlling the vehicles. It would be preferable to provide the trainees with opponents who use the tactical doctrine of the actual or anticipated enemy.

A second method is to use human instructors who are trained to behave in a way that mimics the desired enemy doctrine. Doing so does not reduce the need for simulators and is expensive in manpower costs because large numbers of trained instructors may be required. Furthermore, the instructors must be retrained for each new enemy's doctrine. This method is seldom used.

The third technique is to use a computer system that generates and controls multiple simulation entities using software and possibly a human operator. Such a system is known as a semi-automated force (SAF or SAFOR) or a computer generated force (CGF).

CGF systems are important for several reasons. First, they lower the cost of a DIS system by reducing the number of standard simulators that must be purchased and maintained and by

reducing the number of humans required to operate the system for a given scenario size. Second, a CGF system can be programmed, in theory, to behave according to the tactical doctrine of any desired opposing force, and so eliminate the need to train and retrain human operators to behave like the current enemy. Finally, because a CGF system can be easier to control by a single person than an opposing force made up of many human operators, it may give the training instructor greater control over the training experience [Fishwick,1991]. With all of this in mind, [Oswalt,1993] identifies the development of CGF capabilities as a trend with "significant impact" on military simulation and gaming.

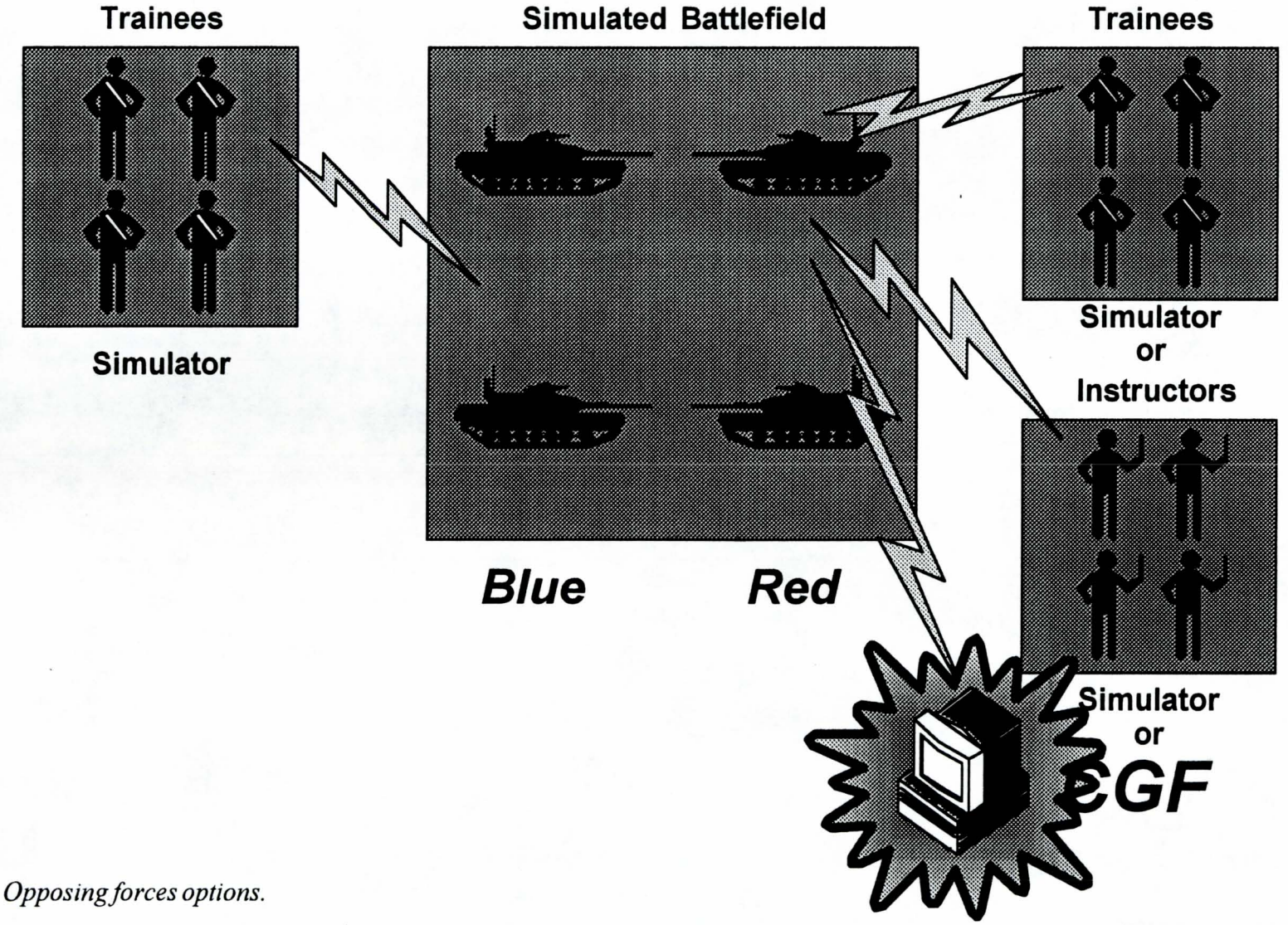
Note the assumption that CGF systems operate at the entity level, i.e. a single virtual entity is either controlled by a CGF system or by humans in a simulator, but not both. This assumption is in fact true for all existing CGF systems of note, but some work is starting on replacing individual crew members within a crewed simulator with CGF-like algorithms; e.g. see [Gagné,1995]. This document will not consider those systems.

2.1.2 CGF system characteristics

Certain characteristics are common to all existing CGF systems, and are essentially inherent in the context in which those systems are used. Some of the most important of those characteristics are listed here; each will be described in turn.

1. Network connection and protocol
2. Battlefield environment simulation
3. Support for multiple entities
4. Operator control of behavior
5. Representation of the military organizational hierarchy
6. Autonomous behavior generation

Network connection and protocol. Clearly, because DIS-type simulations are networked, a CGF system needs both a physical connection to the network and the appropriate software to send and receive network packets. Furthermore, the system must conform to the network protocol that has been defined for the simulation. It needs to correctly interpret the data in the packets it receives and format the data in the packet it sends. A CGF system is required to send network packets when specified by the protocol; such actions may be time or event triggered. Finally, the arrival of incoming network packets sent by other simulation entities should be handled correctly, as per the protocol. For example, if a CGF system receives a packet that signifies that one of its controlled entities has been hit by an anti-tank missile, it should assess the damage that may result from that impact. [Cheung,1994] analyzes the characteristics of the network packet stream typically produced by some CGF systems and compares it to that of crewed simulators.



2.1 Opposing forces options.

Battlefield environment simulation. The entities controlled by the CGF system exist in a battlefield which is a simulated subset of the real-world battlefield. As such, the CGF controlled entities should obey the laws of physics relevant to the activities occurring in the battlefield. Often, this means that one must use physical laws to model such behavior [Barr,1989], although lower-cost solutions may sometimes be appropriate (i.e. simplified kinematics instead of physical modeling). Using physics, the vehicle dynamics of the CGF entities can be modeled, including acceleration, deceleration, turn rates, and vehicle performance characteristics. (For an example of vehicle dynamics modeling, see [Cimini,1992] for a presentation of a flight dynamics model used for CGF aircraft.)

The CGF system usually includes a terrain database that provides the terrain over which the battle will be fought; it may be a detailed representation of an actual piece of terrain, or a large featureless plane corresponding to the surface of the ocean. The effects of the terrain on the simulation events should be modeled, including terrain effects on movement and observation.

Because the world being simulated is a battlefield, combat interactions need to be modeled in accordance with the physics of weapon and armor performance characteristics. For example, in both DIS and SIMNET, a CGF tank that fires on a hostile vehicle determines if a hit was achieved using a set of factors that include range, exposure of the target, and performance of the tank's weapon and sighting systems. If a hit is inflicted, the impacted vehicle considers munition type, range, impact angle, and armor protection to assess the damage it suffers. The accuracy of these calculations is of central importance to the validity of the simulation.

Support for multiple entities. CGF systems typically provide support for multiple entities simultaneously. Their usefulness is due in large part to this characteristic. The CGF system's architecture must provide a means to allocate processing resources to all of its supported entities.

Operator control of behavior. In addition to the autonomous behavior, every production CGF system should include an operator interface that allows a human operator to control the CGF entities. The operator may override autonomously generated behavior, or he or she may initiate and control behavior in situations that are beyond the CGF system's capabilities. Existing CGF systems typically provide a map display of the battlefield that shows the battlefield terrain and the simulated entities on it, together with a human command interface.

Representation of the military organizational hierarchy. Military units have hierarchical organizations. As CGF systems are designed to control larger numbers of entities, it becomes increasingly important to represent the military hierarchy of those entities in the system. With the representation of the hierarchy in place, the operator can give orders to higher level units, or the CGF system can autonomously generate behavior for a unit. Then, the unit level order could be automatically interpreted and passed down to the constituent entities for execution.

Autonomous behavior generation. A CGF system will use built-in behavior to react autonomously to the simulation situation or to carry out orders given by its operator. Its behavior may be encoded as algorithms, production rules, formal behavior specifications, or some other form. The intent is for the CGF system's behavior to be autonomous (i.e. not requiring human

control) and realistic (i.e. true to doctrine, physics, and human responses) to the greatest extent possible.

Figure 2.2 shows a notional decision making process for a CGF system.

It is in the area of autonomous behavior generation that most current CGF research is focused. The Institute for Simulation and Training (IST) and other research laboratories are attempting to increase the level of autonomy of CGF systems. This area will be reviewed in more detail later.

2.1.3 Behavior specification and generation for CGF systems

Behavior refers to actions or reactions by an entity that are the result of a cognitive or decision making process; this is contrasted with actions by an entity that are governed by the laws of physics and have no cognitive involvement.

Of course, as mentioned earlier, the laws of physics must operate in a realistic way in a simulated environment (at least in one that purports to simulate reality). However, simulating behavior, i.e. the intentions, goals, and intelligence of autonomous agents, within simulation is a separate and more uncertain matter. There has arguably been less progress made in representing and specifying intentional, intelligent behavior than physical behavior, and there is certainly much less agreement among researchers on how best to generate such behavior [Wallich,1991], at least in the general case. However, in the more focused area of CGF systems there has been more agreement and progress.

CGF systems produce autonomous behavior for the simulation entities they control. To do so, the desired behavior must be specified and generated. *Behavior specification* is the encoding, in a form usable by the CGF system, of the specific behaviors that CGF system is expected to be able to produce. Most often, the behavior specified for a CGF system reflects military tactical doctrine, and behavior specification for CGF systems is a process of encoding tactics in a form useful to algorithms. Behavior specification is a knowledge engineering problem. *Behavior generation* is the run-time execution of the specified behavior so as to produce useful tactical behavior in real-time that is responsive to the battlefield situation.

2.1.3.1 Behavior specification

The notion of separating the specification of behavior from the generation or execution of behavior is familiar; it is analogous, for example, to a production rule system where behavior (or knowledge) is specified in the form of rules, and the execution of the behavior is performed by an inference engine. Similarly, a robot's movements might be specified with a English-like scripting language which is interpreted at run-time to generate the behavior [Bourne,1982]. However, the degree to which behavior specification and generation are actually separated is quite variable in CGF systems, and none fully succeed in separating them completely in a satisfactory way.

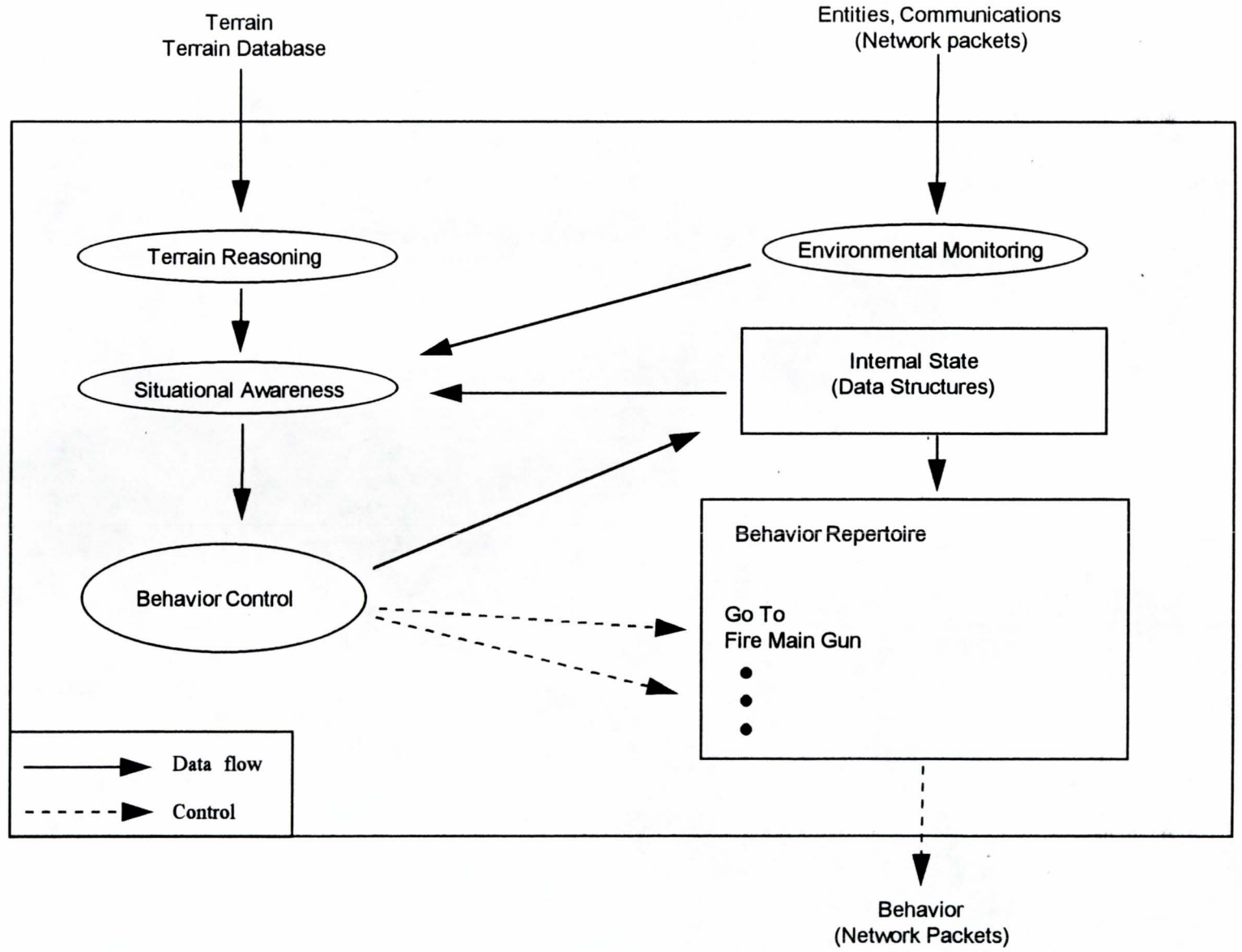


Figure 2.2 CGF autonomous behavior generation..

Work has been done on encoding general autonomous behavior in various forms; these include production rules [Zisman,1978], structured English text [Bourne,1982] [Stevens,1989], and Petri nets [D'Angelo,1983] [Blitz,1988]. Prototype and production CGF systems have used expert system rules [SOGITEC,1989], map overlay symbols and battle drills [Crooks,1990], Petri nets [Madni,1987] [Moshell,1989], and contingent hierarchical scripts [Lockheed,1990].

The three most commonly used methods of CGF behavior specification are:

1. Algorithms and finite state machines
2. Behavior specification languages
3. Combat instruction sets

Algorithms and finite state machines. Typically, CGF systems are written in programming languages such as C, Lisp, or Ada [Bailey,1989] [Booker,1993] [Petty,1995b]. For some CGF systems, the patterns and rules of behavior for CGF entities are essentially encoded directly in the algorithms of the CGF system. One example is the IST CGF Testbed, where CGF behavior is specified in C code, organized with a technique referred to as *Finite State Machines*, which will be explained later. Clearly, in this case the behavior specification and behavior generation mechanisms have not been separated.

Behavior specified directly in a programming language is almost always inaccessible to subject matter experts (SMEs). Specifying doctrine, or general behavior, for implementation as program code in a CGF system requires both a SME and a skilled programmer [Lattimore,1993]. It is a classical knowledge engineering task, which can potentially be quite difficult [Sargeant,1990]. Furthermore, because the SME cannot read the behavioral descriptions, he or she must validate them by observing the generated behavior of the CGF entities in the simulation, a time consuming and unreliable procedure.

Behavior specification languages. A formal language that is precise enough to specify CGF behavior, translatable into machine executable form, and also understandable by SMEs would be very useful [Kornell,1987]. Such a language has been called a *behavior specification language*.

Military doctrine for combat units is usually recorded as text in a training manual; doctrine for other domains and entities may not even be documented to that extent. [Fishwick,1991] asserts that "An ideal solution with respect to automatic control over [CGF entities] is one where commands may be expressed directly in language specified within the training doctrine documentation." For example, a doctrine that specifies a tank platoon to "move at slow speed to the edge of the mine field" would then be translated automatically into an intermediate language for planning and execution. This process, termed by him as "doctrinal language processing", is a subset of the more general natural language processing problem in artificial intelligence. It is clear that some SME accessible language for expressing CGF behavior is desirable.

A survey of eight formal behavior specification languages designed with the intent of expressing tactical doctrine is found in [Petty,1993]. (Three of the languages surveyed are documented in more detail elsewhere in [Smith,1993], [Moshell,1989], and [Smith,1992c].) Unfortunately, the conclusion of that survey was that none of the designs were completely satisfactory. The survey

relates the difficulties that are seemingly inherent in designing a behavior specification language that both has sufficient expressive power to specify complex tactics in an unambiguous way and that remains accessible to SMEs (i.e. to non-programmers). For example, Figure 2.3 gives an example of VBL (Virtual Battlespace Language), a CGF behavior specification language used in the ACBM family of systems (to be described later), and Figure 2.4 shows ILLISH (Intermediate Level Language, Interpreted, for Script Handling), a language designed for use in the IST CGF Testbed. Both VBL and ILLISH are powerful enough to specify CGF behavior, and ILLISH was selected as the best all-around behavior specification language design surveyed in [Petty,1993], but neither are accessible to SMEs. As might have been expected, the languages studied in [Petty,1993] that were most SME accessible were least powerful and the most powerful languages were least SME accessible. This problem remains unsolved.

Combat instruction sets. A Combat Instruction Set (CIS) is a functional element of CGF tactical behavior. The term CIS can variously refer to:

1. A tactical behavior at the entity or small unit level, i.e. the behavior itself
2. A written structured English description of that basic element of behavior
3. A computer representation of that behavior (i.e. program code that executes that behavior).

Note that these three forms are ostensibly different representations of the same thing; which one of the three fors is meant by the term CIS should be clear from the context of its usage.

As a basic element of behavior, a CIS can be a maneuver, battle drill, or a patterned response to a condition. Example CIS-level behaviors include an entity planning a route, a platoon changing to column formation, or a company dispersing in response to an air raid. CISs can be defined for entities, platoons, companies, and battalions, though typically the majority of CISs are at the platoon and company level. According to [McEnany,1994], company and especially battalion level CISs are most often assembled from lower level CISs.

CISs are often used as the unit of CGF behavior specification. A CIS in the first form, a basic element of tactical behavior, is written out explicitly by an SME in the second form, a structured English description of the tactical behavior. That structured English CIS is then encoded by a Software Engineer in the third form, computer program code, and compiled into a CGF system. Figure 2.5 suggests the process. Formal procedures for the transformations labeled (A) and (B) in Figure 2.5 are documented in [McEnany,1994] and [Ourston,1995] respectively.

CISs typically have names, such as Execute_Column_Formation. A CIS consists in essence of three components:

1. Initiating conditions; A set of conditions that must exist for the CIS to become active.
2. Steps; A sequence of detailed steps to be taken to perform the task.
   These steps are either subordinate CISs or primitive actions. Primitive actions are atomic (i.e. not decomposable), executable by individual entities, and do not require any decision making. Each primitive action is an entity action in one of four categories: move, shoot, communicate, or search/observe.
3. Terminating conditions; A set of conditions which if present terminate the CIS.

```
FIRING-START low_on_ammo
    TGT-TYPE tank
        WPN-TYPE main_gun
            USE CANDIDATES FOR FILTER 1
                2D-DISTANCE < 1200. METERS
                AND ORDNANCE > 3 ROUNDS OF-TYPE heat
                    OR
                2D-DISTANCE < 1000. METERS
                AND REL-TGT-HDG > 25. DEGREES
                AND ORDNANCE > 2 ROUNDS OF-TYPE heat
            USE FILTER 1 SELECTIONS FOR FILTER 2
                2D-DISTANCE < 850. METERS
            CHOOSE FROM FILTER 2 SELECTIONS
                FIRE 1 ROUND
                SELECT AT-MOST 2 main_gun
END FIRING-START low_on_ammo
```

Figure 2.3 Example behavior specification language: VBL [Lattimore,1993].

```
DEFINE TARGET;
    REMEMBER SELF:CURRENT_THREAT;
    SUBSCRIBE SELF:CURRENT_THREAT
    TELL SELF AWAKEN THIS_SCRIPT
    GOTO FOUND_ONE WITH SELF:CURRENT_THREAT;
    TELL SELF BEGIN SCAN_FOR_THREATS
    POST SELF:CURRENT_THREAT;
    REMEMBER SELF:ARRIVED;
    SUBSCRIBE SELF:ARRIVED AWAKEN THIS_SCRIPT AT NOW_THERE;
LETSGO:
    TELL SELF BEGIN PLAN_AND_GO 1438 2234 POST SELF:ARRIVED;
    SUSPEND;
NOW_THERE:
    TELL SELF END SCAN_FOR_THREATS;
    POST INPUT SUCCESS;
    END;
FOUND_ONE:
    ASSIGN TARGET INPUT;
    TELL SELF END PLAN_AND_GO;
    REMEMBER SELF:TARGET_DEAD;
    SUBSCRIBE SELF:TARGET_DEAD RESUME THIS_SCRIPT AT LETSGO;
    TELL SELF ATTACK TARGET SELF:TARGET_DEAD;
    SUSPEND;
```

Figure 2.4 Example behavior specification language: ILLISH [Smith,1993].

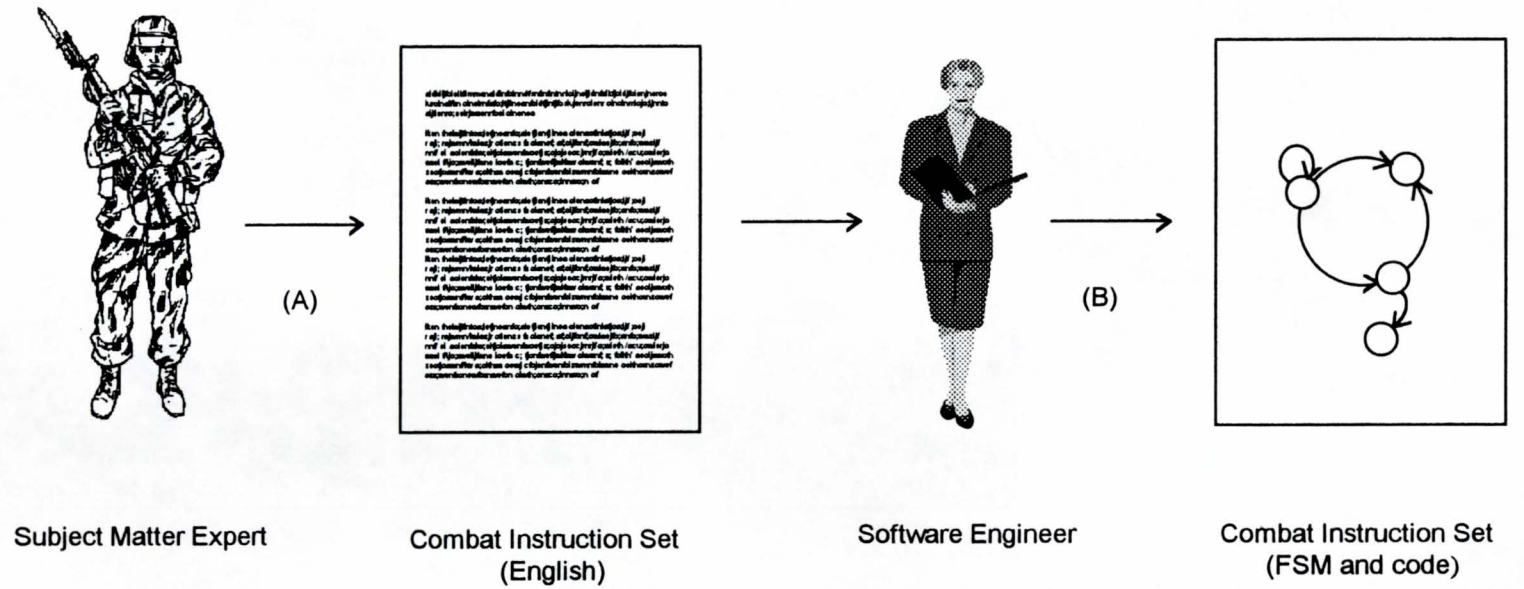


Figure 2.5 CIS development process.

A CGF unit's operations order can be constructed and expressed as a sequence or set of CISs. Associated with the CISs of the mission are triggers for the invocation of each CIS in the mission. Those triggers might be:

1. Completion of another CIS
2. Arrival by the unit at a particular geographical control measure, such as an objective or phase line
3. A specified time
4. The existence of a particular battlefield condition.

CISs were originally used in the first production CGF system, the SIMNET SAF, and are being used presently in ModSAF and the CCTT SAF, two of the three most important existing CGF systems. [McEnany,1993] defines CISs and the structure of a structured English CIS in some detail and gives example CISs in that form.

2.1.3.2 Behavior generation

The most common single paradigm for organizing and controlling behavior generation within simulation has been the Finite State Machine (FSM). (It is assumed that the reader is familiar with FSMs as they are defined in automata theory; if necessary, see [Hopcroft,1979] or [Lewis,1981] for good introductions.) The simulation behavior generation techniques, though referred to as FSMs, always include capabilities outside the bounds of formal automata theory FSMs. Hereinafter "FSMs" refers to the behavior generation paradigm rather than the automata theory construct.

Many FSM variants have been used in simulations of various types; often the researchers reinvent the idea quite unaware of other similar applications (e.g. [Petty,1988a]). The popularity and repeated use of this idea suggests its intuitive appeal and effectiveness.

The common idea is that a simulation entity's behavior is decomposed into a finite set of behavior patterns or states, with identifiable and discrete conditions for transitioning between the states. Typically the FSMs are used as an organizing mechanism for structuring or encapsulating procedures or functions written in a lower level programming language. Associated with each state is an implementation (e.g. a function or procedure in the underlying language) of that state's behavior; while in the state, that implementation is executed. The current state of the FSM therefore determines what behavior the simulation entity executes. Transitions between states are triggered by events or conditions in the simulation.

FSMs have been used in both CGF and non-CGF applications. Representative examples of FSM use in non-CGF systems include a wide range of applications; three examples will be given. First, [Maruichi,1987] describes how the behavior of fish in an ocean environment simulation was defined using FSMs. The underlying implementation language is Lisp. Second, The Zaroff planning system selects behaviors for players in a "hide and seek" game; the behaviors are then animated by Jack, a human modeling and simulation program [Badler,1993]. Once selected, the behaviors' execution is controlled by FSMs [Moore,1995]. Each state corresponds to and controls a distinct temporal component of the behavior. Finally, the Iowa Driving Simulator is a

high-fidelity driving simulator [Cremer,1994] [Papelis,1994]. The operator's station, built around the cab of an actual automobile, includes a motion platform, force feedback controls, and high-resolution image generation. The operator is faced with driving scenarios that include other vehicular traffic. As explained in [Ahmad,1994], the other vehicles in the simulation are controlled by "hierarchical concurrent state machines", a variant of FSMs that allow states to be defined as sets of subordinate states and permits more than one state to be simultaneously active. The implementation language is C.

[Fishwick,1993] describes the use of hierarchically organized FSMs to control simulated behavior in the context of multimodeling. In multimodeling a system is modeled at different levels of abstraction and different modeling formalisms, such as FSMs, may be used at each level. In [Fishwick,1993], the states of an FSM are represented as either more detailed FSMs or low-level continuous models implemented as sets of equations.

As for CGF-type applications, FSMs have been used often. [Petty,1988a] and [Petty,1988b] report on a project where FSMs were used to generate the behavior of combat aircraft in the vicinity of an aircraft carrier. Defensive fighters (F-14 Tomcat) and attacking bombers (Tu-95 Bear), each with different behaviors, were controlled with FSMs. The underlying language was Lisp and the FSM states corresponded to Lisp functions. Although this application (air combat) is one that might be ordinarily associated with a CGF system, the system was developed to test an animated graphics programming environment and lacked several of the characteristics of a CGF system as defined earlier (e.g. a network interface).

FSMs were used to both specify and generate entity-level behavior in SAIC's SimCore simulation, which is an analytic-style simulation interfaced with DIS [Aronson,1994]. The SimCore "Hierarchical, Concurrent Finite State Machines" can be organized hierarchically, with each FSM state containing either one or more sub-FSMs, algorithmic procedural code, a rule-based system, or a linear program. Multiple FSMs can be active concurrently for a single entity.

Though it is a constructive wargame and not a CGF system as defined in this document, it is worth mentioning that FSMs are also used to control entity-level behavior in the U.S. Marine Corps' MWARS simulation [Parsons,1994].

FSMs are used to specify and generate the behavior of CGF entities in all three of the most important CGF systems: the IST CGF Testbed, ModSAF, and the CCTT SAF. These three CGF systems will be examined in some detail later; as an example, the use of the FSM mechanism in the IST CGF Testbed will be presented in some detail here.

In the IST CGF Testbed, the primary means of behavior specification and generation is a code structuring technique based on FSMs. Behavior in the CGF Testbed is ultimately encoded as algorithms written in C. However, the C code is organized using the FSM mechanism. The basic idea is that atomic units of behavior, implemented as C functions, become states in an FSM. In other words, each state in an FSM corresponds to either a C function or another, lower-level FSM. FSMs exist as actual data structures in the CGF Testbed, with each state containing a pointer to the function corresponding to the state. When an FSM enters a particular state, one of

two actions may occur. If that state corresponds to a function, that function is called. If the state corresponds to a FSM, that FSM is started.

Each state determines the next state to be entered by testing simulation conditions; thus transitions may be triggered indirectly by simulation events. The C function for a state in an FSM contains code for both the behavior associated with that state and the conditions for selecting the next state. The transition conditions associated with each state, expressed as C conditions, test conditions in the simulation to determine the next state to be entered. Time delays may be associated with the transitions in the FSM to produce realistically timed behavior.

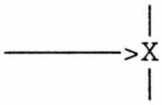
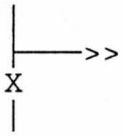
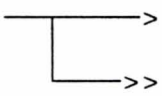
A simulation entity may have multiple independent FSMs controlling various aspects of its behavior executing concurrently in an asynchronous fashion. However, it is the responsibility of the programmer to ensure that FSMs that may execute concurrently do not interfere with each other. This is sometimes a problematic task.

More complex behavior can be constructed, bottom up, by combining simpler FSMs. The FSM mechanism has been extensively used to build up a variety of complex autonomous behavior patterns for CGF entities.

Figure 2.7 gives an example of an FSM from the IST CGF Testbed. The notation used in the example is first defined in Figure 2.6. Both Figure 2.6 and Figure 2.7 are drawn from [Smith,1992c]; it, as well as [Petty,1992c] and [Karr,1992b], contains additional example FSMs from the IST CGF Testbed.

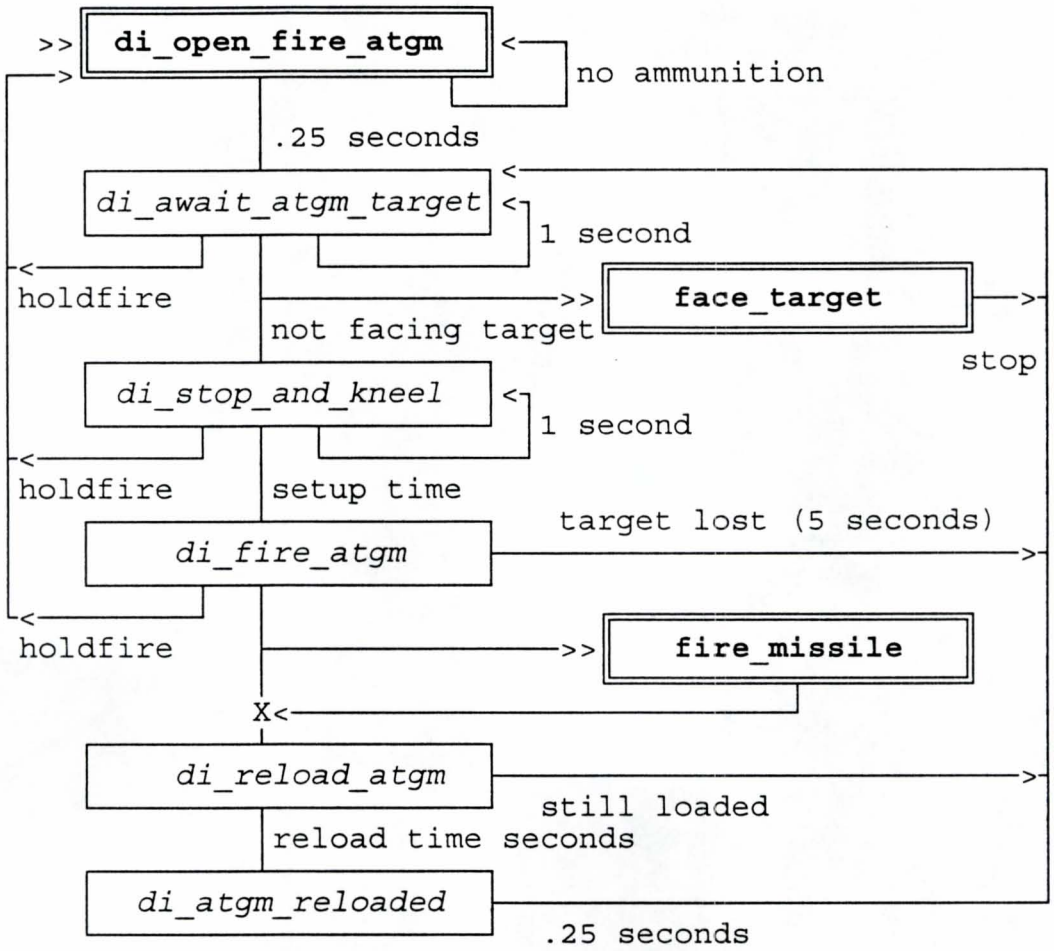
The IST CGF Testbed has the capability to support multiple CGF entities. To do so, it was built around an executive that provides a non-preemptive task scheduling capability. The executive maintains a message queue that identifies entity processes waiting to execute. It gives control to a process on that queue, which executes. Upon completion, that process must identify the next process to execute for its entity and add that process to the executive's message queue before returning control to the executive. The process, i.e. the unit of execution, is an FSM state. That is, when an entity gains control of the processor, it executes one state of one of its currently active FSMs. The state performs its computations by calling its associated function. That function either includes a determination of the next state to execute for the entity or the FSM will become dormant when the current state completes its execution. That next state's identity is placed on the executive's message queue. See [Danisas,1990] and [Smith,1992b] for more details on the IST CGF Testbed's executive; its FSM mechanism is described in detail in [Smith,1992c].

As mentioned earlier, FSMs are also used for behavior generation and control in ModSAF [Calder,1993] [Pratt,1995a] and the CCTT SAF [Marshall,1994]. Those mechanisms will be discussed later.

Symbol	Meaning
>> FSM name	The entry state of an FSM. The name of the FSM is shown in the box. If this symbol appears in an FSM diagram as other than the first state, it implies the invocation of another FSM.
state name	State within an FSM. The name of the state is enclosed in the box.
	Transition to another state within the same FSM. Flow is from top to bottom.
	Transition to another state within the same FSM. Flow is left to right.
	Start an FSM.
	Send an AWAKEN_FSM message to sleeping FSM.
	FSM makes a transition from the upper state to the lower state, starts a new FSM, and goes to sleep awaiting an AWAKEN_FSM message from the new FSM.
	FSM makes a transition in direction of the arrow, starts a new FSM, and goes to sleep awaiting an AWAKEN_FSM message from the new FSM.

Labels on the transition arrows describe transition conditions. Time labels, e.g. "1 second", give delays between states. Only one transition from a state may be taken.

Figure 2.6 IST CGF Testbed FSM diagram notation [Smith, 1992c].



When a CGF infantry fireteam is given permission to fire antitank missiles, the **di_open_fire_atgm** FSM is started. The start state, *di_open_fire_atgm* immediately transitions to the *di_await_atgm_target* state. The *di_await_atgm_target* state performs target acquisition and selection. The *di_await_atgm_target* state repeats every second until a target is found. When a target is found, *di_await_atgm_target* has two actions. First, it starts the **face_target** FSM which causes the fireteam to face the intended target. Second, the FSM transitions to the *di_stop_and_kneel* state, which brings the fireteam to a halt and then transitions to the next state after a delay corresponding to the weapon setup time. The *di_fire_atgm* state launches a missile the target is still visible and if the fireteam is not suppressed. The missile is launched by starting the **fire_missile** FSM; that FSM generates the missile launch flash, controls the missile in flight, and handles the impact at the end of the missile's flight. The **di_open_fire_atgm** FSM sleeps until **fire_missile** reports that the missile flight is finished, whereupon the *di_reload_atgm* state is awakened. The *di_reload_atgm* state reloads the fireteam and transitions to the *di_await_atgm_target* state for another cycle.

Figure 2.7 Example IST CGF Testbed FSM [Smith,1992c].

Although the idea of using FSMs as a behavior control mechanism has been repeatedly reinvented and widely applied, notably in the most important CGF systems, there have been a few dissenting opinions expressed regarding them. For examples, see [Harmon,1991] and [Harmon,1994] for criticisms of their use or [Ahmad,1994] for an expression of their limits (though the latter simply proposes an enhanced version of FSMs). Other behavior generation techniques have been used in some CGF systems; those will be identified later.

2.1.3.3 Artificial intelligence in CGF systems

As might be expected, many researchers have applied artificial intelligence (AI) techniques to the problems of behavior specification and generation, or tactical decision making, in CGF systems. CGF would seem to be a natural application for AI techniques.

As an example, one interesting way to view an autonomously behaving entity is as an active version of an expert system. Expert systems encode knowledge in a way that could be considered passive, in that such systems typically wait for a user to consult the knowledge [Harmon,1986]. Computer generated forces entities instead use their encoded knowledge to act and react to situations in the simulated battlefield, in an active or goal-seeking manner. This difference is a matter of how the knowledge is accessed and triggered; some systems for controlling autonomous entities use behavioral knowledge encoded in forms, such as production rules, that are very much like the knowledge bases of expert systems.

Many other AI techniques have been applied as well. However, a thorough survey of that work is beyond the scope of this document. Instead, Table 2.1 lists many of those efforts.

In spite of the intuitively reasonableness of applying AI to CGF, the results of these applications of AI to CGF have been decidedly mixed. Two problems bedevil the straightforward application of AI to CGF. The first problem, mentioned earlier, is the real-time aspect of CGF systems. CGF systems must make tactical decisions in very short time frames; classical AI techniques can require too much processing time. [Petty,1995b] identifies this problem for general CGF processing and [Hayslip,1988] makes the same comment in regards to AI techniques for terrain reasoning.

The second problem is that the tactical situation, or more precisely the data representing the tactical situation that a CGF decision making algorithm must process, is rather "messy"; i.e. voluminous, continuous, and represented in several different formats. The decision making algorithm must consider the terrain, the other entities in the battlefield, the military mission, and tactical doctrine of the forces it is simulating. The terrain might be made up of thousands of polygons or elevation posts supplemented with linked lists of features such as buildings or trees, all located continuously in three dimensional space. The other entities are represented by a set of attributes that include location, orientation, velocities, damage status, equipment type, and force alignment. The military mission, if organized as Operations Orders, might be a frame-like structure with slots for objectives, phase lines, and intelligence objectives, encoded in a structured English text. The tactical doctrine could be encoded in any of the forms described earlier.

AI Technique	CGF System	CGF Application	Reference(s)
Search (A*, Iterative Deepening A*)	None (proposed application)	Entity route planning (cross country)	[Holmes,1992] [Marti,1994]
	CGF Testbed	Unit route planning (cross country)	[Rajput,1994b] [Karr,1995d] [Karr,1995e]
	None (proposed application)	Unit route planning (cross country)	[Cunningham,1993]
	None (proposed application)	Unit route planning (road nets)	[Benton,1987]
	SIMNET SAF	Unit route planning (road nets)	[Stanzione,1989]
	ODIN SAF	Unit route planning (road nets)	[Stanzione,1993]
	ModSAF	Concealed route planning	[Longtin,1995]
	CCTT SAF	Entity and unit route planning (road nets)	[Campbell,1995]
Behavior-based control	CAAT	Air combat maneuvering	[Keirsey,1994]
	SIMNET SAF	Entity driving, resolution between conflicting goals	[Harmon,1991] [Harmon,1994]
Blackboard	Command Decision System	Blackboard used to integrate results of disparate "knowledge sources", i.e. CGF behavior modules implemented as rule-based expert systems or procedural algorithms	[Gates,1990] [Braudaway,1992] [Braudaway,1993
Fuzzy sets	MWARS	Unit command decision making; selection among tactical actions given by human during scenario setup	[Parsons,1994]
	ModSAF	Target threat evaluation	[Cisneros,1995]

Table 2.1 (Part 1 of 4) Some CGF applications of AI techniques.

AI Technique	CGF System	CGF Application	Reference(s)
Neural networks	Air Combat Maneuvering Expert System (ACMES)	1-vs-1 air combat maneuvering	[Crowe,1990]
	DeSim	Generic military decision making; all decisions at all scales, with parametric inputs to the net	[Weaver,1994]
	None (proposed application)	Tactical decision making for entities and units, based on abstracted physical information	[Jaszlics,1993] [Jaszlics,1994]
Planning (Optimization)	VCom	Entity and unit route planning	[Cunningham,1994]
Planning (Simulation based)	None (proposed application)	Unit mission planning	[Lee,1994a] [Lee,1994b] [Lee,1994c]
Planning (Simulation based)	ModSAF	Unit mission planning	[Karr,1995b]
Planning (State-space search)	Captain	Selecting subunit defensive positions	[Hille,1994] [Hieb,1995] [Hille,1995]
	Soar/IFOR	Planning air combat maneuver and action sequences	[Jones,1993b] [Johnson,1994] [Jones,1994b] [Jones,1994c] [Laird,1994] [Rosenbloom,1994] [Tambe,1994] [Laird,1995] [Nielsen,1995] [Tambe,1995a] [Tambe,1995b]
Planning (Universal plans)	MAXIM	Aircraft and missile maneuvering in air combat	[Dyer,1993]

Table 2.1 (Part 2 of 4) Some CGF applications of AI techniques.

AI Technique	CGF System	CGF Application	Reference(s)
Rule-based expert system	Command Decision System	Cavalry platoon commander for reconnaissance mission (total of 38 rules)	[Braudaway,1992] [Braudaway,1993]
	CCTT SAF	Platoon and company unit command	[Bimson,1994] [Ourston,1994]
	CGF Testbed	Single entity control during reconnaissance operation	[Gonzalez,1991]
	ITEMS	Several aspects of entity and small unit behavior, including air combat maneuvers and weapons use, and unit (company and battalion) command and control	[Siksik,1993] [Kocabas,1995]
	None (proposed application)	Target identification	[Vrba,1988]
	Piastre	Tactical movement of platoons of target vehicles	[SOGITEC,1989] [Huon,1989] [Kada,1994]
Game-tree lookahead	Game Commander	Unit command and control designed for ModSAF	[Katz,1994]
	Intelligent Player	Controlling helicopter movement in air combat	[Katz,1989] [Katz,1991] [Katz,1992] [Katz,1993] [Schaper,1994] [Pandari,1995]

Table 2.1 (Part 3 of 4) Some CGF applications of AI techniques.

AI Technique	CGF System	CGF Application	Reference(s)
Learning (Case-based)	CAAT	Air combat maneuvering	[Keirsey,1994]
Learning (Explanation-based)	ITEMS	1-vs-1 air combat maneuvering	[Kocabas,1995]
Learning (Multistrategy)	Captain	Company and battalion command agents that learn tactical rules and behaviors based on both SME and autonomous performance in simulation	[Hille,1994] [Hieb,1995] [Hille,1995]
Natural language processing	Soar/IFOR	Communication between CGF entities and humans	[Rubinoff,1994] [Lehman,1995]
Backward reasoning from goals	ModSAF	Prolog-based backward reasoning for unit control	[Kwak,1995]
Semantic net	None (proposed application)	Tactical state representation in unit command entity	[Mall,1995]
Distributed AI	None (proposed application)	Distributed tactical decision making	[Le,1990]

Table 2.1 (Part 4 of 4) Some CGF applications of AI techniques.

Figure 2.8 shows the CGF decision making process in the context of the transformations that must occur. In order to apply a typical AI technique, the data describing the tactical situation must be transformed into the format that the technique can process; e.g. input vectors for neural nets, states and operators for search, or facts and rules for a rule-based system. This transformation is an abstraction process, where the large amounts of raw data are abstracted into a classification of the tactical situation, with key aspects of the situation identified and irrelevant ones discarded. This abstraction process is sometimes called *Situational Awareness*.

The center process in Figure 2.8, *Tactical Decision Making*, is where most AI techniques are applied to CGF systems. The techniques operate reasonably well in a conceptual decision space to produce conceptual decisions.

Then, once a conceptual tactical decision has been made, it must again be transformed from the output format of the AI technique into specific commands and actions for the CGF entities; e.g. a decision for a unit to attack an objective must be transformed into a set of specific routes for the unit's component entities to follow. This transformation is referred to as *Order Generation*.

The input and output transformations, especially the former, can be quite difficult. It is often the input transformation task that limits the success of applying a particular AI technique. Direct application of AI techniques is frequently most successful in domains where the tactical situation is closest to the abstract representation used by the AI technique. For example, state-space search (Soar/IFOR, [Laird,1995]) and game-tree generation (Intelligent Player, [Katz,1993]) have been applied to the domain of air combat, where the small numbers of entities and minimal terrain interaction make the input transformation easier. As another example, an effective use of the A* search algorithm for unit route planning described in [Rajput,1994b] [Karr,1995d] depends on an input transformation wherein the essentially continuous polygonal terrain database is cleverly discretized into an array of abstract terrain cell types. The underlying A* search algorithm is defined in [Nilsson,1980] and [Winston,1984].

Other attempts to use AI techniques for CGF that did not understand the importance of the transformations, especially Situational Awareness, have been less successful. The point of this discussion is to emphasize the importance of the transformations. Given an effective transformation, AI can be applied to CGF successfully.

2.1.4 Verification, validation, and accreditation of CGF systems

The process of evaluating a simulation or simulation system and certifying it for use is known as verification, validation, and accreditation (VV&A). [Goldiez,1991] provides a general introduction to the VV&A of simulation systems. Those terms are defined as follows:
Verification; determining if a simulation performs as specified and designed.
Validation; determining if a simulation has sufficient fidelity for its intended purpose.
Accreditation; certification by an authorizing organization that a simulation may be used for its intended purpose.

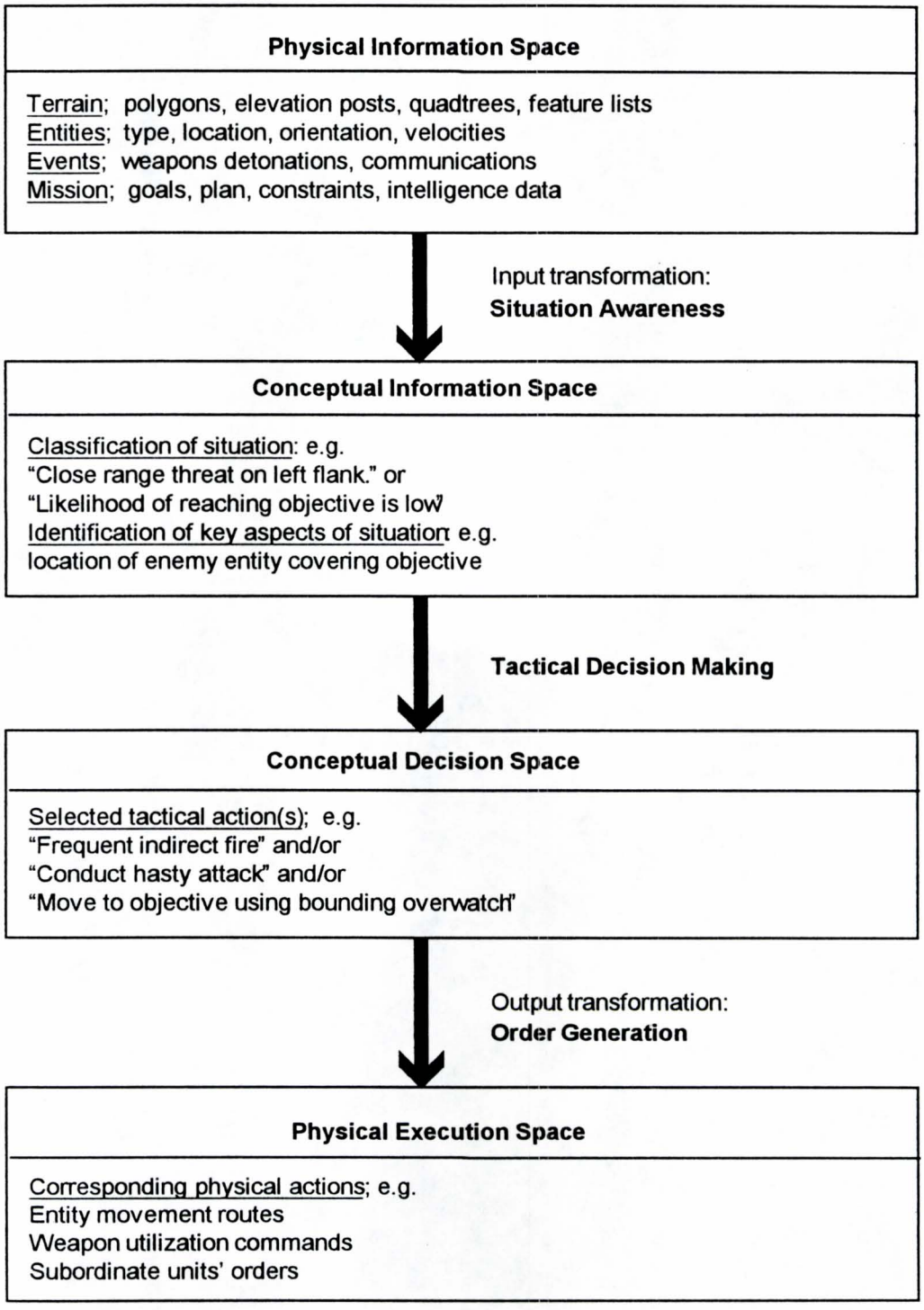


Figure 2.8 CGF decision making in context (adopted from [Jaszlics, 1993]).

