GENETIC ALGORITHMS BASED SCENARIO GENERATION FOR NETWORKED SIMULATIONS

J.W. Bala and B.K. Gogia
Datamat Systems Research, Inc.
8260 Greensboro Drive
MacLean, Virginia 22102

William Lee
Naval Air Systems Command
1421 Jefferson Drive Highway
Arlington, VA 22243-2160

ABSTRACT

To automatically generate simulated scenarios an algorithm is needed to search for the optimal subset of scenario parameters. For most simulated environments the scenario search space is complex and populated with discontinuities, multimodality, and noise. Complexity is especially evident in networked simulations, where the search space can be enormous. Some high-fidelity, large scale network simulation may require specifications of millions of parameters to describe all entities at a high level of resolution. In this paper we present the application of the Genetic Algorithms search technique for scenario optimization in network simulations. Genetic Algorithms as optimization and adaptation techniques, maintain a constant-sized population of candidate solutions known as individual scenarios. At each iteration, known as a generation, each scenario is evaluated and recombined with others on the basis of its overall quality or fitness in solving the simulation task. New scenarios are created using two main genetic recombination operators known as crossover and mutation.

ABOUT THE AUTHORS

Dr. Jerzy Bala is a Senior Scientist with Datamat Systems Research, Inc. He has an extensive background in the field of machine learning. Dr. Bale received his M.S. degree in Electrical and Computer Engineering and Ph.D. in Information Technology from George Mason University. He was a member of the machine learning group at George Mason University, where he pursued his Ph.D. In May 1993 he was awarded a postdoctoral research grant by the National Science Foundation in Computational Science and Engineering. His research has led to over 30 publications. He is a member of ACM, American Association for Artificial Intelligence, and ADPA.

Mr. B.K Gogia is a President of Datamat Systems Research, Inc. His expertise is in Software Reuse and BPR of large software development projects. He holds M.S. in Computer Science and MBA. He is a member of IEEE, IEEE-CS, Society for Enterprise Engineering, ACM, and an active member in ACM SIGs.

Mr. William D. S. Lee is the Naval Air Systems Command's training manager for the AH-1W Attack Helicopter and the UH-1N Utility Helicopter. He has sponsored numerous Small Business Innovative Research Phase I and Phase II efforts to include Alternative Motion Systems, Interactive Embedded Training, and Direct Manipulation Interface. Mr. Lee holds a BS in Systems Engineering from University of Virginia.

GENETIC ALGORITHMS BASED SCENARIO GENERATION FOR NETWORKED SIMULATIONS

J.W. Bala and B.K. Gogia
Datamat Systems Research, Inc.
8260 Greensboro Drive
MacLean, Virginia 22102

William Lee
Naval Air Systems Command
1421 Jefferson Drive Highway
Arlington, VA 22243-2160

1. INTRODUCTION

Current DIS exercises, while large in scale (e.g., REFORGER and ULCI FOCUS LENS), have had relatively few entities in their scenarios. This is primarily because the systems used to conduct the DIS exercises have limited capacity to support individual entities without over-loading their computational capacity. As such, it was feasible to manually plan and script the exercise for each player and entity. As the number of players increases this will no longer be an option. In currently planned exercises the number of entities will expand from a few hundred to tens or hundreds of thousands. As a result there is a pressing need for a new method of creating scenarios that will greatly simplify the process and reduce the vast amount of resources required to generate mission rehearsals.

This paper presents a high-level master-slave interface architecture for intelligent control of scenario generation in distributed interactive simulation. It addresses the complexity problem of current and future networked simulation systems. It is based on a principle that utilizes Genetic Algorithm search techniques. Genetic Algorithms efficiently search the scenario space and generate close to optimal scenarios. The generated scenario performance measure is calculated based on a distance measure to a predefined scenario classes.

2. SCENARIO GENERATION

A simulation may be regarded as composed of interacting objects that represent real-world entities. These objects can correspond to physical entities at various levels of abstraction. Interaction between objects is achieved by different execution mechanisms.

The entities, or simulated components, must have sufficient “identity” to know what they are (ball, aircraft, tree, etc.), their capabilities (can they fly, how fast they can travel, etc.) and how they are to respond to other objects (engage another entity or not engage, react to a collision with another entity, etc.) The entities may be expressed by the following properties:

1. State - a set of different properties, including static (e.g., size, color, length) and dynamic (e.g., current speed) properties.

