Proceedings of the 1988 Winter Simulation Conference
M. Abrams, P. Haigh, and J. Comfort (eds.)

Utilizing abstraction and perspective in battle simulation

Paul A. Fishwick and Steven M. Walczak
Department of Computer and Information Science
University of Florida
Bldg. CSE
Gainesville, FL 32611

Abstract

Scenarios involving battle simulation require much overhead in terms of computer resources. The rules for strategy and equations for vehicle motion can overwhelm a typical engineering workstation. We present key methods based on the represention and manipulation of battle objects using levels of abstraction and varying perspectives. These methods serve to reduce the workstation resources and provide for a more flexible training simulation environment. We illustrate our methods by describing the initial design and implementation of a multi-level tank simulator.

1 INTRODUCTION

Recently, we have discussed formal methods of process abstraction[Fis87,Fis88] and have illustrated the advantages of integrating several methods of abstraction in a simulation environment[Fis86a]. We also discussed the need for simulation environments to make available to the simulation analyst three types of abstraction: *process*, *object*, and *report*. We now overview the advantages of providing explicit abstraction control in a simulation.

An abstract model is usually less computationally complex than the base model. As long as the tradeoff between complexity and data sufficiency is favorable, then the abstract model remains useful. We define "data sufficiency" in terms of the availability of sufficient data for the analyst to use a simulation effectively for useful feedback. The abstract model is also easier to understand than the base model in most cases. Since an abstraction involves a reduction in process components, the model is easily created and modified. Different users will require different degrees of comprehension and therefore different forms of simulation input and output.

What are definitions for "abstraction" and "perspective?" An abstract model is often obtained from a more detailed base model through homomorphic mapping[Fis88]; the mapping attempts to satisfy the requirements of behavior and structure preservation. A perspective, on the other hand, relates to a system "view" taken by a specific analyst. Different analysts will want to see input, model parameters, and output in a certain abstract form. In the battle management domain, we define lieutenants and generals, for instance, as having fundamentally different perspectives for the same underlying system to be simulated. In previous research we have identified a conceptual methodology for simulating a dynamic system through abstraction and perspective. Now, we are applying this methodology to battle simulation so demonstrate the utility of utilizing abstraction and perspective. In the following sections, we define two distinct methods that result in better utilization of simulation resources: *model abstraction* and *model integration*. Then we present the initial design and implementation of a tank simulator program called *SIMTANK*. *SIMTANK*, by itself, is capable of multi-level simulation; however, an analyst is also able to use *SIMTANK* in conjunction with other simulation programs using the central abstraction processor defined in section 3.

2 MODEL ABSTRACTION

2.1 Previous Related Research

Our research, so far, has resulted in a LISP based multi-level simulation language called HIRES (HIerarchical REasoning System)[Fis86a,Fis86b] in which programs can be created to take advantage of process abstraction in particular. The user may manually or automatically weave through levels of abstraction previously defined by the model designer. The HIRES research showed how process abstraction could be applied to scene animation containing a set of moving articulated human bodies. This paper, on the other hand, will discuss our next generation system which will have the novel features of 1) permitting, not only model design via process abstraction, but also object[Cla76,Fei85] and report abstraction integration, and 2) a more detailed application scenario such as battle simulation.

Our original research[Fis88] described a domain involving

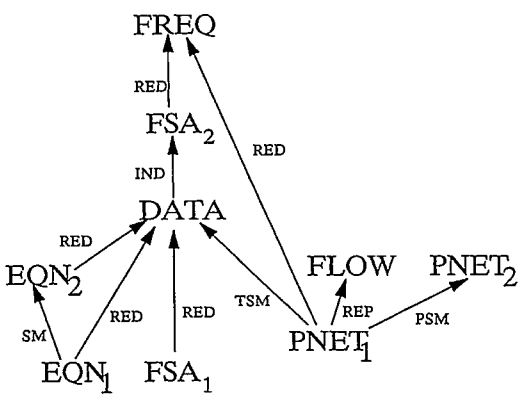

Figure 1: Abstraction Network for **DP**

articulated figures at multiple abstraction levels: the dining philosophers (referred to as **DP**). We defined a multi-level model as shown in figure 1. Each vertex within the graph shown in fig. 1 is a distinct simulation model in itself. *FREQ* designates a model based on frequency of the basic eating activity; *FLOW* provides for a simple representational abstraction of **DP**, *PNET*'s are Petri nets defining resource contention and synchronization, *FSA* is a finite state machine representing a simple synchronization mechanism; *DATA* is a table of events or states, and the *EQN*'s are equations of spline motion for all degrees of freedom associated with the articulated figures. Figure 2 displays the three dimensional geometrical model used for the simulation.