The first requirement of CGF systems is that they work; i.e. that they generate plausibly human and tactically reasonable behavior in real-time situations in the virtual battlefield. Until recently, the difficulty of meeting that basic requirement has held the attention of CGF system developers. However, though building a working CGF system is a challenging task, simply doing so is not enough. To be useful a CGF system must meet standards of quality.

Because the attention of CGF developers had been focused on producing working systems, VV&A of the behavior and capabilities of CGF systems had often been haphazard or informal (with a few exceptions). However, as CGF systems become more important to the utility of current and planned simulation systems a more methodical and quantitative approach to VV&A of CGF systems is needed. In the last few years some preliminary efforts to VV&A CGF systems in a more organized manner have been performed.

This subsubsection will informally present some CGF VV&A issues. It first outlines some CGF fidelity requirements. Following that, several CGF VV&A experiences are described and commented on. Finally, one VV&A method, specifically the Turing Test as applied to CGF systems, is discussed at some length.

2.1.4.1 CGF fidelity requirements

The claimed benefits of a CGF system in DIS are all based on the assumption that the CGF entities can be made to behave in a usefully realistic manner. If human trainees are to experience positive training from interacting with a CGF opponent, that force must provide valid and useful opposition. To do so, the CGF entities must act in a manner that meets three criteria:

1. Physical realism
2. Behavioral intelligence
3. Doctrinal accuracy

Physical realism. The behavior generated for a CGF must be physically realistic in the sense that it provides a level of realism appreciated by the trainee. This requirement was discussed earlier.

Behavioral intelligence. The second and most problematic criterion of CGF behavior is reasonable behavioral intelligence. This means that the CGF controlled entities must react to a given situation in a manner similar to the entities being simulated. Because the simulated entities are often controlled by humans, the CGF behavior must appear to be similar to, and thus as intelligent as, human behavior in each situation. Of course, the intelligence requirement is easier to meet if the CGF is simulating entities that consist of non-human entities such as pilotless drone aircraft. Note that the granularity and fidelity level of the simulation normally keep the intelligent behavior requirement from becoming as difficult as the general AI problem. In the DIS world the repertoire of behaviors available to humans who are acting as members of a tank crew is much smaller than the repertoire of general human behavior, so intelligent behavior by a CGF tank is easier to generate than intelligent human behavior. Even so, producing intelligent behavior in a CGF is still a formidable task and the subject of much research.

In the area of reasonable intelligence, the real-time aspect of DIS becomes particularly relevant. Many AI techniques exist that may be able to contribute to the process producing the desired level of intelligence, but that do not execute with enough speed.

Doctrinal accuracy. CGF behavior must be doctrinally correct in the sense that the actions of the CGF should be believable in terms of the entities the CGF is simulating. Consider the SIMNET SAF; it is often used to control tanks and other vehicles intended to represent those of an army equipped and trained by the former Soviet Union. For the SIMNET SAF vehicles to be accepted by SIMNET trainees as elements of such an army, they must maneuver and act according to Soviet tactical doctrine. The issue in this example goes beyond simple believability; an important goal of SIMNET as a training system is to give the trainees an opportunity to engage an opponent that uses Soviet tactics.

In other simulation domains, doctrinal correctness, where doctrine is defined as the behavioral norms for the class of entities being simulated, remains important. In an air traffic control simulation that includes aircraft controlled by a CGF system, for the simulation to be useful those aircraft should in most cases re-create the behavior of aircraft flown by actual commercial pilots. Fidelity to doctrine is a significant goal to be addressed by the results discussed later in this survey.

Interestingly, [Hunter,1991] asserts that CGF fidelity requirements can be seen as variable depending on the simulation situation. According to [Hunter,1991], CGF entities remote from and not involved with human participants can safely be simulated with a lower degree of fidelity (and therefore with less computational overhead). For example, a simple and inexpensive probabilistic calculation could be used to resolve CGF-vs-CGF missile combat instead of the complex and costly missile flyout process currently used by CGF systems. However, difficulties arise with this idea, for example, in determining whether another entity is in fact CGF controlled or in analytical uses of the CGF system where high fidelity is needed even when humans are not involved. For these and other reasons few CGF developers take this viewpoint and there has been little or no attempt to vary representational fidelity within existing CGF systems.

2.1.4.2 Some CGF VV&A experiences

Here a number of CGF VV&A experiences are briefly recounted; this material follows [Petty,1995f]. The experiences reviewed fall into several categories. Each will be described in turn, followed by a summary of CGF VV&A lessons learned.

1. SME observation
2. Turing test
3. Measured comparison
4. Statistical comparison
5. DIS testing
6. Other techniques
7. Common factors

SME observation. Perhaps the most commonly used VV&A method for CGF systems is observation of CGF entities' behavior by subject matter experts (SMEs), who are typically military officers (either active or retired). In this technique, SMEs would simply observe the CGF entities in a test exercise and intuitively decide, based on their expertise, that the behaviors were (or were not) good enough.

This classical method is applied informally by every CGF developer, but it can be done in a more formal fashion. A particularly well organized and thorough example of this method was the evaluation conducted at the U.S. Army Infantry School of the IST's SAFDI system. At the Infantry School's SIMNET facility the SAFDI system "fought" with and against a U.S. Army unit in a series of three carefully designed battles that were observed and analyzed by an invited panel of SMEs. The SMEs had prepared in advance specific sets of performance criteria that they wished to observe and evaluate. The results of the evaluation are reported in [Chervenak,1993] and [D'Errico,1994].

[Jones,1993a] asserts that using a CGF system in a training environment is a useful means of informal validation, in that the trainees and instructors will necessarily observe the behavior and performance of the CGF entities and are likely to provide feedback on problems.

Turing Test. One particular form of SME observation is the CGF Turing Test. Because of the large amount of attention the CGF Turing Test has received in the CGF literature, it will be discussed separately; it is listed here for completeness.

Measured comparison. The measured comparison CGF VV&A technique involves the measurement of some set of quantifiable aspects, or metrics, of CGF behavior and the comparison of the measured values for those metrics with values for the same metrics derived from a different source that is assumed to be valid.

An experiment to evaluate the effectiveness of a behavior-based driver module that had been added to the SIMNET SAF is detailed in [Harmon,1991]. To conduct the evaluation twenty different movement scenarios were designed so as to test the new driver module under many different circumstances. The scenarios were organized as a taxonomy of driving situations, a method that helped to develop a more complete set of scenarios. The scenarios were run 10 times each with both the baseline and the modified versions of the SIMNET SAF. The results of each exercise were logged. Specific performance metrics, such as vehicle collisions and route efficiency, were defined and calculated from the exercise logs and compared between the two versions. Though this experiment was a comparison of two versions rather than a VV&A effort as such, it is a useful example of the measured comparison technique.

The Unit Performance Assessment System (UPAS) is a personal computer-based system for collecting and analyzing SIMNET network traffic (i.e. PDUs) [Meliza,1991] [Meliza,1995]. It was developed with the intent of measuring how well vehicle crews perform cooperatively as part of a unit. UPAS calculates various performance metrics by analyzing exercise logs. [Vaden,1994] describes how UPAS was used in 1993 to evaluate ModSAF by comparing the actions of ModSAF entities with the actions of entities controlled by humans in crewed

simulators. A series of 10 company sized exercises were logged and then analyzed by UPAS with two goals: first, to evaluate control of ModSAF behavior by physical parameter, such as line of sight and range, and second, to identify specific behaviors that could be used to distinguish between human-controlled and ModSAF-controlled entities. ModSAF's performance was found to differ from the humans' in several areas, including firing range, firing strategies, and scanning (turret azimuth movement).

In 1994 the same methodology, UPAS analysis of network traffic logs, was used to compare ModSAF and the SIMNET SAF in order to select one for the Army's Synthetic Theater of War-Europe (STOW-E) exercise. In a rather critical evaluation, [Meliza,1995] indicates that both CGF systems displayed "inadequate sensitivity" to factors such as mission, enemy, time, terrain, and troops. The UPAS analysis also revealed unrealistic engagement ranges, rates of fire, and firing ranges for direct fire actions, as well as no use of cover and concealment. ModSAF has been improved substantially since these evaluations were conducted (see [Courtemanche,1995a]).

Statistical comparison. Statistical comparison techniques for CGF VV&A attempt to apply proven statistical methods, such as hypothesis testing, to data representing CGF behavior.

Algorithms which planned military reconnaissance routes were implemented in the IST CGF Testbed. The algorithms were given as input a defined area of a polygonal terrain database and produced as output a series of waypoints that defined effective reconnaissance routes for that terrain. Independent of the algorithms, human SMEs chose locations in the input terrain areas for enemy vehicles in defensive positions. The goal for the reconnaissance route planning algorithms was to plan routes based on the terrain that would allow a reconnaissance vehicle following the route to sight as many enemy vehicles as quickly as possible.

The algorithms were validated by statistically comparing their performance with that of human SMEs (military officers) performing the same task. Reconnaissance vehicles followed routes produced by the algorithms and by the SMEs. The time at which each enemy vehicle was sighted was recorded. A statistical hypothesis test was used to compare the sighting times resulting from the algorithms' routes with those resulting from the SMEs' routes. The specific test used was the Wilcoxon Signed-Rank Test, a non-parametric test useful for comparing paired observations. The vehicle sighting times were compared in a pairwise fashion, first-sighted to first sighted, second-sighted to second-sighted, and so on. The best of the algorithms performed at a level comparable to the human SMEs. This effort is documented in [Van Brackle,1993a], [Van Brackle,1993b], and [Petty,1994a].

A similar method was used to VV&A the Security Exercise Evaluation System (SEES), a variant of the U.S. Army's Janus entity-level constructive simulation. Though SEES has limited CGF functionality (most entity behavior is controlled by operators), the technique used is nonetheless instructive. A scenario involving an attempt by a well-armed terrorist group to steal a nuclear warhead from a warhead storage facility at Wurtsmith Air Force Base was defined. The scenario was run a number of times as a live simulation with human soldiers equipped with MILES laser training weapons attacking and defending the actual warhead storage facility. Field instruments

recorded key simulation events. The same scenario was then run repeatedly in SEES and the results logged.

The results of the SEES simulation and the live simulation, including force ratios and the times of key events, were then compared statistically. The Kolmogorov-Smirnov (K-S) statistic was used to determine if the live simulation results and the SEES simulation results came from the same underlying distribution. The formulation of the test used assumed (had as the null hypothesis) that they did. K-S values were computed for the attacker-defender force ratio, which changed over time as soldiers were "killed", at 30 second time intervals for the duration of the exercise. The computed K-S statistics would not support rejection of the null hypothesis for most time intervals. The SEES analysis is reported in [Friedman,1993a] and [Friedman,1993b].

DIS testing. CGF systems and the DIS networked simulation protocol are closely bound, not because CGF systems are only relevant to DIS, but because most existing CGF systems have been developed for use within DIS. The special characteristics of CGF systems affect the way they are tested, i.e. validated, for DIS compliance. Test procedures specific to CGF systems when testing for DIS compliance are given in [Vanzant-Hodge,1994a], where advantages and disadvantages specific to testing CGF systems are identified. The converse operation, using a CGF system as a tool in DIS compliance testing, is discussed in both [Loper,1993] and [Vanzant-Hodge,1994b]. The former reference observes that at the 1992 DIS Interoperability Demonstration the CGF systems tended to pass the DIS compliance tests more readily than the other simulator types.

Other techniques. A CGF system can be incrementally or partially validated through the inclusion of separately validated component models. As part of the Anti-Armor Advanced Technology Demonstration (A2ATD), a set of component models accredited by the Army Material Systems Analysis Activity (AMSAA) were incorporated into ModSAF. Those models were:

1. Direct fire delivery accuracy
2. Direct fire rate of fire
3. Direct fire vulnerability
4. Indirect fire vulnerability
5. Target acquisition
6. Mobility

[Courtemanche,1994] explains the models themselves and their incorporation into ModSAF. The model validation process is described in [Thomas,1995a] and [Thomas,1995b].

After the physical models were incorporated, ModSAF was further validated for A2ATD with a series of comparison trials. Two company-sized scenarios (a hasty attack and a hasty defense) were run 24 times each using ModSAF and the results were logged. The exercise outcomes were compared with the outcomes of the same scenarios produced earlier during the M1A2 Initial Operational Test and Evaluation (IOT&E), a live simulation test of a new variant of the M1 tank. The ModSAF outcomes were also compared to exercise outcomes from CASTFOREM (Combined Arms Support and Task Force Evaluation Model), a constructive force-on-force combat simulation that has been used by the Army for years. The comparisons showed some discrepancies, especially in ModSAF's tactical behavior, which were overcome via operator

workarounds. ModSAF was eventually validated for use in A2ATD. This experiment is discussed in [Harkrider,1995], [Thomas,1995a], and [Thomas,1995b].

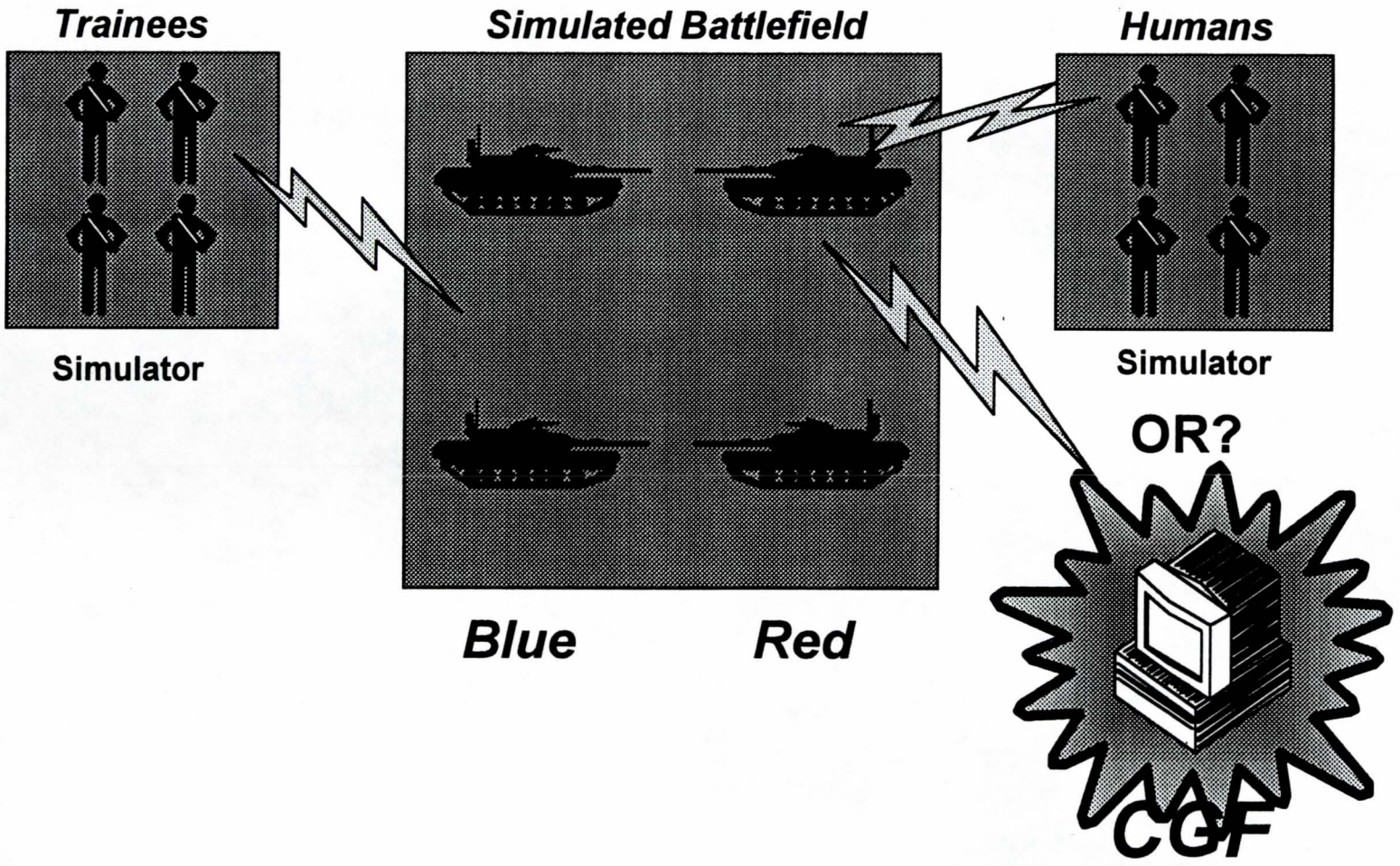
[Monday,1995] reports on automated test procedures for ModSAF. The procedures are of two types: regression testing, which is determining if new changes have compromised existing functionality, and VV&A testing, which is determining if the system conforms to specific modeling criteria. The later testing involves logging test runs and analyzing the logs for events of interest.

Common factors. The first attempts at formal VV&A of CGF systems are of interest because they provide guidelines and lessons that contribute to the design of an overall VV&A methodology for CGF systems. From those experiences, three common factors of general applicability to CGF VV&A can be discerned. First, note that in almost every case, the VV&A process proceeds by logging and then analyzing network traffic (DIS or SIMNET PDUs). This technique appears to be fundamental. Second, the CGF system's behavior is usually being compared with something else; either the behavior of humans, humans in a simulation, or some other simulation. The choice of what to compare the CGF system with seems to depend on the goal of the VV&A. Finally, all of these efforts are essentially "black box" validations, in that the CGF behavior is recorded and analyzed from the outside. Almost no organized CGF VV&A is done by validating the behavior specifications or the performance parameters which are input to the CGF system.

2.1.4.3 The CGF Turing Test

As alluded to earlier, it is important that the users of virtual battlefield simulation, i.e. the trainees who face CGF opponents, accept the behavior of those opponents as plausibly human and reasonably close to the doctrine of the enemy force being simulated. For that reason, many CGF researchers have suggested or assumed a CGF equivalent of the Turing test as a measure of CGF system quality or realism. The issue of the usefulness of a CGF Turing Test is examined in detail in [Petty,1994c] and [Petty,1995c]; those arguments are summarized here.

In [Turing,1950], Alan Turing proposed his now famous test of intelligence for non-human systems. The essential idea of the test, as it is commonly reformulated, is that a system is said to be intelligent if an observer can not reliably determine if its observed behavior is produced by the system or by a human. Whether or not the Turing Test is actually a valid test of intelligence is still hotly debated (e.g., see [Johnson,1992], [Harnad,1992], and [Shapiro,1992]); luckily, that question need not be answered here. Instead, we are concerned with a version of the Turing Test applied to CGF systems: *Can observers of entities in a simulated battlefield reliably determine whether any given entity is controlled by humans or by a CGF system?* (See Figure 2.9.) Like the original Turing Test, the CGF Turing Test is purely operational in that it deliberately ignores the question of how the CGF behavior is generated; it is interested only in the quality of the generated behavior.



2.9 The CGF Turing Test.

Who are the "observers" in this context? [Petty,1994c] and [Petty,1995c] define the trainees in the simulated environment as the observers, for two reasons. First, because the goal of a CGF system in a training simulation is to produce positive training benefits in its participants, a test for quality of the CGF system should be based on the participants. Second, the simulation participants are the observers most often identified by CGF researchers with invoking the CGF Turing Test.

An experimental application of the CGF Turing Test is described in [Potomac,1990] and summarized in [Wise,1991]. In the experiment, two platoons of soldiers fought a series of tank battles in the SIMNET simulation. In the battles, one of the platoons defended a position against an attack; the attacking tanks were controlled by either the other platoon of soldiers, a CGF system (the SIMNET SAF system, described later) or a combination of the two. Each platoon fought in two different scenarios against each of the three attacking forces, for a total of twelve engagements. The two platoons of soldiers had no contact with each other before or during the experiment outside of their simulated battlefield encounters. They were not able to correctly identify the attackers at a rate significantly different from random chance. Thus the CGF system of the experiment (the SIMNET SAF) appears to have passed the CGF Turing Test. The authors of [Wise,1991] seem to think so; they describe the experiment as evidence that "... it is plausible to conduct the Turing Test for computer generated forces, not just individual vehicles, and further that it is now possible to pass it."

Passing the CGF Turing Test is considerably easier than passing the original Turing Test, for at least three reasons. First, the domain of interactions between the observers and the CGF system is more limited; questions and answers are replaced by tactical actions and responses. Second, participants in battlefield simulation typically have a restricted view of the battlefield, severely compromising their ability to evaluate their opponents' actions for plausibility. For example, soldiers inside a simulated tank can only see the portion of the battlefield visible from their location through the narrow vision blocks of the tank. Aggravating the situation is that the enemy entities that the trainees might observe are normally doing their best to remain hidden! Finally, because the observer is a simulation participant, he or she is likely to be more intent on some battlefield activity, such as survival or destroying the opponent, than on observing the opponents' behavior for signs of artificiality.

Nevertheless, many CGF researchers argue or assert that it is sufficient, or at least necessary, for a useful CGF system to be able to pass the CGF Turing Test. A few examples from recent CGF research literature will illustrate this. [Wise,1991] asserts that "In designing computer generated forces, the ultimate goal is to simulate plausible human behavior, ...". According to [Bockstahler,1991], "... manned simulators and computer driven forces should be able to interact without the human operators being able to distinguish between manned or automated forces." [Harmon,1991] criticizes an existing CGF system because "... its performance falls far short of the goal of exhibiting behavior which is indistinguishable from that of humans for a wide range of common situations." One of the architects of the original CGF system, the SIMNET SAF, asserts in [Downes-Martin,1992] that the CGF forces are "...required to be indistinguishable from manned simulators." [Smith,1992b] says that "... simulated entities should be indistinguishable from manned simulators." [Braudaway,1993] states that "One challenge of these computer

generated forces is to emulate human behavior so that the human controlled and computer controlled entities are indistinguishable." [Marti,1994] praises paths generated by a route-planning algorithm by asserting that they "...are reasonably difficult to distinguish from human generated plans." [Katz,1994] uses the indistinguishability criterion in evaluating a CGF system. In [Weaver,1993], passing the Turing Test is presented as an "Underlying Principle" of CGF; the statement "The behavior of the CGF, as seen through the network, should be indistinguishable from that of human participants" is the first such principle listed. [Jones,1993b] ties CGF performance to the Turing Test as directly as possible. In that paper, the assertion "... automated agents should be indistinguishable from other human pilots taking part in the simulation." is immediately followed by the phrase "To construct such intelligent, automated agents ..."; thus the authors not only advance the Turing Test as valid for a CGF system, but also attribute intelligence to a CGF system that passes the test. [Deutsch,1993] and [Webber,1993] also emphasize the importance of realistic, human-like behavior by CGF entities, as perceived by users of the simulation. Additional examples are available, but this should be enough; many CGF researchers believe that the Turing Test is relevant, even central, to evaluating CGF systems. Looking at these statements another way, the authors are implying that passing the Turing Test strongly suggests, or even demonstrates, the quality of a CGF system.

Despite the cited opinions of the majority of CGF researchers, the position taken in [Petty,1994c] and [Petty,1995c] is that the CGF Turing Test is *neither necessary nor sufficient* to establish the quality of a CGF system. To show that this is so, two points must be made; first, that a CGF system that does pass the Turing Test might not produce positive training benefits (i.e. not sufficient), and second, that a CGF system that does not pass the CGF Turing Test can produce positive training (i.e. not necessary).

As for the question of sufficiency, consider two arguments. First, different armies use observably different tactical doctrines. For useful training benefit from a battlefield simulation, it is not enough that the opponent must behave like humans; it should act like humans that are employing the tactical doctrine of the enemy the soldiers are training to defeat. A CGF system that used generic tactics might not improve the training of soldiers who were to face a specific distinctive enemy. Second, consider the case of two groups of friendly soldiers opposing each other in a simulated battlefield. The soldiers would pass the CGF Turing Test by definition, but the resulting training experience would be suboptimal, and possibly negative, because each group of soldiers faced opponents who used friendly doctrine. Such training could be negative because the behaviors the soldiers might learn to defeat friendly tactics could be detrimental against the enemy's tactics.

As for the question of necessity, consider three examples of CGF systems that do not pass the CGF Turing Test but do produce positive training benefit. The first is in the area of "above real-time training". [Guckenberger,1992] describes an experiment in which subjects conducted tank gunnery in simulations that had varying levels of time acceleration (either 1x, 1.6x, or 2x real time). The targets in the simulations were produced by a simple CGF system. The subjects' gunnery skills at standard real-time were measured before and after the training. The accelerated conditions produced better training and transfer of the gunnery task than the standard real-time conditions. What does this have to do with the CGF Turing Test? Gunnery targets moving at 2x

real-time will not pass the CGF Turing Test, yet they did produce a positive training effect. Hence it is possible to produce positive training without passing the test.

A second example is the U.S. Army's Platoon Gunnery Trainer (PGT) training system. The PGT trains platoon gunnery and battle command skills. For several reasons, including the repetitive standardized scenarios, the PGT system's targets would not pass the CGF Turing Test. In spite of that, a careful study reported in [Sterling,1994] shows that improved performance in PGT scenarios is "substantially related" to performance in the U.S. Army's tank gunnery proficiency tests. In other words, though the PGT would not pass the CGF Turing Test, it does produce positive training.

A final example is the IST Semi-Automated Forces Dismounted Infantry (SAFDI) system. The SAFDI system is a version of IST's Computer Generated Forces Testbed that has been enhanced with capabilities and behaviors specialized for dismounted infantry fireteams [Franceschini,1994a] [Franceschini,1994b]. The SAFDI system was evaluated at the U.S. Army Infantry School's SIMNET site (at Ft. Benning, Columbus GA) by an independent team of experts in a series of training scenarios that included SAFDI-generated entities fighting with and against soldiers of the U.S. Army (A Company 1/29 Infantry) [Chervenak,1993] [D'Errico,1994]. While the evaluation did not include a formal CGF Turing Test, it is quite obvious that the SAFDI system would not have passed that test. The reasons include a visual fireteam icon that represents a five man fireteam with a single human icon, a lack of self-preservation on the part of the SAFDI entities, and blind adherence by SAFDI entities to doctrinal firing priorities regardless of available targets. Nevertheless, in spite of its inability to pass the CGF Turing Test the SAFDI system did produce positive training benefits. Those benefits are detailed in [Chervenak,1993] and [D'Errico,1994]. A telling evaluation was made by the commander of A Company 1/29 Infantry in an unsolicited memorandum that is included in the evaluation report; he said that "... the SAFDI greatly increased my unit's training."

From these examples it appears to be demonstrably possible to produce positive training benefits using a CGF system that does not pass the CGF Turing Test; i.e. passing the Turing Test is not necessary for CGF system quality. Taken with the previous arguments that passing the CGF Turing Test is not sufficient, the conclusion is that the test is neither necessary nor sufficient in training applications.

2.1.5 Key research directions for CGF

With the preceding comments in mind, it is now possible to ask directly: What are the key research areas upon which CGF technology depends? They include:

1. Planning of CGF actions
2. Model networks and variable granularity simulation
3. Knowledge base representation
4. Autonomous agent modeling
5. System and network architecture
6. Validation
7. CGF operator interface
8. Terrain representation and reasoning
9. Situational awareness and environmental monitoring
10. Advanced route planning, including formation movement
11. Real-time coordination of cooperative behavior
12. Intelligent target acquisition and selection
13. Adaptive (learning) behavior by CGF entities
14. Modeling fear, self-preservation, and fallibility in CGF
15. Behavior specification for CGF

The first seven topics are discussed in [Fishwick,1991]; some are first identified in [McKeown,1990]. One item on the list, terrain representation and reasoning, is the specific focus of this survey.

2.2 Existing CGF systems

This subsection lists and describes existing CGF systems. It begins with a compendium, or brief descriptions, of a set of CGF systems. The set is intended to be complete, though there may be CGF systems not included here. Following the compendium are detailed descriptions of some of the most important or interesting CGF systems.

Many constructive simulations include automated opponents. This document focuses on CGF systems for virtual environments, so those constructive systems are not included here. A partial list of constructive systems with automated opponents is given in the appendices.

2.2.1 A compendium of CGF systems

The CGF systems in the compendium are listed alphabetically by CGF system name. A set of data fields are given for each CGF system in the compendium; they are defined as follows:

Field	Definition
Name:	CGF system name
Developer:	Agency or company that is developing/developed the system
Started:	Year work began
Status:	Current development status
Language:	Programming language used for implementation
Operating system:	Operating system under with the CGF system runs
Computer:	Computer platform(s) upon which the CGF system runs
Domain(s):	Combat domain (ground, air, sea) of generated entities
Capacity:	Number of entities that can be generated in real-time
Protocols supported:	Network protocols (DIS and/or SIMNET) that the system can use
Primary use(s):	Uses of the system
Real-time:	Does the system run real-time?
Behavior specification:	Mechanism or formalism used to specify CGF behavior
Behavior generation:	Mechanism or algorithm used to generate CGF behavior
Terrain database:	Format of the terrain database
Terrain reasoning:	Major terrain reasoning capabilities of the system
Comments:	Remarks describing the system
Reference(s):	References for more detail

A data field is given as "na" (not available) if information for that field could not be obtained.

Name:	**A-6/A-14 Aircrew Trainer Suite**
Developer:	AAI
Started:	na
Status:	na
Language:	na
Operating system:	na
Computer:	na
Domain(s):	Air combat
Capacity:	120 "threats" which may be platforms, sensors, or weapons
Protocols supported:	None
Primary use(s):	Training system component
Real-time:	Yes
Behavior specification:	Expert system rules
Behavior generation:	Expert system rules
Terrain database:	na
Terrain reasoning:	na
Comments:	Current status not known. Generates both aircraft and surface-based anti-aircraft radar and weapons systems. The threat model evaluates the distance and involvement of a threat with the trainees' aircraft and varies the simulation fidelity of the threats as needed. Threat behavior is encoded as "reaction algorithms", which consist of production rule-like statements.
Reference(s):	System overview [Hunter,1991]

Name:	**Action/Cognition Behavior Model (ACBM)**
Developer:	BDM Federal
Started:	1973
Status:	Active
Language:	Various, recently C++
Operating system:	na
Computer:	na
Domain(s):	Air and ground combat
Capacity:	Over 200
Protocols supported:	DIS
Primary use(s):	Analysis system component
Real-time:	Yes
Behavior specification:	Virtual Battlespace Language
Behavior generation:	Action/Cognition Behavior Model
Terrain database:	Polygonal (SIF/HDI)
Terrain reasoning:	na
Comments:	ACBM, SWEG (Simulated Warfare Environment Generator), and CIMUL8 (among others) are members of a family of simulations developed by BDM since 1973 and used for a wide range of military analysis projects. They include complete battlefield simulation capabilities, not just CGF functions.
Reference(s):	See **2.2.8**

Name:	**Automated Force (AF)**
Developer:	Naval Postgraduate School
Started:	1993
Status:	Inactive
Language:	C and CLIPS
Operating system:	na
Computer:	na
Domain(s):	Ground combat
Capacity:	12
Protocols supported:	DIS
Primary use(s):	Research testbed
Real-time:	Yes
Behavior specification:	Expert system rules
Behavior generation:	Expert system rules
Terrain database:	na
Terrain reasoning:	Route planning
Comments:	Naval Postgraduate School CGF work has moved to ModSAF.
Reference(s):	1. NPSNET overview [Zyda,1992] 2. Description of AF reasoning capabilities [Pratt,1994a]

Name:	**CCTT SAF**
Developer:	Loral ADS, SAIC
Started:	1993
Status:	Under development
Language:	Ada
Operating system:	AIX
Computer:	IBM RISC System/6000
Domain(s):	Ground combat, some air combat
Capacity:	60
Protocols supported:	DIS
Primary use(s):	Training system component
Real-time:	Yes
Behavior specification:	Combat Instruction Sets and Expert System rules
Behavior generation:	Finite State Machines
Terrain database:	Gridded and quadtree (Model Reference Terrain Database)
Terrain reasoning:	Route planning Obstacle avoidance Intervisibility Cover and concealment
Comments:	Production CGF system.
Reference(s):	See **2.2.5**

Name:	**CGF Testbed**
Developer:	Institute for Simulation and Training
Started:	1989
Status:	Active
Language:	C
Operating system:	MS DOS
Computer:	IBM-compatible personal computers
Domain(s):	Ground and air combat, with some sea combat and electronic warfare capabilities
Capacity:	36-40
Protocols supported:	SIMNET and DIS
Primary use(s):	Research testbed, DIS testing
Real-time:	Yes
Behavior specification:	Finite State Machines
Behavior generation:	Finite State Machines
Terrain database:	Polygonal (SIMNET SAF)
Terrain reasoning:	Entity route planning Unit route planning Intervisibility Helicopter terrain avoidance
Comments:	The IST CGF Testbed is the only personal computer-based CGF system. It is widely used outside IST for DIS testing.
Reference(s):	See **2.2.2**

Name:	**Command Decision System**
Developer:	IBM Federal Systems
Started:	1992
Status:	Inactive
Language:	C, Lisp
Operating system:	AIX
Computer:	IBM RISC System 6000
Domain(s):	Ground combat
Capacity:	na
Protocols supported:	None
Primary use(s):	Research prototype
Real-time:	No
Behavior specification:	Rules, C and Lisp procedural code
Behavior generation:	A blackboard controller integrates results of disparate "knowledge sources", i.e. CGF behavior modules implemented as rule-based expert systems, Lisp functions, or C procedures.
Terrain database:	na
Terrain reasoning:	Route planning
Comments:	This system was a research prototype to test applicability of blackboard paradigm to CGF control. It was linked to IBM's Combined Arms Combat Simulator, a self-contained entity level simulation that performed simulation functions.
Reference(s):	1. System overview [Braudaway,1992] 2. Blackboard for CGF behavior control [Braudaway,1993]

Name:	**Intelligent Player (IP)**
Developer:	University of Alabama, East Tennessee State University
Started:	1989
Status:	Active
Language:	C
Operating system:	AIX
Computer:	IBM RISC System/6000
Domain(s):	Air combat (helicopter)
Capacity:	2
Protocols supported:	None
Primary use(s):	Research prototype
Real-time:	Yes
Behavior specification:	C code
Behavior generation:	Game tree lookahead
Terrain database:	Quadtree
Terrain reasoning:	Terrain avoidance
Comments:	This system is a research prototype to test the applicability of game tree lookahead to movement control of a CGF helicopter in air combat.
Reference(s):	1. System overview [Katz,1989] [Katz,1991] [Katz,1992] 2. Details of game tree lookahead mechanism [Katz,1993] 3. Using game tree lookahead in real-time [Schaper,1994] 4. Terrain reasoning with game tree lookahead [Pandari,1995]

Name:	**Interactive Tactical Environment Management System (ITEMS)**
Developer:	CAE Electronics
Started:	na
Status:	Active
Language:	FORTRAN
Operating system:	IRIX
Computer:	SGI Onyx
Domain(s):	Air, ground, and sea combat
Capacity:	100+ entities
Protocols supported:	SIMNET and DIS
Primary use(s):	Training system component, research testbed
Real-time:	na
Behavior specification:	Expert system rules
Behavior generation:	Expert system rules
Terrain database:	na
Terrain reasoning:	Terrain following and avoidance during flight Road following Intervisibility
Comments:	ITEMS emphasizes air entities and behaviors.
Reference(s):	1. System overview and expert system [Siksik,1993] 2. Explanation based learning within ITEMS [Kocabas,1995]

Name:	**MAXIM**
Developer:	Air Force Institute of Technology
Started:	1992
Status:	na
Language:	Lisp (CLOS)
Operating system:	Unix
Computer:	Sun SPARCstation 2
Domain(s):	Air to air combat
Capacity:	na
Protocols supported:	DIS
Primary use(s):	Research testbed, training system component
Real-time:	Yes
Behavior specification:	Universal plans expressed in Lisp
Behavior generation:	Universal plan selection
Terrain database:	None
Terrain reasoning:	None
Comments:	Current status unknown.
Reference(s):	Universal planning applied to air combat [Dyer,1993]

Name:	**ModSAF (Modular Semi-Automated Forces)**
Developer:	Loral Advanced Distributed Systems (LADS)
Started:	1992
Status:	Active
Language:	C
Operating system:	Unix
Computer:	Various; includes SGI, Mips, Sun Sparc, IBM RISC System 6000
Domain(s):	Ground combat and air combat
Capacity:	36-80
Protocols supported:	DIS and SIMNET
Primary use(s):	Training system component, research testbed
Real-time:	Yes
Behavior specification:	Combat Instruction Sets
Behavior generation:	Finite State Machines
Terrain database:	Gridded and quadtree (Compact Terrain Database)
Terrain reasoning:	Route planning Intervisibility
Comments:	Probably the most widely distributed and used CGF system.
Reference(s):	See **2.2.3**

Name:	**Piastre**
Developer:	SOGITEC
Started:	1989
Status:	Active
Language:	Kee Lisp
Operating system:	Unix
Computer:	Sun 3/160
Domain(s):	Ground combat
Capacity:	8
Protocols supported:	DIS
Primary use(s):	Training system component
Real-time:	Yes
Behavior specification:	Expert system rules
Behavior generation:	Expert system rules
Terrain database:	"Object oriented"
Terrain reasoning:	Intervisibility Route planning Road following
Comments:	This system controls opposing force targets for the French army's Leclerc tank turret simulator.
Reference(s):	1. System overview [SOGITEC,1989] [Kada,1994] 2. Expert system to generate CGF behavior [Huon,1989]

Name:	**Semi-Automated Forces Dismounted Infantry (SAFDI)**
Developer:	Institute for Simulation and Training
Started:	1992
Status:	Development complete, delivered to training sites
Language:	C
Operating system:	MS DOS
Computer:	IBM-compatible personal computers
Domain(s):	Ground and air combat
Capacity:	12
Protocols supported:	SIMNET
Primary use(s):	Training system component
Real-time:	Yes
Behavior specification:	Finite State Machines
Behavior generation:	Finite State Machines
Terrain database:	Polygonal (SIMNET SAF)
Terrain reasoning:	Entity route planning Unit route planning Intervisibility Helicopter terrain avoidance
Comments:	A specialized version of the IST CGF Testbed with enhanced and extended dismounted infantry capabilities. Delivered for use to SIMNET training sites to complement the existing SIMNET SAF.
Reference(s):	See **2.2.7**

Name:	**SimCore Tactics**
Developer:	Science Application International Corporation
Started:	na
Status:	na
Language:	na
Operating system:	na
Computer:	na
Domain(s):	Ground and air combat
Capacity:	na
Protocols supported:	DIS
Primary use(s):	Analytical system component
Real-time:	Yes
Behavior specification:	Finite State Machines
Behavior generation:	Finite State Machines
Terrain database:	na
Terrain reasoning:	na
Comments:	SimCore is an analytical-style simulation linked to DIS. It was used for the DARPA Warbreaker project.
Reference(s):	System overview [Aronson,1994]

Name:	**SIMNET SAF**
Developer:	Bolt, Beranek, and Newman Systems and Technologies
Started:	1986
Status:	In use as training system component, no further development
Language:	Lisp and C
Operating system:	Unix
Computer:	Symbolics and Masscomp
Domain(s):	Ground combat, with some air combat
Capacity:	40
Protocols supported:	SIMNET
Primary use(s):	Training system component
Real-time:	Yes
Behavior specification:	Combat Instruction Sets
Behavior generation:	Finite State Machines
Terrain database:	Polygonal and quadtree (SIMNET SAF)
Terrain reasoning:	Vehicle level obstacle avoidance Entity and unit route planning Intervisibility Concealment location
Comments:	This was the first production CGF system. It is still used daily at SIMNET training sites. Considerable operator control can be required for typical training scenarios. The ODIN SAF is an enhanced version of the SIMNET SAF.
Reference(s):	See **2.2.6**

Name:	**Soar/IFOR**
Developer:	University of Michigan
Started:	1993
Status:	Active
Language:	C
Operating system:	Unix
Computer:	SGI Indy 4400
Domain(s):	Air combat
Capacity:	4-10 aircraft
Protocols supported:	DIS (via ModSAF)
Primary use(s):	Research testbed, training system component
Real-time:	Yes
Behavior specification:	Search operators and rules
Behavior generation:	State space search and rule firing
Terrain database:	Gridded
Terrain reasoning:	Terrain avoidance
Comments:	A general AI system applied to air combat.
Reference(s):	See **2.2.4**

Name:	**Team Target Engagement System Computer Controlled Hostiles (TTES CCH)**
Developer:	Institute for Simulation and Training
Started:	1993
Status:	Active
Language:	C
Operating system:	OS/2
Computer:	IBM-compatible personal computers
Domain(s):	Ground combat; specifically individual combatants in urban terrain
Capacity:	12
Protocols supported:	DIS
Primary use(s):	Training system component
Real-time:	Yes
Behavior specification:	Finite State Machines
Behavior generation:	Finite State Machines
Terrain database:	Polygonal
Terrain reasoning:	Entity route planning Intervisibility
Comments:	TTES is a U.S. Marine Corps system for training individual Marines in both marksmanship and urban combat tactics. The CCH component provides individual combatants as opponents and neutrals. It is a development of the IST CGF Testbed. It has been extensively enhanced to generate individual combatants in urban combat.
Reference(s):	1. TTES overview and CCH requirements [Wysocki,1994] 2. Need for individual combatant simulation [OTA,1994] 3. Individual human figures in DIS [Reece,1994a] [Pratt,1994b]

2.2.2 CGF Testbed

2.2.2.1 Overview

Under the sponsorship of ARPA and STRICOM, IST has been conducting research in the area of CGF systems, seeking to increase the realism and autonomy of CGF behavior. A key product of that sponsorship is the IST CGF Testbed. The IST CGF Testbed is a low-cost CGF system that provides an environment for testing CGF behavioral control algorithms. It is documented in [Danisas,1990], [Gonzalez,1990], [Petty,1992c], [Smith,1992b], [Smith,1992c], and [Reece,1994b], and critiqued in [Booker,1993].

IST's research into Computer Generated Forces has had two primary goals: first, to increase the realism and autonomy of Computer Generated Forces (CGF) behavior through the application of artificial intelligence techniques, and second, the development and testing of efficient algorithms for CGF behavior generation and physical modeling.

2.2.2.2 System architecture

Because one of the goals of this research project was to demonstrate the feasibility of low-cost CGF systems, the IST CGF Testbed was developed and runs on IBM-compatible personal computers. A basic IST CGF Testbed installation consists of two standard IBM-compatible personal computers. Each runs one of the two software components of the CGF Testbed; those components are the "Simulator" and the "Operator Interface" (or OI).

The Simulator performs the computations for vehicle dynamics, battlefield environment simulation, and behavior generation for the computer controlled CGF entities. The OI provides an operator interface to the CGF system which consists of a plan view display battlefield map and a menu-based mouse and keyboard input system. The CGF operator enters commands for the CGF entities using the OI, which passes those commands to the Simulator, where they are executed by the CGF entities. The OI and the Simulator communicate via the main simulation network, exchanging packets which are sent point-to-point and are not part of the DIS or SIMNET protocol.

The basic configuration of one OI and one simulator can support approximately 40 CGF entities, which is roughly the number of vehicles in a military unit of battalion size. The simulation network is used for communication between the Simulator and the OI so as to permit easy scaling of the CGF Testbed. If more than 40 entities or more than one OI is required, additional personal computers may be attached to the network. The Simulator and OI software both adjust automatically to the presence of more than one of either component.

The Testbed software is written in ANSI C but is compiled with a C++ compiler to take advantage of the stronger type checking in that language. The IST CGF Testbed can be compiled so as to communicate on the simulation network using either the DIS or SIMNET network protocols.

2.2.2.3 Behavior generation

Like most other CGF systems, the IST CGF Testbed depends on the operator to perform high level planning and mission control. However, the IST CGF Testbed contains a range of flexible intermediate level behaviors that operate automatically and realistically.

One example of such intermediate level behavior is the Testbed's Fire_Weapons behavior for infantry fighting vehicles (IFVs). IFVs, such as the M2 Bradley or the BMP, typically have a variety of weapons systems, each suited to a particular type of target and tactical situation. M2 Bradleys are armed with a coaxial machinegun, used against infantry at short range, a 25mm cannon, used against infantry at longer range and vehicles other than tanks at short and medium range, and TOW anti-tank missiles, used against vehicles other than tanks at long range and tanks at all ranges. The IFV Fire_Weapons behavior performs the following actions without operator intervention:

(1) Scan the terrain for visible enemy targets.
(2) Select the most threatening enemy target from among those visible, based on threat analysis rules encoded in the behavior.
(3) If more than one enemy target falls into the same threat category, select one from among those available based on a fire distribution scheme that considers nearby friendly entities.
(4) Select the appropriate weapon for that target.
(5) Prepare the weapon for firing (aim the turret and possibly raise the TOW launcher).
(6) Fire the weapon and determine if a hit was scored.
(7) Reload the weapon.

For low level behavior, the CGF Testbed includes several excellent algorithms. For example, the route planner is fast and effective and will find routes around terrain obstacles. The primary means of behavior specification of the IST CGF Testbed is a code structuring technique based on finite state machines (FSMs). Behavior in the Testbed is encoded as algorithms, written in C. However, that C code is given structure using the FSM mechanism. The essential idea is that atomic units of behavior, implemented as C functions, become states in a FSM. The CGF Testbed's FSM mechanism was described in more detail earlier.

2.2.2.4 Status and applications

The IST CGF Testbed has been used as both an environment in which to conduct CGF research and as a basis or component of other systems. It continues to be actively used for both purposes.

Examples of the latter use include the Semi-Automated Forces Dismounted Infantry (SAFDI) system and the Team Target Engagement Simulation Computer Controlled Hostiles (TTES CCH) system, where the IST CGF Testbed was enhanced to produce specialized CGF systems, and the Integrated Eagle/BDS-D systems, where the Testbed serves as a crucial component in the linkage between a constructive and a virtual simulation.

Table 2.2 lists applications of the IST CGF Testbed. The SAFDI and TTES CCH systems are also described separately in this document.

System/Application	Application or Experiment	References
CGF Testbed	Low cost CGF testbed	[Danisas,1990] [Gonzalez,1990] [Petty,1992c] [Smith,1992b] [Smith,1992c]
	CGF behavior generation techniques	[Coleman,1990] [Gonzalez,1991] [Clarke,1991] [Fishwick,1991]
	Behavior specification languages for CGF tactics	[Smith,1993] [Petty,1993]
	Terrain reasoning for reconnaissance planning	[Van Brackle,1993a] [Van Brackle,1993b] [Petty,1994b]
	Execution control schemes for CGF systems	[Reece,1993]
	Computational geometry algorithms for intervisibility determination	[Petty,1992a] [Petty,1992b]
	Heuristics to reduce number of required intervisibility determinations	[Rajput,1994a] [Rajput,1995a] [Rajput,1995b]
	CGF aircraft flight dynamics	[Cimini,1992]
	Weapons system evaluation	[Karr,1993b]
	Verification of CGF behavior	[Petty,1994c] [Petty,1995c]
	Entity route planning in the presence of dynamic (moving) obstacles	[Craft 1994b] [Karr,1995a] [Karr,1995c]
	Unit route planning that considers terrain and enemy positions	[Rajput,1994b] [Karr,1995d]
	Comparison of A* and Iterative Deepening A*	[Karr,1995e]
	Precision gunnery target generation	[Krecker,1994]

Table 2.2 (Part 1 of 2) IST CGF Testbed applications.

System/Application	Description	References
CGF Testbed (continued)	Experimental evaluation of the Ada programming language for CGF system implementation	[Craft,1994a] [Craft,1995a]
	Development of DIS standards for Electronic Warfare PDUs	[McDonald,1993] [Wood,1994] [Wood,1995]
Semi-Automated Forces Dismounted Infantry (SAFDI)	Specialized CGF system for dismounted infantry	[Petty,1991] [Karr,1992a] [Petty,1992d] [Franceschini,1993a] [Franceschini,1993b] [Chervenak,1993] [Petty,1994a] [Franceschini,1994a] [Franceschini,1994b] [D'Errico,1994] [Parra,1994]
Integrated Eagle/BDS-D	Integration of constructive and virtual simulations	[Karr,1992b] [Franceschini,1992] [Powell,1993] [Karr,1993a] [Karr,1994a] [Karr,1994b] [Root,1994] [Franceschini,1995a] [Franceschini,1995b] [Franceschini,1995c] [Franceschini,1995d] [Stober,1995] [Petty,1995d]
	Terrain avoidance algorithms for CGF helicopters	[Schricker,1995a]
	Performance metrics for CGF systems	[Schricker,1995b]
Team Target Engagement Simulation Computer Controlled Hostiles (TTES CCH)	CGF systems for individual combatants in urban terrain	[Wysocki,1994] [Reece,1994a]
DIS Testbed	DIS compliance testing	[Loper,1993] [Vanzant-Hodge,1994a] [Vanzant-Hodge,1994b]
	DIS standards for Simulation Management PDUs	[Cox,1995]
	DIS standards for Laser PDUs	[Giroux,1995]

Table 2.2 (Part 2 of 2) IST CGF Testbed applications.

2.2.3 ModSAF

2.2.3.1 Overview and capabilities

ModSAF (for Modular Semi-Automated Forces) is being developed by Loral Advanced Distributed Systems (Loral ADS) under the sponsorship of ARPA and STRICOM. ModSAF is the intellectual descendant of earlier CGF systems developed by the Loral ADS Semi-Automated Forces group (previously part of Bolt, Beranek, and Newman) including the SIMNET SAF and the ODIN SAF. ModSAF serves as both a working CGF system, capable of populating a virtual DIS battlefield with large numbers of computer controlled entities, and as a tool and framework for CGF research. [BBN,1992], [Booker,1993], [Ceranowicz,1994a], and [Ceranowicz,1994b] give overviews of ModSAF's structure and capabilities. [Courtemanche,1995a] reports on the recent ModSAF developments, including several new platoon and company level behaviors.

According to [Ceranowicz,1994b], version 1.2 of ModSAF can generate any of 24 different DIS entity types, including fixed and rotary wing aircraft, tanks, infantry fighting vehicles, other vehicles, and groups of dismounted infantry. ModSAF can also control platoon and company sized units.

The ModSAF software is written in C. It runs on Unix workstations, typically Silicon Graphics and Sun systems. [Vrablik,1994] estimates the capacity to simulate entities of a single SAFsim on such a workstation at between 36 and 80 vehicles, depending on system configuration and network load.