2. Behavior - a description of changes to the object's state (e.g., moving left and accelerating, responding to other entities, etc.).

3. Identity - a name that identifies an object and distinguishes it from all others.

Given that objects provide the fundamental units of simulations, the high-level architecture must indicate the basic ways in which these objects will be specified and will interact with one another in all situations (Figure 1). The Execution and Simulation Representation module specifies timings, event occurrences, and all other processes involved in the simulation. The operational support module is needed to initialize the simulation by generating scenarios and distributing relevant data to all objects involved in the simulation.

Although simulated objects and execution mechanisms are the central components of a simulation, other components are also necessary in forming an overall simulation system. These components should include tools for scenario generation and configuration management. Scenario generation is the first phase of any simulated environment. It creates entities and describes their initial behaviors. Scenario generation requires the specification of a complete and unambiguous set of specifications (rules) describing scenario parameters (objects, paths, events, and timings). For some high-fidelity simulation this may require specifications of millions of parameters to describe all of the entities at a high level of resolution [Barr and Clark, 1993].

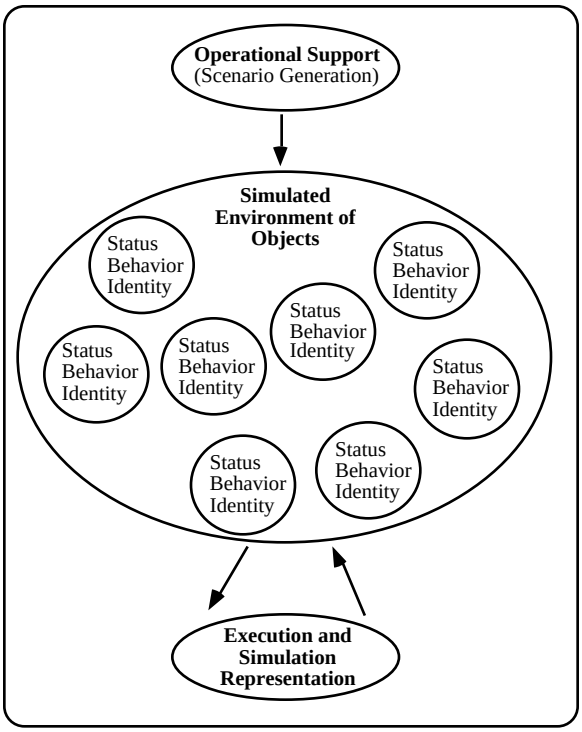

Figure 1. High Level Simulation Architecture

Historically, scenario generation has been a labor-intensive and time consuming process. As the complexity and number of simulations per system increase, a more timely and efficient method to develop scenarios is needed. This is especially evident in networked simulations where the complexity of scenarios may be enormous. To fully utilize the capabilities of current and future simulation systems it is vital to develop user interfaces for scenario generation that will permit the user to communicate easily and quickly with the computer and construct/alter scenarios (i.e., to generate various scenarios with the same simulation process). Many existing systems do not automate the scenario generation process. In most instances, a system developer encodes scenarios, a process that can literally take days. A rapid scenario generation engine is needed.

Another motivation for automated scenario generation is the fact that current simulation systems have a strong tendency to replay the same exercise scenarios over and over. This leads to users acquiring specific skills (by

repeating the same scenarios) when the acquisition of general skill is required.

Scenario generation can be viewed as the search process for the optimal solution (scenario). The search space is defined by a set of all parameters that describe objects' properties (State, Behavior, and Identity). An important property of most of the search techniques is that they suffer from *combinatorial explosion*.

3. SEARCH STRATEGIES

Various strategies for effective search have emerged from the fields of mathematics and computer science over the years. These range from totally uniformed search methods with no knowledge of the domain being searched to well-informed techniques in which knowledge of the domain is used effectively to speed the search. In mathematics three main types of search methods are identified: calculus-based, enumerative, and semi-random.

- Calculus-based search methods seek local extrema by solving a set of usually equations. These methods depend upon the existence of derivatives (well-defined slope values). Even if we allow the numerical approximation of derivatives, this has a severe shortcoming. Many practical parameter spaces have little respect for the concept of derivatives and the smoothness they imply (e.g., an object's name parameter in a scenario space). The real world of search is populated with discontinuities, multimodality, and noise. For example, for almost every possible scenario there exists some parameter (or subset of parameters) that when its value is slightly changed the entire scenario shifts to a completely different category.