2.2 Current Research

We plan on integrating process, object, and report abstraction methods within the simulation environment. Process abstraction allows the simulation model to be multi-layered so that several levels of abstraction are encoded within the model. This means that depending on viewing parameters and trainee selection, parts of the overall simulation process (a process being defined as a scenario involving warfare or other physical movement involving many agents) will involve varying levels of structure. For instance, the trajectory of an aircraft may have several levels of abstraction; the lowest being the system of differential equations and the highest being either a linear equation or a motion heuristic. Object abstraction, on the other hand, enables us to build the geometry of the model according to a tree of structures. We might view a tank as a simple rectangular polyhedron or more completely as a structure defined using constructive solid geometry (a much finer level of visual detail).

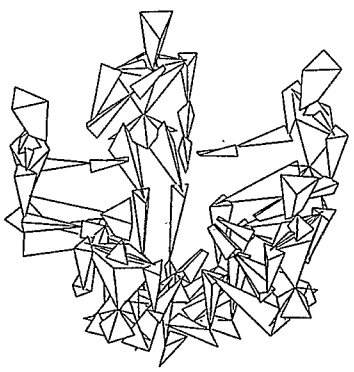

Figure 2: **DP**: A Geometrical Model

The simulation user is free to control process and object abstraction via either manual or automatic methods. A manual method involves the use of an input device such as a joystick to zoom in or out" of portions of the model. The incorporation of both abstraction types will allow for extremely complex scenarios to be used; detail is presented to the user only where appropriate. The current iconic displays of some currently available visually oriented simulation systems can be seen as a special case of our integrated environment: namely, an aerial view of the scene using the highest level of object abstraction representing the objects as simple icons.

Report abstraction was discussed in [Fis86a] and refers to the level of report available to a simulation user. Does the user wish to see a visual, textual or graphical report on simulation variables and objects? A visual report refers to the animation. A textual or graphical report can be obtained by pointing the mouse at the simulation object in question. A hypertext approach may be appropriate to allow a user to obtain the maximum amount of information on a chosen object; the first pop-up window would display immediate data such as ID, object type and velocity vector. Other information can be obtained through sub-menu interaction.

The simulation allows for dynamic control; that is, a user may stop a simulation while it is running, obtain variables and possibly backtrack. The backtracking feature will be useful if the trainee makes an error in judgment and then decides to backtrack to a control point so that another decision can be made. As mentioned, dynamic control includes the control of level of process and object abstraction. The mouse has demonstrated its utility as an effective input device and so we plan on making the mouse a central input device.

The addition of constraints will permit an analyst or a trainee to automatically construct scenarios. For instance, specific scenarios involving aircraft, ground vehicles, and terrain may be completely different on each simulation run by setting constraints on terrain creation and the position, number and speed of objects. Also, the objects in the simulation can arbitrarily be assigned heuristics for movement and other functions. Each object can be assigned a level of intelligence. This allows the trainee to actively work against a set of "smart" objects.

The method of model abstraction permits the analyst to use one computer program that is constructed to provide multiple abstraction levels. The analyst may dynamically switch to different process, object and report levels. We should also consider the capability of switching among different *programs*, each of which represents a user perspective and a possible abstraction level. For instance, we might use a package that contains continuous modeling capabilities (such as TUTSIM[RW88], ENPORT[Inc87] or EASY5/W[Ser]) and then switch to a discrete package when the simulation requires the use of queueing. We term this alternate method (integration among programs) *model integration* and also allow this capability to permit incorporation of perspectives within a battle simulation.

3 MODEL INTEGRATION

Modeling is being applied to an ever growing number of problems [Cel84]. As simulation systems become larger and more complex, a need for using multiple levels of abstraction to display information at an appropriate level for a particular user becomes advantageous. Users of a single simulation system often have varying requirements of the simulation system due to different backgrounds and desired outcomes. A sample population of users of a battle management simulation is given in figure 3. One method of using multiple abstraction levels is to incorporate all desired levels into a single process as discussed in [WF88]. This is the approach taken by *SIMTANK*. *SIMTANK* allows the user to choose the abstraction levels desired for each of the process, object, and report abstraction types from a mouse driven menu. After selecting the desired abstraction levels the simulation is then performed for the user at the selected levels of abstraction. Changing abstraction levels mid-simulation

can be done by dynamically choosing the new abstraction levels desired. *SIMTANK* can then be started from the previous state, at the new abstraction levels, from the simulation *state* information saved by the process.