2.2.3.2 System architecture

ModSAF has three primary software components: the "SAFstation", which is an operator interface allowing a human operator to direct the ModSAF entities; the "SAFsim", which simulates the entities, units, and environmental processes; and the "SAFlogger", which logs, compresses, and plays back exercise network traffic. The SAFstation and SAFsim will be discussed in more detail later.

The ModSAF SAFsim component simulates all of the vehicles and units generated by ModSAF. For entities, it performs both physical simulation (e.g. vehicle dynamics and weapons effects) and behavioral simulation (e.g. route planning and mission execution). For units, only behavioral simulation is required. The SAFsim is a "real-time time-stepped simulation" [Ceranowicz,1994b]. This means that ModSAF does not attempt to enforce a constant update rate for its generated entities; rather, the update rate varies depending on simulation load. The DIS protocol's remote entity approximation helps to make this approach possible.

Two significant features of ModSAF's architecture distinguish it from its predecessors; the first, its modular software structure, makes ModSAF useful as a CGF research tool and DIS system component; the second, the Persistent Object Protocol, provides important performance, checkpointing, and fault-tolerance capabilities. Each will be discussed in turn.

Nearly all of the ModSAF software is implemented as library modules. The modules have precisely defined and documented public interfaces and access to the data structures and routines of the modules is through those interfaces. Software layering techniques are used to reduce module interdependence and binding. Sets of related modules are grouped into libraries, each of which provides a service such as vector math utilities, network interface, map drawing, physical simulation, or behavior simulation [Ceranowicz,1994b]. The goal of this structure is that the ModSAF libraries constitute a "repository of useful capabilities" that can be used in different ways in different DIS and CGF systems, and easily replaced by researchers experimenting with different or improved ways to provide CGF functionality.

The modular structure is exploited to make it easy to add new entity types (e.g. a new tank) to ModSAF's repertoire. It is possible to define in a parameter file which of ModSAF's set of entity simulation modules will be used for the new entity. A variety of entity simulation modules are available in several categories: dynamics models, turret models, weapons models, sensor models, and damage models. If a new entity type cannot be assembled from the existing modules, new ones can be developed and linked in using the parameter file.

The Persistent Object (PO) Protocol is used by ModSAF to transmit information about entities it is controlling on the network. The PO Protocol, which is not part of the DIS protocol, supplements the physical state data normally present in the DIS packets. It includes information about the entities' behavioral state, including the entities' missions, tasks, and status. The PO Protocol packets also describe in a similar fashion the state of ModSAF units.

The ModSAF SAFsims on a network all maintain PO databases that contain the information received from the network about the ModSAF entities and units via the PO Protocol. Because of this, if a SAFsim should fail during the exercise, other SAFsims on the network can take over simulation of the entities previously simulated on the failed node. Load balancing of simulated entities between SAFsims is automatic, both at scenario start-up and during execution.

ModSAF uses the Compact Terrain Database (CTDB) format to represent the battlefield terrain. CTDB is a compressed representation based on elevation posts for most of the terrain surface, supplemented with explicit polygons in areas where greater detail is needed (microterrain) [Smith,1992a]. The CTDB also includes terrain features such as buildings, roads, rivers, trees, treelines, and so on. It is used by ModSAF for point elevation lookup, vehicle orientation, intervisibility calculation, and generation of graphic representations of the terrain [Stanzione,1993]. (The reference actually presents the terrain representation and reasoning approaches in the ODIN SAF system, a predecessor of ModSAF; most of the content also applies to ModSAF.) An updated description of the CTDB format and capabilities can be found in [Smith,1995b]. The CTDB format will be covered in considerably more detail later in this document.

ModSAF relies on a human operator for two functions; first, to set up preplanned missions for ModSAF entities and units, and second, to provide supervisory control of the simulated entities. The SAFstation component allows the operator to perform those functions. The SAFstation software is written in C using X-Windows and Motif. It provides a 2-dimensional on-screen map

of the virtual battlefield that shows the terrain and the entities. Terrain analysis tools for intervisibility evaluation and terrain cross-section display are available on the map. Drawing tools permit the operator to create movement routes and military control measures, such as phase lines, control points, and battle positions, and associate them with specific locations in the terrain.

Using the SAFstation, the operator can create preplanned missions for ModSAF vehicles and units. A mission is divided into a number of phases; for each phase the operator defines the tasks a unit is to perform and the criteria for a transition to the next phase. The transition criteria may involve the geographic control measures. This data is entered into an execution matrix. (Tasks and missions and how they control unit and vehicle behavior will be explained later.)

The operator can also give commands for immediate execution by entities and units via the SAFstation interface. Such intervention may be necessary when ModSAF's automated decision logic is not handling a situation correctly or when a scenario calls for a specific event that must be arranged by the operator. The operator may modify executing preplanned missions or replace them with new tasks. [Ceranowicz,1994c] presents ModSAF's operator command capabilities and the execution matrix in more detail.

2.2.3.3 Behavior generation

A stated goal of the ModSAF behavior specification and generation mechanism, described in [Calder,1993], is to provide a framework for expansion by the developers of ModSAF and an environment for research by other CGF researchers. The mechanism is intended to give a useful structure in which to place behavior generation algorithms without overly constraining the implementation of those algorithms. The presentation of the mechanism given here is based on [BBN,1992] and [Calder,1993].

ModSAF processes each of the entities it is generating in a cyclical fashion, continuously looping through its entity list and processing each one. Going through the entity list once is referred to as a *tick*, and performing the computation needed for a single entity is called *ticking* the entity. As each entity on the list is ticked, all of the processing needed for that entity is completed. If the entity is very busy, e.g. moving, scanning for targets, and handling incoming events, the processing associated with its tick may be large. If it is not busy, as would be the case for a destroyed tank, its tick will be small. Because ModSAF's scheduler is non-preemptive each entity's tick takes as long as needed. The frequency at which an entity is ticked is an indication of run-time load in ModSAF [Vrablik,1994].

Figure 2.10 shows the processing steps for a single entity tick. First, the entity's movement dynamics are computed. Second, incoming event PDUs associated with the entity (e.g. a weapons impact) are resolved. Then the entity's sensor scans (e.g. intervisibility determinations) are processed.

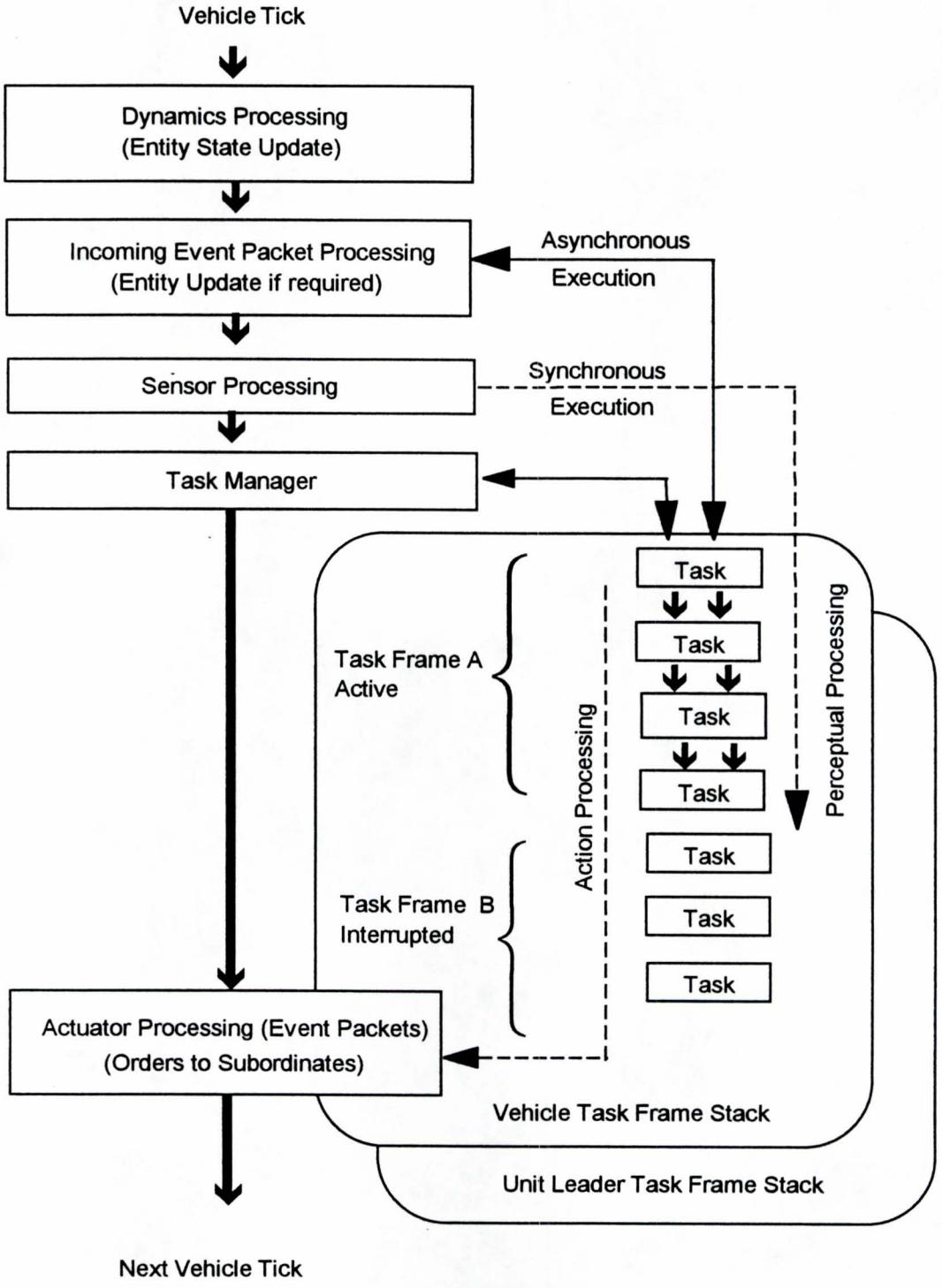


Figure 2.10 MODSAF entity processing [BBN,1992].

Following those steps the entity's behavior is generated in two parts; first, the task manager determines what tasks are active for the entity, and second, the currently active tasks are executed to generate the entity's behavior. The basic building block of ModSAF's behavior generation mechanism is the *task* (which is not to be confused with a task as commonly defined in the context of operating systems). A task is a single non-composite behavior performed by an entity or unit. There are several types of tasks in ModSAF:

1. Entity tasks
2. Unit tasks
3. Reactive tasks
4. Enabling tasks
5. Arbitration tasks

Entity tasks. Behavioral tasks that an individual entity (vehicle or fireteam) will perform. Entity tasks often use information from and control the physical subsystems (sensors and weapons) of the entity. Example entity tasks are Follow Route, Keep Formation, Avoid Collisions, and Spot Enemy Vehicles.

Unit tasks. Behavioral tasks associated with military units. Units tasks often create, monitor, and delete tasks for subordinate units and vehicles, following the hierarchical structure of military units. Example unit tasks are Company Road March, Company Attack, and Platoon Bounding Overwatch.

Reactive tasks. Behavioral tasks used to trigger reactions to battlefield events and situations. Example reactive tasks are Air Raid Happening, Target Meets Commit Criteria, and Hasty Attack Needed.

Enabling tasks. Behavioral tasks used to trigger mission contingencies. They are defined during the construction of a mission by the operator and allow conditional response by a unit to events during the mission. Example enabling tasks are Crossed Phase Line, Detected Enemy Unit, and Reached H-Hour Time.

Arbitration tasks. Special tasks that arbitrate between differing behavioral recommendations from multiple simultaneously active tasks. Example arbitration tasks are Vehicle Movement Arbitration, Vehicle Sensor Arbitration, and Vehicle Targeting Arbitration.

Tasks take as input task parameters which control the execution of a task within a particular mission. For example, the Follow Route task would have a route as a task parameter.

Tasks are implemented within ModSAF as augmented, asynchronous, finite state machines (AAFSMs). An AAFSM realizes a task as a set of states, each encoding a component action of the task, a set of transfer functions that determine and cause transitions between the states, and a set of inputs and outputs for the task. The states and transfer functions are implemented as C code.

A given entity or unit may have more than one task active for it simultaneously; for example, a tank might be executing both Follow Route and Spot Enemy Vehicles. A group of one or more tasks which are to be run at the same time is referred to as a *task frame*. Task frames are assembled by the operator while defining a mission by picking from the available tasks. Each task frame represents a phase of a mission. Specific task parameters for the tasks within a task frame are connected to the tasks by the operator when the mission is assembled.

The *task frame stack* is a run-time data structure that represents the set of task frames assigned to an entity or unit. The task frame at the top of the stack is the set of tasks that the entity or unit is currently executing. As the entity or unit changes from one phase of a mission to the next the top task frame is replaced with the task frame defined for the next phase (changes from one mission phase to another are detected by enabling tasks).

In addition to replacing the top task frame, new task frames may be pushed onto the stack in two ways; some battlefield condition may trigger a reactive task frame, or the operator may issue an immediate command which is assigned to the entity or unit as a task frame. In those cases, the previously executing task frame (and thus its component tasks) are suspended while the newly pushed task frame executes. When it completes it is removed from the task frame stack and the suspended task frame resumes.

The task frame stack also supports transparent task frames, which are task frames that, when pushed onto a stack, do not suspend the entire task frame below it. Tasks in a transparent task frame are merged with the tasks in the task frame below and all are executed simultaneously.

A *mission* is a set of task frames that are linked together in a sequence. A mission is divided into phases, each of which is associated with a task frame. Each task frame specifies a task in the previous task frame that must complete before it begins. Transitions from one phase to the next may also be triggered by enabling tasks. Enabling tasks link the task frames that make up the various alternate courses of action in the mission.

The *task manager* controls the execution of tasks within this structure. It determines which tasks to run for an entity or unit each time that entity or unit is updated. It also manages the task frame stack and handles the transitions between phases by pushing and popping tasks frames on the stack.

Note how this behavior generation mechanism attempts to meet the goal of the ModSAF developers to provide a framework for other CGF researchers and developers. An outside researcher or developer may implement a new algorithm for a specific task and embed it in the task and task frame control structure, thereby freeing the him or her from worrying about those issues outside the task. On the other hand, a researcher interested in experimenting with some other execution control scheme for CGF systems (e.g. [Reece,1993]) would have a difficult time doing so within ModSAF.

2.2.3.4 Status and applications

More detailed descriptions of specific aspects of ModSAF are available:

1. Terrain representation format and capabilities [Smith,1992a] [Smith,1995b]
2. Finding cover and concealment [Longtin,1994]
3. Algorithms to find and select concealed movement routes [Longtin,1995]
4. Other terrain reasoning functions [Stanzione,1993]
5. Near-term entity movement control [Smith,1994]
6. Simulation of missiles in ModSAF [Courtemanche,1995b]
7. ModSAF system capacity in terms of entities [Vrablik,1994]
8. VV&A of ModSAF [Courtemanche,1994] [Harkrider,1995] [Thomas,1995a] [Thomas,1995b] [Vaden,1994] [Meliza,1995] [Courtemanche,1995a]
9. Operator interface and execution matrix [Ceranowicz,1994c]
10. Automated testing and VV&A for ModSAF software integrations [Monday,1995] [Courtemanche,1995a]
11. Application of ModSAF's FSM mechanism to company control [Pratt,1995a]
12. An incomplete automatic company order generation capability [Pratt,1995b]

ModSAF has reached a level of maturity and stability that makes it a useful basis for CGF research or a component of other projects. Consequently, it has been widely distributed throughout the DIS community and at least eleven different agencies have developed ModSAF capabilities. The software development and integration process used to control and integrate the code produced by those agencies is summarized in [Courtemanche,1995a].

Table 2.3 lists a number of ModSAF applications and gives references for them. One ModSAF application, the Soar/IFOR project, is important enough that it is also described separately in this survey.

System	Application	Reference(s)
ModSAF	Prototyping environmental extensions to DIS such as atmospheric haze, battlefield obscurants, smoke, rain, and vehicular dust clouds	[Schaffer,1994] [Haque,1995] [Robasky,1995]
	Test environment for detailed verification of CGF behavior and performance	[Vaden,1994] [Meliza,1995]
	Automated military mission planning	[Sherman,1994] [Mohn,1994]
	Simulation-based unit mission planning	[Karr,1995b]
	Unit mission planning and run-time control using game tree lookahead	[Katz,1994]
	CGF design prototype for CCTT SAF	[Pope,1995a] [Pope,1995b]
	Development environment for company and battalion command agents that learn tactical rules	[Hille,1994] [Hieb,1995] [Hille,1995]
	Test environment for Prolog-based backward reasoning from goals for unit control	[Kwak,1995]
	Incorporation of previously validated and accreditated combat models	[Courtemanche,1994]
	Test environment for case-based learning and behavior based control for air combat	[Keirsey,1994]
	Baseline infrastructure layer for executing generated by Command Forces command entities	[Salisbury,1995]
	Test environment for DIS based on multicast instead of broadcast UDP/IP	[Smith,1995a]
	Test environment for target threat evaluation using fuzzy sets	[Cisneros,1995]
	Test environment for methodologies to compare different vehicle variants using simulation	[Craft,1995b]
	Test environment for a distributed control architecture for cooperative behavior based on formal finite state machines	[Rajput,1995b]

Table 2.3 (Part 1 of 2) ModSAF applications.

System	Application	Reference(s)
LeatherNet	Implementation of Marine Corps individual combatants (simulation and behavior)	[Howard,1995] [Hoff,1995]
DIS Common Database Toolset	Training scenario development and testing	[Butler,1995a] [Butler,1995b]
Corps Level CGF	Integration of constructive and virtual simulation	[Calder,1994] [Raytheon,1994] [Calder,1995a] [Calder,1995b] [Stober,1995]
Anti-Armor Advanced Technology Demonstration (A2ATD)	Weapons system evaluation	[Courtemanche,1994] [Harkrider,1995] [Thomas,1995a] [Thomas,1995b]
Soar/IFOR	Integration of CGF with an AI system for tactical behavior control	[Jones,1993b] [Johnson,1994] [Jones,1994b] [Jones,1994c] [Koss,1994] [Laird,1994] [Rosenbloom,1994] [Rubinoff,1994] [Schwamb,1994] [Tambe,1994] [van Lent,1994] [Laird,1995] [Lehman,1995] [Nielsen,1995] [Tambe,1995a] [Tambe,1995b]

Table 2.3 (Part 2 of 2) ModSAF applications.

2.2.4 Soar/IFOR

Soar is an integrated cognitive architecture based on heuristic state space search and automated learning [Laird,1987]. ModSAF and Soar are being linked in the Soar/IFOR project. The project is developing autonomous intelligent agents for air combat. The project originally was focused on agents for naval air engagements beyond visual range (the system was known as TacAir-Soar at that time); recently it has expanded its goals to include the development of agents for nearly all air missions flown by the U.S. military [Laird,1995].

In the Soar/IFOR system, ModSAF provides the DIS network interface, aircraft dynamics simulation, and low-level behavior execution functions, while Soar provides high-level planning and decision making for the ModSAF-controlled aircraft.

According to [Jones,1993b], the long-term intent of the Soar/IFOR project is to develop capabilities that will:

1. Integrate planning and reactive behavior in real-time
2. Respond to unexpected situations
3. Learn from experience
4. Exhibit human-like cognitive limitations

The Soar/IFOR system can control 4-10 aircraft (depending on network load) operating alone, in sections (2 aircraft), and in divisions (4 aircraft). They can fly a variety of formations and missions including several types of air-to-air and air-to-ground missions.

Soar divides knowledge into problem spaces, which are the state spaces through which Soar searches to find a plan (a sequence of operations to reach a goal state). In CGF terms, the operators are the actions that the CGF entity might perform. When Soar cannot find a plan, it creates a subgoal and shifts to a new problem space, finer grained and more specific to the subgoal, and searches that problem space. This subgoaling happens automatically. Note that this produces a hierarchy of goals. In contrast to hierarchical goals, tactical behavior can often produce conflicting goals (e.g. "destroy bogey" vs. "survive"). Soar is being modified to deal with conflicting goals.

Soar's state space search is guided and supplemented by a rule based component. Soar's rules test the current situation and propose operators, determine preferences between multiple proposed operators, perform procedures associated with the operators, and test that all parts of an operator have been executed. As part of the Soar/IFOR project, a body of search operators and rules for air combat have been developed and encoded in the Soar system. [Laird,1995] reports that 320 operators and 3100 rules have been developed to date.

The Soar/IFOR system has been extensively documented; numerous references are available providing both project overviews and detailed explanations of specific aspects of the system. They include:

1. Project overview [Jones,1993b] [Rosenbloom,1994] [Laird,1995]
2. Autonomous agents that explain their actions [Johnson,1994]
3. Generating agent behavior in response to interacting, and possibly conflicting, goals [Jones,1994b]
4. Operator interface design and development [van Lent,1994]
5. Natural language communication between Soar/IFOR agents (entities) and human participants in an exercise [Rubinoff,1994] [Lehman,1995]
6. Interfacing Soar with ModSAF [Schwamb,1994]
7. Integrating world state information from a variety of sources [Jones,1994c]
8. Coordinating tactics among multiple agents [Laird,1994]
9. Tracking and inferring events from observable cues with internal models of other agents' thought processes [Tambe,1994] [Tambe,1995b]
10. Development of a hypertext system for eliciting and communicating subject matter expertise [Koss,1994]
11. Development of operators and rules for rotary wing aircraft behaviors [Tambe,1995a]
12. Command and control agents for air mission controllers of various types [Nielsen,1995]

2.2.5 CCTT SAF

2.2.5.1 Overview and capabilities

The U.S. Army's Close Combat Tactical Trainer (CCTT) project is a large simulation development effort. CCTT is the first component and the technology base of the Army's Combined Arms Tactical Training (CATT) system, a massive DIS-compatible simulation system for training soldiers from several branches of the Army. CCTT will be a networked real-time battlefield simulation that includes ground and air combat vehicles, dismounted infantry, other weapons systems, and command and control elements. It will be a production system, intended for daily use as a battlefield training environment. CCTT will be fully DIS compliant; in fact, the needs of CCTT are driving large parts of the DIS standardization efforts. [Pope,1995a] and [Pope,1995b] give technical overview of CCTT; the former also addresses how CCTT's requirements are affecting the DIS standards.

CCTT will include a CGF system to serve the essential CGF function of populating the battlefield with computer controlled enemy and supplemental friendly forces. The CCTT CGF system is known as the "CCTT Semi-Automated Forces" or "CCTT SAF". Much of this presentation is based on information in [Marshall,1994].

It is important to observe that as a component of CCTT, the CCTT SAF is a production system, with stringent requirements for performance, reliability, and behavioral fidelity that are not necessarily applied to research oriented CGF systems. Table 2.4 lists some of the requirements for the CCTT SAF.

Characteristic	Requirement
Different entity types	
Friendly forces	53
Opposing forces	47
Miscellaneous	117
Combat Instruction Sets	
Friendly forces	700
Opposing forces	500
Terrain and environment	
Features	High density
Weather	Rain, Haze, Fog, Cloud
Visibility effects	Continuous time of day, Tactical smoke, Flare illumination
Tactical realism	
Representational scope	From entity to Friendly battalion and Opposing regiment
Traceability	All entity models and unit behaviors
VV&A	All entity models and unit behaviors
SAF Simulation features	
Control options	Autonomous, Operator, Command from simulator
Autonomous behaviors	Obstacle avoidance, Detect opposing forces
Miscellaneous features	Damage effects, Stochastic failures, Jamming, Logistics

Table 2.4 CCTT SAF requirements (adapted from [Marshall,1994]).

The CCTT SAF runs on IBM RISC System/6000 workstations under the AIX operating system. One such workstation is planned to support approximately 60 simulated entities. As a Department of Defense production system, the CCTT SAF software is written in Ada. The reader interested in simulation software development in Ada is referred to:

1. General overview [Devlin,1990]
2. In-depth coverage of a large Ada simulation project (the B-2 Aircrew Training Device) [Gross,1991] [Zink,1991] [Bedford,1991] [McMahon,1991] [Weiss,1991] [Croucher,1991]
3. CGF system development in Ada [Craft,1994a] [Craft,1995a]

2.2.5.2 System architecture

To a certain extent, the Loral ADS ModSAF system is serving as a design prototype for the CCTT SAF. According to [Pope,1995a] and [Pope,1995b] the CCTT SAF is "based fundamentally" on the ModSAF model and ModSAF architectural ideas and algorithms are being adapted whenever possible for the CCTT SAF.

Like the IST CGF Testbed and ModSAF, the CCTT SAF consists of two main software components. The "SAF Workstation" is an operator interface that allows the operator to monitor and control the CCTT SAF generated entities and the "CGF Simulator" performs the dynamics and behavioral simulation for those entities.

The SAF Workstation uses X-Windows and Motif as the basis for its operator interface. The interface provides an electronic map of the battlefield and accepts keyboard and mouse input. The designers of the SAF Workstation's User Computer Interface worked to consider the expected demands on the operator (e.g. controlling up to 120 entities at one time) and the lessons learned from previous CGF system operator interfaces. It includes a Unit Editor to create and modify hierarchical unit relationships, an Overlay Editor to prepare geographic input to mission plans using military overlay symbols, and an Exercise Editor to record preplanned behaviors for mission phases.

The CGF Simulator is built around the CGF Application, a single AIX process. The CGF Application is organized as five software modules: Terrain, Behaviors, Vehicle Simulation, DIS Manager, and SEOD Manager.

The Terrain module processes the terrain database and provides information about the terrain to the other modules. The CCTT SAF uses a terrain format known as the "Model Reference Terrain Database" (MRTDB). The MRTDB is a further development of ModSAF's CTDB format [Smith,1992a], with extensions and enhancements designed to conserve space and add features specifically required for CCTT. The MRTDB format is described in [Watkins,1994]; some of the terrain reasoning challenges that the MRTDB was designed to address are defined in [Campbell,1994].

The Vehicle Simulation module performs physical modeling and vehicle dynamics simulation of the CCTT SAF entities. The software design of this module depends on the use of generic data

driven models for common entity components. Entity models are composed of parameterized sets of generic models for hulls, turrets, sensors, weapons, resources, and communications. The physical models attempt to use a continuous simulation approach as much as possible.

The DIS Manager module sends DIS network packets for the CCTT SAF entities and receives the packets for entities external to the system. It also performs DIS interface functions such as remote entity approximation.

The SAF Entity Object Database (SEOD) is a run-time database used by CCTT simulator nodes (including the CCTT SAF) to "store, retrieve, modify, delete, and share CGF command and control information among multiple simulation participants" [Horan,1994]. It is similar in intent to ModSAF's Persistent Object Protocol. The simulator nodes maintain the SEOD's database (each has an identical copy) by way of network traffic sent between nodes. The SEOD database contains information about CGF entities and their behavior states. It can be thought of as both a distributed entity database, with its content updated via SEOD network traffic, and an interface through which applications share command and control information. The SEOD is the medium used by the SAF Workstation and the CGF Simulator to communicate with each other; orders assigned at the SAF Workstation for a unit are placed in the SEOD for that unit to respond to. The SEOD module is the application program interface between the node and the SEOD. SEOD data can be saved and used to restart CCTT scenarios.

The Behaviors module produces unit and vehicle level behavior. Its functioning will be described in detail later.

2.2.5.3 Behavior generation

Because the CCTT SAF is a production system, the behavior generated by it for its controlled entities must adhere to strict standards of tactical fidelity. Furthermore, each behavior generated must be traceable, in the sense that it must be possible to establish the source of a behavior. In response to these requirements, the CCTT SAF developers have instituted a methodical and rigorous knowledge engineering process to identify, specify, and document the tactical behaviors for the CCTT SAF that will be implemented within the system. CCTT CISs include both low-level (entity), doctrine-independent behaviors such as "Fire at target" and "Follow route" and higher-level (platoon) doctrine-dependent behaviors such as "Execute bounding overwatch" [Marshall,1994].

Tactical behavior in the CCTT SAF is specified as CISs (defined earlier). The source data for the CCTT SAF CISs for U.S. forces is the U.S. Army's Training Evaluation Program (ARTEP) doctrine, and the derived CISs are traceable back to doctrinal elements expressed in the ARTEP material. Enemy force CISs are based on Russian tactical doctrine, and Russian military manuals and journals are used as source documents. Every CIS includes, in its structured English representation, pointers to the source documents upon which the behavior is based. [McEnany,1994] discusses the knowledge engineering process of translating CIS from source document to structured English, and estimates the number of CISs to be documented and implemented for the CCTT SAF at 1200 (700 U.S. and 500 enemy).

Once the CISs are expressed in structured English form, they must be translated into program code. In the CCTT SAF, CISs at the vehicle level are implemented as Finite State Machines (FSMs) within Ada packages. Both doctrine-independent and doctrine-dependent CISs are implemented in this way. [Ourston,1995] elaborates on how the English CISs are implemented as software.

At the platoon level and above, a different implementation technique is used. An expert system and tactical rule database is used to select a CIS for execution by a platoon. CISs may be selected based on an operations order (entered through the SAF Workstation), a direct command from the operator, or a development in the battlefield situation. Once a CIS is selected for a platoon it is executed by the FSM mechanism. [Bimson,1994] describes how the expert system makes the selection and [Ourston,1994] explains how the expert system is integrated with the algorithmic CIS execution.

2.2.5.4 Status and applications

More detailed descriptions of specific aspects of the CCTT SAF are available:

1. CCTT SAF system architecture [Marshall,1994]
2. Current CCTT SAF terrain representation format [Watkins,1994] [Campbell,1994]
3. Relationship of CCTT and the CCTT SAF to DIS standards [Pope,1995a] [Pope,1995b
3. Route planning [Campbell,1995]
4. SEOD [Horan,1994]
5. Behavior specification methodology [McEnany,1993] [McEnany,1994] [Ourston,1995]
6. Expert system for tactical decision making [Bimson,1994] [Ourston,1994]

Like the overall CCTT system, the CCTT SAF is still under development, and therefore the information given is subject to change.

2.2.6 SIMNET SAF

As mentioned earlier, SIMNET was the first production version of a networked virtual battlefield simulation. SIMNET included a CGF system developed by Bolt, Beranek, and Newman known as the SIMNET SAF (Semi-Automated Forces).

The SIMNET SAF system uses two minicomputers and a single human operator to generate and control up to approximately 40 vehicles. It connects to the SIMNET network, generates network packets appropriate for the vehicles it is simulating, and processes incoming network packets for other vehicles in the simulation. It uses a version of the SIMNET terrain database that correlates to those used in the crewed simulators.

The SIMNET SAF system controls tanks (e.g. M1s and T-72s), infantry fighting vehicles (M2 Bradleys and BMPs), and other similar vehicles in the SIMNET battlefield. Those vehicles can oppose or cooperate with the vehicles controlled by the human users of SIMNET. They have a repertoire of autonomous behavior that includes following preplanned routes, simple route

planning, automatic formation changes in response to geographic control measures or battlefield events, target selection, and direct fire. However, the SIMNET SAF system has considerable dependence on a human operator for both high-level behavior, such as mission planning, and very low-level behavior, such as route planning across bridges.

The SIMNET SAF was a noteworthy accomplishment in its time, but its capabilities have been surpassed by more recent CGF systems, including ModSAF, its direct descendant. Nevertheless, it is used extensively for training purposes at U.S. Army SIMNET sites, such as Ft. Knox KY and Ft. Rucker AL.

More information on the SIMNET SAF system is available; the references include:

1. System overview [Ceranowicz,1988] [Crooks,1990] [Downes-Martin,1990] [Jacobs,1990]
2. Terrain representation and reasoning capabilities [Stanzione,1989]
3. Evaluation of the CGF entities' behavior [Potomac,1990]
4. Supplemental dismounted infantry workstations [Fraser,1990a] [Fraser,1990b]
5. CGF research using the SIMNET SAF as a testbed [Harmon,1991] [Harmon,1994]

The SIMNET SAF was enhanced for use in DARPA's Project ODIN; in that system it was known as the ODIN SAF. The terrain reasoning capabilities of the ODIN SAF are given in [Stanzione,1993] and an adaptation of the ODIN SAF to produce 10,000 entities is described in [Vrablik,1993]. The SIMNET SAF also continues to be used for large simulation development projects such as "Synthetic Theater of War-Europe", an attempt to combine elements from virtual, live, and constructive simulation into an integrated networked brigade training system [Johannesen,1995].

2.2.7 Semi-Automated Forces Dismounted Infantry

Dismounted infantry, in useful numbers, is conspicuously absent from the SIMNET battlefield. SIMNET contained only very limited dismounted infantry capabilities in the form of manned workstations that permitted operator control of single fireteams [Fraser,1990a] [Fraser,1990b]. The SIMNET SAF system did not generate dismounted infantry entities. That absence created an unrealistic, and possibly negative, training environment. Dismounted infantry, both in reality and in SIMNET, is difficult to see and very dangerous to vehicles when armed with anti-tank missiles. Tank crew trainees in SIMNET were not forced to consider this threat and consequently could learn tactical behaviors that would increase their vulnerability to dismounted infantry.

In response to this need, IST developed a CGF system capable of generating useful numbers of computer controlled dismounted infantry fireteams at minimal cost. This was done by making extensive specialized enhancements to IST's CGF Testbed. The resulting system, known as Semi-Automated Forces Dismounted Infantry (SAFDI), can generate dismounted infantry fireteams and their associated fighting vehicles in the SIMNET battlefield.

The enhancements consisted of entity types, physical models, and behaviors specific to dismounted infantry and supporting entities. More specifically, the infantry functionality implemented for the SAFDI system includes:

1. Hybrid representation of fireteams and individual soldiers [Petty,1991] [Petty,1992d] [Petty,1994a]
2. Infantry ATGM and small-arms weapons utilization behaviors [Petty,1991] [Petty,1992d] [Petty,1994a]
3. Physical models of infantry movement and exhaustion [Karr,1992a]
4. Parametric composition of fireteam size and weapons [Parra,1994]
5. Forward observer behavior, including autonomous call for fire [Franceschini,1994a] [Franceschini,1994b]
6. Use of man-portable air defense weapons [Franceschini,1994a] [Franceschini,1994b]
7. AMSAA-provided indirect fire damage resolution model [Franceschini,1994a] [Franceschini,1994b]

Extensive User and Systems documentation was provided with the SAFDI system at the sites where it was installed [Franceschini,1993a] [Franceschini,1993b].

The SAFDI system has been delivered to U.S. Army sites for evaluation and experimental use in training and development, where it has proven to be both useful and stable. The results of its evaluation at the U.S. Army Infantry School at Ft. Benning are documented in [Chervenak,1993] and [D'Errico,1994].

2.2.8 Action/Cognition Behavior Model

The Action/Cognition Behavior Model (ACBM), Simulated Warfare Environment Generator (SWEG), C++ SWEG (CSWEG), and CIMUL8 are all members of a family of simulations developed by BDM Federal since 1973. A key common ancestor of these simulations is Supressor. They have been used for a wide range of military analysis projects. A typical application of these simulations is to test the effectiveness of new, modified, or proposed vehicle, weapon, electronic warfare, or sensor systems. For example, according to [Jones,1993a] ACBM has been recently used to test the military worth of the Advanced Self-Protection Jammer for aircraft, an electronic support measure system for shipboard air defense, the Airborne Survivability Suite for the RAH-66 light attack/reconnaissance helicopter, and the Reconnaissance, Surveillance, and Target Acquisition aircraft.

The members of the ACBM family are complete simulations in that they include not only CGF functions but full battlefield representation (at least those aspects of the battlefield needed for the application). Over much of their existence these simulations have been used as stand-alone analytic simulations. As is typical for that simulation type, they are event-driven. To make them interoperable with DIS simulation, two enhancements have been made; first, their event handling has been synchronized to a real-time clock to give real-time execution [Landweer,1993b], and second, a DIS protocol interface has been added [Landweer,1994b]. With these enhancements they have been linked into DIS networks and have participated in DIS scenarios in real-time [Landweer,1994a] [Landweer,1994b] [Landweer,1994c].

ACBM uses a formal language (called VBL) of considerable power and complexity to specify both performance and behavior for the entities and systems it is simulating [Lattimore,1993]; an example of VBL was given earlier. CGF entity behavior is generated by a model of cognition also referred to as ACBM [Landweer,1993b].

References to CGF aspects and applications of ACBM include:

1. Utility of CGF systems for analysis purposes, with ACBM as an example [Jones,1993a]
2. Use of SWEG in two tests of the effectiveness of various electronic counter measures and E-2C communications systems involved in Navy air strike missions [Landweer,1993a]
3. Explanation of the ACBM model of cognition [Landweer,1993b]
4. Design of a formal language for expressing CGF performance and behavior, used as input to ACBM [Lattimore,1993]
5. Participation by CIMUL8 in the 1993 I/ITSEC DIS Interoperability Demonstration, including the DIS conversion of CIMUL8 and the specific CGF behaviors implemented [Landweer,1994b]
6. Use of CSWEG in a large scale (regiment-vs-regiment) scenario that included CGF modeling of command and control entities [Jones,1994a]
7. Integration of constructive, virtual, and live simulation via DIS with a CGF system (CIMUL8) in a central role [Landweer,1994a] [Landweer,1994c]

2.2.9 Non-military CGF systems

Simulations in other, non-military, domains also have their own computer generated entities. Some interesting examples include predatory fish in an ocean simulation [Maruichi,1987], an autonomous land vehicle on the surface of another planet in a simulation that provides the context for a machine learning experiment [Petty,1990], and fire fighters and fire in a high-rise fire incident command training system [Altman,1991]. [Warren,1995] offers two examples of how CGF systems that perform military functions (e.g. control adversary aircraft in an air combat simulation) are used for entertainment purposes. In entertainment applications, presentation fidelity is often more important than CGF behavioral fidelity.

[Petty,1995a] describes the adaptation of a military constructive simulation (Janus) to Emergency Management training and [Loper,1995a] examines how well DIS supports Emergency Management simulation. With those ideas as background, [Petty,1995e] explores how CGF requirements for Emergency Management simulation differ from combat simulation.

3. Terrain representation in CGF

This section moves from CGF systems in general to terrain representation in CGF systems. First, background concepts of terrain representation are introduced. Then terrain representation formats used by existing CGF systems and other related systems are surveyed and explained in detail. The explanations discuss the data structures for each of the representational formats and compare their strengths and weaknesses in the CGF context.

3.1 Terrain representation preliminaries

Terrain representation is the representation of a piece of terrain, either actual or imaginary, in a digital computer-readable form. The *format* of a terrain representation defines what data is stored for the terrain and how it is organized into data structures. A particular set or instance of data for a terrain is a terrain database. If that terrain database represents an actual area on the earth, it is *geospecific*; if not, it is *notional.*

Automated terrain analysis is the automated analysis of a digitized terrain representation for the purposes of making or assisting tactical decisions involving the terrain. Automated terrain analysis is not necessarily connected to CGF systems. For example, [Benton,1991] surveys several automated terrain analysis research tasks where the stated goal is to "...let the battlefield commander make more effective use of the terrain through computer analysis...". [Werkheiser,1991] also surveys automated terrain analysis methods.

Of course, terrain analysis algorithms may also be applicable to terrain reasoning within a CGF system. For CGF systems then, *terrain reasoning* is defined as automated analysis of a digitized terrain representation for the purposes of making behavioral decisions involving the terrain [Petty,1994b]. Hereinafter in this document, terrain reasoning will refer to terrain reasoning within a CGF system unless otherwise stated. The reader should note that some sources use the terms terrain analysis and terrain reasoning synonymously, without making the non-CGF vs. CGF distinction made here. Another related distinction between terrain analysis and terrain reasoning is that terrain analysis connotes a planning activity and as such has no real-time response constraints, whereas terrain reasoning suggests real-time activity, usually done during the execution of a simulation.

This survey will consider a set of terrain representations includes both those that have been used for terrain reasoning within CGF systems and those used and proposed for automated terrain analysis systems that are not a component of CGF systems. The latter are interesting in that they may offer ideas for terrain reasoning that can be applied to CGF systems.

It is often the case that terrain databases designed for terrain reasoning have structures that differ greatly from those designed for other uses (image generation, map display, and so on) [Stanzione,1989]. The focus here is on terrain representations used for terrain reasoning. Readers interested in broader issues of terrain representation are referred to:

1. Catalog of digitized terrain data available from the Defense Mapping Agency [DMA,1995]
2. Survey of digital terrain formats and recommendations for a standard DIS terrain representation format [Trott,1995]
3. Generating terrain databases for image generators from source data in standard formats [Roback,1995]

Terrain reasoning algorithms operate on data structures that represent the terrain; those data structures constitute a *terrain database*. It is intuitively clear that the terrain database used by CGF systems "plays an important part in the efficiency and realism of their terrain reasoning algorithms" [Stanzione,1994]. The terrain database must represent the 3D surface of the Earth, referred to as the *terrain surface*, as well as the roads, bridges, buildings, trees, forests, rivers, and so on that are present on the terrain, known as *terrain features*. Terrain features that are artificial objects, such as buildings and roads, are sometimes referred to as *cultural features* or *culture* to distinguish them from *natural features*.

The terrain itself may change during a simulation exercise due to the actions of agents in the simulation (artillery shell bursts create craters, combat engineers blow bridges or construct emplacements); the capability is known as *dynamic terrain*. Simulating dynamic terrain includes the issues of computing the changes to the terrain, updating the terrain database to reflect those changes, and communicating the changes among the nodes of a distributed simulation. Dynamic terrain is largely beyond the scope of this document, though some terrain representations or representational aspects intended to support dynamic terrain capabilities will be mentioned. A reader interested in more information about dynamic terrain is referred to:

1. Survey of dynamic terrain issues and solutions [Moshell,1994]
2. Dynamic terrain server architecture for DIS [Lisle,1994] [Kilby,1994]
3. Dynamic terrain capabilities in the ODIN SAF, based on destructible entities [Stanzione,1993]
4. Dynamic terrain capabilities for CCTT, based on model placement and switching [Campbell,1994]
5. Dynamic terrain database design for image generators [Li,1994]
6. Dynamic terrain approach using a variable resolution gridded terrain representation [Kendall,1995]
7. Server architecture and artillery shell cratering with parameterized hill shapes in the variable resolution gridded terrain representation [Purnell,1995]
8. Representing entity knowledge about the state of dynamic terrain, e.g. "Should the CGF unit know that the bridge has been blown?" [Watkins,1995]
9. Simulation of fluid flow and terrain effects in DIS [Chen,1994] [Chen,1995]

3.2 Elevation posts and gridded terrain

3.2.1 Definition

In an *elevation post* terrain database, the elevation of the Earth's surface is measured relative to a vertical origin at points uniformly spaced in two dimensions relative to a horizontal origin. The vertical origin is typically mean sea level. The horizontal spacing may be at regular latitude-longitude increments or based on meters from a given origin. The elevation values form and are stored as a matrix; the values are called elevation posts as a mnemonic to suggest regularly spaced posts extending from the vertical origin elevation up (or down) to the terrain surface at each measured point. The elevation of the terrain surface to be represented is specified at the elevation posts by the data; the elevation of points between elevation posts must be interpolated.

Terrain surface data given as elevation posts is available in standard forms and resolution from public sources. The Defense Mapping Agency (DMA) provides Digital Terrain Elevation Data (DTED) at resolutions of 3 arc seconds (90-meter spacing between posts) and 1 arc second (30-meter spacing) for many parts of the world [Schiavone,1995]. Those resolutions are referred to as Level 1 and Level 2, respectively. DMA DTED is often used as source data for terrain databases. [DMA,1995] is a complete catalog of terrain data available from the DMA.

Gridded terrain is terrain that is represented as a regular array, or grid, of uniformly sized cells. The cells of the grid are almost always squares and square cells will be assumed herein unless otherwise stated. The grid cells are assigned attributes that are assumed to hold for the entire grid. Elevation is usually only one of those attributes; the grid cells may have many additional attributes, such as terrain surface type, trafficability, features, and many others. See Table 3.1 for a list of example grid cell attributes.

Attribute	Values
Elevation	Meters
Vegetation height	Meters
Urban	One of: None, Present
Hydrology	One of: None, Fordable river, Non-fordable river, Lake
Soil type	One of: Muskeg, Fine grained, Coarse grained, Heavy clay
Power lines	One of: None, Present
Bridges	One of: None, Present
Land use code	One of: Open water, Cropland, Pasture, Coniferous forest, Deciduous forest, Forest clearing, Orchard/vineyard, Dense brushland, Open brushland, Wetlands, Peat cuttings, Abandoned agriculture, Bare ground or sand dunes, Urban
Road type	One of: None, Autobahn, Primary, Secondary, Trail
Obstacles	One of: None, Embankment or ditch, Wall or fence, Other manmade, Military

Table 3.1 Example terrain grid cell attributes [Powell,1988b].

Gridded terrain is often implemented as an extension of elevation post terrain data where the grid cells are aligned with the elevation posts (often the post marks the southwest corner of the cell) and each elevation post's elevation is the elevation of the corresponding grid cell. Of course, there are elevation post terrain databases with no additional attributes and there are gridded

terrain databases that are not aligned with specific elevation posts, but the two representations are so similar in concept and usage that the terms elevation post terrain and gridded terrain are sometimes used synonymously. This document will also do so, referring to the terrain representation format as gridded terrain, with elevation posts giving the elevation attribute of such terrain. The *x,y* coordinates of the elevation posts in gridded terrain are sometimes referred to as *grid points*.

3.2.2 Example

Because of its importance, one specific gridded terrain representation, the SIF/HDI format, will be described in some detail; this description follows [Stanzione,1994]. The U.S. Air Force's Project 2851 set out to develop a standard terrain database format to allow interchange of terrain database data between image generator (IG) manufacturers. The scope of the project was then expanded to include non-IG users of terrain databases, including CGF systems. The DMA set up a Simulator Database Facility (SDBF) as a central repository for terrain database data. The SDBF uses an internal terrain database format, known as the Standard Simulator Database (SSDB), for its data. Three formats for data interchange have been developed for use in collecting and disseminating data for the SDBF. One of them, the SSDB Interchange Format/High Detail Input/Output (SIF/HDI), will be described here.

The SIF/HDI format is important because it is often used as a common source data format for simulators and CGF systems with different internal terrain database representations. For example, a SIF/HDI format terrain database for Ft. Hunter-Liggett in California was used as the common source data for the first DIS Interoperability Demonstration. That demonstration linked 39 different simulators, simulation nodes, and CGF systems from 20 different vendors and agencies [Loper,1993]. (A historical aside; [Wever,1989] describes a standard interchange format for SIMNET terrain databases. It was superseded in importance by SIF/HDI as DIS replaced SIMNET.)

The SIF/HDI format (it is often referred to simply as SIF) has three primary data types:

1. Gridded elevation data
2. Model libraries
3. Cultural features

Gridded elevation data. SIF/HDI stores the terrain elevation as an array of elevation posts. SIF/HDI uses resolutions (post spacings) of 3, 1, 0.3, and 0.03 arc seconds; a single SIF/HDI terrain database may contain elevation data in one or more of those resolutions. Elevation values are given to a precision of one millimeter.

Model libraries. The SIF/HDI model libraries contain models (detailed descriptions) of features that might be located on the terrain. The models in the model libraries are archetypical, in that they are examples or centralized descriptions of features that may exist at any location, or many locations, on the terrain database. The 2D Static model library contains models of two-dimensional (2D) terrain and cultural features that do not change during execution, including roads, rivers, and railroads. The 3D Static model library contains models for unchanging three-

dimensional (3D) features, such as buildings, trees, and bridges. The 3D Dynamic model library contains models for entities that can change or move during execution, such as artillery shell craters and vehicles. The feature models can be defined as polygons, solid geometry, or both. Each model may be given in up to nine levels of detail. The models use the Feature Attribute Coding Standard (FACS), which encodes physical, cultural, and sensor-response data about the feature being modeled. Model attributes include color, texture, and collision test points.

Cultural features. In the SIF/HDI format, cultural features are those features that are location specific, such as a forest with its own unique shape, or are instances of the modeled features, such as a bridge of a particular type (found in the model library) at a specific location. The SIF/HDI format supports six classes of cultural features; they are defined in Table 3.2. [Stanzione,1994] observes that the point light and point light string cultural feature classes are redundant; features of those classes can be represented as point features with the addition of a light emittance attribute. Features are located in the terrain database by latitude and longitude to a precision of one ten-thousandth of an arc second. The SIF/HDI format allows homogenous sets of features to be aggregated into superfeatures, which can simplify terrain reasoning.

In a terrain database, *topology* refers to the spatial relationships between features and how the terrain database organization represents those relationships; in contrast, *geometry* refers to the precise location or shape of a feature. The SIF/HDI has some topological representation, in that segments and vertices between adjacent features are shared, thereby implying adjacency. However, there is no spatial organization to the feature data; features that appear sequentially in the database need not be located in proximity in the terrain.

Feature class	Representation	Attributes	Example(s)
Areal	Line segments that describe a closed polygon, separated at intersections with other features	Layer Inside segments Direction	Ponds Forests
Linear	Line segments, separated at intersections with other segments	Layer Direction	Roads Walls
Point	Vertex or non-connected vertices		Power poles
Point light	Vertex, assumed to be light source		Searchlight
Point light string	Non-connected vertices, assumed to be light sources		Runway lights
Model reference	Vertex and pointer to model in model in model libraries	Orientation Scale	Buildings Bridges
All classes		Feature Descriptor Code Predominant height Bounding rectangle	

Table 3.2 SIF/HDI cultural terrain features.

There have been some criticisms leveled at SIF/HDI as a general purpose terrain database interchange format. [Hardis,1994] and [Pope,1995a] give some of those. Along the same lines, after surveying the range of available terrain representation formats, [Trott,1995] suggests certain additions and updates to the SIF/HDI format which would make it, in the authors' opinion, "a standard format that will be capable of addressing all DIS Synthetic Environment requirements".

While the suggested enhancements have merit, the reference does not discuss terrain reasoning, and the recommended format is likely not the optimum solution in that area.

More specific to CGF systems, [Stanzione,1994] analyzes the applicability of the SIF/HDI format to CGF systems' terrain reasoning, concluding that the data needed for CGF terrain reasoning is available in the SIF/HDI format but that additional feature attribute standardization and spatial organization of the data is needed to efficiently support CGF terrain reasoning. Because SIF/HDI is typically used as a source data format and converted to each CGF system's internal terrain database format the needed spatial organization can be computed during the conversion process.

[Pope,1995a] describes how CCTT visual databases are created from DMA formats. Once created they are converted into SIF/HDI format; the SIF/HDI terrain data is then input to the CCTT SAF terrain database compiler, which generates the CCTT SAF's MRTDB terrain reasoning terrain database and Quadtree plan view display terrain database (both to be described later) from it. The intent of this process is twofold: first, to increase correlation between the CCTT visual database and the CCTT SAF's terrain reasoning terrain database (correlation is further discussed later), and second, to insulate the CCTT system's visual terrain database and the CCTT SAF's terrain databases from changes to each other. According to [Pope,1995a], CCTT is the first major DIS project to use the SIF/HDI format. In discussing the same process, [Watkins,1995] indicates that the intermediate form is actually SIF/HDI with some extensions. It is not clear from the reference whether a strict SIF/HDI file would be readable by the CCTT SAF terrain database compiler.