- In enumerative search methods, an algorithm with a finite search space starts looking at objective function values at every point in the space one at a time. Although the simplicity of this type of search is attractive, and enumeration is a very “human” kind of search (when the number of possibilities is small), such methods must eventually be discounted for one simple reason: lack of efficiency. Many practical spaces are simply too large to search one at a time.

- Semi-random search methods have achieved increasing popularity as the shortcomings of calculus-based and enumerative methods have been recognized. Random searches, in the long run, can be expected to do no better than enumerative methods. The Genetic Algorithm is an example of the search procedure that uses semi-random choice as the tool to guide a highly exploitative search through a coding of a parameter space [De Jong, 1988]. Each point in the problem space can be considered as an individual represented uniquely within the system by a string generated by some alphabet. This alphabet is often taken to be {1,0}. (Some evidence exists that the binary alphabet is optimal). At any instance in time, the system maintains a population of strings representing the current set of solutions to the problem. The process begins by random generation or designer specifications of a starting population. The only feedback available to an adaptive strategy is the value of the process performance measure (fitness). A Genetic Algorithm is highly applicable to multimodal and multidimensional search spaces in which no a priori information is required.

4. SCENARIO SEARCH METHOD

We propose a high-level master-slave interface architecture for intelligent control of scenario generation in distributed interactive simulation (Figure 2).

The interface module (master module) processes information concerned with the global tactical picture. It has a full access and control of those attributes of each simulation

platform that are essential for interaction between platforms during the simulation process. We call such attributes -- global attributes, in contrast to - local attributes. The global attributes mainly represent various parameters obtained from sensor environments. The global attributes are highly interdependent (e.g., correlated sensor readings between two simulator platforms).

A generation engine is responsible for global attributes initialization and control. It works in a semi-automatic fashion. Some parts of global attributional representation are supplied by users (User Defined Global Control) and some parts are automatically tuned. The tuning process is considered to be an optimization problem. We propose to use Genetic Algorithms optimization techniques [Holland, 1975] to optimize global attributional representation.

5 GENETIC ALGORITHMS BASED GENERATION ENGINE

Genetic Algorithms, (Figure 3) as optimization and adaptation techniques, maintain a constant-sized population of candidate solutions known as individuals [De Jong, 1988]. The initial seed population can be chosen randomly or on the basis of heuristics. At each iteration, known as a generation, each individual is evaluated and recombined with others on the basis of its overall quality or fitness in solving the task. The expected number of times an individual is selected for recombination is proportional to its fitness relative to the rest of population. The power of a genetic algorithm lies in its ability to exploit in a highly efficient manner information about a large number of individuals. The search underlying Genetic Algorithms is such that breadth and depth are balanced according to the observed performance of the individuals evaluated so far. By allocating more reproductive occurrences to above average individuals, the overall effect is to increase the population's average fitness. New individuals are created primarily using two genetic recombination operators known as crossover and mutation. Crossover operates by selecting a random location in the genetic string of the parents (crossover point) and concatenating the initial segment of one parent with the final segment of the other parent to create a new child. A second child is simultaneously generated using the remaining segments of the two parents. Mutation provides for occasional disturbances in the crossover operation by inverting one or more genetic elements during reproduction. This operation ensures diversity in the genetic strings over long periods of time and prevents stagnation in the evolution of optimal individuals. The individuals in the population are typically represented using a binary notation to promote efficiency of the genetic operations and application independence.