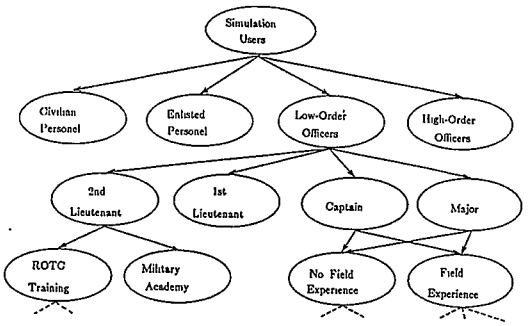

Figure 3: Sample population of users of a Battle Simulation system

Another method of achieving multiple abstraction levels is to use the Centralized Abstraction Processor (**CAP**) presented in [WF88]. The **CAP** stores information about the base model which will drive the simulation system in a central database. This model is transformed into the different specific levels of abstraction required by the simulation systems currently on the network (see figure 4). This methodology does not preclude the use of multi-level simulation systems like *SIMTANK*. Any simulation system on the network will be given an appropriately abstracted version of the base model stored in the central database from which to execute. Periodically, state information will be retrieved from each of the simulation systems to keep the base model current. Simulation systems which are capable of interactively switching abstraction levels will automatically be given the proper abstraction from the base model through the model abstraction window (see figure 5). The model abstraction window uses a table of current simulation systems on the network and the abstraction levels required of each system. Multiple abstraction simulation systems have multiple entries in the table and changing the current abstraction level involves changing the current pointer into the table for that particular system. Section 5 contains an example of this method.

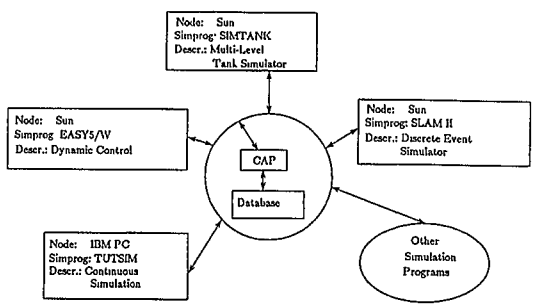

Figure 4: **CAP** simulation network environment

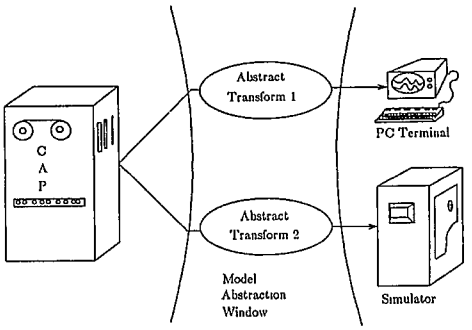

Figure 5: Model abstraction window

This centralized approach will help reduce the cost of complex simulation systems by permitting the re-use of existing simulation systems with only an update to the table of current systems table needed (if not already present). Allen[All87] indicates that modern computer wargames can involve multiple users stationed around the world. The **CAP** methodology supports this geographically dispersed configuration of users and furthermore would permit the use of different simulation systems at each of the sites. An inherent increase in the speed of development of simulation models is an additional benefit gained from the use of a single base model to drive a large number of simulation systems having different abstraction requirements.

4 SIMTANK: AN INITIAL IMPLEMENTATION

SIMTANK, as described in section 3, is a simulation system which incorporates multiple levels of abstraction. The multiple abstraction levels occur for all of the abstraction types: process, object, and report. The user is given the flexibility of choosing which levels of abstraction are desired for each of the three abstraction types via a menu. Input to *SIMTANK* from the user is primarily mouse driven. The program is being developed on a SUN 3 workstation using SunCore and PHIGS. Once the user has selected the current abstraction levels, the simulation is automatically started. At any time the user may halt the simulation which will cause the simulation *state* to be saved automatically. The saved state can then be used to restart the simulation after choosing different abstraction levels for the ensuing simulation.