3.2.3 Additional applications

This subsubsection briefly presents additional gridded terrain representations. They are:

1. Eagle
2. PATH PLAN
3. JPL Mobile Robot
4. Martin Marietta SAFOR
5. Stealth terrain navigation
6. Compact Terrain Database
7. NASA Ames LOS attachment
8. Model Reference Terrain Database
9. Iowa Driving Simulator
10. RAND
11. ARL Variable Resolution Terrain

Eagle. In the Eagle constructive simulation, terrain gridded at 100 meter resolution, with 10 attributes associated with each grid, is used as input to a unit route planning algorithm [Powell,1987] [Powell,1988a] [Powell,1988b] [Powell,1989] [Wright,1990].

PATH PLAN. A simple terrain grid, with elevation and explicit obstacles as the only grid cell attributes, is used to represent a portion of Ft. Hunter-Liggett to test a robot motion planning algorithm [Ok,1989].

JPL Mobile Robot. A terrain representation of uniformly sized square grid cells is used to receive and fuse sensor information for a mobile robot [Slack,1989] [Ewing,1992]. The terrain grid cells' attributes are elevation, slope, roughness, location and height of obstacles, presence or absence of vegetation, and composition (grass, dirt, leaves). The terrain database is centered on the robot, rather than being of fixed extent and location, so as the robot moves the terrain grid cells' attribute values are updated by the sensors. The terrain data is supplied as input to an abstraction process that computes a set of terrain features for use in the robot's route planning.

Martin Marietta SAFOR. Polygonal terrain (defined later) is discretized into a hexagonal grid. Each hexagonal grid cell has three attributes: elevation, mobility, and exposure [Bockstahler,1991].

Stealth terrain navigation. A gridded terrain representation, with elevation as the sole attribute of each grid cell, is used as a test environment for route planning, intervisibility, and bounding overwatch terrain reasoning algorithms designed for a parallel machine architecture [Teng,1992].

Compact Terrain Database. The Compact Terrain Database (CTDB) is a gridded terrain database format used in the ODIN SAF and ModSAF [Smith,1992a] [Stanzione,1993]. CTDB elevation posts are spaced at 125m intervals. By using elevation posts over much of the database extent the CTDB format achieves considerable terrain database size reduction over the polygonal format of its predecessor (the SIMNET SAF terrain database, to be described later); hence its "Compact" appellation. CTDB also uses fixed point numeric representations to save space. [Smith,1995b] observes that the compressed format positively affects run-time performance in that more of the terrain database can be retained in memory. However, [Watkins,1994] comments that some of the CTDB compression is at the cost of additional run-time processing as compared to other formats. The CTDB does include some polygonal data as well; that will be described later.

A polygonal (triangular) representation of the terrain surface is induced from the elevation posts by assuming a diagonal bisecting each of the squares formed by the elevation posts in the x,y plane; see Figure 3.1. In early CTDB terrain databases, the diagonal was always Northwest to Southeast; in the current format, it may be either Northwest to Southeast or Northeast to Southwest, though the direction is specifiable only for a database as a whole, not for individual squares.

Microterrain and terrain features are represented in ModSAF as polygons; they will be discussed later. The CTDB format continues to be actively enhanced; [Smith,1995b] presents recent enhancements and a list of projected future capabilities.

NASA Ames LOS Attachment. The NASA Ames Research Center's Vertical Motion Simulator (a flight simulator) uses a gridded terrain representation with elevation posts only to perform point-to-point intervisibility determinations [Sansom,1993]. The gridded representation is constructed in a preprocessing step from the polygonal terrain database used in the flight simulator's image generator.

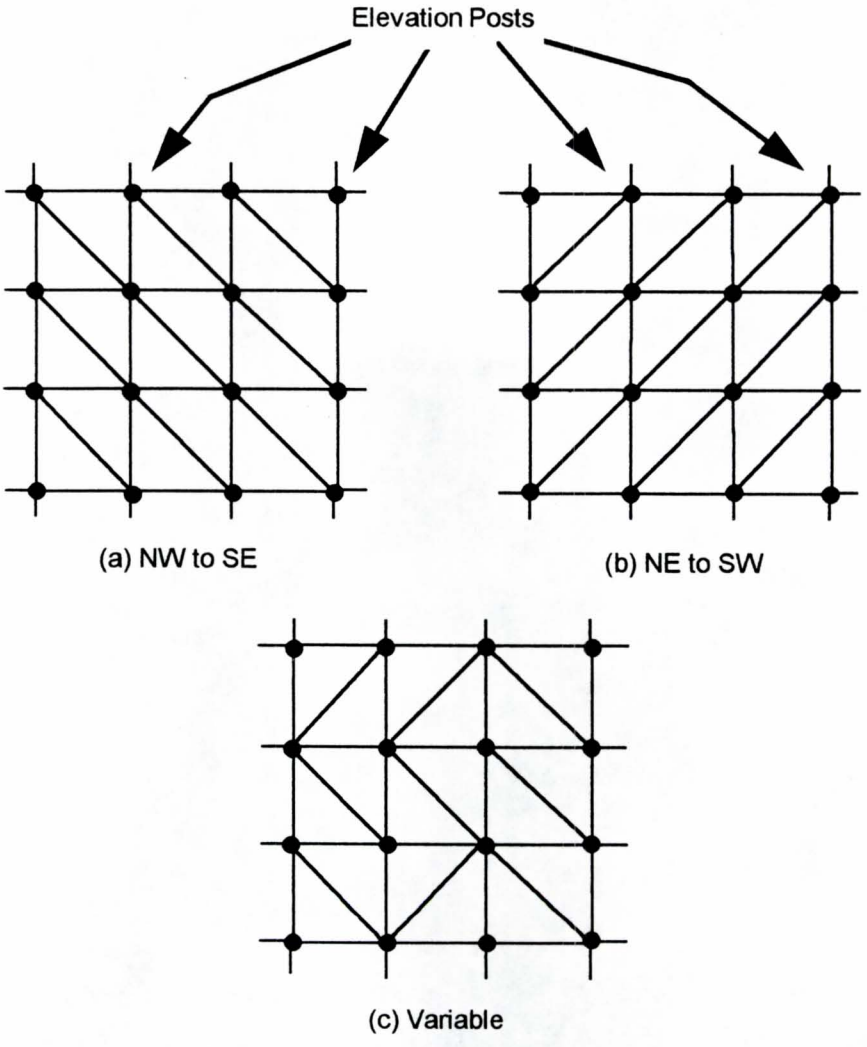


Figure 3.1 Inducing polygons from elevation posts.

Model Reference Terrain Database. The Model Reference Terrain Database (MRTDB) is a gridded terrain database format used in the CCTT SAF (as well as other simulation nodes in the CCTT system) [Watkins,1994] [Watkins,1995]. Driving the MRTDB database design are CCTT's demanding terrain database requirements, which are well beyond those imposed on the SIMNET or CTDB formats [Watkins,1994]. A few CCTT requirements are:

1. Terrain database size; 100 Km x 150 Km
2. Elevation post spacing; 30 m
3. Feature density; 30,000 structures (10,000 destructible) and 10,000,000 individual trees

The MRTDB format improves upon the CTDB's already compact format with additional data compression techniques in order to meet CCTT's large terrain database requirements. The improved compression allows even more of the database to be retained in memory.

As in the CTDB format, the terrain surface in the MRTDB format is taken to be the triangular surface induced by the elevation posts. However, in MRTDB, the diagonalization is variable; the diagonal direction for each square is stored with the elevation post at the square's Southwest corner.

The MRTDB design is object-oriented and it is implemented in Ada. It permits different elevation post spacings within the format. The dominant feature of the MRTDB is its extensive use of models to describe terrain features such as trees and buildings. A set of such features are described in the MRTDB's Feature Model Library. Instances of those features are located in the terrain by a reference to the library entry. This method avoids a detailed description of a feature, such as a building, at every feature location in the database. Different versions of the features can be defined in the Feature Model Library to represent the features in different states, such as normal, damaged, and destroyed, and the database reference switched when a particular feature's state is changed by exercise events [Campbell,1994]. [Pope,1995a] and [Pope,1995b] provide lists of enhancements of MRTDB relative to CTDB.

Terrain reasoning capabilities provided by the MRTDB include elevation, collision detection, munition impact detection, intervisibility road route planning, static obstacle avoidance, area intervisibility, and cover finding [Watkins,1995]. That reference also discusses three novel terrain representation issues dealt with in the MRTDB. They are:

1. Bi-level terrain; representing overpasses and bridges
2. Penetrable forests; representing very large numbers of individual trees without exceeding space limits
3. Terrain awareness; representing the state of entity knowledge about dynamic terrain

Iowa Driving Simulator. A variable resolution gridded representation is used in the Iowa Driving Simulator (IDS) [Papelis,1994]. The IDS has sufficient fidelity in its vehicle dynamics models and terrain representation to be applied to virtual prototyping [Kuhl,1994]. The terrain database is partitioned in the x,y plane into arbitrary rectangles, called *datazones*, whose sides are aligned with the x and y axes. Within each datazone the terrain elevation (and other attributes) are given by regularly spaced *datasets*, which are essentially elevation posts. The dataset spacing is constant within a datazone, but variable among datazones; it may be set as needed to represent

arbitrarily high resolution terrain. The IDS terrain database format supports overlapping datazones at a given *x,y* location (with different *z* values) to model bridges, overpasses, and similar structures. Finally, the format includes a second partitioning of the terrain in the *x,y* plane into square sectors, each one with a list of the datazones it overlays, so as to speed terrain query access.

RAND. In a quantitative analysis of route planning algorithms reported in [Marti,1994] a gridded terrain representation is used, with elevation the only attribute of the elevation posts.

ARL Variable Resolution Terrain. Gridded terrain based on the idea that "terrain is built up of many hills, each with its own set of shape parameters" is used to support dynamic terrain [Purnell,1995]. The elevation at a particular grid point is the sum of the heights of the hills that are present at that grid point. The terrain is changed (e.g. artillery cratering) by adding additional hills to the terrain database. [Kendall,1995] goes on to describe how this gridded terrain format can be mathematically manipulated to provide gridded terrain with variable resolution. The basic idea of the technique, called *adaptive grid generation*, is that the grid resolution (spacing) varies in space, with closer grid spacings around simulation entities, and in time, with the areas of increased resolution moving to follow the simulation entities. This dynamic respacing of the grid is accomplished by using elliptical Poisson equations with entity locations as point attractor source terms to calculate the *x,y* coordinates of the grid points prior to calculating the elevations of those points by applying the parametric hills.

[Kendall,1995] proposes adaptive grid generation as a standard terrain model for DIS. While mathematically interesting, the adaptive grid generation method has a number of unresolved problems that call into question its applicability to DIS. Two will be considered here: computational expense and terrain correlation.

The computational expense of the adaptive grid generation method may be prohibitively high for DIS. To find the grid points a system of partial differential equations must be solved. In considering the problem of computation time, [Kendall,1995] states: "The time between distributed interactive simulation (DIS) entity state protocol data unit (PDU) updates, approximately five seconds, is available for updating the locations of the grid points and the corresponding terrain elevations." This statement is erroneous on several counts:

1. DIS Entity State (ES) PDUs for a single entity are typically sent five times per second, not once every five seconds [Cheung,1994].
2. The DIS ES PDU sending rate is variable and unpredictable, and so no fixed amount of time can be assumed to be available.
3. Entity locations must be dead reckoned between ES PDU arrivals and those entities may be involved in intervisibility determinations during the interarrival periods, so the detailed terrain recalculations may be needed even more often than ES PDUs arrive.

Furthermore, the grid recalculation times given in [Kendall,1995] for four entities exceed five seconds, and the given results appear to not include the calculation of the elevation at the grid points. Nearly all DIS exercises will involve many more than four entities.

Ensuring terrain database correlation is another problem area with adaptive grid generation. If every node that may sight a particular entity does not perform the grid recalculations at exactly the same time and with exactly the same entity locations, then slightly different terrain forms will result, possibly giving different intervisibility results.

3.3 Polygonal terrain

3.3.1 Definition

Elevation posts approximate the terrain surface by giving its elevation at a set of fixed points. [Schiavone,1995] asserts that "DIS resources that interact with and/or represent the terrain require a common, exhaustive, unambiguous representation of the surface of the earth." By exhaustive and unambiguous the reference is referring to the terrain surface between the elevation posts. The most common method of providing such a continuous 3D terrain representation is *polygonal terrain*. In polygonal terrain, the terrain surface is represented as a set of 3D polygons. The polygon set is contiguous, in that adjacent polygons share edges, and complete, in that for every x,y coordinate within the geographic extent of the terrain database there is a terrain surface polygon that includes that point. The z coordinate of the polygon at any given x,y coordinate is the terrain elevation at that point. Figure 3.2 is an example.

Gridded terrain can be converted to polygonal terrain; this conversion is often done with SIF/HDI data as input (e.g. see [Loper,1993]). The obvious way to do so is by taking the elevation posts as the vertices of polygons. Typically the polygons are triangles. However, naive application of this simple procedure can produce certain problems. First, large regions with constant elevation or constant slope can be converted into many coplanar polygons, needlessly increasing both storage and processing requirements for the terrain database. Second, differing polygonalizations may derive from identical elevation post data. An example is given in Figure 3.3. The four elevation posts in (a) can be triangulated in two different ways, shown in (b) and (c); in the absence of other information, both are equally valid. Such ambiguity can be avoided in at least two ways. One is by adopting a convention that all terrain databases derived from the same gridded data and that are intended to correlate be triangulated using the same procedure, either (a) or (b); ModSAF's CTDB uses that method. The other is to explicitly indicate which is correct; the CCTT MRTDB uses a bit indicator for that purpose [Watkins,1994].

[Schiavone,1995] discusses the creation of polygonal terrain from elevation posts in somewhat more detail, indicating that a Delaunay triangulation algorithm is commonly used. The reference also describes the process of manually editing and enhancing a terrain database that was originally created from elevation posts using a triangulation algorithm. [Sundaram,1994] presents an algorithm to generate polygonal terrain from arbitrary elevation posts in real-time.

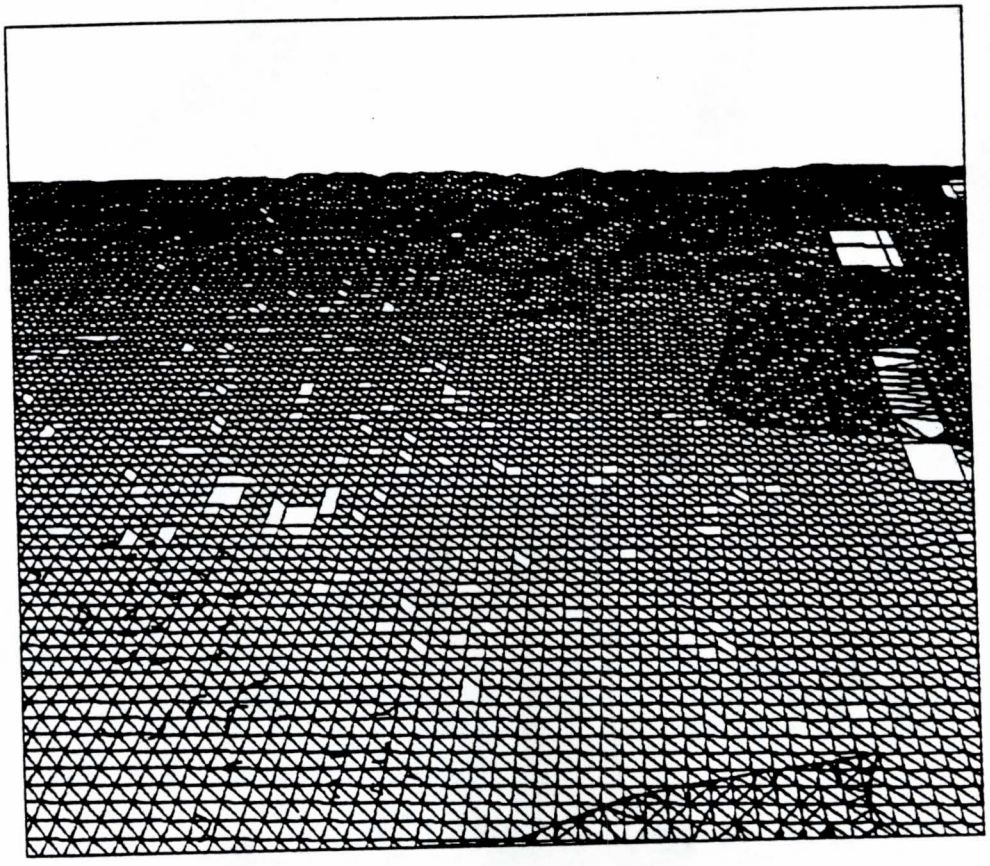
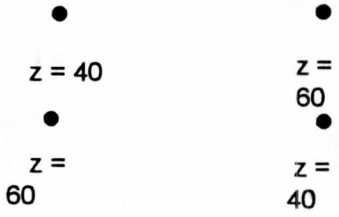
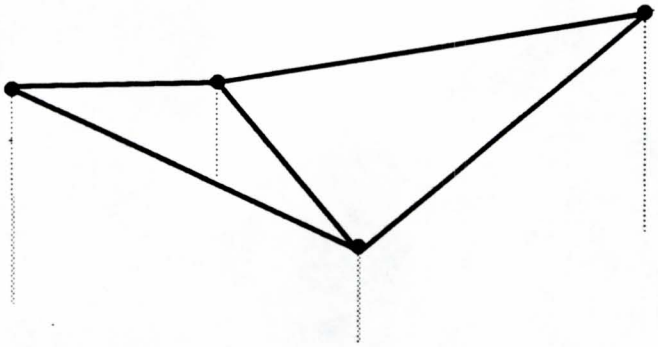


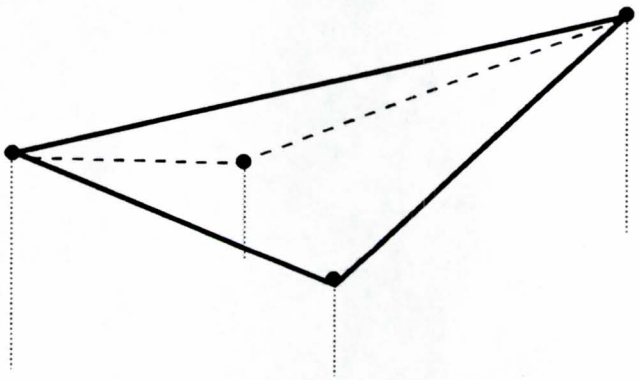
Figure 3.2 Polygonal terrain.



(a) Elevation Post Z values



(b) "Saddle"



(c) "Ridge"

Figure 3.3 Different polygonal surfaces (b) and (c) derived from the same elevation posts (a).

Polygonal terrain is also often converted to a gridded form. This is usually done to provide a simpler, more regular data structure for a terrain reasoning algorithm. For example, [Sansom,1993] details the process of constructing a gridded terrain representation from a polygonal IG terrain database; the former is used for intervisibility determination. A simple grid abstraction of polygonal terrain is fairly easy to compute from a terrain database using well-known methods. See [Nagy,1979b] for a general approach. The exact process used varies considerably depending on the needs of the particular terrain reasoning algorithm; specific examples will be given later when terrain reasoning algorithms are discussed.

A *Triangulated Irregular Network* (TIN) is a terrain surface composed of a set of polygons (triangles) that was not derived from an initial grid of elevation posts. Consequently, the polygon vertices in a TIN may be arbitrarily located in the *x,y* plane.

The term *microterrain* refers to high-resolution terrain formations such as valleys, gullies, ridges, peaks, pits, and craters. As microterrain, these formations are formed from polygons that are usually much smaller than the *surface* (or *macroterrain*) polygons resulting from triangulating elevation posts. Furthermore, whereas the macroterrain polygons' vertices will be located in a regular grid in the *x,y* plane because of their alignment with elevation posts, the vertices of microterrain terrain polygons will typically be arbitrarily located. Table 3.3 compares the characteristics of typical macroterrain polygons with microterrain polygons. (The values given are notional and are not derived from a specific polygonal terrain representation format.) Clearly, microterrain and TINs are related, as microterrain is often constructed from TINs, simply because representing microterrain in a gridded or elevation post format by closely spaced posts can be extremely expensive in terms of memory requirements.

Polygons	Size	Vertices	Terrain Database Coverage
Surface, macroterrain	125 meters	Grid-aligned in *x,y* plane	98% of geographic extent
Microterrain	1 meter	Arbitrary in *x,y* plane	2% of geographic extent

Table 3.3 Macroterrain and microterrain polygon characteristics.

3.3.2 Example

Although polygonal terrain is conceptually just a set of polygons with 3D coordinates for their vertices, polygonal terrain databases almost always use some organizational scheme on the polygon set so as to speed access to and processing of specific parts of the terrain surface. As an example, the polygonal terrain database format used in the SIMNET SAF and other SIMNET simulators, as well as the IST CGF Testbed, will be examined. The CGF Testbed's polygonal format was derived from the format used in SIMNET and is nearly identical to it. This discussion follows [Smith,1992b]. (The SIMNET polygonal format is also known as *libTDB*, according to [Smith,1995b], but this document will refer to it as the SIMNET polygonal format.)

The SIMNET polygonal format superimposes a regular partitioning of the terrain onto the polygons. The overall terrain area is first divided into *patches*, which are square areas 500 meters on a side. Each patch is further divided into a 4x4 array of 16 *grid cells*, which are square areas

125 meters on a side. (A terminological note: the developers of the SIMNET format actually and confusingly call the grid cells "grids", contrary to both normal usage and this document. In this document the term "grid cells" will be used to avoid confusion, but the reader should be aware that the SIMNET grid cells are called grids in the references.) The polygon for a given point, e.g. for an entity elevation and orientation computation, can be found by first performing arithmetic on the *x,y* coordinates of the point to determine the patch and grid cell that contain the polygon and then searching a list of the terrain surface polygons associated with the selected patch and grid cell, performing a polygon inclusion test on the point for each polygon. (The reader familiar with computational geometry will no doubt recognize this as a form of the point location problem; see [Preparata,1988].)

The terrain surface polygons within a patch are stored in three arrays, repeated for each patch. Each vertex in the patch is an entry in the vertex array, which holds the *x,y,z* coordinates for the vertices. Likewise, each polygon edge is an entry in the edge array; those entries contain the indices of the edge's vertices in the vertex array. Finally, each polygon is an entry in the polygon array, where the entries store the indices of the polygon's edges in the edge array. The polygon entries also contain bit masks that indicate whether the polygon overlaps with each of the 16 grid cells in the patch. Thus, once the grid cell containing a point is determined, the bit masks can be used to reduce polygon inclusion tests to only those polygons that overlap that grid cell.

Features are also primarily represented as polygons in this format. Roads, rivers, and lakes are polygons, with appropriate polygon type codes, that lie on the terrain surface polygons. Man-made structures, such as buildings, trailers, and water towers, are closed sequences of polygons with an associated type. Treelines are open sequences of polygons. Tree canopies are closed treelines with additional polygons to form the canopy's "roof". Individual trees are given as open polygons with an associated radius [Smith,1992b].

The SIMNET format includes several useful filters [Watkins,1994]. They are:

1. Grid cell masks; allow rapid filtering of features by grid cell, as mentioned earlier
2. Grid cell maps; provide direct access to certain features by grid cell and type
3. Minimum and maximum *x,y,z* values; quickly eliminate patches from consideration in some algorithms, such as intervisibility determination
4. Patch guards; control caching of patches based on a summary of patch data.

Note that in the polygonal format there is no abstract or object representation of features. The abstract notion of a road is represented as a set of triangles and quadrilaterals of a certain polygon type that happen to be geographically adjacent, with no topological or object relationship between them. [Stanzione,1989] criticizes this format for that reason, saying that it "... is not suited for either reasoning or drawing", correctly observing that because there is no connection between the many individual polygons of terrain features, reasoning about them is problematic. Road following, for example, is very difficult using this format because consecutive road polygons are not linked or even necessarily closely stored in the terrain database.

The difficulty of reasoning on the polygonal terrain database led to the use of the supplementary quadtree terrain database (described later) in the SIMNET SAF. However, the developers of the

CGF Testbed chose to perform terrain reasoning on the polygonal terrain, in spite of its difficulty, so as to avoid all possibility of internal terrain correlation errors (terrain correlation will be defined later).

3.3.3 Additional applications

This subsubsection briefly presents additional polygonal terrain representations. They are:

1. Compact Terrain Database
2. Mesh
3. Integrated Computer Generated Forces Terrain Database

Compact Terrain Database. As mentioned earlier, the Compact Terrain Database (CTDB) is a gridded terrain database format used in the ODIN SAF and ModSAF [Smith,1992a] [Stanzione,1993]. In the CTDB format, most of the terrain surface is represented with elevation posts. However, polygons (squares and triangles) can be used for microterrain in areas where the regular elevation post grid does not satisfactorily describe the desired terrain. That might be the case for data not derived from a regular grid, such as TINs. In the CTDB format, microterrain polygons can be used for terrain configurations such as river beds and multi-level terrain (e.g. tunnels or bridges) [Smith,1995b].

Polygons are also used in the CTDB format to represent many of the terrain features. Table 3.4 lists ModSAF's terrain feature types and identifies those which have a polygonal representation.

Feature category	Feature	Polygonal representation?
Terrain surface	Ground	Yes
	Water	Yes
Structures	Buildings	Yes
	Pipelines	No
	Power pylons	No
	Other opaque, non-penetrable structures	Yes
Trees	Individual trees	No
	Tree lines	Yes
	Tree canopies	Yes
Linear features	Roads	
	Rivers	

Table 3.4 CTDB terrain features (adapted from [Longtin,1994]).

Mesh. An arbitrary polygonal terrain database can be converted into a specialized polygonal format called a *mesh* [Cunningham,1993]. In the mesh format, all of the polygons are convex, and all terrain features (such as rivers, tree canopies, and buildings) are embedded in the polygons rather than being situated on top of surface polygons. An algorithm to produce a mesh from an arbitrary polygonal terrain database is given in the reference.

Integrated Computer Generated Forces Terrain Database. The Integrated Computer Generated Forces Terrain Database (ICTDB) is a terrain database project being conducted to satisfy the CGF terrain and environmental reasoning requirements for ARPA's Synthetic Theater of War

(STOW) program. The ICTDB designers intend to address the perceived shortcomings in current CGF terrain databases and to include a richer set of terrain features and terrain reasoning attributes in their format. Presently the ICTDB project has completed requirements analysis, data source investigation, and preliminary design, but development and demonstrations are just beginning. The ICTDB design uses polygons, in the form of TINs, to represent the terrain surface. Pointers are maintained from each polygon to topologically adjacent polygons. The polygons are also organized into square patches, which are the units of storage. Each patch is subdivided into a virtual grid with a variable number of rows and columns, dependent on the number of polygons in the patch. In the patch data structure, each grid cell's entry contains a pointer to the polygon that occupies the largest portion of that grid cell. The patch and virtual grid structure is used to speed point location operations (i.e. determining the polygon that includes a given *x,y* point). Terrain features are stored in a quadtree and graph structure, similar to those used in the SIMNET SAF and ODIN SAF terrain databases. [Stanzione,1995] reports the current status of the ICTDB project.

3.4 Quadtrees

3.4.1 Definition

As classically defined, a region quadtree (or simply a quadtree) is a hierarchical data structure for representing a 2D surface, such as a terrain database (which is 2D when projected into the *x,y* plane). The surface to be represented is to be square (without loss of generality, in that a non-square surface can be enclosed in a square). Given a surface where each point is categorized into one of two or more categories, a quadtree partitions the surface into four equally sized quadrants, or *quads*, with the partition continuing recursively until each lowest-level quad is entirely of one category. Alternatively, if the surface includes features or objects with known locations and non-zero extent, the surface is recursively partitioned into quads until no lowest-level quad contains more than one feature (although a feature may span more than one quad). The quads' sizes can vary from the size of the entire surface down to the minimum resolution of the categorization.

Quadtrees can also be used to represent elevation. For elevation, the quads are recursively subdivided until a quad is entirely at a single elevation, or alternately, is planar within a specified tolerance.

Each node of the quadtree data structure correspond to a quad; leaf nodes of the quadtree correspond to quads that are not further partitioned. The nodes of the quadtree contain data for the quad such as its category or the feature it contains.

Figure 3.4 gives an example of a classical quadtree applied to terrain representation. In the figure, (a) shows a gridded terrain area with each grid cell assigned one of two types of terrain, and (b) shows a quadtree representation of that terrain. The example of Figure 3.4 is typical of quadtrees derived from gridded terrain. The terrain grid cells partition the terrain at the highest level of resolution, corresponding to the lowest level of the quadtree, thereby defining the smallest quad size. In such a quadtree unpartitioned quads represent sets of grid cells with the same value for the attribute that determines the quadtree partitioning. In some cases the quad partitioning is

performed at every level down to a predetermined quad size, without regard for any quad attribute, to produce a *complete quadtree*.

[Antony,1988] provides information about quadtree notation, operators for moving within a quadtree, and quadtree memory requirements. More extensive information on the general topic of quadtrees can be found in [Samat,1984].

3.4.2 Example

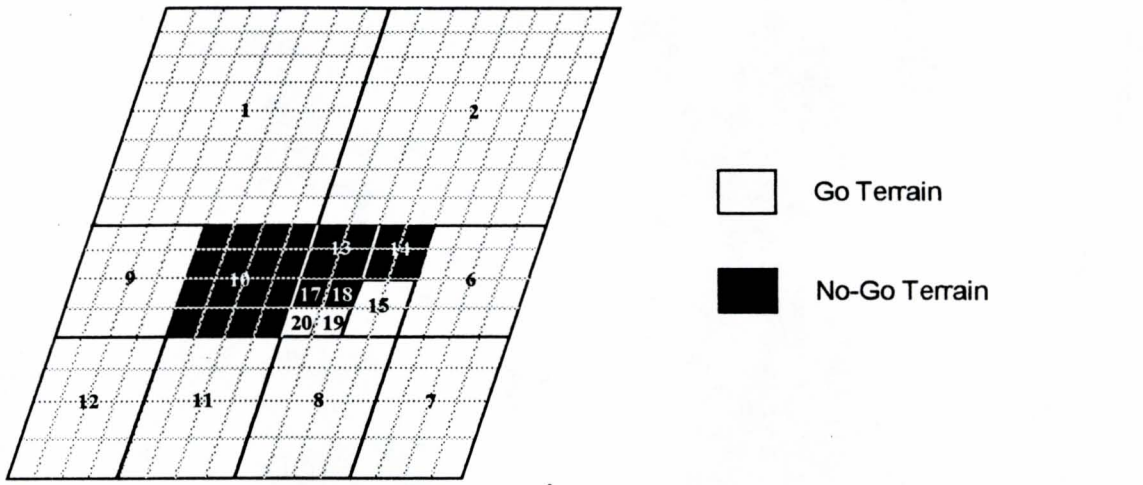
The SIMNET SAF uses a complete quadtree, referred to as the *libQuad*. It will be presented as an example of quadtree use in CGF systems. This explanation is largely adapted from [Stanzione,1989].

As implemented, the SIMNET SAF libQuad quadtree is complete; that is, all of the quadtree nodes are expanded at every level except the lowest, or equivalently, all of the leaf nodes are at the same level of the tree and correspond to terrain quads of identical geographic size. The size of the lowest level quads is 2500m x 2500m. That quad size was determined by experimental analysis weighing memory required for the quadtree, which increases for smaller quads, against object search time, which increases for larger quads.

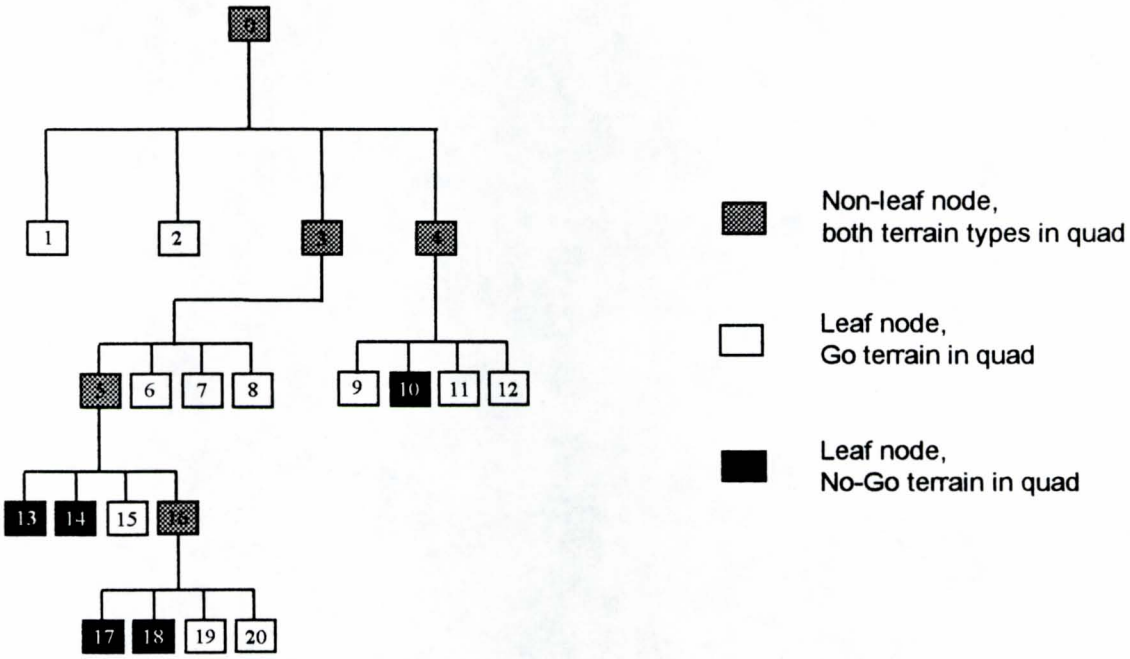
The nodes of the quadtree contain pointers to terrain objects that are located within the node's corresponding quad. There are five classes of terrain objects; they are:

1. Network (e.g. roads, rail lines, bridges)
2. Area object (e.g. forests, bodies of water)
3. Linear object (e.g. treelines, contour lines)
4. Point objects (e.g. trees, buildings)
5. Dynamic terrain (e.g. battlefield control measures, minefields)

A terrain object is pointed to by the node corresponding to the smallest quad that contains the object. The object pointers within a node point to the objects' representation, which are arrays with a structure depending on object class. Figure 3.5 shows a an example of terrain represented in this way.



(a) Simple terrain with two terrain types



(b) Corresponding quadtree

Figure 3.4 Quadtree applied to terrain representation (adapted from [Stanzione, 1989]).

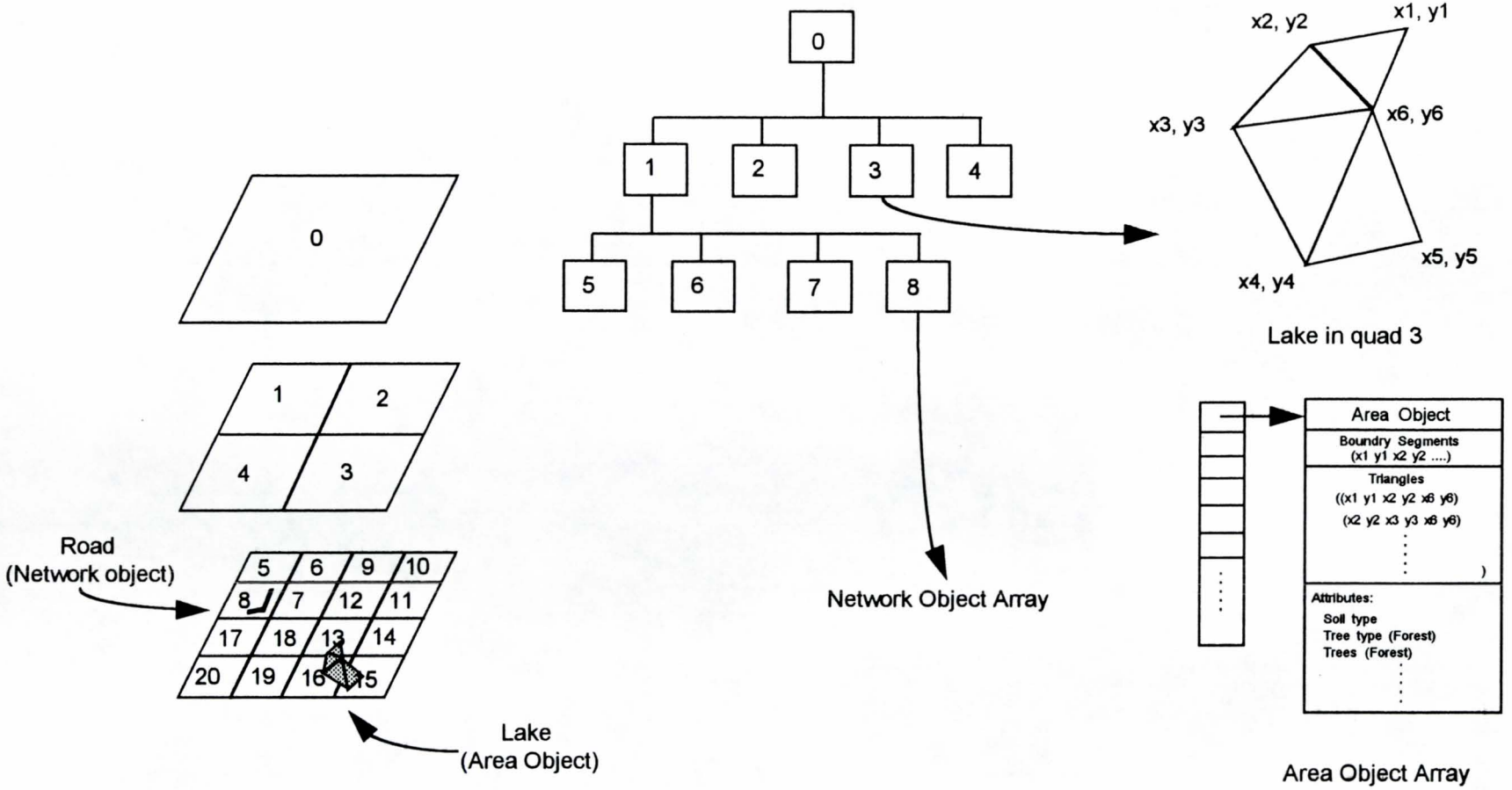


Figure 3.5 Simnet SAF libQuad quadree example (adapted from [Stanzione, 1989]).

3.4.3 Additional applications

This subsubsection briefly presents additional quadtree terrain representations. They are:

1. Pyramid
2. REACT
3. QUILT
4. ODIN SAF libquad
5. Compact Terrain Database
6. Integrated Computer Generated Forces Terrain Database
7. Intelligent Player

Pyramid. A pyramid (complete quadtree) supplemented with a frame based object representation is proposed as a spatial (terrain) database design [Antony,1988]. The objects, such as lakes, roads, or entities, are bidirectionally linked to the quadtree nodes at each level that correspond to the quad that includes the objects.

REACT. A quadtree is used to represent terrain for automated terrain analysis in a low-altitude air-to-air combat application called REACT [Hayslip,1988]. The quads of the terrain are assigned abstract terrain types (corridor, low flat, mountain, hilly, plateau) which are determined by preprocessing DMA elevation post data. Each quad is subdivided if its subordinate quads do not all have the same abstract type. The size of the smallest quads is determined by the variability of the terrain.

QUILT. A quadtree-based Geographic Information System (GIS) called QUILT is used in an expert system designed to predict minefield sites [Doughty,1988]. The quadtree quads have four attributes: proximity to nearest road, area, mobility type (go, restricted, slow, very slow, no go, built up, and open water), and degree of canalization. The latter attribute measures how restricted movement is through the area represented by the quad.

ODIN SAF libQuad. The same quadtree as described for the SIMNET SAF was also used in the ODIN SAF, a further development of the SIMNET SAF [Stanzione,1993], as well as early versions of the ModSAF CTDB [Smith,1995b]. A list of specific quadtree terrain objects and their classes is given in the reference.

Compact Terrain Database. The quadtree which was a separate database in the SIMNET SAF, ODIN SAF, and early versions of ModSAF was eliminated in favor of a quadtree integrated into the CTDB data structures in recent versions of ModSAF [Smith,1995b]. The nodes of the ModSAF quadtree may be expanded using any criteria, and are not all expanded to the same level, as was the case for its SIMNET SAF and ODIN SAF predecessors. In addition to storing linear features where topological information is of paramount importance, such as roads, the ModSAF quadtree is a repository for abstract terrain features, such as tree canopies, areas of steep slope, and political boundaries. The features may be stored at both interior nodes and leaf nodes; they are accessed via iterative fetching [Smith,1995b].

Integrated Computer Generated Forces Terrain Database. As in the SIMNET SAF libQuad, the ODIN SAF libQuad, and the early versions of the CTDB, the ICTDB database design uses a quadtree to store terrain features [Stanzione,1995]. The ICTDB quadtree may contain features at any level of the quadtree. The features are referenced both explicitly and implicitly; the explicit reference is at the level of the quadtree whose corresponding quads' side dimension are closest to the feature's size. The feature is implicitly referenced (by pointer to the explicit reference) at lower levels of the quadtree in nodes that correspond to quads that overlap the feature. The quadtree also contains aggregate features, which are features that exist as collections of other features, that may themselves be aggregate features. For example, a group of features such as buildings, may be grouped into an aggregate feature like a village. Aggregate features are intended to be used in terrain reasoning by units.

Intelligent Player. Intelligent Player is a research system that uses game tree lookahead to perform real-time control and planning of movement for a CGF helicopter in air-to-air combat [Katz,1989] [Katz,1991] [Katz,1992] [Katz,1993] [Schaper,1994]. Intelligent Player uses a quadtree terrain representation. The quadtree is used for terrain elevation only; each quad is recursively subdivided until all points in the quad fall within a single plane, within a predefined tolerance level. The quadtree structure, its construction, and its use for terrain avoidance and intervisibility determination is described in [Pandari,1995].

3.5 Graphs

3.5.1 Definition

Terrain may be represented as a graph. Typically, the vertices of the graph correspond to significant features of the terrain, such as road intersections, junctions in mobility corridors, or local elevation maxima. The edges connect vertices that are related in a way relevant to the definition of the vertices; for example, in a road net graph, the vertices are road junctions and an edge connects two vertices if and only if the corresponding road junctions are connected by a road segment. Weights are often assigned to the edges to represent a quantity or attribute of interest within the representation scheme. In the road net graph example, the edges might be weighted with the length of the road segment they represent. Other possible attributes upon which edge weights might be based are trafficability, cover, and concealment.

Terrain graphs are usually searched to find a route or path, generally with standard graph search algorithms such as A* [Nilsson,1980] [Winston,1984].

3.5.2 Example

A route planner developed at the Swedish National Defence Research Establishment (FOA) uses a connectivity graph [Holmes,1992]. Vertices in the connectivity graph represent traversable regions in a digital map and edges signify adjacency. Subdividing the overall traversable area in the map into distinct regions is done using a vertical scan algorithm. The separation of a single traversable region into two regions occurs at a local minimum of a non-traversable obstacle. The

algorithm depends heavily on the binary nature (traversable or non-traversable) of the source map. The route planner searches the resulting graph using an A* algorithm.

3.5.3 Additional applications

This subsubsection briefly presents additional graph terrain representations. They are:

1. Multiple route finder
2. Eagle
3. SIMNET SAF libQuad
4. Neighborhoods
5. ODIN SAF libQuad
6. Compact Terrain Database
7. CGF Testbed line of sight graph
8. Integrated Computer Generated Forces Terrain Database

Multiple route finder. A straightforward graph representation of a road network is searched to find multiple road routes for unit route planning [Benton,1987].

Eagle. A graph representing unit mobility corridors is produced from a Delaunay triangulation of gridded source terrain data [Powell,1987] [Powell,1988a] [Powell,1988b] [Powell,1989] [Wright,1990].

SIMNET SAF libQuad. A graph representing a road network, with road segments and road junctions represented separately, is embedded in a quadtree in the SIMNET SAF [Stanzione,1989].

Neighborhoods. A hierarchically organized set of graphs is used to represent urban terrain at different scales in a format that is more symbolic than numeric [Goel,1991]. At a high level, a vertex is a neighborhood and edges are major roads or adjacency relationships. The neighborhood vertices at the high level are expanded into more detailed graphs, with street intersections as vertices and streets as edges. At the lowest level, individual buildings are vertices and their connections to streets are edges. Graph search is used to find progressively more detailed routes by descending into the hierarchy.

ODIN SAF libQuad. The same quadtree-embedded graph used for road route planning in the SIMNET SAF is also used in the ODIN SAF [Stanzione,1993].

Compact Terrain Database. The CTDB uses a graph to represent linear terrain features (roads and rivers) [Smith,1992a] [Stanzione,1993] [Longtin,1994] [Smith,1995b]. Edges are road or river segments and vertices are intersections. In the latest version of the CTDB format, the graph has been moved out of the supplemental quadtree of earlier CTDB versions and integrated into the terrain data patches [Smith,1995b].

CGF Testbed line of sight graph. A graph where the vertices are terrain points of tactical importance (e.g. ridge crests, treeline endpoints) and the edges are unobstructed lines of sight is used to plan reconnaissance routes [Van Brackle,1993a] [Van Brackle,1993b] [Petty,1994b].

Integrated Computer Generated Forces Terrain Database. Roads and rivers are also represented as graphs embedded in the feature quadtree in the ICTDB design [Stanzione,1995].

3.6 Other terrain representations

This subsection briefly presents additional terrain representation methods that do not fall into one of the four primary categories. They are:

1. Scale-space filtering
2. JPL Mobile Robot
3. Run-Length-Code
4. Parameterized microterrain
5. Captain abstract geometric model and semantic net
6. Obstacle segment abstraction

Scale-space filtering. [Keirsey,1988] describes the application of scale-space filtering, a signal processing technique, to identify important terrain features (defined in the reference as local elevation minima and maxima). The terrain elevation data is treated as a 2D signal and processed using a gaussian filter. Varying the filter parameter produces representations of the significant terrain features at different levels of abstraction as small variations in the terrain are smoothed and merged.

JPL Mobile Robot. A gridded terrain representation containing robot sensor data is used as input to an abstraction process that computes a set of abstract spatial terrain features for use in the robot's route planning [Slack,1989] [Ewing,1992]. The abstract terrain features can be primitive, which are computed directly from the attributes of subsets of the terrain grid cells by filtering functions, or composite, which are derived from combinations of primitive or other composite features. The set of abstract features constitutes an alternate terrain representation. [Slack,1989] gives the average slope of the area under the robot as an example of a primitive feature and the preferred escape direction as an example of a composite feature.

Run-Length-Code. Binary (traversable or non-traversable) 2D terrain represented in Run-Length-Code format is used as source data for a entity route planner developed at the Swedish National Defense Research Establishment (FOA) [Holmes,1992]. In the RLC format the terrain is pixelized, i.e. it consists of a rectangular grid of discrete x,y locations. The non-traversable areas are defined using RLC lines, where each RLC line is given by its x,y coordinates and its length. The RLC lines are assumed to run from their given coordinates parallel to the x axis for their given length. Non-traversable areas with a size in the y direction greater than the width of one RLC line are built up from a "stack" of such lines.

Parameterized microterrain. [O'Byrne,1993] suggests representing microterrain (1 meter scale) not explicitly as many small polygons or closely spaced elevation posts, but rather implicitly with

numeric parameters as attributes of macroterrain polygons or grid cells. The reference proposes wavelength and roughness (amplitude) parameters to represent the microterrain existing in the macroterrain area (polygon or grid cell) with which they are associated. The specific models potentially affected by microterrain, such as movement, intervisibility, and combat resolution, would consider the implicit microterrain in their calculations. Though not stated in the reference, it seems obvious that the microterrain parameters for a macroterrain area could be determined by that area's terrain type; for example, a road polygon would have less microterrain than a brush polygon.

Captain abstract geometric model and semantic net. Captain is an automated knowledge acquisition system designed to allow a SME to teach an automated command agent tactical behavior [Hille,1994] [Hieb,1995]. Captain uses two internal terrain representations, creating them in a process termed semantic terrain transformations by its developers [Hille,1995]. First, an input CTDB terrain database is transformed into an abstract geometric model by applying abstraction, generalization, aggregation, and simplification operators that are relevant to the tactical context. The resulting abstract geometric model omits much of the specific detail contained in a CTDB terrain database. Instead, terrain regions are created which have a discrete value in one or more of five classes: relief, cover, mobility, avenue of approach, area of responsibility, and subunit area of responsibility. The abstract geometric model is then transformed into a semantic net, where the classified regions of the abstract geometric model become named objects (e.g. hill-863, avenue-of-approach-2) in the net that are associated with each other by relationships chosen from a predefined set of terrain object relationships (e.g. IN-FRONT-OF, WITHIN). The semantic net is used by a set of inference rules in the automated command agent.

Obstacle segment abstraction. [Rajput,1994b] and [Karr,1995d] describe a hybrid terrain representation combining elements of both gridded and graph representations. Cells of a regular square grid overlaid on polygonal terrain are assigned obstacle segments, which are abstractions of obstacles that block movement contained in the terrain underlying the grid cell. Each cell also is overlaid with 8 to 12 points that become vertices of a graph representing movement routes in the terrain; vertices on opposite sides of an obstacle segment are not connected by an edge in the graph. The graph is searched with the A* algorithm to plan unit routes.

3.7 Summary of CGF terrain representations

Table 3.5 gives a summary of the terrain representations surveyed in this document. Each entry in the table has six components:

1. Category; one of: Gridded, Polygonal, Quadtree, Graph, or Other
2. Application; one of: CGF system, Automated terrain analysis, or Other
3. System/Format; name of system using the representation, e.g. ModSAF, or name of the terrain representation, e.g. SIF/HDI
4. Description; comments elaborating on the Category to describe the representation
5. Reference(s); papers or other references describing the representation

Terrain representations that are used strictly as intermediate working representations in terrain reasoning algorithms will only be listed if they are interestingly different from any of the described representational formats.