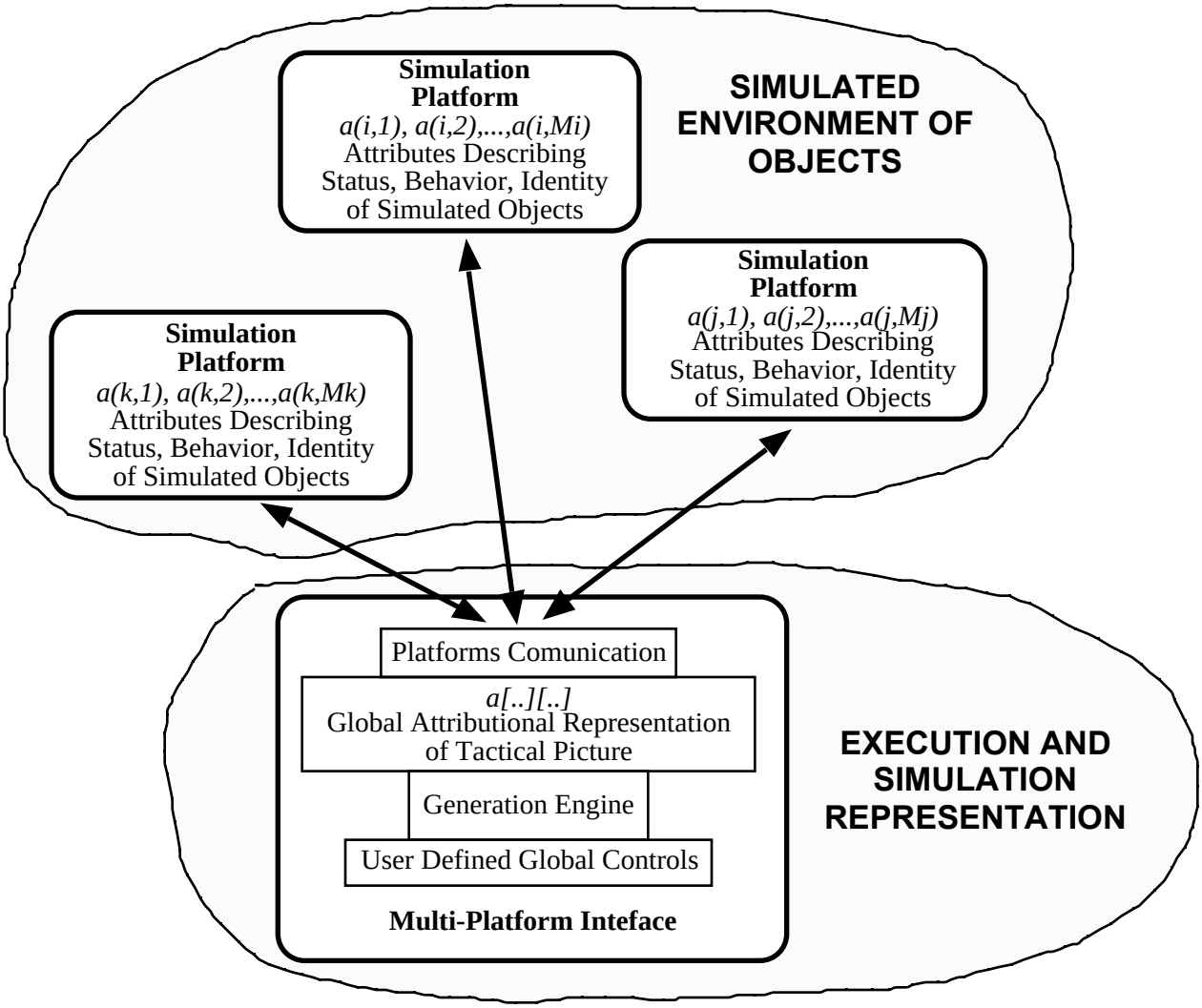

Figure 2. An Architecture for Scenario Generation in Networked Simulations

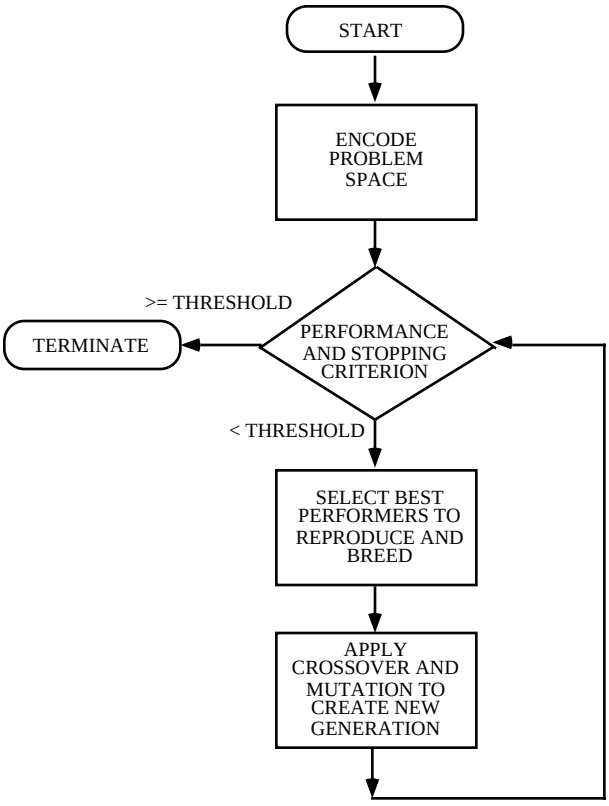

Figure 3. Genetic Algorithms

A Genetic Algorithm is very effective at finding optimal solutions to a variety of problems. It performs especially well when solving complex "real world" problems because it does not have many of the limitations of traditional techniques. Due to its nature, a Genetic Algorithm will search for solutions without regard to the specific inner working of the problem. This ability lets Genetic Algorithms perform well on large, complex scheduling problems, the design of communication networks, nuclear plant fuel configurations, financial portfolios management, and various small engineering problems.

The following considerations motivated us in the choice of Genetic Algorithms as the interface generation engine:

- Genetic Algorithms do not require specification of many parameters (usually a user specifies only the population size, crossover and mutation rate, and stopping criterion).
- The search space, as considered in our proposed architecture, is highly multi-modal. That is, there exist many optimal/suboptimal solutions. Genetic Algorithms search techniques are very efficient in such a search space.
- The search space is also highly multi-dimensional, thus yielding a very complex search problem (Distributed Interactive Simulations may involve thousands of objects with hundreds of attributes). Genetic Algorithms search techniques are very efficient in searching complex representation spaces.

In the proposed approach Genetic Algorithms search the space of possible mission rehearsal scenarios to find the most optimal scenario for the exercise problem based on their proximity to predefined scenario classes and specifications that have been initially set up by the instructors/system managers. An example would be "A scenario class Hard" (Table 1).

ATTRIBUTE	VALUE
A1 -> Number_of_moving_objects	2 to 10
A2 -> Number_of_static_objects	more than 2
A3 -> Behavior_type_object_1	type_C
A4 -> background_type	type_A

Table 1. An Example of the Criteria Table.