Category	Application	System/Format	Description	Reference(s)
Gridded	Automated terrain analysis	Eagle	Fixed size 100 meter square grid cells with several attributes, including elevation.	[Powell,1987] [Powell,1988a] [Powell,1988b] [Powell,1989] [Wright,1990]
	Other (Robotics)	PATH PLAN	Simple gridded terrain with elevation and explicit obstacles as attributes.	[Ok,1989]
	Other	JPL Mobile Robot	Gridded terrain is used to receive and fuse environmental data from the robot's sensors.	[Slack,1989] [Ewing,1992]
	CGF system	Martin Marietta SAFOR	Polygonal data discretized into hexagons with elevation mobility, and exposure attributes for each hexagon.	[Bockstahler,1991]
	Other	Stealth Terrain Navigation	Gridded with elevation only attribute.	[Teng,1992]
	CGF system	ODIN SAF ModSAF	CTDB format achieves considerable database size reduction by using elevation posts to implicitly define polygons over much of the database extent. Microterrain and TIN polygons are given explicitly.	[Smith,1992a] [Stanzione,1993] [Longtin,1994] [Smith,1995b]
	Other (Flight simulator)	NASA Ames LOS	Gridded with elevation only attribute.	[Sansom,1993]

Table 3.5 (Part 1 of 6) Summary of CGF and terrain reasoning terrain representations.

Category	Application	System/Format	Description	Reference(s)
Gridded (cont'd)	CGF system	CCTT SAF	MRTDB is even more compact than CTDB, due to feature representation by reference to a library of archetypical features.	[Watkins,1994] [Campbell,1994] [Watkins,1995] [Pope,1995a] [Pope,1995b]
	CGF system	Iowa Driving Simulator	Elevation post spacing may vary in different portions of the database, and can be set arbitrarily small to represent high-resolution terrain.	[Papelis,1994] [Kuhl,1994]
	Other (Algorithm analysis)	RAND	Gridded with elevation only attribute.	[Marti,1994]
	Other (Data interchange)	SIF/HDI	Gridded elevation with detailed associated feature data.	[Stanzione,1994]
	Other (DIS server)	ARL Variable Resolution	Gridded terrain where elevation is found as sum of parameterized hills.	[Purnell,1995] [Kendall,1995]

Table 3.5 (Part 2 of 6) Summary of CGF and terrain reasoning terrain representations.

Category	Application	System/Format	Description	Reference(s)
Polygonal	CGF system	SIMNET SAF CGF Testbed	Polygons organized into regular square patches and grid cells for faster access.	[Stanzione,1989] [Smith,1992b]
	CGF system	ODIN SAF ModSAF	Though the CTDB is primarily a gridded format, explicit polygons are used for microterrain and TINs.	[Smith,1992a] [Stanzione,1993] [Longtin,1994] [Smith,1995b]
	CGF system	VCom	Polygons in mesh, where all polygons are convex and terrain features are embedded in the polygons.	[Cunningham,1993]
	CGF system	None (proposed)	ICTDB has TIN polygons organized into regular square patches and virtual grid cells for faster access.	[Stanzione,1995]

Table 3.5 (Part 3 of 6) Summary of CGF and terrain reasoning terrain representations.

Category	Application	System/Format	Description	Reference(s)
Quadtree	Automated terrain analysis	None (proposed)	Complete quadtree with terrain object frames bidirectionally linked to nodes.	[Antony,1988]
	Automated terrain analysis	REACT	Quads are assigned abstract terrain types based on DMA elevation data.	[Hayslip,1988]
	Automated terrain analysis	MSPES	Quads have several attributes, including terrain type.	[Doughty,1988]
	CGF system	SIMNET SAF ODIN SAF	Complete quadtree with fixed size quads. Nodes point to arrays representing terrain objects, such as roads and buildings. Stored separately from associated polygonal (SIMNET) or gridded (ODIN) database.	[Stanzione,1989] [Stanzione,1993]
	CGF system	ModSAF	CTDB quadtree represents topological relationships of linear features (e.g. roads) as well as enumerated abstract features. Quadtree is contained within the CTDB patch data structures.	[Smith,1995b]
	CGF system	None (proposed ICTDB)	ICTDB quadtree contains individual and aggregate features. Road and river networks are also stored as graphs within the quadtree.	[Stanzione,1995]
	CGF system	Intelligent Player	Quadtree stores elevation only.	[Pandari,1995]

Table 3.5 (Part 4 of 6) Summary of CGF and terrain reasoning terrain representations.

Category	Application	System/Format	Description	Reference(s)
Graph	Automated terrain analysis	None (proposed)	Edges are roads, vertices are intersections.	[Benton,1987]
	Automated terrain analysis	Eagle	Vertices represent junctions of mobility corridors; edges are weighted for tactical factors (distance, traversal, time, cover, and concealment).	[Powell,1987] [Powell,1988a] [Powell,1988b] [Powell,1989] [Wright,1990]
	CGF system	SIMNET SAF ODIN SAF	Graph embedded in quadtree represents road and river networks.	[Stanzione,1989] [Stanzione,1993]
	None (proposed)	Neighborhoods	Hierarchically organized graphs represent terrain. Vertices are distinct locations, edges show adjacency.	[Goel,1991]
	CGF system	ModSAF	Graph embedded in quadtree represents road and river networks.	[Smith,1992a] [Stanzione,1993] [Longtin,1994] [Smith,1995b]
	Automated terrain analysis	FOA	Vertices represent traversable regions, edges represent region adjacency.	[Holmes,1992]
	CGF system	CGF Testbed line of sight	Vertices are terrain locations of tactical significance, edges represent unobstructed lines of sight.	[Van Brackle,1993a] [Van Brackle,1993b] [Petty,1994b]
	CGF system	ICTDB	Graph embedded in quadtree represents road and river networks.	[Stanzione,1995]

Table 3.5 (Part 5 of 6) Summary of CGF and terrain reasoning terrain representations.

Category	Application	System/Format	Description	Reference(s)
Other	Automated terrain analysis	None (proposed)	Scale-space filtering reveals existence and structure of terrain features at different scales.	[Keirsey,1988]
	Other	None (proposed)	Abstract spatial feature set is derived from gridded terrain using filter functions.	[Slack,1989] [Ewing,1992]
	Automated terrain analysis	FOA	Binary (traversable or non-traversable) 2D terrain is represented in Run-Length-Code format; it is converted to a graph for terrain reasoning.	[Holmes,1992]
	CGF system	None (proposed)	Microterrain is represented implicitly by numeric parameters (wavelength and roughness).	[O'Byrne,1993]
	CGF system	Captain	Abstract geometric model classifies terrain into regions with discrete values for five terrain types.	[Hieb,1995]
	CGF system	Captain	Semantic net relates named terrain objects with predefined topological relationships.	[Hieb,1995]
	CGF system	CGF Testbed	Terrain grid cells are assigned obstacle segments representing impassable obstacles and points that serve as vertices of a graph used for route planning.	[Rajput,1994b] [Karr,1995d]

Table 3.5 (Part 6 of 6) Summary of CGF and terrain reasoning terrain representations.

3.8 Terrain representation comments

3.8.1 Geographic data processing

[Nagy,1979a] is an early survey of geographic data processing; it discusses issues relating to data structures for terrain, geometric relations and operators for terrain objects, and coordinate transformations between terrain representations, some of which is useful background for terrain representation.

3.8.2 Ocean and littoral terrain

By intent, this section has dealt almost entirely with representing land terrain. What of water? The Navy has had some success in modeling deep water, primarily for anti-submarine warfare applications. However, a terrain modeling area which has yet to be fully understood is the ocean littoral, or coastal, zone. [Donner,1991] presents some methods for mathematical and polygonal modeling of the ocean surf zone. [Haeger,1994] outlines some requirements for a littoral terrain representation, but does not provide any terrain representation design. [Craft,1995b] lists some enhancements to the CTDB format that would be useful to support amphibious vehicle operations in littoral terrain.

3.8.3 Multiple representations and terrain correlation

CGF systems sometimes use multiple terrain representations (and these CGF terrain representations are almost always different from the terrain database used by the image generators in the simulation system). The intent is that each CGF terrain reasoning task is performed on the representation that best supports that task. Table 3.7 shows how the tasks are assigned to the different terrain databases in five CGF systems. Recall that each of the CGF systems in the table has two components: a simulator, that simulates the dynamics and behavior of the CGF entities, and an operator interface, that allows a human operator to control the CGF entities. For ease of reference, Table 3.6 identifies those components for each of the CGF systems. Table 3.7 shows the terrain representation formats used by the two components for each system. The table is meant to give examples of how different terrain reasoning tasks are performed using different terrain representation formats, and is not an exhaustive list of terrain reasoning capabilities of any of the CGF systems contained therein.

CGF System	Sim = Simulator	OI = Operator Interface	Reference
SIMNET SAF	Simulation Host	SAFOR Workstation	[Downes-Martin,1990]
CGF Testbed	Simulator	Operator Interface	[Smith,1992b]
ODIN SAF, ModSAF	SAFsim	SAFstation	[Stanzione,1989]
ModSAF	SAFsim	SAFstation	[Ceranowicz,1994a]
CCTT SAF	CGF	SAF Workstation	[Marshall,1994]

Table 3.6 CGF system component names.

CGF System	Cmpnt *1	Gridded	Polygonal	Quadtree	Graph *2
SIMNET SAF	Sim	na	Orientation and elevation Intervisibility determination	Entity route planning Obstacle avoidance	na
	OI	na		Unit route planning Road route planning 2D map display	na
CGF Testbed	Sim	na	Orientation and elevation Intervisibility determination Ground collision detection Entity route planning Unit route planning Reconnaissance route planning Obstacle avoidance	na	na
	OI	na	2D map display Contour line display	na	na
ODIN SAF	Sim	Elevation and orientation Intervisibility determination Vehicle mobility Ground collision detection	na	Entity route planning Finding cover and concealment	Vehicle route planning
	OI	Intervisibility display Contour line display Hypsometric elevation display Shaded relief display Terrain cross-section display	na	Unit route planning Road route planning Route checking 2D map display	Unit route planning Road route planning Route checking

*1 Sim=Simulator component, OI=Operator Interface component; see Table 3.6 for system-specific names.
*2 In most of the instances listed in this table, the Graph representation is embedded in the Quadtree representation.
na = not applicable

Table 3.7 (Part 1 of 2) Terrain reasoning task assignment by terrain reasoning format in CGF systems.

CGF System	Cmpnt *1	Gridded	Polygonal	Quadtree	Graph *2
ModSAF	Sim	Elevation and orientation Intervisibility determination Vehicle mobility Ground collision detection	na	Entity route planning Finding cover and concealment	Vehicle route planning
	OI	Intervisibility display Contour line display Hysometric elevation display Shaded relief display Terrain cross-section display	na	Unit route planning Road route planning Route checking 2D map display	Unit route planning Road route planning Route checking
CCTT SAF	Sim	Elevation and orientation Intervisibility determination Entity route planning Obstacle avoidance Unit route planning Area intervisibility Finding cover and concealment	na	na	na
	OI	na	na	2D map display	na

*1 Sim=Simulator component, OI=Operator Interface component; see Table 3.6 for system-specific names.
*2 In most of the instances listed in this table, the Graph representation is embedded in the Quadtree representation.

Table 3.7 (Part 2 of 2) Terrain reasoning task assignment by terrain reasoning format in CGF systems.

References for Table 3.7

SIMNET SAF [Stanzione,1989]
CGF Testbed [Smith,1992b] [Petty,1994b]
ODIN SAF [Stanzione,1993]
ModSAF [Stanzione,1993] [Smith,1995b]
CCTT SAF [Watkins,1994] [Campbell,1994]

Of course, having multiple representations introduces the possibility that the different representations of the same terrain may be inconsistent; that possibility is observed in [Papelis,1994] and [Schiavone,1995], for example. The potential for terrain inconsistencies also exits between different simulator nodes that may use different terrain representation formats. The issue of whether two terrain databases that purport to represent the same terrain are consistent is known as *terrain correlation*, and inconsistencies are referred to as *terrain correlation error*. Terrain correlation error can seriously erode the realism of a simulation. For example, a terrain feature such as a treeline that is differently located in two terrain databases can result in an entity concealed on one database and exposed to enemy direct fire on another, producing an unfair fight situation.

The general terrain correlation problem, while important and interesting, is beyond the scope of this document. The interested reader is referred to:

1. Survey and tutorial on terrain correlation [Schiavone,1995]
2. Terrain correlation definitions and metrics [Zvolanek,1992] [Zvolanek,1993]
3. Mechanism through which correlation errors between the two complementary representations in the ODIN SAF and ModSAF (CTDB gridded and quadtree) are avoided [Stanzione,1993]
4. Terrain correlation certification using statistical hypothesis testing [Schiavone,1994] [Goldiez,1994]
5. Terrain correlation testing associated with the I/ITSEC DIS Interoperability Demonstrations in 1993 [Goldiez,1994] and 1994 [Nelson,1995]
6. Early ideas on measures of terrain correlation [Wever,1989]
7. Terrain correlation errors due to differences in terrain accessing and process algorithms, rather than data discrepancies, and how those errors are avoided in the CCTT terrain modules [Watkins,1995]
8. Terrain correlation errors due to overly simplistic feature representations [Watkins,1995]
9. Avoiding terrain correlation errors by using common source data for terrain databases intended to correlate [Stanzione,1989] [Loper,1993]
10. Recommendation for a standard terrain representation for DIS so as to avoid terrain correlation errors [Trott,1995]
11. Correlation of terrain databases for different sensor types [Fawcett,1991]

4. Terrain reasoning in CGF

In this section CGF terrain reasoning algorithms are examined. In order to organize the exposition, special attention will be given to terrain reasoning algorithms for three crucial CGF terrain reasoning tasks: route planning, intervisibility, and finding cover and concealment. In each case the task is defined and then the existing CGF terrain reasoning algorithms for the task are presented. Finally, after route planning, intervisibility, and finding cover and concealment have been examined, several other interesting terrain reasoning problems will also be briefly surveyed.

4.1 Terrain reasoning in military tactics

Military terrain reasoning, which informally is the analysis and understanding of terrain so as to increase tactical effectiveness, is of paramount importance in military operations, especially ground operations. Historically, the military significance of terrain and terrain reasoning has been observed by soldiers, historians, and military experts for centuries. Numerous examples could be cited; three, arbitrarily chosen, will be mentioned here. First, Sun Tzu's timeless classic *The Art of War* [Sun,600BC] devotes an entire chapter to terrain reasoning. Second, in their descriptions of some of the first actions fought in World War I by units of the French Foreign Legion newly transferred to France from Algeria, both [Porch,1991] and [Reybaz,1932] identify lack of combat experience in European terrain and a lack of training in "the utilization of terrain" as one reason for the heavy losses suffered by those units. Finally, in [Keegan,1994], which is a panoramic and thematic overview of the entire history of human military conflict, the importance of the effective tactical use of terrain is observed in discussions of the startling successes of the Arab armies carrying Islam through the ancient world, the tactics of Renaissance-era Swiss pikeman versus musketeers, and the victories of the Viet Minh against the French in Indochina.

Moving from military history to current military practice, the Gulf War made apparent the fact that modern weapons are accurate and lethal to an unprecedented (and unexpected) degree [Bonsignore,1992]. This fact has made terrain reasoning even more crucial in that suboptimum use of terrain can result in the abrupt destruction of a fighting force. The importance of terrain reasoning is directly asserted in [Schmitt,1988], a U.S. Marine Corps training manual for company commanders:

"Terrain has an immense influence on how the battle will be fought. Proper evaluation and utilization of terrain may reduce the disadvantage of incomplete information of the enemy. Terrain provides opportunities and imposes limitations, giving a decisive advantage to the commander who uses it best. Many battles are won or lost by the way in which the commander uses terrain to protect his force and to bring effective fire to bear on the enemy."

The corresponding U.S. Army manual [U.S. Army,1988], echoes the admonition:

"Master the art of clever use of terrain." "Proper use of terrain, ... is crucial to the company team's survival on the battlefield."

With the understanding that terrain reasoning is essential for the success and survival of actual military forces, it becomes clear that effective terrain reasoning by CGF systems for their CGF entities is also important if the CGF systems are to usefully simulate their real-world counterparts.

4.2 Terrain reasoning definitions

A number of terms are used frequently in terrain analysis and terrain reasoning; they will be defined here. Some (but not all) of these definitions are adapted from [Powell,1987] and [Schmitt,1988].

1. Terrain reasoning
2. Intervisibility
3. Observation
4. Field of fire
5. Cover
6. Concealment
7. Obstacles
8. Key terrain
9. Mobility corridor
10. Avenue of approach
11. OCOKA
12. No-go, Slow-go, and Go terrain
13. Chokepoint
14. Dynamic terrain

Terrain reasoning. As defined earlier, terrain reasoning in a CGF system is the automated analysis of a digitized terrain representation for the purposes of making behavioral decisions involving the terrain. The overall intent of CGF terrain reasoning is that the behavior of the CGF entities be based on the terrain to the extent dictated by doctrine. The CGF terrain reasoning problem is that of developing algorithms to perform terrain reasoning tasks that are computationally efficient and that lead to CGF behavior that is realistic and tactically effective. CGF terrain reasoning tasks include route planning, seeking cover and concealment, and finding locations that maximize fields of fire.

Intervisibility. The fact of whether or not an unobstructed line of sight exists from one entity to another, or the process of making that determination. Unobstructed means that the line of sight does not intersect intervening terrain (surface or features) or entities.

Observation. The presence of an unobstructed line of sight from a given location to a location (or set of locations, i.e. an area) in question. (Compare observation with intervisibility; observation refers to location to location, while intervisibility is entity to entity.)

Field of fire. The terrain area which, for a given entity and from a given location, is visible and within effective weapon range.

Cover. Protection by terrain from observation and direct fire. For example, a ridge might provide cover to an entity behind it. Cover can be provided by ditches, defiles, river banks, craters, buildings, and so on.

Concealment. Protection by terrain from observation (but not direct fire). For example, a treeline obstructs observation, but can be fired through. Note that both cover and concealment are relative to a direction (or range of directions); a treeline provides concealment from one direction, but not the opposite direction. The direction for which cover and concealment are defined is usually the direction of known or expected enemy forces.

Obstacles. Natural or man-made terrain features that slow, stop, or deflect movement. Examples include rivers, embankments, and mine fields. An obstacle may also be simply an object to avoid while moving, such as another entity.

Key terrain. A terrain area whose seizure or control offers significant advantage to the possessor. Key terrain is typically characterized as having observation or fields of fire to nearby avenues of approach (defined later). Key terrain areas are often final or intermediate mission objectives. It is possible that a key terrain feature or area dominates the battlefield to such a degree that the overall outcome of the battle depends on its control; such terrain is known as *decisive terrain.*

Mobility corridor. A relatively open terrain area through which a military unit can move.

Avenue of approach. A set of one or more mobility corridors of sufficient size for a given military unit to move. Avenues of approach are often defined relative to a destination, an objective, or key terrain. They must be broad enough to allow the unit to maneuver and bypass obstacles and enemy centers of resistance. Clearly, larger units require larger avenues of approach (and mobility corridors). Good avenues of approach offer both speed of movement and cover and concealment.

OCOKA. An acronym for the five characteristics for which military planners are trained to analyze terrain: Observation and fields of fire, Cover and concealment, Obstacles, Key terrain, and Avenues of approach. The terrain reasoning algorithms to be examined later should also consider the OCOKA characteristics.

Chokepoint. A point or small area through which multiple alternate movement routes are forced to pass due to obstacles or no-go terrain. A single bridge over an unfordable river is an obvious example of a chokepoint.

Dynamic terrain. Terrain that may change during a simulation exercise due to the actions of agents or environmental effects in the simulation. The terrain reasoning algorithms to be covered in this section will usually treat dynamic terrain, which can change over time, as instantaneously static. This means that the terrain may change between terrain reasoning computations, and the changes to the terrain will be considered by the terrain reasoning algorithms when they are invoked, but it is assumed that the terrain does not change during a single terrain reasoning computation.

No-Go, Slow-Go, and Go terrain. Terrain which significantly hinders movement, somewhat hinders movement, or does not hinder movement, respectively. Characteristics that are used to categorize a given terrain area as No-Go, Slow-Go, or Go are given in Table 4.1.

Characteristic	No-Go	Slow-Go	Go
Urban buildup	Present, > 500 meters wide	Present, < 500 meters wide	None
Rivers and waterways	Present, fordable or spannable nowhere	Present, fordable or spannable in several places	None; or Present, fordable or spannable everywhere
Uphill slopes	> 45% uphill	30% to 45% uphill	< 30% uphill
Elevation variation per kilometer	> 200 meters	100 to 200 meters	< 100 meters
Obstacles	Man-made or military	None	None
Trees	> 15 cm thick and < 6 meter spacing	5 to 15 cm thick and < 6 meter spacing	< 5 cm thick or > 6 meter spacing
Hard surface roads per kilometer	0	1	2 or more

Table 4.1 No-Go, Slow-Go, and Go terrain (adapted from [Powell,1987]).

4.3 Route planning

This subsection examines route planning, which is perhaps the fundamental CGF terrain reasoning task. It begins by defining route planning, describing the two basic approaches to route planning, and commenting on how route planning problems and algorithms differ at different hierarchical levels and geographical scales. Then route planning algorithms are surveyed in three categories: entity route planning, unit route planning, and reconnaissance route planning.

The algorithms surveyed focus almost exclusively on ground vehicles; they plan routes that follow the surface of the terrain. Throughout this subsection it should be assumed that the entities are ground entities and the routes follow the terrain's surface unless stated otherwise.

4.3.1 Definition

In general terms, CGF *route planning* is the process of algorithmically generating a movement path, or route, for a CGF entity or unit from a given starting location to a given destination location across the surface of a given terrain database. The route as generated should avoid obstacles, which may be impassable terrain features, no-go terrain areas, or other entities. A *waypoint* is a point on a route. If given as input to a route planner, a waypoint is a point through which the route is constrained to pass; if returned by a route planner, it is usually a point at which the route changes direction. Route planners often return the routes they produce as a sequence of waypoints, under the assumption that the route proceeds along a straight line from each waypoint to the next waypoint in the sequence. The waypoint list is then handed over to a route follower, which actually moves the simulated entity along the route over time. This subsection will focus on route planning.

Route planning was chosen for examination in this survey because it is a ubiquitous terrain reasoning problem for CGF systems, both essential to a CGF system's behavior generation and closely linked to the terrain. Furthermore, route planning has been widely studied, as evidenced by the large body of literature on the subject. This is likely true for several reasons; first, route planning is important in several application areas, including CGF and robotics; second, route planning is a primitive behavior for CGF entities, which must be available before higher level cognitive behavior in the general category of movement planning can be built; and finally, the route planning problem is often easily simplified or abstracted into a form amenable to solution using precise algorithmic methods.

In regards to abstract route planning algorithms, [Mitchell,1988] is an excellent theoretical and analytical survey of route planning algorithms considered under idealized geometric circumstances. Several of the route planning algorithms used in CGF systems and described in this subsection can be seen as based on those found in [Mitchell,1988] and adapted to the particular terrain representations used by the CGF system. As for the suitability of the ideal algorithms, [Karr,1995d] mentions two problems with applying the idealized algorithms to the "gritty" details of an actual CGF system's terrain representation, which are typical of the type and scope of such difficulties.

4.3.2 Route planning approaches

[Benton,1991] separates route planning algorithms into two categories, grid-based and graph-based, and nearly all route planning algorithms do in fact fall into those categories. In grid-based route planning, the terrain is first discretized into a grid, if it is not already in a grid or cellular representation. Once the grid is available, its cells are then searched, usually by exploring all cells adjacent to a partial route in a depth-first manner. When the cell containing the destination is reached a route is constructed as a sequence of segments from one grid cell to another.

In contrast, in a graph-based route planner the terrain is first abstracted into a graph, as described earlier, with the edges typically weighted to reflect parameters relevant to route planning, such as length, trafficability, exposure to enemy fire, and so on. The graph is then searched (often using an A* algorithm; for examples see [Benton,1987], [Powell,1988b], [Stanzione,1989], [Holmes,1992], [Cunningham,1993], [Stanzione,1993], [Campbell,1995], [Longtin,1995] and [Karr,1995e]); when the destination is found by the search a route is returned as a sequence of graph edges. Depending on what information is used to weight the edges, the graph may be computed as a pre-processing operation and stored as an alternative or supplement to another terrain database in a different format. Clearly, graph-based planning is more general; a terrain grid representation is a special case of a terrain graph where the vertices correspond to the grid cells' center points and the edges correspond adjacent grid cells. In [Marti,1994], a terrain grid is searched with an A* algorithm using precisely that interpretation.

A simple application of grid-based planning of long routes can lead to two significant problems. First, a grid sufficiently fine-grained to permit passage between narrowly spaced obstacles can require excessive memory to store. Second, searching the grid in a simple breadth-first manner

can be prohibitively costly in terms of processor utilization. Hierarchical and hybrid methods are often used to avoid those problems; they will be explained next.

4.3.3 Route planning levels

Route planning is often conducted at two different scales or organizational levels in CGF systems, high-level and low-level. Generally speaking, *high-level* route planning algorithms are used to plan routes that:

1. are long (> 1 Km)
2. follow roads over much of their length
3. are more often for units than entities
4. avoid large terrain obstacles, such as rivers, urban areas, and forests
5. do not consider small terrain obstacles, such as individual trees and buildings
6. do not consider dynamic obstacles, such as moving entities

On the other hand, *low-level* route planning algorithms plan routes that:

1. are short (μ 1 Km)
2. seldom follow roads, are primarily cross-country
3. are more often for entities than units
4. avoid small terrain obstacles, such as individual trees and buildings
5. consider large terrain obstacles at the component level, i.e. a forest is a collection of individual trees to be avoided
6. predict the movement of and avoid dynamic obstacles, such as moving entities

The high-level and low-level route planning problems differ fundamentally in physical scale; long vs. short distances, large vs. small obstacles, units vs. entities moving. They also differ in time scale, both in traversal time and planning time. High-level routes take longer to traverse than low-level routes, simply as a function of their greater length. Additionally, there is almost always more time available to plan a high-level route than a low-level route. A movement route for a battalion-sized unit may be generated as part of a battalion plan which in reality can take minutes or hours to produce, whereas a low-level route for a single entity must often be planned in less than a second to preserve realistic CGF response times.

The difference between the two levels of route planning has resulted in something of a dichotomy in route planning algorithms. Many route planning algorithms are specifically designed for either low-level or high-level route planning. Some CGF systems use different algorithms at the different levels; both ModSAF [Smith,1994] and the CCTT SAF [Campbell,1995] operate in that manner. A recurring (but not universal) theme is a highly structured algorithm, such as A*, used for high-level route planning to produce high-level routes that are completely planned in advance, combined with a low-level route planning scheme that either produces low-level plans for segments of the high-level route as those segments are traversed or possibly even traverses the segments in a reactive heuristic manner without detailed advance planning.

For example, in the IST CGF Testbed, long routes are partitioned into a series of *segments*, each of which is assumed to be traversable. (Here a segment is defined as a portion of a route, rather

than having its geometric definition as a portion of a line.) A detailed route is planned for each segment using the Testbed's grid-based route planner as the moving entity reaches the waypoint that ends the previous segment and begins the next. If a segment turns out to be untraversable due to an impassable obstacle (e.g. a river without bridges or fords) the route planning process fails (and the CGF operator is so notified). As another example, the CCTT SAF also uses a high-level route planner to plan to produce overall routes around large scale obstacles, such as rivers, forests, and urban areas, and then applies a different low-level route planner to produce routes that avoid individual trees, buildings, and entities [Campbell,1995]. Finally, [Benton,1991] describes the Hierarchic Route Planner, a route planner that combines a grid-based and a graph-based route planner. A cross-country mobility graph is searched to produce the large scale route and a grid-based planner finds precise paths and traversal times between the nodes of the graph.

Of course, algorithms have been designed that use the same route planning approach as both levels; e.g. the unit route planner described in [Rajput,1994b] and [Karr,1995d] can be applied to entity level route planner by simply varying the algorithm's parameters.

The next two subsubsections will survey route planning algorithms for low-level route planning and high-level route planning, respectively. Following the descriptive distinction most often made in the literature between the two levels, they are referred to as *entity route planning* and *unit route planning* respectively.

4.3.4 Entity route planning

Entity route planning is the simplest and most thoroughly studied form of the route planning problem. A obstacle-free route must be found from the given starting location (usually the entity's current location) to the destination location on a given terrain database. The mobility capabilities of the specific entity for which the route is being planned must be considered.

This subsubsection will describe several entity route planning algorithms. They are:

1. Potential fields
2. REACT
3. PATH PLAN
4. Martin Marietta SAFOR
5. Wavefront expansion
6. Stealth terrain navigation
7. FOA
8. RAND
9. ModSAF Near Term Navigation

Potential fields. Though it is more of a reactive route finding method than a route planning algorithm, CGF movement control based on potential fields has been proposed repeatedly. The potential field scheme was originally devised for robot movement control; for example, see [Arkin,1987] or [NASA,1993]. [Le,1991a] and [Le,1991b] advance potential fields as specifically applied to CGF. The idea is based on the notion of artificial charged force fields with two possible charge polarities; like charges repel and opposite charges attract. Things a moving

CGF entity should avoid, such as terrain obstacles, areas exposed to enemy observation, and other entities, are assigned a charge that matches the moving entity, and thus repel it. Things that the moving entity should move towards, such as its movement goal or its place in a formation, are given the opposite charge, and thus attract it. At each time step, the direction (based on location) and force (based on assigned charge strength and distance) of each repeller and attracter are summed to produce a total movement vector, which is assigned to the moving CGF entity.

The potential field scheme has appeal in that it unifies into one method a number of route planning and movement control considerations, including movement towards a goal avoidance of terrain obstacles, avoidance of terrain areas exposed to enemy observation, dynamic obstacle avoidance, and formation keeping. However, the approach has not been applied in any production CGF system due to several problems. First, determining the correct relative values for the potential field strengths of the repellers and attracters is difficult. Second, it is computationally expensive, requiring a recalculation of potential field values at each time step, as compared to a route planner that generates a route once and saves it. Finally, it is quite possible for the moving entity to be routed to a local maxima of the potential fields and become trapped, unable to make progress towards its final destination [NASA,1993].

REACT. The algorithm described in [Hayslip,1988] plans movement direction for a single aircraft using a table look up based on a quadtree of abstract terrain types.

PATH PLAN. [Ok,1989] presents an entity route planning algorithm which searches gridded terrain. It differs from classic search algorithms, such as A*, in several ways. No a priori knowledge of the terrain is assumed; instead, the algorithm uses only terrain information acquired by a short range sensor while the entity moves. If follows therefore that only local terrain data is used in the search. The search (and the entity's movement) is guided by simple heuristics. While the algorithm does not produce an optimum route, in the cases tested it did come close to the optimum route found by A* while requiring much less computation.

Martin Marietta SAFOR. In [Bockstahler,1991] vehicle routes are found by converting a polygonal terrain database into a hexagonal grid, with the hexagons weighted for elevation, mobility, and exposure. The hexagonal grid is searched with an A* algorithm to find a minimum weighted route.

Wavefront expansion. The IST CGF Testbed route planner uses a wavefront expansion algorithm, previously described in [Moore,1959], to plan entity routes from a given starting location to a given ending location. [Smith,1992b] explains the process in detail. The algorithm proceeds as follows (also see Figure 4.1):

(1) Create a route planning grid. Overlay a square array, or grid, on the underlying SIMNET format polygonal terrain, with the size of the grid cells approximately equal to the size of the entity for which the route is being planned. The grid boundaries are oriented parallel with the north-south and east-west axes of the terrain and is large enough to encompass the starting and ending locations. The cells of the grid are all initially considered to be unobstructed.

(2) Access the terrain database's feature lists to find terrain features that fall within the

grid's boundaries. For features defined as obstacles to movement (rivers, buildings, treelines, canopies, and others), their locations given in the terrain database are used to determine the grid cell(s) they occupy. Mark those grid cells as obstructed. A diagonal buffering process is used to fill in grid cells adjacent to diagonal obstacles to prevent unintended passage through diagonal obstacles.

(3) Use Bresenham's algorithm [Foley,1982] to traverse the grid from the starting location to the ending location to determine if a direct route of unobstructed grid cells exists. If so, return the starting and ending locations as waypoints and stop.

(4) Assign the grid cell containing the starting location the number 1.

(5) Repeat until the grid cell containing the ending location has been assigned a number: For every unnumbered grid cell adjacent to a numbered grid cell, assign the unnumbered grid cell a number equal to the number of the lowest numbered adjacent grid cell plus 1.

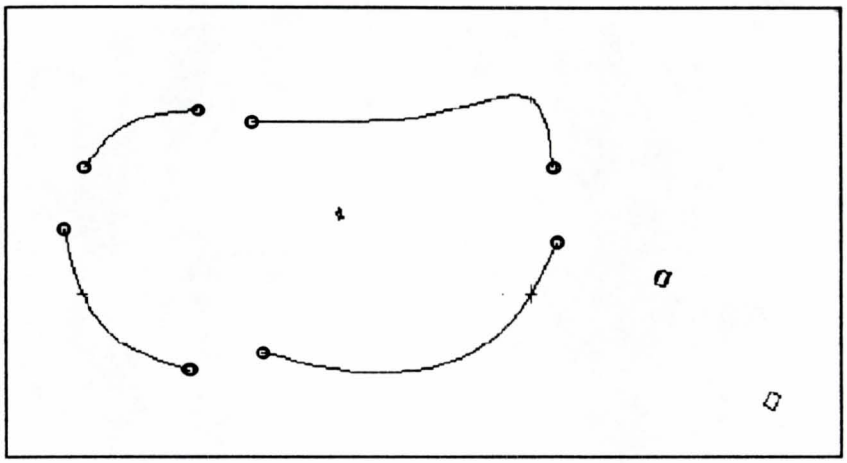
(6) Beginning with the grid cell containing the end location, track back from each grid cell to an adjacent grid cell along decreasing numbered grid cells to the starting location. While doing so, if there are more than one like numbered grid cells to choose from, select the one closest in Euclidean distance to the grid cell containing the starting location. Record the sequence of grid cells so chosen as a sequence.

(7) Eliminate from the sequence any grid cells that fall on a straight line between two other grid cells in the sequence.

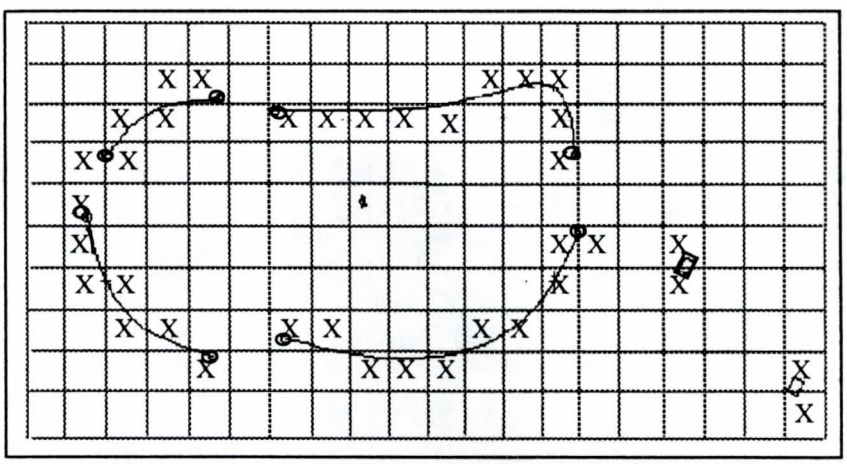
(8) Convert the list of grid cells to a waypoint sequence by taking as waypoints the point corresponding to the center of the grid cells.

(9) To avoid "stair-stepping" in the route caused by grid granularity ([Mitchell,1988] calls the effect "digitization bias"), apply a route relaxation test to each sequence of three waypoints on the list, removing the middle waypoint of the sequence if a direct unobstructed path exists from the two endpoints of the sequence. Return the relaxed waypoint sequence.

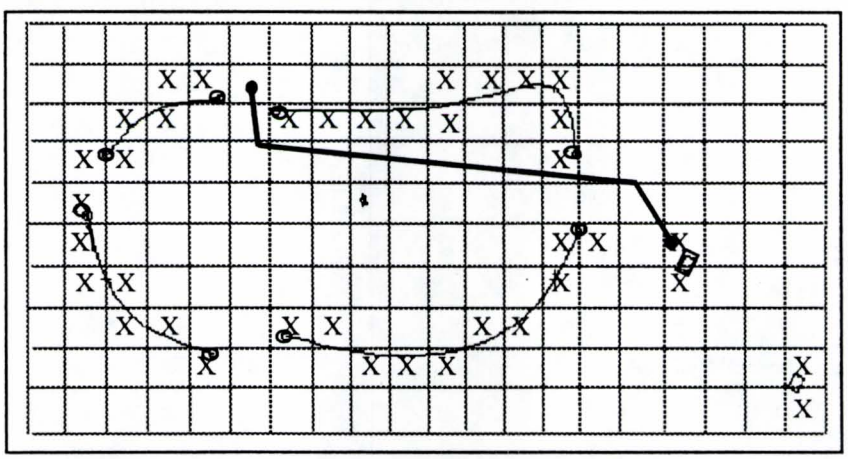
This algorithm is generally fast and effective, but it does have some limitations. First, as mentioned earlier, it may be necessary to break long routes up into segments and plan the route using the wavefront expansion algorithm along each segment; this is due to the potentially large amount of memory required to store the route planning grid. The waypoints that are the ending waypoint of one segment and the starting waypoint of the next may be given by the CGF operator or determined by the algorithm. Second, it is possible that the grid granularity can result in small passages between obstacles being missed by the algorithm; this problem may be exacerbated by the diagonal buffering process. Bridges especially are prone to being overlooked. Nevertheless, the algorithm has operated satisfactorily and reliably in practical application [Chervenak,1993] [D'Errico,1994].



(a) Terrain



(b) Terrain with grid and obstruction cells marked.



(c) Terrain with grid, obstruction cells marked, and route.

Figure 4.1 Wavefront expansion algorithm example.

Stealth terrain navigation. [Teng,1992] details an entity route planning algorithm based on dynamic programming that was designed for a parallel machine architecture. Within the context of several simplifying assumptions, the algorithm selects optimum routes that consider terrain traversibility and exposure to enemy observation. The algorithm assumes a highly discretized environment where the terrain is gridded, entities are located discretely in the terrain grid cells (i.e. the location of an entity is specified precisely as a grid cell), entities move discretely from grid cell to grid cell only in orthogonal directions, and time proceeds discretely in time steps that are the minimum amount of time an entity might require to move from one grid cell to another.

Given a starting position and a fixed time interval, the algorithm calculates the grid cells reachable during that interval and assigns each grid cell a numeric evaluation of the best route to that grid cell. The evaluation of a grid cell is the amount of time not exposed to enemy observation spent moving along the best route to that grid cell. The exposure to enemy observation is based on projecting enemy entity movements over the time interval. The evaluation is calculated using a dynamic programming approach. It examines for each grid cell (the subject grid cell) at each time step within the interval the previously found best routes to each of the subject grid cell's neighbors. Those best routes are considered at the previous time step corresponding to the length of time required to move from the neighboring grid cell to the subject grid cell and the exposure to enemy observation that would occur when moving from the neighboring grid cell to the subject grid cell. It then extends the route from the neighbor to the subject grid cell that produces the least total exposure, keeping grid cell-to-grid cell pointers so that the route can be followed. The algorithm depends heavily on a parallel architecture for practicality; the route evaluation and the exposure to enemy observation of a grid cell are both recalculated for every grid cell in the terrain at every time step when planning a route, each by a separate (virtual) processor. This entity route planning algorithm is used for both a unit route planning algorithm and a bounding overwatch algorithm, both of which will be described later.

FOA. [Holmes,1992] provides extensive details on an entity route planning algorithm developed at the Swedish National Defence Research Establishment (FOA) that searches a graph terrain representation using an A* algorithm. Vertices in the graph represent traversable regions rather than points, so a route in the graph is a sequence of regions through which the entity must move. Such a sequence is called a symbolic route and uses a rule-based inference process to classify and describe the route in symbolic terms, e.g. "left turn". Because it consists of regions, the symbolic route must be refined to give a specific sequence of points, i.e. a geometric route, for an entity to follow; that refinement process is also described in the reference.

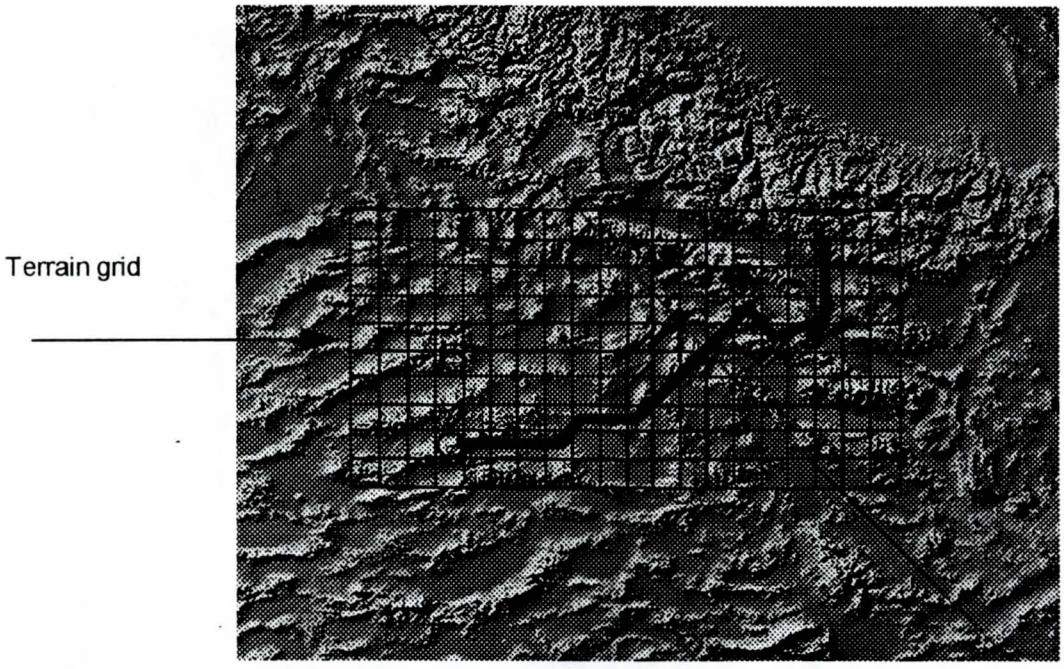
RAND. The entity route planning algorithm described in [Marti,1994] searches a terrain grid using an A* algorithm. The cost function is a composite function of distance and slope between elevation posts. This research is interesting because its intent was not to develop a new route planning algorithm but to quantify the relations between terrain resolution, route planning computation time, and route quality. The reference reports a number of valuable results, including the observation that using higher resolution terrain data (i.e. more closely spaced elevation posts) provides a linear improvement in path quality for an exponential increase in data space. The reference shows that for a route planning algorithm that uses an overlaid grid the grid should be aligned with the line connecting the route's start and end locations rather than with the coordinate axes of the terrain database; see Figure 4.2. [Marti,1994] also includes quantitative values for optimum size and resolution of the route planning grid.

ModSAF Near Term Navigation. As mentioned, ModSAF partitions "movement control", or route planning, into unit-level (high-level) and entity-level (low-level) [Smith,1994] planning. Long-term route planning done at the unit level is not time critical and is performed with traditional planning techniques. However, ModSAF's entity-level short-term route planning is carried out with an interesting and impressive technique described in [Smith,1994]. The method allows moving entities to avoid terrain obstacles (such as treelines, trees, lakes, rivers, and buildings), to cross bridges, to avoid collision with other moving entities, and to keep their places in a formation. Note that the method as described assumes that the moving entity is moving along the surface of the terrain.

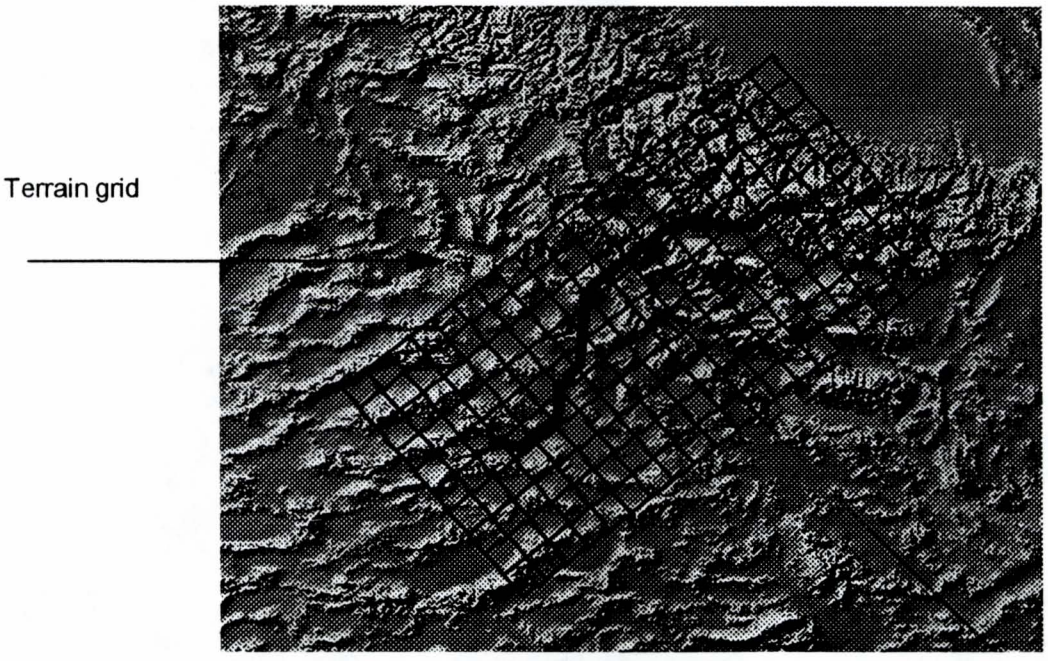
The method depends on the use of an internal representation of space and time, referred to as the *map*, that combines all relevant movement constraints (such as obstacles and moving entities) and goals (such as the destination location and roads). The map is a three dimensional representation where the first two dimensions are spatial and the third is temporal. Points in the map are denoted with the triple *(x,y,t)* and represent a specific point *(x,y)* on the surface of the terrain at a specific moment in time *(t)*.

While obstacles are typically three dimensional objects in the simulated world, their third dimension is not represented in the map. Instead they are represented by their 2D bounding volume. Stationary obstacles form infinitely tall vertical "towers" in the map, corresponding to occupying their 2D bounding volume in the *x,y* plane at all times in the map. Moving obstacles form "leaning towers" in the map as their *x,y* locations change over time. A route is a curve through the map that does not intersect any of the obstacles' volumes and connects the starting and ending locations of the routes. Figure 4.3 suggests an example of the map with a stationary obstacle, a moving obstacle, and a curve corresponding to a route that avoids those obstacles.

Unlike many route planning methods in which movement speed is represented in a way distinct from the spatial route (if at all), the ModSAF method includes speed in the route representation directly as the curve corresponding to the route slopes through the third dimension of the map. Routes planned by this method use speed changes to avoid moving obstacles. A route that slows down will increase its slope in the map relative to the *x,y* plane, whereas a route that speeds up will decrease it.



(a) Grid aligned with coordinate axes



(b) Grid aligned with start and end locations

Figure 4.2 Two different route planning grids for the same route end points (adapted from [Marti, 1994]).

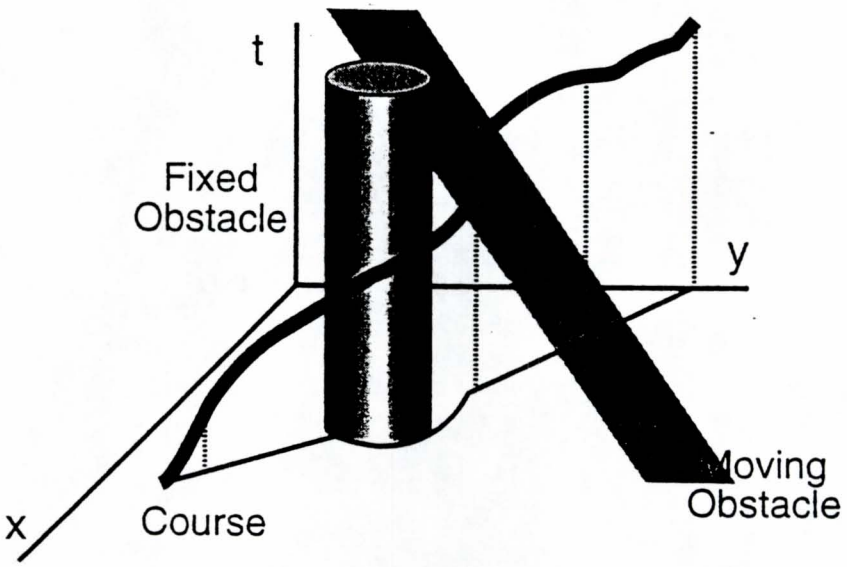


Figure 4.3 ModSAF route planning map [Smith,1994].

The route planner plans a route by finding a curve in the map from the entity's current location to its goal location. The initial curve is the straight line connecting the current and goal locations. The curve is modified by adding control points (points through which the curve must pass) to the curve until it does not intersect any obstacles in the map. Control points that lead the curve around obstacles in the spatial dimensions are computed using cubic splines; in particular, a generalized form of the Catmull-Rom spline, in which the derivative at each control point is parallel to the line connecting the previous control point to the next one, is used. Obstacle avoidance in the temporal dimension is performed by varying the entity's movement speed, and is computed by integrating a series of successive speed-related derivatives. See [Smith,1994] for the equations.

Once an acceptable curve (i.e. one that does not intersect any obstacles) has been found, it is reexamined for violations of physical performance constraints of the moving entity. The violations considered are excessive speed, excessive turn rate, excessive deceleration, excessive acceleration, and sharp turns. Additional spatial and temporal control points are added as needed to the curve so as to remove the violations.

The goal location for each route planner execution is either the entity's final destination, or a location corresponding to the location the entity should occupy along the previously planned high-level route 2.5 seconds into the future. The 2.5 second interval is the *planning horizon* of the route planner. The *execution horizon* of the route planner is 0.5 seconds. The route planner will replan an entity's route as soon as either the execution horizon has elapsed since the last route plan was produced for that entity or a new obstacle has appeared within the 2D area bounded by the map. Using a planning horizon larger than the execution horizon provides smoothness and continuity between successive routes.

The curve that results from the planning process specifies a route completely, giving both the x,y route to follow and the speed at which the entity should be as it follows the route. Once calculated, the route is passed to the ModSAF vehicle dynamics routines, which move the entity along the route.

As far as could be determined from the reference, the ModSAF route planning method ignores the z coordinate (the elevation) of the terrain. Consequently, the effects of terrain slope on entity speed are neglected. Furthermore, areas of extreme slope that should thereby constitute obstacles are not so treated by the method.

4.3.5 Unit route planning

Unit route planning differs from entity route planning in several ways. One is that width of the route becomes an important consideration. A gap between two obstacles sufficient to allow passage for an individual entity may be a tactically unacceptable chokepoint for a company. A second is that units, more often than entities, are constrained to have all or part of their route on roads. Finally, units normally move in formation and a route that permits the unit to maintain its formation while moving is generally desirable.

This subsubsection will present several unit route planning algorithms, all of which account for one or more of these special considerations. They are:

1. Multiple route finder
2. Eagle
3. SIMNET SAF
4. Stealth terrain navigation
5. Mesh
6. Obstacle segment abstraction
7. ModSAF concealed routes

Multiple route finder. [Benton,1987] describes a variation of the A* algorithm to perform unit route planning in a road network. The A* algorithm was modified to find and return multiple non-interfering routes through the graph.

Eagle. [Powell,1988a], [Powell,1988b], [Powell,1989], and [Wright,1990] explain an interesting unit route planning algorithm used in an automated terrain analysis application. ([Powell,1987] describes a less developed predecessor of the algorithm.) In summary, the algorithm proceeds as follows:

(1) Given a gridded terrain database, categorize each grid cell as Go (passable) or No-Go (impassable) based on grid cell attributes in the database.
(2) Aggregate the No-Go grid cells into No-Go regions, considering a parametric minimum separation distance. The boundaries of the No-Go regions are given as points.
(3) Compute the Delaunay triangulation of the plane based on the No-Go regions' boundary points. Triangles inside the No-Go regions are discarded.
(4) Create a graph based on the triangulation, with the circum-centers of the remaining triangles as vertices and edges connecting adjacent triangles. The edges are weighted for length and other factors of interest to route planning based on the triangles.
(5) Simplify the graph by discarding vertices with exactly two edges and replacing the incident edges with a single edge connecting the two neighboring vertices.
(6) Find unit mobility corridors by searching the graph using an A* algorithm.
(7) Combine unit mobility corridors into unit avenues of approach using a simple distance metric.

SIMNET SAF. The SIMNET SAF uses the A* algorithm to search road nets stored in a quadtree to find unit routes [Stanzione,1989]; this approach is also used in the ODIN SAF [Stanzione,1993]. The references indicate that both the SIMNET SAF and the ODIN SAF needed an avenue of approach generation capability; their developers were apparently unaware of the avenue of approach algorithm, described earlier, given in [Powell,1988a], [Powell,1988b], [Powell,1989], and [Wright,1990].

Stealth terrain navigation. The entity route planning algorithm presented in [Teng,1992] and summarized earlier can be adapted to plan routes for small units such as platoons that move in formation. A desired characteristic for a platoon route is that the entities of the platoon be able to move in formation and maintain intervisibility with each other. Recall that the entity route planning algorithm evaluated and extended partial routes based on the exposure to enemy

observation of the terrain grid cells. For the unit route planning version, formation mutual intervisibility is also considered. As each grid cell is evaluated, the mutual intervisibility of a platoon formation centered at that grid cell is calculated. A weighted sum of unexposed time and formation mutual intervisibility is used to evaluate a grid cell for inclusion in the route. The route found by this method is a sequence of grid cells, corresponding to the center of the formation; from it routes for the individual entities of the formation are found by simple formation offsets from the center's route.

Mesh. A unit route planner detailed in [Cunningham,1993] also uses the A* algorithm, but the routes are not confined to road nets. The A* algorithm searches a specialized polygonal terrain database format known as a mesh (described earlier; in a mesh, all of the polygons are convex) minimizing a cost function that considers factors important to movement, such as exposure to enemy observation, distance travelled, or penetration of excluded areas. The vertices of the mesh polygons are the vertices of the search, though movement is not constrained to be along the edges of the mesh. Additional vertices may be interpolated as needed along the existing edges and edges may be interpolated from such an interpolated vertex to a mesh vertex, thereby allowing the movement route to traverse the interiors of the mesh polygons. The movement cost is assumed to remain constant within a mesh polygon. Once a unit's route is found by searching the mesh with A*, the entities of the unit follow the route by moving in formation.

Obstacle segment abstraction. [Rajput,1994b] and [Karr,1995d] describe a route planner for units that uses a hybrid terrain representation that combines elements of both gridded and graph representations. The grid generation process is a sophisticated one; the grid cells are assigned abstract terrain types that represent topological connectivity and traversibility relationships of the terrain in the grid cell that affect unit route planning. The method proceeds as follows:

(1) Overlay (conceptually) a regular square grid on the polygonal terrain. The grid extent is determined by the unit's boundaries, and oriented to include the route's starting and destination points. The grid cell size is a function of the unit's size; 125m for platoons, 500m for companies, and 1000m for battalions.

(2) Analyze the terrain underlying each grid cell. Terrain features that constitute obstacles to movement (rivers, treelines, canopies) are abstractly encoded by assigning zero or more obstacle segments to the grid cell. Figure 4.4 (a) shows the different obstacle segment types, (b) is a notional terrain example, and (c) gives the resulting obstacle segment abstraction grid.

(3) Assign either 8 or 12 sample points to each grid cell (8 if the grid cell has no tunnel obstacle segment, 12 if it does). The sample points become vertices of a graph representing movement routes in the terrain; vertices on opposite sides of an obstacle segment are not connected by an edge in the graph. Figure 4.4 (d) illustrates the placement of the sample points in the grid cell.

(4) Search the resulting graph with the A* algorithm to plan a unit route. The route cost of each edge in the graph considers distance, trafficability, and concealment. Distance is calculated as the Euclidean length of the edge. Trafficability is found as a function of the polygon type and average slope of a 5x5 array of points around the edge's end vertex. Concealment is the percentage of a circular area centered on the edge's end vertex that is observable from known enemy locations.

The obstacle segment abstraction unit route planner is both flexible and efficient. It's flexibility is due to the scalability. The size of the terrain grid cells can be changed to make the algorithm more or less sensitive to smaller obstacles; hence it can be used to plan routes for units of any size or even entities. It is efficient in that the obstacle abstraction scheme captures the key topological aspects of the terrain without retaining and processing unnecessary detail. Finally, the route planner can plan multiple distinct routes in a given terrain area by increasing the cost of sample points that have been used in previous routes.

ModSAF concealed routes. [Longtin,1995] presents ModSAF's algorithm to plan unit routes that take advantage of regions that are concealed from enemy observation. The algorithm will be covered in detail later in the subsection on cover and concealment.

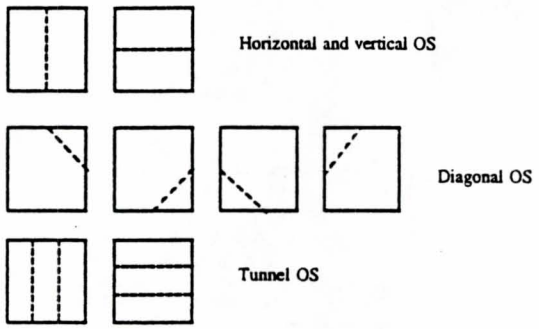
4.3.6 Reconnaissance route planning

The third and final variation of route planning to be surveyed is reconnaissance route planning. The nature of the reconnaissance route planning task differs somewhat from entity and unit route planning, where starting and ending points are given and the route planner generates a route between them. Instead, a reconnaissance route planner is given a starting point and an area of terrain, and it must produce a militarily effective reconnaissance route for that terrain area. A militarily effective reconnaissance route is one that enables a reconnaissance vehicle following it to observe (i.e. have an unobstructed line of sight to), at some point on its route, as much of the terrain area as possible, especially points that are likely to be locations of hostile entities.

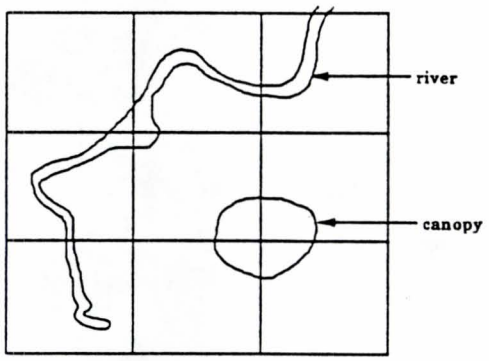
This subsubsection will present two reconnaissance route planning algorithms. They are:

1. Objects and rules
2. All-Points

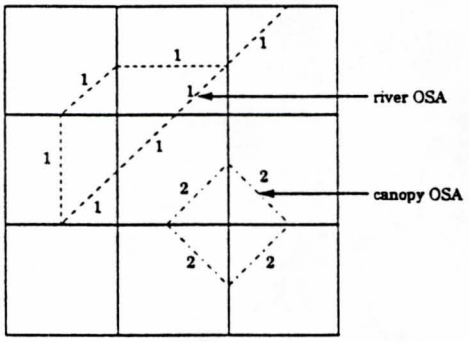
Objects and rules. [Gonzalez,1991] reports on an attempt to perform reconnaissance route planning using expert systems rules that operate on terrain objects. A terrain object is an abstract element or characteristic of the terrain, such as a "hill". Terrain objects are easily identified by human terrain analysts, but are not explicitly represented in most terrain representations; e.g. in a polygonal terrain representation, a hill is a set of contiguous polygons whose vertices' z coordinates are greater than those of the polygons around the set. The approach encountered two difficulties. First, it was much harder than expected to create a set of rules for reconnaissance route planning, even given the ability to operate on objects instead of representational details such as z coordinates, and second, it was extremely difficult to find meaningful terrain objects within the underlying polygonal terrain. The latter problem, abstracting terrain objects from underlying representations, is discussed in more detail later.



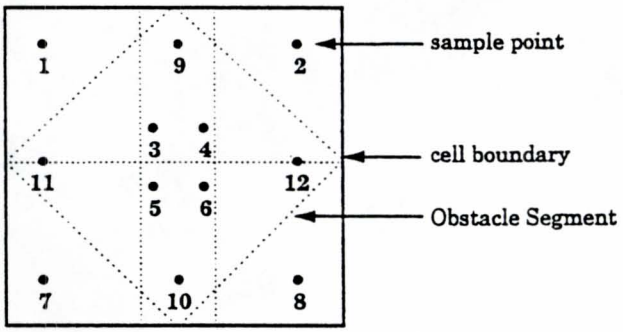
(a) Obstacle segment types



(b) Notional terrain



(c) Obstacle segment obstruction grid



(d) Sample points

Figure 4.4 Obstacle segment abstraction [Karr,1995d].

All-Points. In contrast, a geometric and algorithmic approach to the reconnaissance route planning task was quite successful. In this effort, reported on in [Van Brackle,1993a], [Van Brackle,1993b], and [Petty,1994b], three different reconnaissance route planning algorithms were developed. Given an area of terrain, represented in a polygonal format (to be specific, the SIMNET format), the algorithms produced a reconnaissance route, returned as a sequence of waypoints for a reconnaissance vehicle to move through. The best of the three algorithms, known as "All-Points", operates as follows:

(1) Identify a set of *Important Points* for the given terrain area. Important points are placed at polygon vertices, at treeline endpoints and concavity changes, and centered on each side of features such as buildings. The intent behind identifying Important Points is that if a reconnaissance vehicle has observed all of the Important Points, it will have observed essentially the entire terrain area.

(2) Identify a subset of the Important Points from which all of the Important Points can be observed; that subset is referred to as the Route Points.

  (2.1) Construct a line of sight graph on the Important Points. Treat the Important Points as vertices and place an edge in the graph between each pair of Important Points if and only if an unobstructed line of sight exists between them.

  (2.2) Find the vertex in the line of sight graph with the highest degree (i.e. the Important Point that can see the most other Important Points); call it P. Add P to the Route Point set.

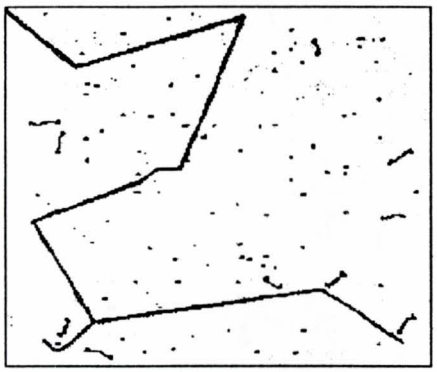
  (2.3) Given vertex P found in step (2.2), for each vertex Q such that (P,Q) is an edge in the line of sight graph and for each vertex R such that (Q,R) is an edge in the line of sight graph, remove the edge (Q,R) from the line of sight graph. The idea is that because P has been added to the Route Point set, and the Important Points Q have been seen from P, the algorithm no longer needs to be concerned with seeing those points from other points (the Important Points R). Note that R may equal P.

  (2.4) If any edges remain in the line of sight graph, go to step (2.2).

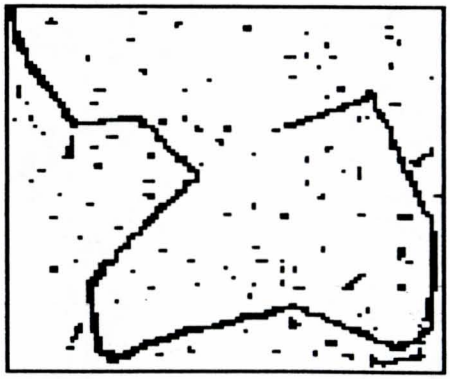
(3) Determine an efficient order for visiting the Route Point set. This is done by treating the points in the Route Point set as the vertices of a complete graph, with the edges weighted according to the Euclidean distances between them, and applying a Traveling Salesperson approximation algorithm (NEARINSERT/ARBINSERT in [Rosenkrantz,1974]).

(4) Return the sequenced Route Point set as a series of waypoints.

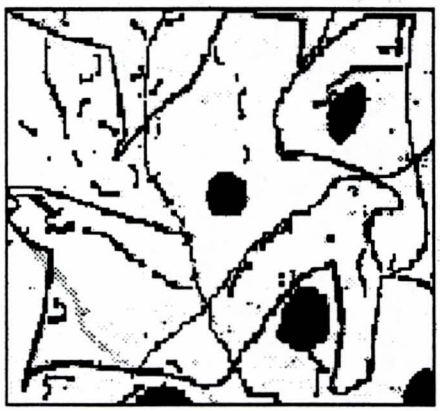
Reconnaissance routes produced by the All-Points algorithm for three test terrain areas were compared to routes planned by human SMEs, who were military officers trained in terrain analysis, on the same terrain areas (see Figure 4.5). The comparison used both measured and statistical comparison techniques, as described earlier in the document in the review of CGF VV&A techniques. The All-Points algorithm performed at a level comparable with the human SMEs.



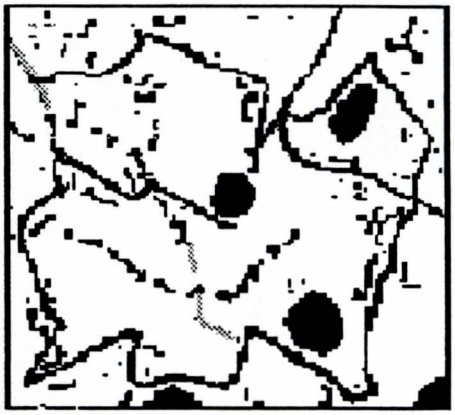
(a) SME1's route on the Clear terrain



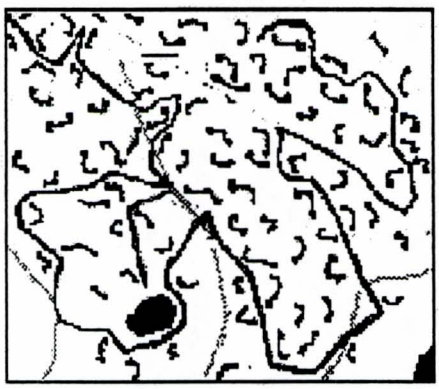
(b) All-Points' route on the Clear terrain



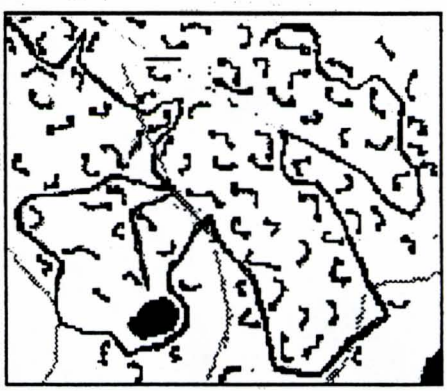
(c) SME1's route on the Mixed terrain



(d) All-Points' route on the Mixed terrain



(e) SME1's route on the Rough terrain



(f) All-Points' route on the Rough terrain

Figure 4.5 Reconnaissance route planning (adapted from [Petty, 1994]).

4.3.7 Dynamic obstacle avoidance

Before leaving the topic of route planning, a closely related subject should be mentioned. Obstacles are impediments to movement, so effective route planning algorithms must avoid obstacles. Some obstacles are motionless, or static; examples include unfordable rivers, treelines, and impassable slopes. Other obstacles can be moving, or dynamic; the most common examples are other moving entities in the simulation. Presumably a moving tank should not collide with another moving tank.

Moving obstacles are referred to as dynamic. *Dynamic obstacle avoidance* is an important aspect of route planning, though it is not an aspect of terrain reasoning (terrain obstacles rarely move). Sometimes a moving entity's route around static obstacles (such as individual trees and buildings) is planned in advance, while moving obstacles are avoided by continuously considering their positions during the movement; e.g. the CCTT SAF [Campbell,1995]. ModSAF, on the other hand, uses an obstacle representation that serves for both static and dynamic obstacles [Smith,1994].

[Roos,1991] describes an algorithmic method, based on dynamic Voronoi diagrams, that could be used for dynamic obstacle avoidance. [Craft,1994b], [Karr,1995a], and [Karr,1995c] analyze an effective CGF dynamic obstacle avoidance algorithm in detail.

4.4 Intervisibility determination

4.4.1 Definition

Intervisibility, in its simplest form, is the task of determining if one entity can see another in the simulated environment. Intervisibility also refers to related problems, such as determining what portion of a region can be seen from a specific location, or what is the cumulative mutual visibility between two regions. Intervisibility determination is of fundamental importance to behavior generation in a CGF system because many important CGF actions and responses are conditional on sighting a hostile entity (e.g. direct fire). Intervisibility is a terrain reasoning problem because it is the terrain that most often obstructs visibility.

Though simple in concept, intervisibility is problematic in implementation simply because of the demands it places on CGF systems' processing capacity. Each intervisibility determination can be quite computationally expensive, and in a simulation exercise with n entities, $O(n^2)$ intervisibility determinations may be required each second [Petty,1992a] [Petty,1992b], producing a serious load on CGF system performance [Sansom,1993] [Kada,1994]. In a CGF system, intervisibility determination is almost always the single most computationally expensive operation, potentially consuming so much of the system's computational resources that CGF entity behavior generation is negatively effected [Rajput,1995a] [Rajput,1995b].

Sophisticated algorithms and terrain representation data structures have been developed to reduce the computational cost of intervisibility. This subsection will survey several of those algorithms. Point-to-point, point-to-region, and region-to-region intervisibility algorithms will be mentioned; each will be defined in turn.

4.4.2 Point-to-point intervisibility

Point-to-point intervisibility is the most basic of the intervisibility operations. Informally, it is the process of determining if the line of sight from one entity to another is obstructed by intervening terrain. More formally, point-to-point intervisibility is the process of determining if a line segment connecting two 3D points in the simulated environment intersects any portion of the terrain database.

Several terms, including a few used in the preceding discussion, should be defined.

Sighter. The entity on whose behalf an intervisibility determination is initiated; the determination is performed to establish if the sighter has intervisibility to a target.

Target. The entity whose intervisibility status, relative to the sighter, is in question.

Line of sight (LOS). A line segment in the simulated environment connecting two given 3D points. The two points are defined as relative to the sighter and the target. Typically, the sighter's LOS endpoint is defined at the point where the entity's commander's head would be, relative to the sighter's location, and the target's LOS endpoint is the center of mass or volume of the entity.

Obstructed. A LOS is obstructed if it intersects an opaque terrain element.

In this formulation of point-to-point intervisibility, several assumptions are made. Two will be identified here. First, it is assumed that only terrain may obstruct the LOS. Of course, that is not in fact true; other entities, battlefield obscurants such as smoke [Bess,1991] and environmental characteristics such as fog may also obstruct the LOS. Many intervisibility algorithms neglect those factors, and this document will do so also, simply because they are not components of terrain representation or reasoning. Second, it is almost universally assumed in point-to-point intervisibility algorithms that a LOS that passes under any part of the terrain is obstructed; while that assumption does not hold true for terrain multi-level terrain features (such as bridges), the assumption is true often enough to make its use nearly ubiquitous.

Numerous variations of point-to-point intervisibility algorithms exist, dependent on both terrain representation and simulation requirements. Several will be described:

1. SIMNET SAF
2. CGF Testbed
3. Algorithms C and P
4. Stealth terrain navigation
5. NASA Ames LOS Attachment
6. Compact Terrain Database

7. CCTT SAF
8. Integrated Computer Generated Forces Terrain Database
9. Intelligent Player

SIMNET SAF. In the SIMNET SAF, intervisibility is probabilistic; some terrain objects, such as treelines, do not automatically obstruct a LOS that traverses them, instead reducing the probability of a LOS existing [Stanzione,1989]. The result of an intervisibility determination is probabilistically chosen from a discrete set of possibilities: visible, partially visible, not visible.

CGF Testbed. The CGF Testbed uses a polygonal terrain representation that is nearly identical to the SIMNET format terrain database, which was described earlier. Both the terrain surface and features (treelines, canopies, and buildings) are constructed from polygons. The polygons are organized spatially and stored in patches, which are square areas in the *x,y* plane of the terrain database, and grid cells, which are subsquares of the patches. Each patch has arrays that store the surface and feature polygons that it contains. Bit maps associated with the polygon array entries indicate which grid cells the polygons overlap.

The CGF Testbed's intervisibility algorithm is described in some detail in [Petty,1992a], [Petty,1992b], and [Smith,1992b]. It operates in two basic steps: point location and traversal. Point location determines the patches and grids containing the LOS endpoints. Because the patches and grid cells are of known fixed size, point location can be performed with simple arithmetic operations on the endpoints' coordinates.

In the traversal step it is necessary to determine which patches and grid cells to search for possible intersections with the LOS. In this algorithm, the patch and grid cells through which the LOS passes (in 2D, projected into the *x,y* plane) are found using a modification of Bresenham's algorithm [Foley,1982], moving from the sighter's grid cell to the target's grid cell. Within each grid cell on the LOS the surface and feature polygons are checked for intersection with the LOS.

Each of the surface polygons that overlaps the grid cells through which the LOS passes is checked to determine if it obstructs the LOS. This is accomplished by testing each of the polygon's edges within the current grid cell for intersection with the LOS. The line intersection is computed using *x,y* coordinates only. Then, if the polygon edge and the LOS are found to intersect in 2D, the *z* coordinates are calculated for the edge and the LOS at the intersection point. If the *z* value for the polygon edge is greater than the *z* value for the LOS, the LOS is taken to be obstructed.

The feature polygons in the patches and grid cells are also checked; recall that arrays containing the features are associated with each patch. After surface polygons have been checked, the treelines and canopies within the patch and grid cell are checked to see whether they obstruct the LOS. The treeline and canopy check proceeds in a manner very similar to the polygon edge check, allowing for the additional height of the features.

Algorithms C and P. [Petty,1992a] and [Petty,1992b] describe research into point-to-point intervisibility algorithms; that research had the goal of producing more time efficient algorithms for point-to-point intervisibility determination. Four point-to-point intervisibility algorithms were

compared experimentally: the CGF Testbed algorithm just described, the algorithm used in the SIMNET Plan View Display (PVD) node, and two new algorithms developed at IST dubbed Algorithms C and P. Both of the new algorithms were faster than both the CGF Testbed's algorithm and the SIMNET PVD algorithm.

Algorithm C stores the terrain polygons in two data structures: a doubly-connected-edge-list (DCEL) and a slab list. Both the DCEL and the slab list are well known data structures commonly used in computational geometry and are defined in [Preparata,1988]. The process of creating the DCEL and the slab list data structures from terrain polygons is detailed in [Petty,1992a]. Given the endpoints of the LOS (i.e. the sighter's location and the target's location), Algorithm C performs point location using the standard slab method operating on the slab list data structure to identify the polygons containing the LOS endpoints. The slab method used is given in [Preparata,1988].

Once the polygons containing the LOS endpoints have been identified, Algorithm C's LOS traversal follows the LOS in 2D from polygon to polygon, starting with the sighter's endpoint. If the LOS has been determined to pass through a given polygon, that polygon's edges are tested for intersection with the LOS to determine through which edge the LOS leaves the polygon. The z coordinates of the LOS and the exit edge at the point of intersection are compared to determine if the LOS is obstructed by that polygon; if the z coordinate of the exit edge is greater than that of the LOS at the intersection point, the LOS is determined to be obstructed. If not, the polygon connectivity information in the DCEL is used to identify the polygon that is adjacent through the exit edge and is thereby the next polygon the LOS passes through. The traversal ends when the polygon containing the LOS endpoint (found in the point location step) is reached.

Algorithm P triangulates the terrain polygons and creates a data structure that is a list of triangles. Each triangle's entry in the list stores its vertices and the adjacent triangles. The triangles containing the LOS endpoints are found using a variation of the slab method described in [Petty,1992a] and [Petty,1992b]. Algorithm P traverses the LOS from triangle to triangle beginning with the sighter's endpoint. At each triangle on the LOS the edge through which the LOS entered the triangle is known. The next triangle along the LOS is determined by computing the 2D parametric equation of the LOS with the x,y coordinates of the triangle's vertex that is not on the edge through which the LOS entered the triangle. As in Algorithm C, the z coordinates of the LOS and the exit edge are intersected to determine if the triangle obstructs the LOS. The traversal ends when the triangle containing the LOS endpoint is reached.

Empirical comparison, related in [Petty,1992a] and [Petty,1992b], determined that Algorithm C had the faster point location and Algorithm P the faster LOS traversal. Both of the algorithms were faster than the CGF Testbed's algorithm and the SIMNET PVD algorithm.

Stealth terrain navigation. [Teng,1992] describes a point-to-point intervisibility algorithm for gridded terrain and a parallel machine architecture. The LOS is converted to a set of terrain grid cells by projecting the LOS into the x,y plane of the terrain. Then the viewing angle from the sighter's grid cell to every other grid cell on the line is sight is computed based on the grid cells' elevation attributes. The LOS is obstructed if and only if the viewing angle from the sighter's grid

cell to the target's grid cell is less than the viewing angle from the sighter's grid cell to any other grid cell on the LOS. On a parallel machine, each grid cell's viewing angle is computed in parallel on a separate processors; the comparison is done with a scan operation.

NASA Ames LOS Attachment. [Sansom,1993] describes the intervisibility determination method used for the NASA Ames Research Center's Vertical Motion Simulator. In a preprocessing phase, the flight simulator's polygonal IG terrain database is converted to a gridded terrain representation with elevation posts only. During run-time, intervisibility determinations are performed using a Bresenham-like 2D digital differential analyzer traversal of the terrain grid along the LOS. The terrain elevation of the elevation posts found by the traversal is compared to the height of the LOS, which is computed by a 3D digital differential analyzer algorithm. If the terrain height is greater than the LOS height, the LOS is obstructed by the terrain. Note that the terrain height is taken to be that of the elevation posts; there is no interpolation between elevation posts when a LOS passes between the posts.

Compact Terrain Database. Intervisibility determinations in the ODIN SAF and ModSAF return a continuous value specifying the portion of the entity that is visible, i.e. that is not obstructed by intervening terrain. Determining the portion of an entity visible requires performing several point-to-point intervisibility determinations to different points on the target entity. Early versions of the algorithm would then decrease the effective visible portion of the target entity to account for cumulative light transmittance through intervening tree foliage [Stanzione,1993] [Smith,1995b]; more recently, the light transmittance model was separated from the portion visible calculations [Smith,1995b]. The reference also mentions the CTDB intervisibility algorithm's assumption regarding multi-level terrain; it assumes that all terrain elements other than the terrain surface are transparent.

CCTT SAF. In CCTT, an intervisibility determination also computes the portion, given as a percentage, of a target entity that is visible from a sighting point. Solid objects and features, such as terrain surface or buildings, obstruct intervisibility, while non-solid objects, such as trees, provide partial transmittance. The complicating effect of dynamic objects on intervisibility determination in CCTT is discussed in [Campbell,1994].

Integrated Compact Terrain Database. The intervisibility algorithm used in the ICTDB terrain database, as given in [Stanzione,1995], appears to be very much like Algorithm P from [Petty,1992a] and [Petty,1992b]. Recall that the ICTDB is a polygonal terrain database format where the terrain surface is defined by triangles, and each terrain surface triangle is linked to its three topologically adjacent neighbors with pointers. The ICTDB intervisibility algorithm uses Algorithm P's traversal method, following the LOS from triangle to triangle. The LOS enters each triangle through an edge of that triangle. Within each triangle it determines the next triangle on the LOS by inserting the coordinates of the third vertex (the vertex that is not an endpoint of the edge through which the LOS entered the triangle) into the equation of the LOS. Once the next triangle is determined the pointer to that triangle is followed.

Intelligent Player. [Pandari,1995] mentions intervisibility determination using a quadtree terrain representation for terrain elevation.

4.4.3 Point-to-region and region-to-region intervisibility

Point-to-region intervisibility determines how much, or what parts, of a terrain region are visible from a given point. Region-to-region intervisibility measures cumulative mutual visibility between two given regions. Point-to-region and region-to-region intervisibility algorithms often operate by systematically applying point-to-point determinations.

The point-to-region and region-to-region algorithms to be presented are:

1. Stealth terrain navigation
2. Simulation based planner
3. Automated mission planner

Stealth terrain navigation. [Teng,1992] provides considerable detail on two intervisibility algorithms that provide point-to-region and region-to-region intervisibility results in gridded terrain. In the point-to-region case, the result returned is a visibility map specifying whether each grid cell in the region is visible from the sighter's grid cell. For the region-to-region case, the algorithm finds for each grid cell in the sighter region the number of visible grid cells in the target region. The algorithms are highly dependent on the gridded terrain representation and on the parallel machine architecture for which they are designed.

Simulation based planner. The automated mission planner described in [Lee,1994a], [Lee,1994b], and [Lee,1994c] uses a region-to-region intervisibility determination in its evaluation of the suitability of unit positions. The region-to-region intervisibility determination is performed on two circular regions. It calculates a probability of an entity in one region being seen from the other region by performing a set of point-to-point intervisibility determinations from points within the two regions.

Automated mission planner. [Karr,1995b] describes an automated mission planner implemented in ModSAF. It includes a region-to-region intervisibility function called "Area Line of Sight". It calculates a percentage of intervisibility between circular regions. Inputs to the function are the centers, radii, and number of sample points to check within each region. If the number of sample points in the two regions are n_1 and n_2, n_1 points within the first region and n_2 points within the second region are selected randomly. The function then performs point-to-point intervisibility determinations from each of the selected points in the first region to each of the selected points in the second region, a total of $n_1 \times n_2$ determinations. The percentage of those that are unobstructed is returned.

4.4.4 Other intervisibility algorithms

[Petty,1992a] and [Petty,1992b] are examples of research work to reduce the computational expense of each intervisibility determination in a CGF system. Work has also been undertaken on the complement of that problem, which is reducing the number of intervisibility determinations made by a CGF system. The essential idea is to use heuristics that suggest when intervisibility determinations can be delayed or skipped in a CGF system without unduly affecting the generated behavior of the generated CGF entities. Some surprisingly effective heuristics have been

developed that reduce intervisibility determinations by up to 50% while delaying first sighting times by less than a second. See [Rajput,1994a], [Rajput,1995a], and [Rajput,1995b] for details.

4.5 Finding cover and concealment

4.5.1 Definition

Cover refers to terrain that protects an entity (or unit) from direct fire; concealment provides protection from observation. Note that by definition, cover and concealment is always relative to an enemy direction (or set or range of directions).

The process of finding and using cover and concealment is of paramount importance in military tactical terrain reasoning. A U.S. Marine Corps field manual [Schmitt,1988] states:

"In the offense, use of cover and concealment allow the attacker to close with the enemy with fewer losses. In the defense, cover and concealment protect the force against enemy preparations and fires in support of the attack and help to deceive the enemy as to the location of the main defensive positions. In both cases, cover and concealment facilitate surprise."

As it is in real-world military tactics, so to for CGF. Realistic and convincing CGF behavior requires effective use of cover and concealment. [Longtin,1994] states that "The need to find covered or concealed locations with respect to enemy locations arises frequently in CGF systems, since there are many real-life tactics which require this ability, such as occupying a battle position or performing a bounding overwatch maneuver." However, doing so is a difficult problem in CGF terrain reasoning, and some CGF systems have not been able to use cover and concealment effectively. [Stanzione,1989] admits that the SIMNET SAF does not use cover and concealment while moving. [Vaden,1994] and [Mengel,1994] mention the failure of an early version of ModSAF to use cover and concealment, except when individual entity positions were selected by the CGF system operator. [Meliza,1995] criticizes a more recent version of ModSAF for not routing entities under its control to cover or concealment when they come under fire. Considerable work remains to be done in this area.

When defining a particular cover and concealment problem or task, there are several important aspects to be specified. Changing any of these aspects produces a different variation of the general cover and concealment problem. The aspects are:

1. Protection desired; cover, concealment, both.
2. Degree of protection; partial, defilade, complete.
3. Relative to; specific enemy location(s), direction (direction of travel, position facing), range of directions.
4. Enemy location(s); stationary, moving projected to a single location, moving projected to a region.
5. Subject; entity, unit.
6. Subject movement; stationary (at a location), moving (along a route)

In this subsection three variations of the cover and concealment problem, and algorithms that address them, will be examined. They are:

1. Finding cover and concealment for one or more stationary entities with respect to a single stationary enemy location.
2. Planning an entity or a unit's route to utilize cover and concealment with respect to multiple stationary enemy locations.
3. Executing bounding overwatch, a combination of the first two.

4.5.2 Stationary cover and concealment

ModSAF includes algorithms to find cover and concealment locations for stationary entities. The algorithms use terrain represented in ModSAF's CTDB terrain database format. Those algorithms are reported in [Longtin,1994] and will be presented in some detail here. The cover-finding algorithm will be described first, followed by the concealment-finding algorithm.

The cover-finding algorithm attempts to find hull-defilade locations for entities. A hull-defilade location is defined as a location such that a vehicle at that location has its hull protected from direct fire by the terrain surface while its turret is exposed. Note that this definition is implicitly relative to a specific direction; a ridge that provides cover for a location from one direction is likely irrelevant from the opposite direction. In fact, the ModSAF cover-finding algorithm finds cover relative to a specific enemy location.

Along with the enemy location, a number of other parameters are input to the cover-finding algorithm. The complete list is:

1. enemy location
2. search area
3. grid spacing
4. hull height of cover-seeking vehicle
5. main gun orientation limits
6. tree opacity
7. minimum allowed visibility
8. hull coverage
9. search state data structure
10. cover locations found array
11. main gun elevation angles array

The role of each of these parameters will be made clear as the algorithm is described.

The algorithm proceeds as follows:

(1) Define *sample points* along the left, back, and right sides of the search area. The search area, given as input to the algorithm, must be a rectangle. The side of the search area closest to the given enemy location is designated as the front. The distance between consecutive sample points is given by the grid spacing parameter.

(2) Define a set of 2D line segments from the enemy location to each of the sample points. These segments are called *profiles*. Figure 4.6 shows an example enemy location, search area, sample points, and profiles.

(3) Construct *profile arrays* for each of the profiles. A profile array is a set of three dimensional points that describes the profile, or shape, of the terrain surface along a profile. Each point in the profile array is a point at which the slope of the terrain surface changes along the profile. Each successive pair of points defines a 3D segment; taken together, those segments describe the 3D shape of the terrain surface along the profile, or line segment from the enemy location to the profile's sample point. Figure 4.7 (a) shows a profile with the connecting segments added.

(4) For each segment of each profile, examine that segment (the *current segment*) for a possible covered location on that segment.

(4.1) Construct a vector from the enemy location to the endpoint of the current segment closer to the enemy location. Compare the slope of that vector with the maximum slope of the vectors similarly constructed for previous segments along the profile, and save the greater as the maximum; the maximum is the current segment's *tangent line*.

(4.2) Construct two line segments, one extending from each endpoint of the current segment perpendicular in the upwards (increasing z) direction with a length of the cover-seeking vehicle hull height. The line segment extending from the endpoint closer to the enemy location is *testline 1*, and the other is *testline 2*.

(4.3) If the tangent line intersects testline 1 but not testline 2, then a candidate cover location exists. Construct a line segment by connecting the upper ends of the testlines. Note that this new line segment, called the *vehicle height segment*, is parallel to the current segment and separated from it by a distance equal to the cover-seeking vehicle hull height. Intersect the tangent line with the vehicle height segment. Project that intersection point to the current segment to find the candidate cover location. Figure 4.7 (b) includes the tangent line, testlines, vehicle height segments, and candidate cover locations for an example profile.

(5) If a candidate cover location is found, test it against the following additional constraints:

(5.1) Is the candidate cover location within the search area? Because part (or all) of the profile may lie outside the search area, the candidate location is tested for inclusion in the rectangular search area.

(5.2) Does the slope of the terrain surface polygon at the candidate cover location allow a vehicle located there to aim its main gun at the given enemy location, given the main gun orientation limits provided as input?

(5.3) Is the enemy location visible from the candidate cover location? As mentioned earlier, the ModSAF CTDB point-to-point intervisibility routines returns a value indicating the portion of the target entity that is visible. Though the reference does not make this clear, it seems reasonable that the minimum allowed visibility parameter is applied to an enemy entity at the enemy location; the enemy location is considered to be visible if and only if the portion of an entity at that location exceeds the parameter. The tree opacity parameter is used in the point-to-point intervisibility calculations to model the effects of trees, which are neither completely transparent nor completely opaque.

If the candidate cover location passes all of these tests, it is added to the cover locations found array. The main gun elevation required to aim the main gun at the enemy location from the cover location is also stored in the corresponding entry in the main gun elevation angles array.

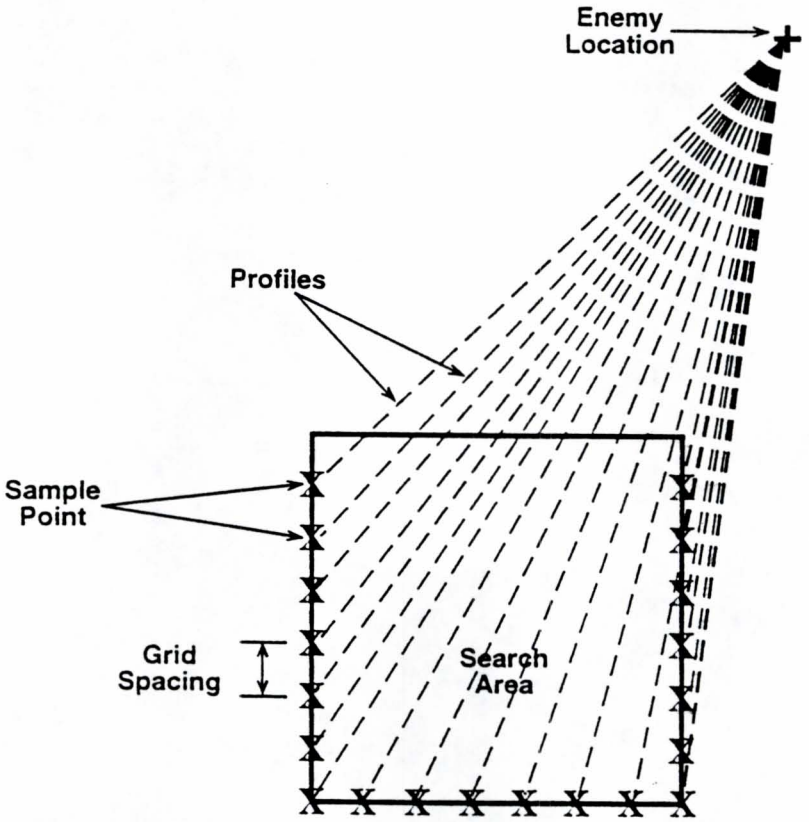
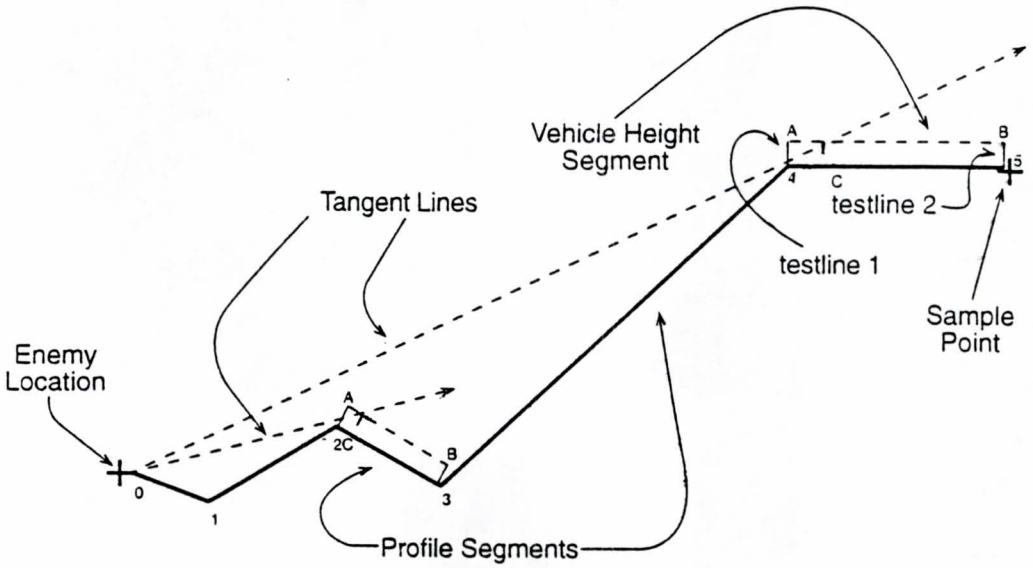
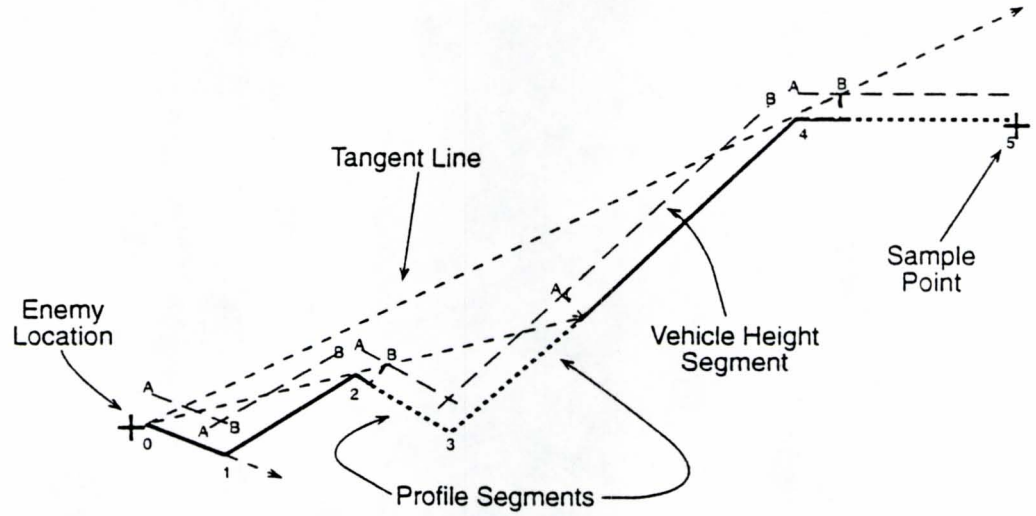


Figure 4.6 ModSAF cover and concealment search area with profile array segments [Longtin, 1994].



(a) Profile for cover



(b) Profile for concealment

Figure 4.7 Example ModSAF profiles for cover and concealment [Longtin, 1994].

The ModSAF concealment-finding algorithm attempts to find locations that are concealed by terrain features (treelines and buildings). It is very similar to the cover-finding algorithm. As with cover, concealment is found relative to a given enemy location. The concealment-finding algorithm differs from the cover-finding algorithm given earlier only in step (4); that portion of the concealment-finding algorithm is given here.

(4) For each segment of each profile, examine that segment (the *current segment*) for a possible concealed location on that segment.
   (4.1) Construct a vector from the enemy location to the endpoint of the current segment closer to the enemy location. Compare the slope of that vector with the maximum slope of the vectors similarly constructed for previous segments along the profile, and save the greater as the maximum; that maximum is the *tangent line* of the current segment.
   (4.2) Construct two line segments, each extending from the current segment perpendicular in the upwards (increasing z) direction with a length of the concealment-seeking vehicle hull height. One line segment, *testline 1*, extends from the current segment's endpoint closer to the enemy location. The other, *testline 2*, extends from the point on the current segment which is a perpendicular projection of the point at which the distance between the tangent line and the current segment becomes greater than the vehicle height. Construct the *vehicle height segment* by connecting the upper endpoints of testline 1 and testline 2.
   (4.3) Test the terrain features in the patch for intersection with vehicle height segment. If the vehicle height segment intersects a treeline or building, calculate a candidate concealment location just behind (relative to the given enemy location) the feature.

The concealment-finding algorithm is otherwise identical to the cover-finding algorithm, including the application of the three additional constraints (search area inclusion, main gun elevation limits, and enemy location intervisibility).

Note that cover, because it protects from both observation and direct fire, is normally to be preferred to concealment, which does not protect from direct fire. When ModSAF attempts to find locations for the entities of a unit, it will first search for covered locations.

Because the time required to completely search the search area for cover and concealment locations could easily exceed the time available for a single entity's execution cycle ("tick") in ModSAF, the cover-finding algorithm is designed to partially execute and then save its state, performing a complete search over the course of several calls. The search state data structure contains the data needed for that capability.

A few evaluative comments regarding these two ModSAF algorithms can be made. Both algorithms will miss cover or concealment locations that happen to fall between the profile array lines. This will occur unless that parameter is set to a fairly small value (such as a typical vehicle's length). Thus the algorithms' effectiveness can be highly dependent on the grid spacing parameter. Of course, a small grid spacing parameter will result in a longer search time. The user

has some control over the fidelity and computation expense of the cover-and-concealment-finding algorithms by adjusting the parameter.

Though the algorithms seem to be focused on vehicles, with their use of hull height, they are actually easily adaptable to dismounted infantry by treating the shoulder height of a standing soldier as the hull height, and the soldier's head as the turret.

Both the cover-and-concealment-finding algorithms operate based on a single enemy location. [Longtin,1994] acknowledges this as a problem, especially for finding cover.

It is not clear why the search area test is not applied until after the considerable computational expense of finding cover or concealment locations along the profiles has been incurred, instead of simply terminating the search along a profile at the point it leaves the search area.

4.5.3 Cover and concealment during movement

Planning a route and executing movement to take advantage of cover and concealment is an important part of CGF terrain reasoning; [Longtin,1995] asserts that "A concealed-route algorithm is a vital component of CGF systems since it is a major contributor toward the realism of the generated forces."

Using cover and concealment during movement can be more difficult than finding covered or concealed locations for stationary vehicles. However, there have been theoretical and practical CGF algorithms developed that use, in some form, cover and concealment during movement. Several will be described here. They are:

1. Theoretical maximum concealment
2. ODIN SAF cover during movement
3. ODIN SAF cover under fire
4. ModSAF concealed routes
5. LeatherNet concealed routes

Theoretical maximum concealment. [Mitchell,1988] describes an algorithm, as a special case of the weighted regions shortest path problem, for finding a route that provides maximum concealment. However, the algorithm operates on a simplified terrain representation that is not found in any existing CGF system.

ODIN SAF cover during movement. [Stanzione,1993] describes, without providing a detailed algorithm, a procedure for use of cover in movement in the ODIN SAF system. Infantry Fighting Vehicles (IFVs), while moving, can detect a sudden increase in its area of observation. This would typically occur when the IFV is cresting a hill. (Note that in order to detect the change the IFV must be performing intervisibility calculations to determine its area of observation on a periodic basis. Presumably the frequency at which the area of observation is determined is related to the IFV's speed of movement, though the reference does not say so.) When the area of observation increases suddenly the IFV will stop. By keeping track of slope of the terrain in front of the IFV, the algorithm can stop the IFV just short of the crest so that only the IFV's turret is

exposed above the crest. From this defilade location the IFV can scan the surrounding area for enemy entities before moving further.

ODIN SAF cover under fire. [Stanzione,1993] also discusses the use of cover in the ODIN SAF system as a behavioral response to enemy fire. According to the reference, "SAF ground vehicles and DI will move away from dropping artillery and look for trees to use for cover." The quadtree terrain database is consulted to find the closest tree to move towards. As described, this behavior does not make sense. First, a tree provides little or no protection from artillery damage. Second, trees are concealment (protection from observation), not cover. Thus, it is not clear why an entity should seek the concealment that a tree offers in response to an artillery barrage, which is usually indirect (i.e. unobserved) fire.

ModSAF concealed routes. [Longtin,1995] provides a focused description of a procedure used in ModSAF to plan a route from a given start point to a given goal location that uses concealment. The procedure operates on ModSAF's CTDB terrain database format. It takes as input the start and goal locations and a set of *enemy descriptors*. Enemy descriptors may be of three types:

1. Enemy direction; 2D vector, giving the general direction of enemy entities.
2. Enemy location; specific location of a known or suspected enemy entity.
3. Enemy area; a 2D *x,y* polygon enclosing a terrain area expected to contain enemy entities.

The input set of enemy descriptors may contain zero or more of each type. Note that the flexibility provided by the different types and arbitrary number of enemy descriptors allows the concealed route procedure to operate under circumstances of varying degrees of specificity with regard to information about enemy locations.

The concealed route plan is divided into two main phases. First a map of concealed areas is found, and then an optimally concealed route is planned using that map. Given the start and goal locations, the procedure operates as follows:

(1) Construct a concealed area map.
  (1.1) Define the search area, a 2D rectangle enclosing the start and goal locations in the *x,y* plane, and a grid of locations within that rectangle; this data structure is the *cumulative concealment map*. The spacing of the grid locations is a parameter to the procedure. The grid locations correspond to locations in the terrain. Associated with each grid location is an exposure attribute that indicates whether it is exposed to enemy observation, with an exposure attribute value of one signifying concealed and a value of zero signifying exposed. All of the exposure attributes for the cumulative concealment map are initially set to one.
  (1.2) For each enemy descriptor, calculate the concealed area relative to that descriptor and combine it with the concealment map.
    (1.2.1) Create a working concealment map for the enemy descriptor, setting all of the working concealment map's exposure attributes to zero.
    (1.2.2) Determine which of the grid locations are concealed relative to the enemy descriptor and set those exposure attributes to one. The manner in which this is done depends on the enemy descriptor's type.
    For enemy directions: Set the exposure attributes of grid locations

"behind" (relative to the direction) treeline, tree, and building features, for a fixed distance, to one. Elevation is not considered, because there is no elevation associated with the enemy direction descriptor.
For enemy locations: Perform an intervisibility determination from the enemy location to each grid location. If the line of sight is obstructed, set the exposure attribute to one.
For enemy areas: Compute the principal axis of the search area, and define a series of line segments within the enemy area, perpendicular (in the *x,y* plane) to that axis, and separated by a distance dependent on the grid spacing. Project each segment on the terrain surface, and find the point on that projection with the highest elevation. Find the concealed locations for each of those highest points as if it was an enemy location.