This proximity is measured by numerically expressing closeness (distance) to a criteria table, in other words, how close does the user's desired scenario match previously defined scenarios created in previous exercises. The criteria table would have been built as part of the acceptance of the proposed interface architecture, or built when the system became operational by the user. It describes initial specifications for a given scenario and defines the scenario category to be generated. The entries of the criteria table are rules that describe the conditions for various scenario class parameters. A scenario class can be expressed by a large number of rules that form a disjunctive normal form (DNF) data

structure. For the Table 1 example the corresponding rule is expressed as follows:

Scenario Class "Hard" if
[A1 =2..10] &[A2 > 2] &[A3 = "C"]&[A4= "A"]

An individual scenario in the population of scenarios is then evaluated and recombined with others on the basis of its overall quality and/or fitness to the current problem, the desired level of quality and/or fitness having been established during system design and refined during user operation. This process is iterative, and the expected number of times a scenario is selected for recombination is proportional to its fitness relative to the rest of the population. By allocating more reproductive occurrences to the above scenarios (relative to quality and/or fitness) an overall increase in the population's average fitness is achieved. The result is consistently different scenarios, thereby preventing the scenario user from "gaming" the system. The Russian submarine that was at some location in the last exercise may now be an Iranian submarine, or may not be there at all, just as in the real world.

For the instructors/system managers, the process of scenario generation is greatly simplified by requiring only one simple process of determining what opponent is desired and giving "orders" to the desired forces (User Defined Global Control module in Figure 2). The system would then begin the process of automatic generation scenario across all platforms. This process leads to generation of all local simulation attributes (Figure 2) that now can be transmitted to individual platforms.