(1.2.3) Intersect the working concealment map with the cumulative concealment map by performing a logical AND of the corresponding grid location's exposure attributes.

(1.3) Polygonalize the cumulative concealment map by computing polygons that enclose each disjoint cluster of adjacent grid locations with exposure attributes of one. Provide the set of concealment polygons to the second phase of the procedure.

(2) Plan a route that utilizes the concealed areas (see Figure 4.8).

(2.1) Construct a graph from the polygonal concealment map. The vertices of the graph will correspond to *x,y* locations in the terrain and the edges to straight segments between those locations. The edges will be assigned weights representing their length exposed to enemy observation.

(2.1.1) Convert the concealment polygons' vertices to a coordinate system with the *x* axis aligned with a line passing through the start and goal locations.

(2.1.2) Add the start and goal locations to the graph's vertex set.

(2.1.3) For each concealment polygon, add two vertices to the graph corresponding to the polygon's vertices with the minimum and maximum *x* coordinates. Add an edge to the graph connecting these two vertices. Assign that edge a weight of zero.

(2.1.4) For each vertex in the graph's vertex set, add edges from that vertex to the three vertices closest to it that have greater *x* coordinates and that are not already connected to it. If there are fewer than three such vertices, than add only the allowed edges. Assign each added edge a weight corresponding to the 2D Euclidean distance along that edge.

(2.2) Search the graph using the A* algorithm, minimizing edge weights, i.e. exposed distance. Use the Euclidean distance from the start location to the goal location as a pruning criteria.

(2.3) For any edges of the optimum route that have weight zero but that are not entirely within a concealment polygon (this might happen for a non-convex concealment polygon), adjust the route by replacing that edge with a series of edges that remain within the concealment polygon.

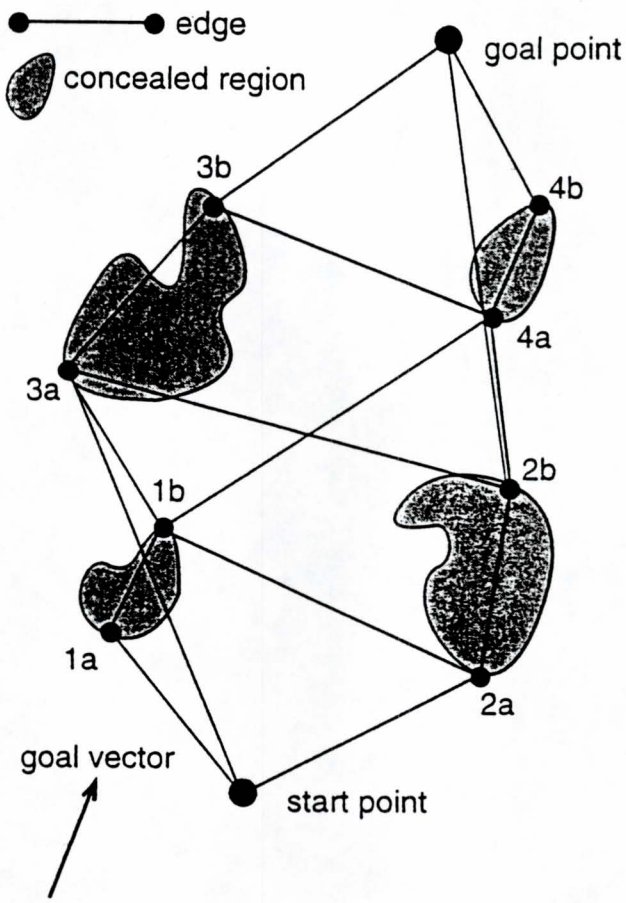


Figure 4.8 ModSAF concealed route planning graph (adapted from [Longtin, 1995]).

Just as in the cases of ModSAF's cover-and-concealment-finding algorithms described earlier, the concealed route planning algorithm may require more computational time than is available in a single ModSAF entity time-slice, or tick. The concealed route planner is designed to save its state and be resumed across multiple ticks.

[Longin,1995] identifies several ways the concealed route planner might be improved. The most important deficiency is that the procedure ignores obstacles to movement such as rivers and steep slopes. Such obstacles are considered only after the concealed route has been planned; when found, the planned route is modified to bypass the obstacles. That modification process may lead to sub-optimal routes because the extra exposed movement added for obstacle avoidance was not considered by the route planner.

LeatherNet concealed routes. LeatherNet is a version of ModSAF with specialized capabilities and behaviors for U.S. Marine Corps individual infantrymen [Howard,1995]. LeatherNet includes a route planning algorithm that takes advantage of concealment [Hoff,1995]. The algorithm applies the shortest-path paradigm (so denoted in [Mitchell,1988]) wherein regions of terrain are weighted with non-negative costs per unit of traversal, the total cost of a route is computed by summing the cost incurred in each section traversed, and the route planning algorithm minimizes the route's cost. The cost values can be computed as a function of a number of different terrain characteristics. For example, higher cost values may be associated with movement obstacles and areas exposed to enemy observation, and lower cost values with passable terrain and concealed areas. Note that the shortest-path paradigm as described and as used in LeatherNet is essentially 2D; terrain elevation is considered in route planning only indirectly, in that it contributes to determining concealed regions.

The shortest-path paradigm allows the terrain representation to be either "grid-based", where the terrain regions are regular (typically square) and of fixed size, or "weighted regions", where the overall terrain is partitioned into contiguous non-overlapping polygonal subdivisions of arbitrary size. The LeatherNet algorithm uses the weighted regions approach, primarily because the application's emphasis on individual combatants requires the ability to represent terrain features of small size relative to the overall terrain extent. Using a grid-based representation would require using a grid spacing as small as the smallest feature to be represented, which would result in a very large grid, expensive in both memory utilization and processing time. Using the weighted regions approach, small features of interest may be represented by small polygons, and large featureless regions may be represented by large polygons. In contrast, the ModSAF concealed route planner described earlier, which is primarily oriented towards units and vehicle-sized entities, determines concealed regions using a grid-based representation.

The algorithm is given start and goal locations and the CTDB terrain database. As is typical, a 2D rectangular search area enclosing the start and goal locations is defined. Then, from the CTDB a weighted regions terrain representation, called the *movemap*, is developed. First, CTDB terrain features that are obstacles to movement are added to the movemap by transferring their polygonal footprint; those regions are assigned a high cost. Second, concealed regions defined as polygons are added to the movemap; they are assigned a low cost. (The source of the concealed region polygons will be addressed later.) Finally, the area of the movemap not enclosed in either

obstacle or concealed region polygons is tessellated to produce a complete partition of the search area into convex polygons; the regions created in this step are assigned medium cost.

Once the movemap is complete, it is passed to the weighted regions route planning algorithm, which is a direct implementation of the "Continuous Dijkstra" algorithm explained in detail in [Mitchell,1991] and summarized in [Hoff,1995]; those descriptions will not be repeated here. The algorithm has worst case complexity of $O(n^8)$, where n is the number of vertices in the movemap, but often runs much faster than the worst case in practice.

The greatest shortcoming in the LeatherNet concealed routes algorithm is that the concealed regions must be identified manually, i.e. by the CGF operator. The concealed regions polygons used to build the movemap are created by a human operator and input to the movemap generation process via a file. Clearly it would be preferable for the concealed regions to be identified automatically. [Hoff,1995] suggests a possible method to do so, and lists several other ways the algorithm could be improved.

4.5.4 Bounding overwatch

Bounding overwatch is a military tactic developed by the U.S. Army for movement by a unit when enemy contact is expected. If executed properly, it combines both the stationary and moving and the entity and unit aspects of the use of cover and concealment.

When moving by bounding overwatch, a unit splits into two elements; the entities of an element operate together. Each element takes a role, either bounding or overwatch. The overwatch element remains motionless, watching the terrain ahead for enemy forces, and stands ready to protect the bounding element with fire if enemy forces are sighted. The bounding element moves forward as far as possible while still remaining within the area that can be protected by the overwatch element. The bounding element moves to a position from which it can survey the terrain in the direction the unit is to move. Once it reaches that position, the elements switch roles, with the former bounding element now performing overwatch and the former overwatch element bounding forward, moving past the overwatch element to the next position in the direction of advance.

Controlling the cooperative behavior of bounding overwatch is a behavioral control problem; see [Rajput,1995b]. However, finding a good overwatch position is clearly a problem in terrain reasoning. Ideally, an overwatch position provides both cover and concealment from most directions and good observation in the direction of advance, and the movement of an element to an overwatch position is along a route that is covered or concealed.

Three bounding overwatch algorithms will be mentioned. They are:

1. Stealth terrain navigation
2. ODIN SAF
3. ModSAF

Stealth terrain navigation. The entity route planning, unit route planning, and intervisibility algorithms presented in [Teng,1992] and described earlier are combined to produce a bounding overwatch algorithm that operates in gridded terrain. In the algorithm, two groups of two entities move from a common initial location to a common goal. The movement is divided in stages; at each stage one group moves while the other remains stationary in the overwatch role.

Recall that the entity and unit route planning algorithms given in the reference find all of the terrain grid cells reachable in a tactically predetermined time interval and evaluate each reachable grid cell for exposure to enemy observation and formation mutual intervisibility along the route to that grid cell. At each bounding overwatch stage, the unit route planning algorithm is applied to the moving group to find the grid cells that it can reach during the given time interval. The moving group's destination is selected from among those candidate grid cells by applying the following criteria:

1. Reachability; reachable at the end of the time interval.
2. Configuration; ahead of the overwatching group by at least half the maximum distance a group can travel during the time interval and within a corridor predetermined relative to the overall movement direction.
3. Route quality; good in terms of exposure to enemy observation and formation mutual intervisibility.
4. Future safety; not exposed to enemy observation during the next time interval, when the moving group will become the overwatching group.
5. Observability; nearby grid cells from which projected enemy movements can be observed.

Evaluating the future safety and observability criteria require determining the intervisibility from the candidate grid cells to the projected locations of the enemy entities. That determination is done using the region-to-region intervisibility algorithm also described in [Teng,1992]. The final destination grid cell is chosen using a weighted sum and thresholds on these criteria and the moving group moves to the destination grid cell. Once that movement is completed, the two groups' roles (moving and overwatching) are reversed and the process is repeated.

ODIN SAF. [Stanzione,1993] briefly summarizes a bounding overwatch algorithm in the ODIN SAF system. A Dismounted Infantry platoon will split into two parts, or sections. The two sections then move alternately (one section moves while the other remains motionless) from covered location to covered location. A covered location is considered to be one that is covered relative to the direction of movement or the direction of known enemy forces. Note that this is a simplified approximation of the actual bounding overwatch maneuver as performed by the U.S. military; in reality, the motionless "overwatching" section should be located where it can observe in the direction of enemy forces, rather than simply be under cover from that direction.

ModSAF. [Courtemanche,1995a] briefly describes ModSAF's "Platoon Overwatch Movement" behavior. A platoon executing overwatch movement is split into two groups, with one group executing a "Hasty Occupy Position" task while the other group travels along a preplanned route. No details are given as to how the overwatching positions are selected. A different bounding overwatch implementation in ModSAF is described in [Rajput,1995c], but that reference also omits description of the terrain reasoning aspects of the problem.

4.6 Other terrain reasoning algorithms

Several other terrain reasoning algorithms will be mentioned in this subsection. They are:

1. Terrain search queries
2. Minefield site prediction
3. Finding observation locations
4. Captain subunit positioning

Terrain search queries. The hybrid quadtree and frame-based object terrain database proposed in [Antony,1988] seems to be particularly well suited for search queries, where the goal is to find all terrain objects of a given class that meet given selection criteria.

Minefield site prediction. [Doughty,1988] describes a combination of a quadtree terrain representation with a rule-based expert system embodying military minefield doctrine that predicts minefield sites.

Finding observation locations. [Keirsey,1988] proposes a heuristic for finding good locations for general observation. Assuming that a uniformly space grid of points is superimposed on the terrain, a visibility metric, measuring the observation area of each point, is computed. The visibility metric is the number of points visible from each point, weighted for the range from the observing point to the observed point. Note that the metric provides no information as the whether a point is useful for observing any particular point.

Captain subunit positioning. Captain is an automated knowledge acquisition system designed to allow a SME to teach an automated command agent tactical behavior [Hille,1994] [Hieb,1995]. Captain transforms a CTDB terrain database into an abstract geometric model that is then transformed into a semantic net [Hille,1995]. The semantic net is input to the automated command agent where a set of inference rules are used to reason about placement and movement of a unit's subunits. A version space of plausible unit plans is generated by applying the inference rules to the semantic net. The generated plans are evaluated by a human SME. Based on the SME's evaluation, additional inference rules are learned by Captain.

5. Conclusions

Computer Generated Forces systems control autonomous entities within virtual simulation systems. This survey began with a broad overview of CGF systems as a group, provided a compendium of CGF systems, and examined several of the more important or interesting examples in some detail. From this review, it should be clear that CGF systems are extremely important to DIS-type virtual simulation.

It then delved more deeply into the terrain representation formats used by CGF systems. A number of different terrain representation formats are in use in CGF systems, each with its own strengths and weaknesses.

Finally, terrain reasoning algorithms were considered. Terrain reasoning is of central importance to CGF systems' autonomous behavior generation. Algorithms to perform terrain reasoning are often heavily dependent on the details of the terrain representation format. Just as the application of AI techniques to CGF behavior generation in general has been hampered by the difficulties of the input transformation, so has the development of object-oriented terrain reasoning algorithms been troubled by the difficulties of finding tactically meaningful objects within the gridded or polygonal terrain representations common to CGF systems. However, effective terrain reasoning algorithms have been implemented based on geometric approaches.

6. Acknowledgments

I am pleased to acknowledge the assistance provided to me by the following people during the preparation of this document: J. Michael Moshell for inspiring me to start it, Barbara J. Schiavone for spurring me to complete it, Clark R. Karr, Robert W. Franceschini, Douglas D. Wood, and Mary P. Slepow for reviewing drafts of the document, and Daniel E. Mullally Jr. for providing subject matter expertise. Each of these people has my sincere thanks.

7. References

Abellanas, M., García-López, J., and Hurtado, F. (1994). "Consecutive guards", *Proceedings of the Sixth Canadian Conference on Computational Geometry*, Saskatoon Saskatchewan Canada, August 2-6 1995, pp. 393-398.

Ahmad, O., Cremer, J., Kearney, J., Willemsen, P., and Hansen, S. (1994). "Hierarchical, concurrent state machines for behavior modeling and scenario control", *Proceedings of the Fifth Annual Conference on AI, Simulation, and Planning in High Autonomy Systems*, Gainesville FL, December 7-9 1994, pp. 36-42.

Altman, M., Frogge, M. A., and Huder, R. (1991). "Simulating Behaviors in the High-Rise Fire Incidient Command Training System", *Proceedings of the Eurographics Workshop on Animation and Simulation*, Vienna Austria, September 1-2 1991.

Antony, R. (1988). "Representation Issues in the Design of a Spatial Database Management System", *Proceedings of the U.S. Army Symposium on Artificial Intelligence Research for Exploitation for the Battlefield Environment*, El Paso TX, November 15-16 1988, pp. 212-222.

Arkin, R. C. (1987). "Motor Schema Based Navigation for a Mobile Robot: An Approach to Programming by Behavior", *Proceedings of the 1987 IEEE International Conference on Robotics and Automation*, Institute of Electrical and Electronics Engineers, Raleigh NC, March 31-April 3 1987, pp. 264-271.

Aronson, J. (1994). "The SimCore Tactics Representation and Specification Language", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 187-193.

Aurenhammer, F. (1991). "Voronoi diagrams - A Survey of a Fundamental Geometric Data Structure", *Computing Surveys*, Vol. 23, No. 3, pp. 345-405.

Avis, D. and Bhattacharya, B. K. (1983). "Algorithms for computing d-dimensional Voronoi diagrams and their duals", in Preparata, F. P. (Ed.), *Advances in Computing Research*, Vol. 1, JAI Press, pp. 159-180.

Badler, N., Phillips, C., and Webber, B. (1993). *Simulating Humans: Computer Graphics, Animation and Control*, Oxford University Press, Oxford UK, 1993.

Bailey, M. M. and Companion, M. A. (1989). "State-of-the-Art Assessment for Simulated Forces", *Technical Report IST-CR-89-18*, Institute for Simulation and Training, 1988.

Barr, A. H. (1989). "Introduction to Physically-Based Modeling", Course #30, *ACM SIGGRAPH 89*, July 31-August 4 1989.

BBN (1992). "ModSAF", *Presentation*, BBN Systems and Technologies Corporation, September 22 1992.

Bedford, B. R. (1991). "Software Reliability Measurement on the B-2 Aircrew Training Device (ATD)", *Proceedings of the 13th Interservice/Industry Training Systems Conference*, Orlando FL, December 2-5 1991, pp. 155-161.

Belleville, P., Bose, P., Czyzowicz, J., Urrutia, J., and Zaks, J. (1994). "K-Guarding Polygons on The Plain", *Proceedings of the Sixth Canadian Conference on Computational Geometry*, Saskatoon Saskatchewan Canada, August 2-6 1995, pp. 381-386.

Benton, J. R. (1987). "Automated route finder for multiple tank columns", *Report ETL-4080*, U.S. Army Engineer Topographic Laboratories, September 1987.

Benton, J. R. (1991). "Automated Terrain Reasoning", *AI Exchange*, U.S. Military Academy, West Point NY, Vol. 5, No. 1, Winter 1991, pp. 8-9.

Bess, R. D. and Soderberg, B. T. (1991). "Battlefield Smoke - A New Dimension in Networked Simulation", *Proceedings of the 13th Interservice/Industry Training Systems Conference*, Orlando FL, December 2-5 1991, pp. 256-261.

Bhattacharyya, G. K. and Johnson, R. A. (1977). *Statistical Concepts and Methods*, John Wiley and Sons, New York NY, 1977.

Bimson, K., Marsden, C., McKenzie, F., and Paz, N. (1994). "Knowledge-Based Tactical Decision Making in the CCTT SAF Prototype", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 293-303.

Blitz, A., Moore, K., Powell, W., and Vail, P. (1988). "A general purpose modeling and simulation tool exploiting the Petri Net paradigm", *Tools for the Simulation Profession*, The Society for Computer Simulation, pp. 21-25.

Bockstahler, D. E. and Mowbray, T. J. (1991). "Autonomous Navigation for Semi-Automated Forces", *Proceedings of the 2nd Behavioral Representation and Computer Generated Forces Symposium*, Institute for Simulation and Training, Orlando FL, May 6-7 1991, pp. A 1-4.

Bonsignore, E. (1992). "Gulf Experience Raises Tank Survivability Issues", *Military Technology*, February 1992, pp. 64-70.

Booker, L. B., Goldberg, D. E., and Holland, J. H. (1989). "Classifier Systems and Genetic Algorithms", *Artificial Intelligence*, Vol. 40, Nos. 1-3, September 1989, pp. 235-282.

Booker, L., Brooks, P, Garrett, R., Giddings, V., Salisbury, M., and Worley, R. (1993). "1993 DMSO Survey of Semi-Automated Forces", *Defense Modeling and Simulation Office Report*, July 30 1993.

Bourne, D. A. (1982). "A Numberless, Tensed Language for Action Oriented Tasks", *Technical Report CMU-RI-TR82-12*, Carnegie Mellon University.

Bouwens, C. L., Matusof, R., and Loper, M. L. (1995). "Application Profiles of Distributed Interactive Simulation Standards for Synthetic Environments", *Proceedings of the 6th International Training Equipment Conference*, The Hague, The Netherlands, April 25-27 1995, pp. 165-172.

Branch, G. (1989). "Simulation Technology: Special Forces Silent Partner", *National Defense*, January 1989, Vol. 73, No. 444, pp. 55-57.

Brand, S. (1987). *The Media Lab: Inventing the Future at MIT*, Viking Penguin, New York NY, 1987.

Brassard, G. and Bratley, P. (1988). *Algorithmics: Theory and Practice*, Prentice Hall, Englewood Cliffs NJ, 1988.

Braudaway, W. (1992). "An Object Oriented, Subsumptive Task-Level Architecture for the Integration of Technology for the Control of Semi-Automated AirLand Battle Forces", *Technical Report*, IBM Federal Systems Company, December 11 1992.

Braudaway, W. (1993). "A Blackboard Approach to Generated Forces", *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, March 17-19 1993, pp. 11-20.

Brooks, R. A., Buchanan, B. G., Lenat, D. B., McKeown, D. M., and Fletcher, J. D. (1989). "Panel Review of the Semi-Automated Forces", *IDA Document D-661*, Institute for Defense Analyses, 1989.

Butler, B. E. and Weihagen, G. (1995a). "Common Database Toolset: Extending and Prototyping the Architecture for Distributed Interactive Simulation", *Proceedings of the 1995 Simulation MultiConference*, Military, Government, and Aerospace Simulation, Phoenix AZ, April 9-13 1995, pp. 129-134.

Butler, B. E. and Weihagen, G. (1995b). "Common Database Toolset: Extending and Prototyping the Architecture for Distributed Interactive Simulation", *Proceedings of the 6th International Training Equipment Conference*, The Hague, The Netherlands, April 25-27 1995, pp. 151-163.

Byers, D. (1988). "Simulator Networking: New Training Horizons", *Military Forum*, May 1988, pp. 46-50.

Calder, R. B., Smith, J. E., Courtemanche, A. J., Mar, J. M. F., and Ceranowicz, A. Z. (1993). "ModSAF Behavior Simulation and Control", *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, March 17-19 1993, pp. 347-356.

Calder, R. B. and Evans, A. B. (1994). "Construction of a Corps Level CGF", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 487-496.

Calder, R. B., Peacock Jr., J. C., Panagos, J., and Johnson, T. E. (1995a). "Integration of Constructive, Virtual, Live, and Engineering Simulations in the JPSD CLCGF", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 71-82.

Calder, R. B., Peacock Jr., J. C., Wise, B. P., Stanzione, T., Chamberlain, F., Panagos, J. (1995b). "Implementation of a Dynamic Aggregation/Deaggregation Process in the JPSD CLCGF", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 83-91.

Campbell, C. E. and McCulley, G. (1994). "Terrain Reasoning Challenges in the CCTT Dynamic Environment", *Proceedings of the Fifth Annual Conference on AI, Simulation, and Planning in High Autonomy Systems*, Gainesville FL, December 7-9 1994, pp. 55-61.

Campbell, C., Hull, R., Root, E., and Jackson, L. (1995). "Route Planning in CCTT", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 233-243.

Ceranowicz, A., Downes-Martin, S., and Saffi, M. (1988). "SIMNET Semi-Automated Forces 3.0: A Functional Description (Revised)", *Report No. 6939*, BBN Systems and Technologies Corporation, October 27 1988.

Ceranowicz, A. (1994a). "ModSAF Capabilities", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 3-8.

Ceranowicz, A. (1994b). "Modular Semi-Automated Forces", *Proceedings of the 1994 Winter Simulation Conference*, Society for Computer Simulation, Orlando FL, December 11-14 1994, pp. 755-761.

Ceranowicz, A., Coffin, D., Smith, J., Gonzalez, R., and Ladd, C. (1994c). "Operator Control of Behavior in ModSAF", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 9-16.

Chazelle, B. and Guibas, L. J. (1988). "Visibility and Intersection Problems in Plane Geometry", *Technical Report CS-TR-167-88*, Princeton University, 1988.

Chen, J. and Sartor, M. (1994). "Fluids in a Distributed Interactive Simulation", *Proceedings of the Fifth Annual Conference on AI, Simulation, and Planning in High Autonomy Systems*, Gainesville FL, December 7-9 1994, pp. 43-45.

Chen, J. and Sartor, M. (1995). "An Approach to Implementing Fluids in a Distributed Interactive Simulation", *Proceedings of the 12th DIS Workshop on Standards for the Interoperability of Defense Simulations*, Institute for Simulation and Training, Orlando FL, March 13-17 1995, pp. 309-317.

Cheng, S. (1994). "Widest Empty Corridor with Multiple Links and Right-angle Turns", *Proceedings of the Sixth Canadian Conference on Computational Geometry*, Saskatoon Saskatchewan Canada, August 2-6 1995, pp. 57-62.

Chervenak, J. and D'Errico, J. (1993). "Advanced Warfighting Experiment, Evaluation of Semi-Automated Forces Dismounted Infantry, Phase I", *Dismounted Warfighting Battle Lab*, November 1993.

Cheung, S. E. and Loper, M. L. (1994). "Traffic Characterization of Manned-Simulators and Computer Generated Forces in DIS Exercises", *Proceedings of the Fifth Annual Conference on AI, Simulation, and Planning in High Autonomy Systems*, Gainesville FL, December 7-9 1994, pp. 70-76.

Cimini, F. C., Campbell, C. E., and Petty, M. D. (1992). "A Simple Flight Dynamics Model for Computer Generated Forces", *Proceedings of the Southeastern Simulation Conference 1992*, The Society for Computer Simulation, Pensacola FL, October 22-23 1992, pp 41-47.

Cisneros, J. E., Karr, C. R., and McCauley-Bell, P. (1995). Intelligent Targeting in ModSAF, *Technical Report IST-CR-95-36*, Institute for Simulation and Training, December 1 1995.

Clarke, T. L. and Otte, J. M. (1991). "Human Behavioral Modeling Using Catastrophe Theory", *Proceedings of the 2nd Behavioral Representation and Computer Generated Forces Symposium*, Institute for Simulation and Training, Orlando FL, May 6-7 1991, pp. C 1-8.

Cole, R. and Sharir, M. (1989). "Visibility Problems for Polyhedral Terrains", *Journal of Symbolic Computation*, Vol. 7, pp. 11-30.

Coleman, V., Gonzalez, G. L., Petty, M. D., Smith, S. H., Vanzant-Hodge, A., Watkins, J. E., and Wood, D. D. (1990). "Alternative Implementations of Knowledge Representation and Acquisition Methods in Distributed Interactive Simulation: Investigations and Findings", *Technical Report IST-TR-90-21*, Institute for Simulation and Training, University of Central Florida, 1990.

Cosby, L. N. (1995). "SIMNET: an insider's perspective", *Distributed Interactive Simulation Systems for Simulation and Training in the Aerospace Environment*, SPIE Critical Review 58, Orlando FL, April 19-20 1995, pp. 59-72.

Courtemanche, A. J. and Monday, P. (1994). "The Incorporation of Validated Combat Models into ModSAF", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 129-140.

Courtemanche, A. J. and Ceranowicz, A. (1995a). "ModSAF Development Status", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 3-13.

Courtemanche, A. J., Hamilton, S. E., and Monday, P. (1995b). "Representation of Missiles in ModSAF", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 267-274.

Cox, A., Gibb, A., and Page, I. (1995). "Army Training and CGFs - A UK Perspective", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 353-363.

Cox, A. (1995). "Simulation Management Functionality in the IST CGF", *Proceedings of the 12th DIS Workshop on Standards for the Interoperability of Defense Simulations*, Institute for Simulation and Training, Orlando FL, March 13-17 1995, pp. 335-341.

Craft, M. A. and Petty, M. D. (1994a). "Experimental Conversion of the IST Computer Generated Forces Simulator from C to Ada", *Technical Report IST-TR-94-13*, Institute for Simulation and Training, April 20 1994.

Craft, M. A., Cisneros, J. E., and Karr, C. R. (1994b). "Dynamic Obstacle Avoidance", *Technical Report IST-TR-94-41*, Institute for Simulation and Training, December 21 1994.

Craft, M. A. and Petty, M. D. (1995a). "Experimental Conversion of the IST Computer Generated Forces Simulator from C to Ada", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 433-442.

Craft, M. A., Franceschini, D. J., Kraus, M. K., Mullally, D. E., Adkins, M. K., Albright, R. L., Nida, J. C., and Napravnik, L. J. (1995b). "Final Report, AAAV: Demonstrating the Feasibility of Using Virtual Simulation for Test and Evaluation", *Technical Report IST-CR-95-32*, Institute for Simulation and Training, October 9 1995.

Cremer, J., Kearney, J., Papelis, Y., and Romano, R. (1994). "The Software Architecture for Scenario Control in the Iowa Driving Simulator", *Proceedings of the Fourth Conference on*

Computer Generated Forces and Behavioral Representation, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 373-381.

Crooks, W. H., Fraser, R. E., Herman, J. A., Jacobs, R. S., McDonough, J. G., Bonanni, P., Harrison, B., Junot, A., and Kirk, J. (1990). "SIMNET Semi-Automated Forces (Version 3.x) Functional Specification", *Technical Report PTR-4043-15-0200-4/90*, Perceptronics, April 18 1990.

Cronin, T. M. (1988). "Allocating Sensor Envelope Patterns to a Map Partitioned by Territorial Contours", *Proceedings of the U.S. Army Symposium on Artificial Intelligence Research for Exploitation for the Battlefield Environment*, El Paso TX, November 15-16 1988, pp. 65-78.

Croucher, G. and Law, D. (1991). "Using Parallel Ada in the Implementation of Simulation and Training Systems", *Proceedings of the 13th Interservice/Industry Training Systems Conference*, Orlando FL, December 2-5 1991, pp. 196-204.

Crowe, M. X. (1990). "The Application of Artificial Neural Systems to the Training of Air Combat Decision-Making Skills", *Proceedings of the 12th Interservice/Industry Training Systems Conference*, Orlando FL, November 6-8 1990, pp. 302-312.

Cunningham, C. T. (1993). "Control of Movement in an Arbitrary Polygonal Terrain", *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, March 17-19 1993, pp. 307-315.

Cunningham, C. T. (1994). "Development of Intelligent Simulations at LLNL", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 345-352.

D'Angelo, G. J. (1983). "Tutorial on Petri Nets", *Simuletter*, Vol. 14, Nos. 1-4, pp. 10-25.

D'Errico, J. (1994). "Evaluation of the SAFDI System at the USAIS, Ft. Benning, GA. SIMNET Site", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 149-154.

Danisas, K., Smith, S. H., and Wood, D. D. (1990). "Sequencer/Executive for Modular Simulator Design", *Technical Report IST-TR-90-1*, Institute for Simulation and Training, 1990.

Devillers, O., Golin, M., Kedem, K., and Schirra, S. (1994). "Revenge of the Dog: Queries on Voronoi Diagrams of Moving Points", *Proceedings of the Sixth Canadian Conference on Computational Geometry*, Saskatoon Saskatchewan Canada, August 2-6 1995, pp. 122-127.

Devlin, M. J. I. (1990). "Ada Technology for Simulation", *National Defense*, November 1990, Vol. 75, No. 462, pp. 41-43.

Deutsch, S. (1993). "Notes Taken on the Quest for Modeling Skilled Human Behavior", *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, March 17-19 1993, pp. 359-365.

DIS Steering Committee (1994). "The DIS Vision: A Map to the Future of Distributed Simulation", *Technical Report IST-SP-94-01*, Institute for Simulation and Training, May 1994.

DMA (1995). *Digitizing the Future*, Fourth Edition, Defense Mapping Agency.

Donovan, K. B. (1990). "Real Time Mission Rehearsal: The Database Challenge", *National Defense*, November 1990, Vol. 75, No. 462, pp. 21-24.

Donner, M. E. (1991). "The Challenges of Simulating a Hovercraft Ocean Environment", *Proceedings of the 13th Interservice/Industry Training Systems Conference*, Orlando FL, December 2-5 1991, pp. 301-313.

Doughty, J. W., Downs, A. L., Gillotte, M. J., and Hirsch, S. A. (1988). "An Expert System for Minefield Site Prediction", *Proceedings of the U.S. Army Symposium on Artificial Intelligence Research for Exploitation for the Battlefield Environment*, El Paso TX, November 15-16 1988, pp. 166-179.

Downes-Martin, S. (1990). "Replacing the Exercise Controller with the Enemy: the SIMNET Semi-Automated Forces Approach", *Proceedings of the 1st International Training Equipment Conference*, UK, April 1990.

Downes-Martin, S. (1992). "Proposed Architecture for the Computer Generated Forces", *Presentation*, Loral ADS, January 30 1992.

Dyer, D. E. and Gunsch, G. H. (1993). "Enlarging the Universal Plan for Air Combat Adversaries", *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, March 17-19 1993, pp. 255-261.

Edelsbrunner, H. and Seidel, R. (1986). "Voronoi diagrams and arrangements", *Discrete and Computational Geometry*, Vol. 1, pp. 25-44.

El Gindy, H. and Avis, D. (1981). "A Linear Algorithm for Computing the Visibility Polygon from a Point", *Journal of Algorithms*, Vol. 2, pp. 186-197.

Everett, E. and Rivera-Campo, E. (1994). "Edge guarding a triangulated polyhedral terrain", *Proceedings of the Sixth Canadian Conference on Computational Geometry*, Saskatoon Saskatchewan Canada, August 2-6 1995, pp. 293-295.

Ewing, T. L. (1992). "Mathematical Modeling of the Terrain Around a Robot", *NASA Tech Briefs*, Vol. 16, No. 6, June 1992, p. i.

Fawcett, D. H. (1991). "Sensor Data Base Correlation", *Proceedings of the 13th Interservice/Industry Training Systems Conference*, Orlando FL, December 2-5 1991, pp. 106-112.

Field, D. (1986). "Implementing Watson's algorithm in three dimensions", *Proceedings of the 2nd Annual Symposium on Computational Geometry*, Association for Computing Machinery, pp. 246-259.

Fishwick, P. A., Petty, M. D., and Mullally, D. E. (1991). "Key Research Directions in Behavioral Representation for Computer Generated Forces", *Proceedings of the 2nd Behavioral Representation and Computer Generated Forces Symposium*, Institute for Simulation and Training, Orlando FL, May 6-7 1991, pp. E 1-14.

Fishwick, P. A. (1993). "A Simulation Environment for Multimodeling", *Discrete Event Dynamic Systems: Theory and Applications 3*, Kluwer, Boston MA, pp. 151-171.

Foley, J. D. and Van Dam, A. (1982). Fundamentals of Interactive Computer Graphics, Addison-Wesley, Reading MA, 1982.

Fortune, S. (1987). "A sweepline algorithm for Voronoi diagrams", *Algorithmica*, Vol. 2, pp. 153-174.

Fortune, S. (1992). "Voronoi diagrams and Delaunay triangulations", *Computing in Euclidean Geometry, Lecture Notes Series on Computing*, Vol. 1, World Scientific, pp. 193-234.

Franceschini, R. W. (1992). "Intelligent Placement of Disaggregated Entities", *Proceedings of the Southeastern Simulation Conference 1992*, The Society for Computer Simulation, Pensacola FL, October 22-23 1992, pp. 20-26.

Franceschini, R. W., Parra, F. R., Watkins, J. E., and Nanda, S. (1993a). "SAFDI User's Guide", *Technical Report IST-TR-93-23*, Institute for Simulation and Training, University of Central Florida, August 31 1993.

Franceschini, R. W., Watkins, J. E., Parra, F. R., McCulley, J. E., Lautenschlager, J. A., Jackson, L. A., and Nanda, S. (1993b). "SAFDI Support Manual", *Technical Report IST-TR-93-24*, Institute for Simulation and Training, University of Central Florida, August 31 1993.

Franceschini, R. W. and Petty, M. D. (1994a). "Dismounted Infantry in DIS Style Scenarios: A SAFDI Project Overview", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 155-167.

Franceschini, R. W., Petty, M. D., and Reece, D. A. (1994b). "Dismounted Infantry in Distributed Interactive Simulation", *Proceedings of the 16th Interservice/Industry Training Systems and Education Conference*, Orlando FL, November 28-December 1 1994, pp. 4-20.

Franceschini, R. W. and Petty, M. D. (1995a). "Status Report on the Development of PDUs to Support Constructive+Virtual Linkages", *Proceedings of the 12th DIS Workshop on Standards for the Interoperability of Defense Simulations*, Institute for Simulation and Training, Orlando FL, March 13-17 1995, pp. 385-388.

Franceschini, R. W. and Petty, M. D. (1995b). "Linking constructive and virtual simulation in DIS", *Distributed Interactive Simulation Systems for Simulation and Training in the Aerospace Environment*, SPIE Critical Review 58, Orlando FL, April 19-20 1995, pp. 281-298.

Franceschini, R. W. and Karr, C. R. (1995c). "Integrating Constructive and Virtual Simulation", *Proceedings of the 6th International Training Equipment Conference*, The Hague, The Netherlands, April 25-27 1995, pp. 425-431.

Franceschini, R. W. (1995d). "Integrated Eagle/BDS-D: A Status Report", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 21-25.

Fraser, R. E. and Herman, J. A. (1990a). "Integration of SIMNET Dismounted Infantry Simulators with SAFOR Workstations, Functional Specification", *Technical Report PTR-4043-06-0000-90/90*, Perceptronics, February 23 1990.

Fraser, R. E. and Herman, J. A. (1990b). "Integration of SIMNET Dismounted Infantry Simulators with SAFOR Workstations, Operator's Manual", *Technical Report PTR-4043-06-0000-90/01*, Perceptronics, February 23 1990.

Friedman, G. and Toms, R. (1993a). *Security Exercise Evaluation System (SEES) V2.0 Accreditation Project Final Report*, Lawrence Livermore National Laboratory, March 1993.

Friedman, G. (1993b). "The Security Exercise Evaluation System (SEES) V2.0 Accreditation Process", *Proceedings of the 1993 Summer Computer Simulation Conference*, Boston MA, July 19-21 1993, pp. 887-893.

Gagné, D. (1995). "Training the Crew Concept via Multi-Agent Systems", *Proceedings of the 6th International Training Equipment Conference*, The Hague, The Netherlands, April 25-27 1995, pp. 175-184.

Gat, E., Fearey, J., and Provenzano, J. (1993). "Semi-Automated Forces for Corps Battle Simulation", *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, March 17-19 1993, pp. 69-74.

Gates, K. and Frantz, F. (1990). "Semi-Automated Force Simulation using a Blackboard", *Proceedings of the 12th Interservice/Industry Training Systems Conference*, Orlando FL, November 6-8 1990, pp. 295-301.

Ghosh, S. K. and Mount, D. M. (1987). "An Output Sensitive Algorithm for Computing Visibility Graphs", *Proceedings of the 28th IEEE Symposium on Foundations of Computer Science*, Institute of Electrical and Electronics Engineers, Los Angeles CA, 1987, pp. 11-19.

Ghosh, S. K. (1988). "On recognizing and characterizing visibility graphs of simple polygons", *Lecture Notes in Computer Science 318*, Springer-Verlag, New York NY, pp. 97-104.

Giroux, R. (1995). "Laser PDU Lessons Learned", *Proceedings of the 12th DIS Workshop on Standards for the Interoperability of Defense Simulations*, Institute for Simulation and Training, Orlando FL, March 13-17 1995, pp. 455-458.

Goel, A. K., Callantine, T. J., Shankar, M., and Chandrasekaren, B. (1991). "Representation, Organization, and Use of Topographic Models of Physical Spaces for Route Planning", *Proceedings of the Seventh Conference on Artificial Intelligence Applications CAIA-92*, Volume I: Technical Papers, Institute of Electrical and Electronics Engineers, Miami Beach FL, February 1991, pp. 308-314.

Gold, C. M. (1994). "Persistent spatial relations - a systems design objective", *Proceedings of the Sixth Canadian Conference on Computational Geometry*, Saskatoon Saskatchewan Canada, August 2-6 1995, pp. 219-224.

Goldiez, B. F. (1991). "The Impact of Verification, Validation, and Accreditation on Simulation and Training", *Military Simulation & Training*, April 1991, pp. 35-37.

Goldiez, B. F. and Nelson, R. S. (1994). "Terrain Database Correlation", *Proceedings of the 5th International Training Equipment Conference*, The Hague, The Netherlands, April 26-28 1994, pp. 337-346.

Goldiez, B. F. (1995). "History of networked simulations", *Distributed Interactive Simulation Systems for Simulation and Training in the Aerospace Environment*, SPIE Critical Review 58, Orlando FL, April 19-20 1995, pp. 39-58.

Gonzalez, A. J., Mullally, D. E., and Gonzalez, G. L. (1991). "A Hierarchical Rule-Based Architecture for Implementing Intelligent Adversaries in a SIMNET Environment", *Proceedings of the 13th Interservice/Industry Training Systems Conference*, Orlando FL, December 2-5 1991, pp. 339-346.

Gonzalez, G., Mullally, D. E., Smith, S. H., Vanzant-Hodge, A. F., Watkins, J. E., and Wood, D. D. (1990). "A Testbed for Automated Entity Generation in Distributed Interactive Simulation", *Technical Report IST-TR-90-15*, Institute for Simulation and Training, August 15 1990.

Green, P. and Sibson, R. (1977). "Computing Dirichlet tesselations in the plane", *Computing Journal*, Vol. 21, pp. 168-173.

Gross, D. C. and Stuckey, L. D. (1991). "Ada Types: The Cornerstone of Simulation Models", *Proceedings of the 13th Interservice/Industry Training Systems Conference*, Orlando FL, December 2-5 1991, pp. 8-18.

Guckenberger, D., Uliano, K. C., and Lane, N. E. (1992). "The Application of Above Real-Time Training (ARTT) for Simulators: Acquiring High Performance Skills", *Proceedings of the 14th Interservice/Industry Training Systems and Education Conference*, San Antonio TX, November 2-4 1992, pp. 928-935.

Guibas, L. J., Hershberger, J., Leven, D., Sharir, M., and Tarjan, R. E. (1987). "Linear Time Algorithms for Visibility and Shortest Path Problems inside Simple Polygons", *Algorithmica*, Vol. 2, pp. 209-233.

Guibas, L. J. and Hershberger, J. (1989). "Optimal Shortest Path Queries in a Simple Polygon", *Journal of Computer and System Sciences*, Vol. 39, pp. 126-152.

Haeger, S. D. (1994). "Modeling the Littoral Ocean for Military Applications", *Proceedings of the 16th Interservice/Industry Training Systems and Education Conference*, Orlando FL, November 28-December 1 1994, pp. 4-12.

Haque, S., Gonzalez, R., and Schaffer, R. (1995). "Environmental Simulation in DIS", *Proceedings of the 6th International Training Equipment Conference*, The Hague, The Netherlands, April 25-27 1995, pp. 249-258.

Hardis, K. C. (1994). "Resolving the Debate Over Standards for the Interchange of Simulator Databases", *Proceedings of the 11th DIS Workshop on Standards for the Interoperability of Defense Simulations*, Institute for Simulation and Training, Orlando FL, September 26-30 1994, pp. 89-93.

Harkrider, S. and Yeakel, W. P. (1995). "Anti-Armor Advanced Technology Demonstration (A2ATD) Experiment One Or What We Did On Our Summer Vacation", *Proceedings of the 12th DIS Workshop on Standards for the Interoperability of Defense Simulations*, Institute for Simulation and Training, Orlando FL, March 13-17 1995, pp. 733-743.

Harmon, P. and King, D. (1986). *Expert Systems*, John Wiley and Sons, New York NY, 1986.

Harmon, S. Y., Yang, S. C., Howard, M. D., and Tseng, D. Y. (1991). "A Behavior-Based SAFOR and Its Preliminary Evaluation", *Proceedings of the 2nd Behavioral Representation and Computer Generated Forces Symposium*, Institute for Simulation and Training, Orlando FL, May 6-7 1991, pp. G 1-14.

Harmon, S. Y., Yang, S. C., and Tseng, D. Y. (1994). "Command and Control Simulation for Computer Generated Forces", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 263-273.

Harnad, S. (1992). "The Turing Test is not a trick: Turing indistinguishability is a scientific criterion", *SIGART Bulletin*, Vol. 3, No. 4, October 1992, pp. 9-10.

Harrison, M. (1992). *Wing Commander I & II: The Ultimate Strategy Guide*, Prima Publishing, Rocklin CA, 1992.

Hayslip, I. C. and Gilmore, J. F. (1988). "A Multi-Level Knowledge Representation for Reasoning about Terrain", *Proceedings of the U.S. Army Symposium on Artificial Intelligence Research for Exploitation for the Battlefield Environment*, El Paso TX, November 15-16 1988, pp. 123-137.

Held, M. (1994). "On Computing Voronoi Diagrams of Convex Polyhedra by Means of Wavefront Propagation", *Proceedings of the Sixth Canadian Conference on Computational Geometry*, Saskatoon Saskatchewan Canada, August 2-6 1995, pp. 128-133.

Hernández-Peñalver, G. (1994). "Controlling Guards", *Proceedings of the Sixth Canadian Conference on Computational Geometry*, Saskatoon Saskatchewan Cnada, August 2-6 1995, pp. 387-392.

Hershberger, J. (1989). "An Optimal Visbility Graph Algorithm for Triangulated Simple Polygons", *Algorithmica*, Vol. 4, pp. 141-155.

Hieb, M. R., Tecuci, G., Pullen, J. M., Ceranowicz, A., and Hille, D. (1995). "A Methodology and Tool for Constructing Adaptive Command Agents for Computer Generated Forces", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 135-146.

Hille, D., Hieb, M. R., and Tecuci, G. (1994). "CAPTAIN: Building Agents that Plan and Learn", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 411-422.

Hille, D., Hieb, M. R., Tecuci, G., and Pullen, J. M. (1995). "Abstracting Terrain Data Through Semantic Terrain Transformations", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 355-365.

Hoff, B., Howard, M. D., and Tseng, D. Y. (1995). "Path Planning with Terrain Utilization in ModSAF", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 255-263.

Holmes, P. D. and Jungert, E. R. A. (1992). "Symbolic and Geometric Connectivity Graph Methods for Route Planning in Digitized Maps", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 14, No. 5, May 1992, pp. 549-565.

Hopcroft, J. E. and Ullman, J. D. (1979). *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading MA, 1979.

Horan, B. (1994). "A SEOD Sneak Preview (Coming Soon to a Simulator Near You", *Proceedings of the 11th DIS Workshop on Standards for the Interoperability of Distributed Simulations*, Institute for Simulation and Training, Orlando FL, September 26-30 1994, pp. 379-388.

Howard, M. D., Hoff, B., and Tseng, D. Y. (1995). "Individual Combatant Developments in ModSAF", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 479-486.

Hunter, S. J. and Puckett, H. R. (1991). "Modeling of the Intelligent Threat in a Dense Tactical Training Environment", *Proceedings of the 13th Interservice/Industry Training Systems Conference*, Orlando FL, December 2-5 1991, pp. 328-334.

Huon, M. (1989). "Expert system for the simulation of tank platoon behavior in a synthetic scene", *Unpublished report*, SOGITEC, 1989.

Jacobs, R. S., McDonough, J. G., and Crooks, W. H. (1990). "Semi-Automated Forces in Distributed Simulation", *Proceedings of the DARPA Conference on Behavioral Representation in Semi-Automated Forces*, Ft. Knox KY, October 25 1990.

Jaszlics, S. L. (1993). "Artificial Intelligence in Tactical Command and Control Applications: Architecture and Tools", *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, March 17-19 1993, pp. 367-373.

Jaszlics, I. J., Jaszlics, S. L., and Jones, S. H. (1994). "Automated Battlefield Simulation Command and Control Using Artificial Neural Networks", *Proceedings of the Fifth Annual Conference on AI, Simulation, and Planning in High Autonomy Systems*, Gainesville FL, December 7-9 1994, pp. 100-105.

Johannesen, N. P. (1995). "Synthetic Theater of War-Europe (STOW-E) A Brigade Training Environment", *Proceedings of the 6th International Training Equipment Conference*, April 25-27 1995, The Hague, The Netherlands, pp. 115-122.

Johnson, W. L. (1992). "Needed: A New Test of Intelligence", *SIGART Bulletin*, Vol. 3, No. 4, October 1992, pp. 7-9.

Johnson, W. L. (1994). "Agents that Explain Their Own Actions", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 87-95.

Jones, R. E. (1993a). "Using CGF for Analysis and Combat Development", *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, March 17-19 1993, pp. 209-220.

Jones, R. M., Tambe, M., Laird, J. E., and Rosenbloom, P. S. (1993b). "Intelligent Automated Agents for Flight Training Simulators", *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, March 17-19 1993, pp. 33-42.

Jones, R. E. and Lattimore, P. J. (1994a). "An Implementation of Battalion Level C3I for CGF", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 287-292.

Jones, R. M., Laird, J. E., Tambe, M., and Rosenbloom P. S. (1994b). "Generating Behavior in Response to Interacting Goals", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 317-324.

Jones, R. M. and Laird, J. E. (1994c). "Multiple Information Sources and Multiple Participants: Managing Situational Awareness in an Autonomous Agent", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 511-517.

Kada, A. (1994). "Smart Tactical Environments", *Interactions*, No. 10, November 1994, pp. 4-5.

Karr, C. R., Petty, M. D., Van Brackle, D. R., Cross, D. D., Franceschini, R. W., Hull, R. D., Provost, M. H., and Smith, S. H. (1992a). "The IST Semi-Automated Forces Dismounted Infantry System: Capabilities, Implementation, and Operation", *Technical Report IST-TR-92-6*, Insititute for Simulation and Training, University of Central Florida, 28 February 1992.

Karr, C. R., Franceschini, R. W., Perumalla, K. R. S., and Petty, M. D. (1992b). "Integrating Battlefield Simulations of Different Granularity", *Proceedings of the Southeastern Simulation Conference 1992*, The Society for Computer Simulation, Pensacola FL, October 22-23 1992, pp. 48-55.

Karr, C. R., Franceschini, R. W., Perumalla, K. R. S., and Petty, M. D. (1993a). "Integrating Aggregate and Vehicle Level Simulations", *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, March 17-19 1993, pp. 231-242.

Karr, C. R. and Watkins, J. E. (1993b). "Using Distributed Interactive Simulation and Computer Generated Forces to Evaluate and Refine New Weapon Systems", *Proceedings of the 1993 Summer Computer Simulation Conference*, Boston MA, July 19-21 1993, pp. 852-857.

Karr, C. R. and Root, E. D. (1994a). "Integrating Aggregate and Vehicle Level Simulations", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 425-435.

Karr, C. R. and Franceschini, R. W. (1994b). "Status Report on the Integrated Eagle/BDS-D Project", *Proceedings of the 1994 Winter Simulation Conference*, Society for Computer Simulation, Orlando FL, December 11-14 1994, pp. 762-769.

Karr, C. R., Craft, M. A., and Cisneros, J. E. (1995a). "Dynamic Obstacle Avoidance", *Distributed Interactive Simulation Systems for Simulation and Training in the Aerospace Environment*, SPIE Critical Review 58, Orlando FL, April 19-20 1995, pp. 195-219.

Karr, C. R., Rajput, S., Cisneros, J. E., and Nee, H. (1995b). "Automated Mission Planning in ModSAF", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 159-167.

Karr, C. R., Craft, M. A., and Cisneros, J. E. (1995c). "Dynamic Obstacle Avoidance for Computer Generated Forces", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 245-253.

Karr, C. R. and Rajput, S. (1995d). "Unit Route Planning", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 295-304.

Karr, C. R., Rajput, S., and Breneman, L. J. (1995e). "Comparison of the A* and Iterative Deepening A* Graph Search Techniques", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 443-449.

Katz, A. and Ross, A. (1989). "One on one Helicopter Combat Simulated by Chess Type Lookahead", *AIAA Flight Simulation Technologies Conference*, Boston MA, August 1989, p. 357.

Katz, A., Butler, B., and Allen, D. (1991). "A Computer Generated Helicopter for Air to Air Combat", *AIAA Simulation Technologies Conference*, New Orleans LA, August 1991, p. 82.

Katz, A. and Butler, B. (1992). "A Flight Model for Unmanned Simulated Helicopters", *Journal of Aircraft*, Vol. 29, No. 4., July-August 1992, p. 521.

Katz, A. (1993). "Intelligent Player - First Principle Foundations", *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, March 17-19 1993, pp. 329-334.

Katz, A. and Butler, B. (1994). "Game Commander - Applying an Architecture of Game Theory and Tree Lookahead to the Command and Control Process", *Proceedings of the Fifth Annual Conference on AI, Simulation, and Planning in High Autonomy Systems*, Gainesville FL, December 7-9 1994, pp. 106-112.

Keegan, J. (1994). *A History of Warfare*, Alfred A. Knopf, New York NY, 1994.

Keirsey, D. M., Krozel, J., and Payton, D. W. (1988). "Scale-Space Representations for Flexible Automated Terrain Reasoning", *Proceedings of the U.S. Army Symposium on Artificial Intelligence Research for Exploitation for the Battlefield Environment*, El Paso TX, November 15-16 1988, pp. 108-118.

Keirsey, D., Krozel, J., Payton, D., and Tseng, D. (1994). "Case-Based Computer Generated Forces", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 307-316.

Kendall, T. M., Purnell, T. A., and Kaste, V. A. (1995). "Adaptive Grid Generation for Variable Resolution Terrain", *Proceedings of the 12th DIS Workshop on Standards for the Interoperability of Defense Simulations*, Institute for Simulation and Training, Orlando FL, March 13-17 1995, pp. 513-519.

Kilby, M., Lisle, C., Altman, M., and Sartor, M. (1994). "Dynamic Environment Simulation with DIS Technology", *Proceedings of the 16th Interservice/Industry Training Systems and Education Conference*, Orlando FL, November 28-December 1 1994, p. 4-18.

Kocabas, S., Oztemel, E., Uldudag, M., and Koc, N. (1995). "Automated Agents that Learn and Explain Their Own Actions: A progress report", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 63-68.

Kornell, J. (1987). "Reflections on using knowledge based systems for military simulation", *Simulation*, Vol. 48, No. 4, April 1987, pp. 144-148.

Koss, F. V. and Lehman, J. F. (1994). "Knowledge Acquisition and Knowledge Use in a Distributed IFOR Project", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 541-544.

Krecker, D. K. (1994). "Enhancement and Use of the IST CGF Testbed for a Precision Gunnery Demonstration", *Proceedings of the Fourth Conference on Computer Generated Forces and*

Behavioral Representation, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 51-57.

Kreutzer, W. (1986). *System Simulation: Programming Languages and Styles*, Addison-Wesley, Sydney Australia, 1986.

Kuhl, J. G. and Wargo, J. (1994). "High Fidelity Virtual Prototyping to Support Ground Vehicle Acquisition", *Proceedings of the 16th Interservice/Industry Training Systems and Education Conference*, Orlando FL, November 28-December 1 1994, pp. 4-21.

Kwak, S. (1995). "A Comparison Study of Behavioral Representation Alternatives, *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 529-539.

Laird, J. E., Newell, A., and Rosenbloom, P. S. (1987). "SOAR: An Architecture for General Intelligence", *Artificial Intelligence*, Vol. 33, No. 1, 1987, pp. 1-64.

Laird, J. E., Jones, R. M., Nielsen, P. E. (1994). "Coordinated Behavior of Computer Generated Forces in TacAir-Soar", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 325-332.

Laird, J. E., Johnson, W. L., Jones, R. M., Koss, F., Lehman, J. F., Nielsen, P. E., Rosenbloom, P. S., Rubinoff, R., Schwamb, K., Tambe, M., Van Dyke, J., van Lent, M., and Wray III, R. E. (1995). "Simulated Intelligent Forces for Air: The Soar/IFOR Project 1995", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 27-36.

Landweer, P. (1993a). "ACETEF Joint Aeronautical Commander's Group Demo and E-2C Tests", *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, March 17-19 1993, pp. 145-153.

Landweer, P. (1993b). "Action/Cognition Behavior Model for Computer Generated Forces", *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, March 17-19 1993, pp. 335-345.

Landweer, P. (1994a). "Integration of CGF with Fielded Equipment Using DIS", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 59-62.

Landweer, P. (1994b). "Use of an Interactive CGF as a Networked Partner with Local Simulators", *Proceedings of the Fourth Conference on Computer Generated Forces and*

Behavioral Representation, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 63-65.

Landweer, P. (1994c). "Integration of CGF with Fielded Equipment Using DIS", *Proceedings of the Fifth Annual Conference on AI, Simulation, and Planning in High Autonomy Systems*, Gainesville FL, December 7-9 1994, pp. 262-268.

Lankester, H. (1995). "Multi-Application Command Agents", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 169-177.

Lattimore, P. J. and Riecken, M. E. (1993). "A Virtual Battlespace Language", *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, March 17-19 1993, pp. 87-93.

Le, H. T. (1990). "On the Role of Distributed AI in Large Scale Network Simulation", *Proceedings of the 12th Interservice/Industry Training Systems Conference*, Orlando FL, November 6-8 1990, pp. 241-246.

Le, H. T. (1991a). "Multiple Autonomous Combatants: Control and Navigation", *Proceedings of the 2nd Behavioral Representation and Computer Generated Forces Symposium*, Institute for Simulation and Training, Orlando FL, May 6-7 1991, pp. D 1-12.

Le, H. T., Phinney, S. E., and Seward, V. C. (1991b). "Semi-Automated Forces: A Behavioral Modeling Approach", *Proceedings of the 13th Interservice/Industry Training Systems Conference*, Orlando FL, December 2-5 1991, pp. 321-327.

Lee, D. T. (1980). "Two-Dimensional Voronoi Diagrams in the Lp-Metric", *Journal of the Association for Computing Machinery*, Vol. 27, No. 4, October 1980, pp. 604-618.

Lee, J. J. and Fishwick, P. A. (1994a). "Simulation-Based Planning for Computer Generated Forces", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 451-460.

Lee, J. J. and Fishwick, P. A. (1994b). "Real-Time Simulation-Based Planning for Computer Generated Forces Simulation", *Simulation*, The Society for Computer Simulation, Vol. 63, No. 5, November 1994, pp. 299-315.

Lee, J. J. and Fishwick, P. A. (1994c). "Incorporating Simulation-Based Models into Planning Systems", *Proceedings of the Fifth Annual Conference on AI, Simulation, and Planning in High Autonomy Systems*, Gainesville FL, December 7-9 1994, pp. 113-119.

Lehman, J. F., Dyke, J. V., Rubinoff, R. (1995). "Natural Language Processing for IFORs: Comprehension and Generation in the Air Combat Domain", *Proceedings of the Fifth Conference*

on Computer Generated Forces and Behavioral Representation, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 115-123.

Lewis, H. R. and Papadimitriou, C. H. (1981). *Elements of the Theory of Computation*, Prentice-Hall, Englewood Cliffs NJ 1981.

Li, X., Miller, D., Illing, M., Kenworthy, M., and Heinen, M. (1994). "Dynamic Terrain Database Design for Real Time Image Generation", *Proceedings of the 16th Interservice/Industry Training Systems and Education Conference*, Orlando FL, November 28-December 1 1994, pp. 6-3.

Lisle, C., Altman, M., Kilby, M., and Sartor, M. (1994). "Architectures for Dynamic Terrain and Dynamic Environments in Distributed Interactive Simulation", *Proceedings of the 10th Workshop on Standards for the Interoperability of Defense Simulations*, Institute for Simulation and Training, Orlando FL, March 14-18 1994, pp. 89-105.

Lockheed (1990). "ALBM Detailed Design Plan Update", *Technical Report LMSC F405841*, Lockheed Missiles and Space Company.

Longtin, M. J. (1994). "Cover and Concealment in ModSAF", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 239-247.

Longtin, M. J. and Megherbi, D. (1995). "Concealed Routes in ModSAF", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 305-313.

Loper, M. L., Thompson, J. R., and Williams, H. L. (1991). "Simulator Networking: What Can It Offer The Training Community?", *Military Simulation & Training*, Issue 4 1991, pp. 11-14.

Loper, M. L. and Petty, M. D. (1993). "Computer Generated Forces at the DIS Interoperability Demonstration", *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, March 17-19 1993, pp. 155-168.

Loper, M. L. and Petty, M. D. (1994). "Simulator Networking for Medical Simulation", *Proceedings of the 5th International Training Equipment Conference*, The Hague, The Netherlands, April 26-28 1994, pp. 412-419.

Loper, M. L. and Petty, M. D. (1995a). "Distributed Interactive Simulation and Emergency Management", *Proceedings of the 1995 Simulation MultiConference*, Simulation for Emergency Management, Society for Computer Simulation, Phoenix AZ, April 9-13 1995, pp. 326-321.

Loper, M. L. (1995b). "Introduction to distributed interactive simulation", *Distributed Interactive Simulation Systems for Simulation and Training in the Aerospace Environment*, SPIE Critical Review 58, Orlando FL, April 19-20 1995, pp. 3-15.

Madni, A. M., Ahlers, R., Chu, Y. (1987). "Knowledge-Based Simulation: An Approach to Intelligent Opponent Modeling for Training Tactical Decisionmaking", *Proceedings of the 9th Interservice/Industry Training Systems Conference*, Washington DC, November 30-December 2 1987, pp. 179-183.

Mall, H., Bimson, K., McCormack, J., and Ourston, D. (1995). "Command Entity Cognitive Behaviors for SAF and CGF", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 203-209.

Marshall, H., Anderson, C., Baran, T., Berggren, P., Bimson, K., Blanchard, D., Braudaway, W., Burch, B., Colon, J., Cosby, J., Glover, G., King, J., Ourston, D., and Watson, J. (1994). "Close Combat Tactical Trainer Semi-Automated Forces (SAF) Design Overview", *Presentation*, Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation, May 4 1994.

Marti, J. and Christophe, B. (1994). "Automated Path Planning for Simulation", *Proceedings of the Fifth Annual Conference on AI, Simulation, and Planning in High Autonomy Systems*, Gainesville FL, December 7-9 1994, pp. 122-128.

Maruichi, T., Uchiki, T., and Tokoro, M. (1987). "Behavioral Simulation Based on Knowledge Objects", *Proceedings of the European Conference on Object Oriented Programming*, Paris France, June 17-19 1987, pp. 213-222.

McDonald, L. B. and Wood, D. D. (1993). "Standards Development for Distributed Interactive Simulation: Engineering Change Proposal Number 2, Continued Development of SIGINT/EW Capabilities for Distributed Interactive Simulation", *Technical Proposal IST-PR-92-06*, Institute for Simulation and Training.

McEnany, B. R., Jacobs, I. M., and Fonda, G. R. (1993). "Close Combat Tactical Trainer (CCTT) Semi-Automated Forces (SAF) Combat Instruction Set (CIS) Development", *Unpublished report*, IBM Federal Systems, October 21 1993.

McEnany, B. R. and Marshall, H. (1994). "CCTT SAF Functional Analysis", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 195-207.

McKeown, Jr., D. M. (1990). "Some Research Issues for Advance Simulation Technologies", *Presentation (DARPA Conference on Behavioral Representation in Semi-Automated Forces)*, October 25 1990.

McMahon, P. E. and Meehl, D. W. (1991). "Software Metrics, Ada, and the B-2 ATD", *Proceedings of the 13th Interservice/Industry Training Systems Conference*, Orlando FL, December 2-5 1991, pp. 162-171.

Meliza, L. L. and Tan, S. C. (1991). "Application of the SIMNET Unit Performance Assessment System to After Action Review", *Proceedings of the 13th Interservice/Industry Training Systems Conference*, Orlando FL, December 2-5 1991, pp. 136-144.

Meliza, L. L. and Vaden, E. A. (1995). "Measuring Entity and Group Behaviors of Semi-Automated Forces", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 181-192.

Mengel, L. L. (1994). *ModSAF Summer Exercise (SUMEX-I) Final Report.*

Mitchell, J. S. B. (1988). "An Algorithmic Approach to Some Problems in Terrain Navigation", *Artificial Intelligence*, Vol. 37, 1988, pp. 171-201.

Mitchell, J. S. B. and Papadimitriou, C. H. (1991). "The Weighted Region Problem: Finding Shortest Paths Through a Weighted Planar Subdivision", *Journal of the Association for Computing Machinery*, Vol. 38, No. 1, pp. 18-71.

Mohn, H. L., Pratt, D. R., and McGhee, R. B. (1994). "Meta-Level C2/Mission Planning Tool for ModSAF", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 461-472.

Monday, P. and Perneski, J. (1995). "The Use of Automated Regression and VVA Testing in ModSAF, *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 193-196.

Moore, E. F. (1959). "The Shortest Path Through a Maze", *Proceedings of the International Symposium on Switching Theory*, Harvard University Press, Vol. 1, pp. 282-292.

Moore, C. G. and Whiteley, D. I. (1995). "Generic Architectures for Intelligent Computer Generated Forces", *Proceedings of the 6th International Training Equipment Conference*, The Hague, The Netherlands, April 25-27 1995, pp. 185-193.

Moore, M. B., Gieb, C., and Reich, B. D. (1995). "Planning for Reactive Behaviors in Hide and Seek", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 345-352.

Moshell, J. M., Hughes, C. E., and Petty, M. D. (1989). "Constraints as a Specification Mechanism for Automated Opposing Forces in Networked Simulators", *Proceedings of the Interactive Networked Simulation for Training Conference*, Institute for Simulation and Training, Orlando FL, April 26-27 1989, pp. 84-90.

Moshell, J. M., Blau, B., Li, X., and Lisle, C. (1994). "Dynamic Terrain", *Simulation*, Vol. 62, No. 1, January 1994, pp. 29-42.

Nagy, G. and Wagle, S. (1979a). "Geographic Data Processing", *ACM Computer Surveys*, Vol. 11, No. 2, June 1979, pp. 139-181.

Nagy, G. and Wagle, S. (1979b). "Approximation of Polygonal Maps by Cellular Maps", *Communications of the ACM*, Vol. 22, No. 9, September 1979, pp. 518-525.

NASA (1993). "Potential-Field Scheme for Avoidance of Obstacles by a Robot", *NASA Tech Briefs*, Vol. 17, No. 4, April 1993, p. 80.

Nelms, D. W. (1988). "SIMNET II", *National Defense*, July/August 1988, pp. 68-69.

Nelson, R. S. (1995). "A Review of Terrain Database Correlation Test Results for the 1994 I/ITSEC DIS Demonstration", *Proceedings of the 12th DIS Workshop on Standards for the Interoperability of Defense Simulations*, Institute for Simulation and Training, Orlando FL, March 13-17 1995, pp. 405-410.

Nielsen, P. E. (1995). "Intelligent Computer Generated Forces for Command and Control", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 211-218.

Nilsson, N. J. (1980). *Principles of Artificial Intelligence*, Tioga, Palo Alto CA, 1980.

O'Byrne, E. C. (1993). "Dismounted Infantry: Indispensible to the Virtual Battlefield", *Proceedings of the 15th Interservice/Industry Training Systems and Education Conference*, Orlando FL, November 29-December 2 1993, pp. 783-791.

O'Dúnlaing, C. and Yap, C. (1985). "A Retraction Method for Planning the Motion of a Disc", *Journal of Algorithms*, Vol. 6, 1985, pp. 104-111.

O'Rourke, J. (1987). *Art Gallery Theorems and Algorithms*, Oxford University Press, Oxford UK, 1987.

O'Rourke, J. (1994). *Computational Geometry in C*, Cambridge University Press, Cambridge UK, 1994.

OTA (1994). "Virtual Reality and Technologies for Combat Simulation", *Background Paper 301-804 814/27414*, Office of Technology Assessment, Congress of the United States, 1994

Ok, D. K. (1989). "A Computer Simulation Study of a Sensor-Based Heuristic Navigation for Three-Dimensional Rough Terrain with Obstacles", *M.S. Thesis*, U.S. Naval Postgraduate School, June 1989.

Okabe, A., Boots, B., and Sugihara, K. (1992). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, John Wiley, Chichester England, 1992.

Oswalt, I. (1993). "Current Applications, Trends, and Organizations in U.S. Military Simulation and Gaming", *Simulation & Gaming*, Vol. 24, No. 2, June 1993, pp. 153-189.

Ourston, D. and Bimson, K. (1994). "Integrating Heterogenous Knowledge Processes in the SAF Behaviors Implementation", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 333-342.

Ourston, D., Blanchard, D., Chandler, E., and Loh, E. (1995). "From CIS to Software", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 275-285.

Page, I. and Kendall, G. (1995). "An Automated CBS OPFOR", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 149-158.

Pandari, A. and Schaper, G. A. (1995). "Terrain Reasoning by Intelligent Player", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 367-373.

Papadopoulou, E. and Lee, D. T. (1994). "Shortest paths in a simple polygon in the presence of "forbidden" vertices", *Proceedings of the Sixth Canadian Conference on Computational Geometry*, Saskatoon Saskatchewan Canada, August 2-6 1995, pp. 110-115.

Papelis, Y. E. (1994). "Terrain Modeling on High-Fidelity Ground Vehicle Simulation", *Proceedings of the Fifth Annual Conference on AI, Simulation, and Planning in High Autonomy Systems*, Gainesville FL, December 7-9 1994, pp. 48-54.

Parra, F. R., Franceschini, R. W., Jackson, L. A., Vemulapati, J., and Xuxia, Y. (1994). "Parametric Fireteam Makeup for Semi-Automated Forces Dismounted Infantry", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 169-174.

Parsons, J. D. (1994). "Using Fuzzy Logic Control Technology to Simulate Human Decision-Making in Warfare Models", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 519-529.

Petty, M. D. (1988a). "Tactical Simulation in an Object Oriented Animated Graphics Environment", *M.S. Thesis*, Computer Science Department, University of Central Florida, April 25 1988.

Petty, M. D., Moshell, J. M., and Hughes, C. E. (1988b). "Tactical Simulation in an Object-Oriented Animated Graphics Environment", *Simuletter*, Vol. 19, No. 2, June 1988, pp. 31-46.

Petty, M. D., Frederick, T. J., and Moshell, J. M. (1990). "Experiments in routing an autonomous land vehicle with a weakly inductive learning algorithm", *Proceedings of the Third Florida Artificial Intelligence Research Symposium*, Cocoa Beach FL, April 3-6 1990, pp. 159-163.

Petty, M. D., Karr, C. R., Van Brackle, D. R., Cross, D. D., Franceschini, R. W., and Gonzalez, G. L. (1991). "Functional Specification and Implemented Capabilities of the IST Semi-Automated Forces Dismounted Infantry System", *Technical Report IST-TR-91-20*, Institute for Simulation and Training, 1991.

Petty, M. D., Campbell, C. E., Franceschini, R. W., Provost, M. H., and Karr, C. R. (1992a). "Preliminary Investigations into Efficient Line of Sight Determination in Polygonal Terrain", *Technical Report IST-TR-92-5*, Institute for Simulation and Training, February 28 1992.

Petty, M. D., Campbell, C. E., Franceschini, R. W., and Provost, M. H. (1992b). "Efficient Line of Sight Determination in Polygonal Terrain", *Proceedings of the 1992 IMAGE VI Conference*, Phoenix AZ, July 14-17 1992, pp. 239-252.

Petty, M. D. (1992c). "Computer Generated Forces in Battlefield Simulation", *Proceedings of the Southeastern Simulation Conference 1992*, The Society for Computer Simulation, Pensacola FL, October 22-23 1992, pp. 56-71.

Petty, M. D., Karr, C. R., and Smith, S. H. (1992d). "Semi-Automated Forces Dismounted Infantry in the SIMNET Battlefield", *Proceedings of the 14th Interservice/Industry Training Systems and Education Conference*, San Antonio TX, November 2-5 1992, pp. 314-321.

Petty, M. D., Smith, S. H., Reece, D. A., Karr, C. R., Hull, R. D., and Mullally, D. E. (1993). "A Design Study of Behavior Specification Languages for Autonomous Entities in Battlefield Simulation", *Technical Report IST-TR-93-18*, Institute for Simulation and Training, June 15 1993.

Petty, M. D. and Franceschini, R. W. (1994a). "Dismounted Infantry in Distributed Interactive Simulation", *Proceedings of the Individual Combatant Modeling and Simulation Symposium (INCOMSS-94)*, Ft. Benning GA, February 15-17 1994, pp. 288-305.

Petty, M. D. and Van Brackle, D. R. (1994b). "Reconnaissance Planning in Polygonal Terrain", *Proceedings of the 5th International Training Equipment Conference*, The Hague, The Netherlands, April 26-28 1994, pp. 314-327.

Petty, M. D. (1994c). "The Turing Test as an Evaluation Criterion for Computer Generated Forces", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 107-116.

Petty, M. D. and West, P. A. (1995a). "Plowshares: Applying a Military Constructive Simulation Model to Emergency Management Training", *Proceedings of the 1995 Simulation MultiConference*, Simulation for Emergency Management, Society for Computer Simulation, Phoenix AZ, April 9-13 1995, pp. 326-331.

Petty, M. D. (1995b). "Computer generated forces in Distributed Interactive Simulation", *Distributed Interactive Simulation Systems for Simulation and Training in the Aerospace Environment*, SPIE Critical Review 58, Orlando FL, April 19-20 1995, pp. 251-280.

Petty, M. D. (1995c). "Computer Generated Forces and the Turing Test", *Proceedings of the 6th International Training Equipment Conference,* The Hague, The Netherlands, April 25-27 1995, pp. 195-204.

Petty, M. D. and Franceschini, R. W. (1995d). "Disaggregation Overload and Spreading Disaggregation in Constructive+Virtual Linkages", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 103-111.

Petty, M. D., Slepow, M. P., and West, P. D. (1995e). "CGF Opportunities in Plowshares", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 337-334.

Petty, M. D. (1995f). "Case Studies in Verification, Validation, and Accreditation for Computer Generated Forces", *Proceedings of the ITEA "Modeling & Simulation: Today and Tomorrow" Workshop*, Las Cruces NM, December 11-14 1995.

Pickett, H. K. and Petty, M. D. (1995). "Report on the State of Computer Generated Forces 1994", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, 549-558.

Pope, A. R. (1989). "The SIMNET Network and Protocols", *Report No. 7102*, BBN Systems and Technologies Corporation, July 1989.

Pope, C. N., Vuong, M., Moore, R. G., and Cowser, S. S. (1995a). "Cost-Effective Interoperable Distributed Interactive Simulation for Large-Scale Tactical Warfare Simulation", *Proceedings of the 6th International Training Equipment Conference*, The Hague, The Netherlands, April 25-27 1995, pp. 129-142.

Pope, C. N., Vuong, M., Moore, R. G., and Cowser, S. S. (1995b). "A Whole New CCTT World", *Military Simulation & Training*, Issue 5 1995, pp. 6-19.

Porch, D. (1991). *The French Foreign Legion*, Harper Collins, New York NY, 1991.

Potomac Systems Engineering (1990). "Report of the Evaluation of the Representation of Semi-Automated Forces (SAF) in the SIMNET Model", *Potomac Systems Engineering*, Annandale VA, July 1990.

Powell, D. R. (1987). "Computer Based Terrain Analysis for Operational Planning", *Proceedings of 1987 IEEE International Conference on Systems, Man, and Cybernetics*, Institute of Electrical and Electronics Engineers, pp. 1022-1026.

Powell, D. R. and Storm, G. (1988a). "Avenue of Approach Generation", *Proceedings of the U.S. Army Symposium on Artificial Intelligence Research for Exploitation for the Battlefield Environment*, El Paso TX, November 15-16 1988, pp. 203-210.

Powell, D. R., Wright, J. C., Slentz, G., and Kundsen, P. (1988b). "Representations to Support Reasoning on Terrain", *Proceedings of the U.S. Army Symposium on Artificial Intelligence Research for Exploitation for the Battlefield Environment*, El Paso TX, November 15-16 1988, pp. 212-222.

Powell, D. R. (1989). "Object Oriented Terrain Analysis", *LA-UR-89-3665*, Los Alamos National Laboratory, October 1989.

Powell, D. R. and Hutchinson, J. L. (1993). "Eagle II: A Prototype for Multi-Resolution Combat Modeling", *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, March 17-19 1993, pp. 221-230.

Pratt, D. R., Bhargava, H. K., Culpepper, M., and Locke J. (1994a). "Collaborative Autonomous Agents in the NPSNET Virtual World", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 177-186.

Pratt, D. R., Barham, P. T., Locke, J., Zyda, M. J., Eastman, B., Moore, T., Biggers, K., Douglass, R., Jacobsen, S., Hollick, M., Granieri, J., Ko, H., Badler, N. I. (1994b). "Insertion of an Articulated Human into a Networked Virtual Environment", *Proceedings of the Fifth Annual Conference on AI, Simulation, and Planning in High Autonomy Systems*, Gainesville FL, December 7-9 1994, pp. 84-90.

Pratt, D. R., McAndrews, G., and McGhee, R. (1995a). "Autonomous Agent Interactions in ModSAF", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 219-229.

Pratt, D. R., Mohn, H., and McGhee, R. (1995b). "Implementation Of A Tactical Order Generator For Computer Generated Forces", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 287-291.

Preparata, F. P. (1977). "Steps into computational geometry", *Technical Report*, Coordinated Science Laboratory, University of Illinois, 1977.

Preparata, F. P. and Shamos, M. I. (1988). *Computational Geometry: An Introduction*, 2nd Edition, Springer-Verlag, New York NY, 1988.

Purnell, T., Pearson, R., Kendall, T., Kaste, V., Zhou, W., and Qiu, M. (1995). "An Architecture for Dynamic Environments", *Proceedings of the 12th DIS Workshop on Standards for the Interoperability of Defense Simulations*, Institute for Simulation and Training, Orlando FL, March 13-17 1995, pp. 437-444.

Rajput, S., Craft, M. A., Breneman, L. J., Petty, M. D., Holly, T. P., and Ng, J. J. (1994a). "Intervisibility Heuristics for Computer Generated Forces", *Technical Report IST-TR-94-22*, Institute for Simulation and Training, May 16 1994.

Rajput, S. and Karr, C. (1994b). "Unit Route Planning", *Technical Report IST-TR-94-42*, Institute for Simulation and Training, December 21 1994.

Rajput, S., Karr, C. R., Petty, M. D., and Craft, M. A. (1995a). "Intervisibility heuristics for CGF", *Distributed Interactive Simulation Systems for Simulation and Training in the Aerospace Environment*, SPIE Critical Review 58, Orlando FL, April 19-20 1995, pp. 299-327.

Rajput, S., Karr, C. R., Petty, M. D., and Craft, M. A. (1995b). "Intervisibility Heuristics for Computer Generated Forces", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 451-464.

Rajput, S. and Karr, C. R. (1995b). "Cooperative Behavior in ModSAF", *Technical Report IST-CR-95-35*, Institute for Simulation and Training, November 20 1995.

Raytheon (1994). "System Specification for the Corps Level Computer Generated Forces", *CDRL Sequence No. A0011*, Contract No. DACA76-93-D-0007, Raytheon Systems Development Company, July 22 1994.

Reece, D. A. (1993). "Execution Control for CPU-Sharing Agents", *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, March 17-19 1993, pp. 75-84.

Reece, D. A. (1994a). "Extending DIS for Individual Combatants", *Proceedings of the Fifth Annual Conference on AI, Simulation, and Planning in High Autonomy Systems*, Gainesville FL, December 7-9 1994, pp. 91-97.

Reece, D. A. (1994b). "The Architecture of the CGF Testbed", *Unpublished report*, Institute for Simulation and Training, October 5 1994.

Reybaz, J. (1932). *Le 1er Mystérieux: Souvenirs de guerre d'un légionnaire suisse*, André Barry, Paris France, 1932.

Rice, J. A. (1995). *Mathematical Statistics and Data Analysis*, 2nd Edition, Duxbury Press, Belmont CA, 1995.

Roback, C. A. and Lee, R. (1995). "Terrain Database Generation - Methodology and Procedure", *Proceedings of the 12th DIS Workshop on Standards for the Interoperability of Defense Simulations*, Institute for Simulation and Training, Orlando FL, March 13-17 1995, pp. 301-307.

Robasky, K. J., Schaffer, R., Wilbert, D., and Haque, S. (1995). "A Toolkit for Cost Estimating Real and Theoretical Protocols", *Proceedings of the 12th DIS Workshop on Standards for the Interoperability of Defense Simulations*, Institute for Simulation and Training, Orlando FL, March 13-17 1995, pp. 477-479.

Roos, T. and Noltemeier, H. (1991). "Dynamic Voronoi Diagrams in Motion Planning", *Computational Geometry: Methods, Algorithms, and Applications*, Proceedings of the International Workshop on Computational Geometry, Springer-Verlag, Bern Switzerland, March 1991, pp. 227-236.

Root, E. D. and Karr, C. R. (1994). "Displaying Aggregate Units in the Virtual Environment", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 497-502.

Rosenbloom, P. S., Johnson, W. L., Jones, R. M., Koss, F., Laird, J. E., Lehman, J. F., Rubinoff, R., Schwamb, K. B., and Tambe, M. (1994). "Intelligent Automated Agents for Tactial Air Simulation: A Progress Report", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 69-78.

Rosenkrantz, D. J., Stearns, R. E., and Lewis, P. M. (1974). "Approximate Algorithms for the Traveling Salesperson Problem", *Proceedings of the 15th Annual Symposium on Switching and Automata Theory*, New Orleans LA, 1974, pp. 34-42.

Rubinoff, R. and Lehman, J. F. (1994). "Natural Language Processing in an IFOR Pilot", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 97-104.

Salisbury, M. and Tallis, H. (1993). "Automated Planning and Replanning for Battlefield Simulation", *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, March 17-19 1993, pp. 243-254.

Salisbury, M. R., Booker, L. B., Seidel, D. W., and Dahmann, J. S. (1995). "Implementation of Command Forces (CFOR) Simulation", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 423-430.

Samat, H. (1984). "The Quadtree and Related Hierarchical Data Structures", *Computing Surveys*, Vol. 16, No. 2, 1984, pp. 187-260.

Sansom, R. and Darling, D. (1993). "A Radar Altitude and Line of Sight Attachment", *Proceedings fo the AIAA Flight Simulation Technologies Conference*, Monterey CA, August 9-11 1993, pp. 243-250.

Sargeant, J. M. and Schuerger, J. M. (1990). "A Graphically Oriented Automated Knowledge Acquisition Tool", *Proceedings of the Third Florida Artificial Intelligence Research Symposium*, Cocoa Beach FL, April 3-6 1990, pp. 107-111.

Schaffer, R. L. (1994). "Environmental Extensions to ModSAF", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 17-23.

Schaper, G. A., Pandari, S., and Singh, M. (1994). "Lookahead Limits of Intelligent Player", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 401-410.

Schiavone, G. A., Nelson, R. S., and Goldiez, B. (1994). "Statistical Certification of Terrain Databases", *Proceedings of the 16th Interservice/Industry Training Systems and Education Conference*, Orlando FL, November 28-December 1 1994, pp. 4-9.

Schiavone, G. A., Nelson, R. S., and Hardis, K. C. (1995). "Interoperability Issues for Terrain Databases in Distributed Interactive Simulation", *Distributed Interactive Simulation Systems for Simulation and Training in the Aerospace Environment*, SPIE Critical Review 58, Orlando FL, April 19-20 1995, pp. 89-118.

Schmitt, J. F. and Powell, A. W. (1988). *Ground Combat Operations*, United States Marine Corps Operational Handbook 6-1, Marine Corps Combat Development Command, Quantico VA.

Schricker, S. A., Franceschini, R. W., Petty, M. D., and Tolley, T. R. (1995a). "Terrain Avoidance for CGF Helicopters", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 315-325.

Schricker, S. A., Tolley, T. R., and Franceschini, R. W. (1995b). "Benchmarking and Optimization of the IST CGF Testbed", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995 pp. 465-476.

Schwamb, K. B., Koss, F. V., and Keirsey, D. (1994). "Working with ModSAF: Interfaces for Programs and Users", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 395-399.

Seidel, R. (1982). "The complexity of Voronoi diagrams in higher dimensions", *Proceedings of the 20th Allerton Conference on Communications, Control, and Computing*, pp. 94-95.

Shamos, M. and Hoey, D. (1975). "Closest point problems", *Proceedings of the 16th Annual IEEE Symposium on the Foundations of Computer Science*, Institute of Electrical and Electronics Engineers, pp. 151-162.

Shapiro, S. C. (1992). "The Turing Test and The Economist", *SIGART Bulletin*, Vol. 3, No. 4, October 1992, pp. 10-11.

Sherman, R. H. (1994). "Using Computer Generated Forces to Support Cooperative Mission Planning", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 37-49.

Siksik, D. N. (1993). "Intelligent Computer Generated Forces Through Expert Systems", *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, March 17-19 1993, pp. 3-10.

Slack, M. G. (1989). "Space-Time Modeling Using Environmental Constraints in a Mobile Robot System", *Proceedings of SPIE '89 Sensor Fusion II: Human and Mechanical Strategies*, Vol. 1198, Philadelphia PA, November 6-9 1989, pp. 520-528.

Smith, J. E. (1992a). "Compact Terrain Database Library User Manual and Report", *ODIN SAF Documentation*, March 1992.

Smith, S. H., Karr, C. R., Petty, M. D., Franceschini R. W., and Watkins, J. E. (1992b). "The IST Computer Generated Forces Testbed", *Technical Report IST-TR-92-7*, Institute for Simulation and Training, February 28 1992.

Smith, S. H. and Petty, M. D. (1992c). "Controlling Autonomous Behavior in Real-Time Simulation", *Proceedings of the Southeastern Simulation Conference 1992*, The Society for Computer Simulation, Pensacola FL, October 22-23 1992, pp. 27-40.

Smith, S. H. (1993). "ILLISH: Intermediate Level Language, Interpreted, for Script Handling", *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, March 17-19 1993, pp. 95-106.

Smith, J. E. (1994). "Near-term Movement Control in ModSAF", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 249-260.

Smith, J. E., Russo, K. L., and Schuette, L. C. (1995a). "Prototype Multicast IP Implementation in ModSAF", *Proceedings of the 12th DIS Workshop on Standards for the Interoperability of Defense Simulations*, Institute for Simulation and Training, Orlando FL, March 13-17 1995, pp. 175-178.

Smith, J. E. (1995b). "Recent Developments in ModSAF Terrain Representation", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 375-381.

SOGITEC (1989). "An Expert System for Tank Platoon Behavior", *Interactions*, No. 2, June 1989, pp. 6-7.

Stanzione, T. (1989). "Terrain Reasoning in the SIMNET Semi-Automated Forces System", *Geo'89 Symposium on Geographical Information Systems for Command and Control*, SHAPE Technical Centre, The Hague, The Netherlands, October 1989.

Stanzione, T., Smith, J. E., Brock, D. L., Mar, J. M. F., and Calder, R. B. (1993). "Terrain Reasoning in the ODIN Semi-Automated Forces System", *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, March 17-19 1993, pp. 317-326.

Stanzione, T. (1994). "Suitability of the Standard Simulator Database Interchange Format for Representation of Terrain for Computer Generated Forces", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 231-238.

Stanzione, T., Chamberlin, F., Evans, A., and Buettner, C. (1995). "Integrated Computer Generated Forces Terrain Database", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 399-409.

Sterling, B. (1994). "Conditions for Using Performance on Simulations as 'Gates' for Live Fire and Maneuver Training", *Proceedings of the 5th International Training Equipment Conference*, The Hague, The Netherlands, April 26-28 1994, pp. 57-63.

Stevens, S. M. (1989). "Intelligent Interactive Video Simulation of a Code Inspection", *Communications of the ACM*, Vol. 32, No. 7, July 1989, pp. 832-843.

Stober, D. R., Kraus, M. K., Foss, W. F., Franceschini, R. W., and Petty, M. D. (1995). "Survey of Constructive+Virtual Models", *Proceedings of the Fifth Conference on Computer Generated*

Forces and Behavioral Representation, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 93-102.

Sun Tzu. (600BC). *The Art of War*, Translated by Cleary, T., Shambhala, Boston MA, 600BC.

Sundaram, R., McArthur, D., and Devarajan, V. (1994). "Incremental Real Time Delaunay Triangulation for Terrain Skin Generation", *Proceedings of the 16th Interservice/Industry Training Systems and Education Conference*, Orlando FL, November 28-December 1 1994, pp. 6-4.

Tallis, H. (1993). "Flexible Control of Military Subordinates in a Reactive Simulation", *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, March 17-19 1993, pp. 109-120.

Tambe, M. and Rosenbloom, P. S. (1994). "Event Tracking in Complex Multi-Agent Environments", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 473-484.

Tambe, M., Schwamb, K., and Rosenbloom, P. S. (1995a). "Building Intelligent Pilots for Rotary Wing Aircraft", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 39-44.

Tambe, M. and Rosenbloom, P. S. (1995b). "Agent Tracking in Complex Multi-agent Environments: New Results", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 125-133.

Teng, Y. A., DeMenthon, D., and Davis, L. S. (1992). "Stealth Terrain Navigation With Bounding Overwatch", *Proceedings of the DARPA Image Understanding Workshop*, Morgan-Kaufmann, San Diego CA, January 26-29 1992, pp. 979-989.

Thomas, J. G. (1995a). "Verification and Validation of Modular Semi-Automated Forces (ModSAF) in Support of A2ATD Experiments", *Proceedings of the 12th DIS Workshop on Standards for the Interoperability of Defense Simulations*, Institute for Simulation and Training, Orlando FL, March 13-17 1995, pp. 359-361.

Thomas, J. G. (1995b). "Verification and Validation of Modular Semi-Automated Forces (ModSAF) in Support of A2ATD Experiments", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 197-200.

Thorpe, J. A. (1987). "The New Technology of Large Scale Simulator Networking: Implications for Mastering the Art of Warfighting", *Proceedings of the 9th Interservice/Industry Training Systems Conference*, Orlando FL, November 30-December 2 1987, pp. 492-501.

Trott, K. C. and Langevin, T. (1995). "DIS Terrain Database Exchange Format Analysis", *Proceedings of the 12th DIS Workshop on Standards for the Interoperability of Defense Simulations*, Institute for Simulation and Training, Orlando FL, March 13-17 1995, pp. 105-117.

Turing, A. M. (1950). "Computing machinery and intelligence", *Mind*, Vol. 59 No. 236, October 1950, pp. 433-460.

U.S. Army (1988). *Tank and Mechanized Infantry Company Team*, Field Manual 71-1, U.S. Army, November 22 1988.

U.S. Army (1991). *The Soviet Army: Troops, Organization, and Equipment*, Field Manual 100-2-3, U.S. Army, June 6 1991.

Vaden, E. A., Meliza, L., and Johnson, W. R. (1994). "Using the Unit Performance Assessment System (UPAS) to Measure Modular Semi-Automated Force Behavior", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 117-128.

Van Brackle, D. R., Gouge, C. D., Hull, R. D., and Petty, M. D. (1993a). "Terrain Reasoning for Reconnaissance Planning in Polygonal Terrain Cultural Features", *Technical Report IST-TR-93-03*, Institute for Simulation and Training, 1993.

Van Brackle, D. R., Petty, M. D., Gouge, C. D., and Hull, R. D. (1993b). "Terrain Reasoning for Reconnaissance Planning in Polygonal Terrain", *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, March 17-19 1993, pp. 285-305.

van Lent, M. and Wray, R. (1994). "A Very Low Cost System for Direct Human Control of Simulated Vehicles", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 79-85.

Vanzant-Hodge, A. F., Smith, S. H., Cheung, S., and Humphrey, D. (1994a). "Testing a CGF in a DIS Environment", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 141-146.

Vanzant-Hodge, A., Cheung, S., and Smith, S. (1994b). "Testing Conformance for Distributed Interactive Simulation (DIS) Standards", *Proceedings of the 16th Interservice/Industry Training Systems and Education Conference*, Orlando FL, November 28-December 1 1994, pp. 4-1.

Vrablik, R. and Wilbert, D. (1993). "The Use of Semi-Automated Forces to Simulate a 10,000 Entity Exercise", *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, March 17-19 1993, pp. 169-178.

Vrablik, R. and Richardson, W. (1994). "Benchmarking and Optimization of ModSAF", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 25-33.

Vrba, J. A. and Herrera, J. A. (1988). "Improved Expert System Performance Through Knowledge Shaping", *Proceedings of the U.S. Army Symposium on Artificial Intelligence Research for Exploitation for the Battlefield Environment*, El Paso TX, November 15-16 1988, pp. 293-301.

Wallich, P. (1991). "Silicon Babies", *Scientific American*, December 1991, pp. 124-134.

Warren, R., Crowe, M., Shillcutt, D. (1995). "Bi-Directional Technology Transfer Between Government Applications of Computer Generated Agents and Commercial Entertainment", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 329-335.

Watkins, J. and Provost, M. (1994). "Design of Terrain Reasoning Database for CCTT", *Proceedings of the Fifth Annual Conference on AI, Simulation, and Planning in High Autonomy Systems*, Gainesville FL, December 7-9 1994, pp. 62-68.

Watkins, J. (1995). "Terrain Capabilities in CCTT", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 9-11 1995, pp. 411-419.

Weaver, W. B. (1993). "Behavioral Representation Completeness in Computer Generated Forces Implementations", *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, March 17-19 1993, pp. 375-386.

Weaver, W. B. (1994). "Simulating Generic Military Decision Making with an Empirically-Trained Neural Network", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 531-540.

Webber, B. and Badler, N. (1993). "Virtual Interactive Collaborators for Simulation and Training", *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, March 17-19 1993, pp. 199-205.

Weiss, J. M. and Korba, R. E. (1991). "Advantages of an Object-Oriented Design Approach to the Simulation of Leadship Effects", *Proceedings of the 13th Interservice/Industry Training Systems Conference*, Orlando FL, December 2-5 1991, pp. 314-320.

Werkheiser, A. (1991). "Automated Terrain Reasoning", *Unpublished report*, U.S. Army Topographic Engineering Center, 1991.

Wever, P. and Lang, E. (1989). "SIMNET Database Interchange Specification", *Report No. 7108*, BBN Systems and Technologies Corporation, July 1989.

Winston, P. H. (1984). *Artificial Intelligence*, Addison-Wesley, Reading MA, 1984.

Wise, B. P., Miller, D., and Ceranowicz, A. Z. (1991). "A Framework for Evaluating Computer Generated Forces", *Proceedings of the 2nd Behavioral Representation and Computer Generated Forces Symposium*, Institute for Simulation and Training, Orlando FL, May 6-7 1991, pp. H 1-7.

Wood, D. D. and Petty, M. D. (1994). "Development of Signal Intelligence/Electronic Warfare Capabilities in a Computer Generated Forces Testbed", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando FL, May 4-6 1994, pp. 365-371.

Wood, D. D. and Petty, M. D. (1995). "Electronic warfare and DIS", *Distributed Interactive Simulation Systems for Simulation and Training in the Aerospace Environment*, SPIE Critical Review 58, Orlando FL, April 19-20 1995, pp. 179-194.

Wright, J. C. and Powell, D. R. (1990). "Artificial Intelligence Technologies Applied to Terrain Analysis", *Report LA-UR-90-123*, Los Alamos National Laboratory, June 1990.

Wysocki, F. and Fowlkes, D. (1994). "Team Target Engagement Simulator (TTES) Advanced Technology Demonstration", *Proceedings of the Individual Combatant Modeling and Simulation Symposium 1994 (INCOMSS-94)*, Ft. Benning GA, February 15-17 1994, pp. 144-190.

Yiu, S. M. and Choi, A. (1994). "Edge Guards on a Fortress", *Proceedings of the Sixth Canadian Conference on Computational Geometry*, Saskatoon Saskatchewan Canada, August 2-6 1995, pp. 296-301.

Zhu, B. (1994). "Intersection detection and computation of Manhatten terrains", *Proceedings of the Sixth Canadian Conference on Computational Geometry*, Saskatoon Saskatchewan Canada, August 2-6 1995, pp. 256-262.

Zink, W. E. and Neebe, J. M. (1991). "The Challenges of Developing a Real-Time Environment in Ada", *Proceedings of the 13th Interservice/Industry Training Systems Conference*, Orlando FL, December 2-5 1991, pp. 19-34.

Zisman, M. D. (1978). "Use of Production Systems for Modeling Asynchronous, Concurrent Process", in Waterman, D. A. and Hayes-Roth, F. (Eds.), *Pattern-Directed Inference Systems*, Academic Press, New York NY, pp. 53-68.

Zvolanek, B. and Dillard, D. E. (1992). "Database Correlation Testing for Simulation Environments", *Proceedings of the 14th Interservice/Industry Training Systems and Education Conference*, San Antonio TX, November 2-4 1992, pp. 867-875.

Zvolanek, B., Dillard, D. E., Stewart, J. R., and Baumann, E. W. (1993). "Quantitative Correlation Testing from DoD Project 2851 Standard Simulator Data Bases", *Proceedings of the 15th Interservice/Industry Training Systems and Education Conference*, Orlando FL, November 29-December 2 1993, pp. 829-836.

Zyda, M. J., Pratt, D. J., Monahan, J. G., and Wilson, K. P. (1992). "NPSNET: Constructing a 3D Virtual World", *Proceedings of the 1992 Symposium on Interactive 3D Graphics*, March 29-April 1 1992.

8. Appendices

8.1 List of acronyms and abbreviations

A2ATD	Anti-Armor Advanced Technology Demonstration
AAM	Air-to-Air Missile
ACBM	Action/Cognition Behavior Model
ACETEF	Air Combat Environment Test and Evaluation Facility
ACM	Association for Computing Machinery
ADST	Advanced Distributed Simulation Technology
AF	Automated Force
AI	Artificial Intelligence
AMSAA	Army Materiel Systems Analysis Activity
ARL	Army Research Laboratory
ARPA	Advanced Research Projects Agency
ATGM	Anti-Tank Guided Missile
ATR	Anti-Tank Rocket
BBN	Bolt, Beranek, and Newman
BBS	Brigade Battle Simulation
BDS-D	Battlefield Distributed Simulation - Developmental
CASTFOREM	Combined Arms Support and Task Force Evaluation Model
CATT	Combined Arms Tactical Training
CBS	Corps Battle Simulation
CCTT	Close Combat Tactical Training
CFOR	Command Forces
CGF	Computer Generated Forces
cm	Centimeter
CTDB	Compact Terrain Database
DARPA	Defense Advanced Research Projects Agency
DCEL	Doubly Connected Edge List
DI	Dismounted Infantry
DIS	Distributed Interactive Simulation
DTED	Digital Terrain Elevation Data
FACS	Feature Attribute Coding Standard
FWA	Fixed Winged Aircraft
FZD	Fire Zone Defense
GIS	Geographic Information System
HDI	High Detail Input/Output
HOBP	Hasty Occupy Battle Position
IBM	International Business Machines
ICTDB	Integrated Computer Generated Forces Terrain Database
IDS	Iowa Driving Simulator
iff	If and only if
IFOR	Intelligent Forces
IFV	Infantry Fighting Vehicle

IG	Image Generator
I/ITSEC	Interservice/Industry Training Systems and Equipment Conference
ILLISH	Intermediate Level Language, Interpreted, for Script Handling
IOT&E	Initial Operations Test and Evaluation
IP	Intelligent Pilot or Intelligent Player
IST	Institute for Simulation and Training
JPL	Jet Propulsion Laboratory
Km	Kilometer
LADS	Loral Advanced Distributed Simulation
LLNL	Lawrence Livermore National Laboratory
LOS	Line of Sight
m	Meter
ModSAF	Modular Semi-Automated Forces
MRTDB	Model Reference Terrain Database
NASA	National Aeronautics and Space Administration
NE	Northeast
NW	Northwest
OCOKA	Observation and fields of fire, Cover and concealment, Obstacles, Key terrain, and Avenues of approach
PDU	Protocol Data Unit
RWA	Rotary Winged Aircraft
SAFDI	Semi-Automated Forces Dismounted Infantry
SAF	Semi-Automated Forces
SAFOR	Semi-Automated Forces
SAM	Surface-to-Air Missile
SDBF	Simulator Database Facility
SE	Southeast
SIF	SSDB Interchange Format
SIMNET	Simulator Networking
SME	Subject Matter Expert
SSDB	Standard Simulator Database
STOW	Synthetic Theater of War
STRICOM	U.S. Army Simulation, Training, and Instrumentation Command
SW	Southwest
SWEG	Simulated Warfare Environment Generator
TDB	Terrain Database
TIN	Triangulated Irregular Network
TRP	Target Reference Point
UPAS	Unit Performance Assessment System
VBL	Virtual Battlespace Language
VMS	Vertical Motion Simulator
2D	Two dimensions, or two-dimensional
3D	Three dimensions, or three-dimensional
3DAR	Three-dimensional attack route

8.2 Some constructive CGF systems

Name: Red Adversarial Gaming Environment (RAGE)
Developer: MITRE
Simulation: AirLand Battle Management
Domain: Ground combat
Scope: Corps
Reference(s): [Tallis,1993]

Name: Adversarial Planner (AP)
Developer: MITRE
Simulation: Eagle
Domain: Ground and air combat
Scope: Division and brigade
Reference(s): [Salisbury,1993]

Name: System to Automate Force Control Actions (STAFCA)
Developer: Pathfinder Systems
Simulation: BBS
Domain: Ground combat
Scope: Brigade
Reference(s): [Jaszlics,1993]

Name: Virtual Commander (VCom)
Developer: Lawrence Livermore National Laboratory
Simulation: Joint Conflict Model (JCM), a Janus variant
Domain: Ground and air combat
Scope: Battalion
Reference(s): [Cunningham,1994]

Name: GeKnoFlexE
Developer: Defense Research Agency (UK)
Simulation: Corps Battle Simulation
Domain: Ground and air combat
Scope: Corps and division
Reference(s): [Cox,1994] [Page,1995] [Lankester,1995]

Name: SAFOR
Developer: Jet Propulsion Laboratory
Simulation: Corps Battle Simulation
Domain: Ground and air combat
Scope: Corps and division
Reference(s): [Gat,1993]