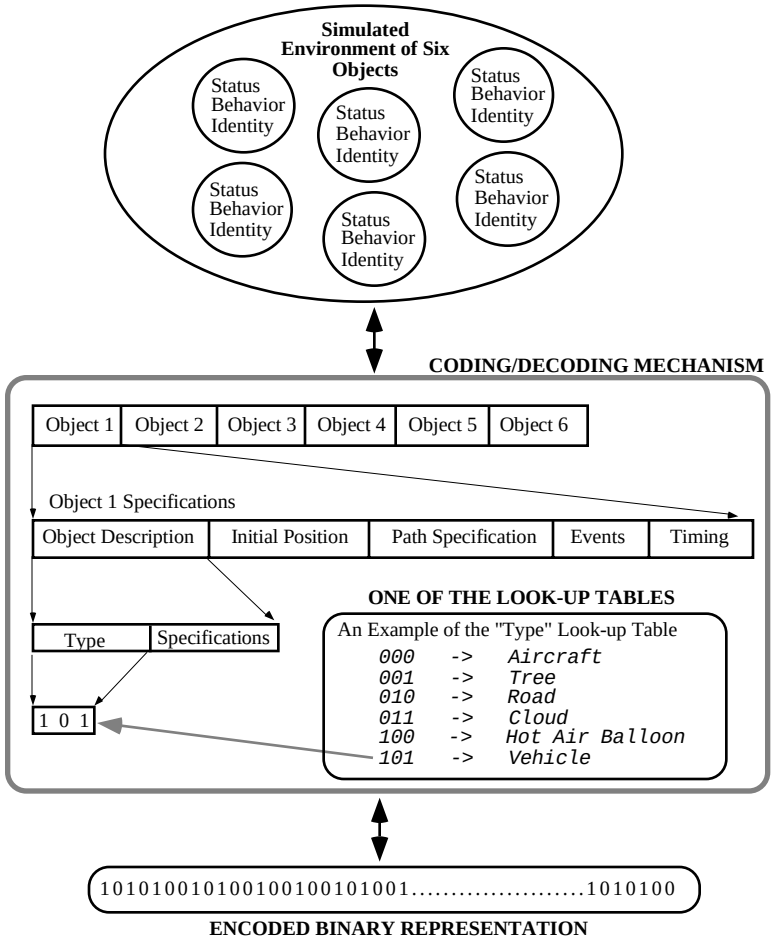

Figure 5. A Scenario Coding Schema

The scenario encoding/decoding used to represent scenarios consists of an ordered list of fields together with the look-up tables which indicate how bit strings are to be decoded to produce information about a given scenario. Figure 4 depicts a simple example of a possible encoding/decoding scheme. All objects are encoded as a concatenated string of bit fields. Each field represents objects' parameters and behavior. It consists of sub-fields that describe the object, its initial position, path, events, and timings. Accordingly each sub-field is further divided into sub-parts that describe the object in a greater detail. Finally, the shortest, non-dividable fields are mapped to different pieces of information by using various look-up tables.

6. CONCLUSIONS

A high-level master-slave interface architecture for intelligent control of scenario generation in distributed interactive simulation has been presented in this paper. It addresses the complexity problem of current and future networked simulation systems. It is based on a principle that utilizes Genetic Algorithm search techniques. Genetic Algorithms efficiently search the scenario space and generate close to optimal scenarios. The generated scenario performance measure is calculated based on a distance measure to a predefined scenario. The presented approach has been developed and is currently being experimentally implemented by Datamat Systems Research, Inc.

REFERENCES

Bala, J., B.K. Gogia, "Genetic Algorithms Based Scenario Generation Approach For Simulated Worlds," International Joint Conference on Artificial Intelligence (IJCAI) Workshop on Entertainment and AI/Alife, Montreal, Canada, August 18, 1995.

Bala J. and Gogia, B.K., "Direct Manipulation Interface for Flight Training Simulators: A Rapid Scenario Generalization and Optimization," Naval Air Systems Command (NAVAIR) PMA205, Phase I Final Report, January 1995.

Barr, T. and Clark, K., "Scenario Preparation of DIS," *Proceedings of the 14th Interservice/Industry Training Systems and Education Conference*, 1993.

De Jong, K., "Learning with Genetic Algorithms: An Overview," in *Machine Learning*, Vol.3, 121-138, 1988.

Feiner, S. and Beshers, C. "Worlds Within Worlds: Metaphors for Exploring n-Dimensional Virtual Worlds." *Proceedings of 3rd Annual Symposium on User Interface Technology,* Snowbird, Utah, 1990.

DEFINITIONS

ACM	Association for Computing Machinery.
ADPA	American Defense Preparedness Association.
DIS	Distributed Interactive Simulation.
MASTER-SLAVE	A two modules control architecture, where a master module is an entity controlling a slave module.
REFORGER	Return of Forces To Germany.