The current version of *SIMTANK* has two levels of abstraction for the three abstraction types. Future effort will increase the number of abstraction levels for each of the abstraction types to at least three so that greater flexibility can be achieved by *SIMTANK*. Terrain effects have been left out for simplicity reasons, but will be included in future versions. Effort will also be focused on the ability of an intelligent coach which will be responsible for heuristically determining the "best" abstraction levels of each of the three abstraction types for new users so that a default selection of abstraction levels will produce the "best" results.

5 A SCENARIO

Figure 6 depicts an aerial view of a battlefield with two participants on either side of intervening river and forest regions. Assume that a command has been given to the tank commander to engage the enemy in battle. The enemy tanks have been detected by reconnaissance aircraft. The act of engagement (depicted in fig. 6 by a line with arrows) will require tank groups to cross the bridge and enter the region separating the two forests. The command to specify the physical path to take may arise from a real time decision (during a training simulation) or as the resultant feedback from a planner. Given the integrated system depicted in fig. 4, an analyst (or trainee) who is executing the simulation of this scenario begins with the *SIMTANK* program. During the course of the simulation, **CAP** will switch to different abstraction levels or application programs that it needs to accomplish the run. *SIMTANK* utilizes a discrete map composed of adjacent cells (organized, for instance, in a hexagonal lattice). Tanks move from one cell to another toward the bridge. Once tanks have begun to cross the bridge, **CAP** switches control to the SLAM II program since the bridge imposes a queueing problem during the run. The SLAM II language is disposed to simulating entities moving through queues, so it serves as a natural choice in this instance. In this manner, it is not necessary for *SIMTANK* to contain queueing capabilities. Also, we might use SLAM II's probabilistic functions when considering situations such as 1) possibilities of an enemy attack during the vulnerable bridge crossing, and 2) a structural bridge failure which incapacitates a tank for a duration. Next, the commander issues an order to engage the enemy by firing missiles. The ballistics and control aspects associated with this firing necessitates **CAP** to switch to a continuous simulation using either TUTSIM or EASY/5W if the analyst is concerned with the precise trajectory. Figure 7 displays the "perspective switching" that occurs during the simulation. The analyst starts in *SIMTANK*, proceeds to SLAM II due to the bridge crossing and then oscillates between *SIMTANK* and a continuous simulation package while the battle continues. Switching perspectives can be manual or automatic, and in this example scenario, the automatic feature was used so that the analyst was completely unaware of the actual computer program being executed at a given time.

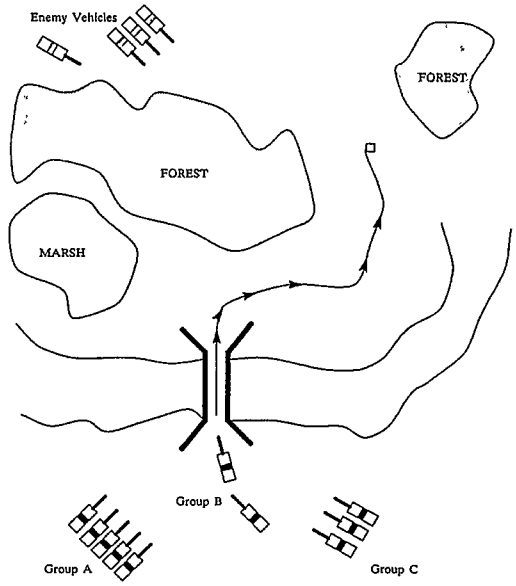

Figure 6: Battlefield Scenario

6 CONCLUSIONS

We have described our research directions and designs for a simulation environment that incorporates capabilities in abstraction and perspective to solve a typical simulation problem, such as one involving battle management. The use of process, object, and report abstractions in a single program gives the analyst much flexibility in viewing simulation results and manipulating simulation objects. *SIMTANK* is an example of such a program that uses abstraction methods to improve the training environment. *SIMTANK* permits the analyst to switch (automatically or manually) among several abstraction levels during the course of the simulation.

We have also defined the construction of a network of simulation programs centered around a "central abstraction processor" that serves to coordinate various aspects of each simulation; existing simulation software is incorporated into the network to serve its most useful purpose. For instance, EASY5/W excels at the process of permitting an analyst to view state variables for a dynamic system — so why should one build this feature into every new simulation program? We have decided to let each simulation program perform the task for which it was originally intended. We hope this effort will help to avoid "reinventing the wheel" every time a new simulation technique is desired within a simulation application.

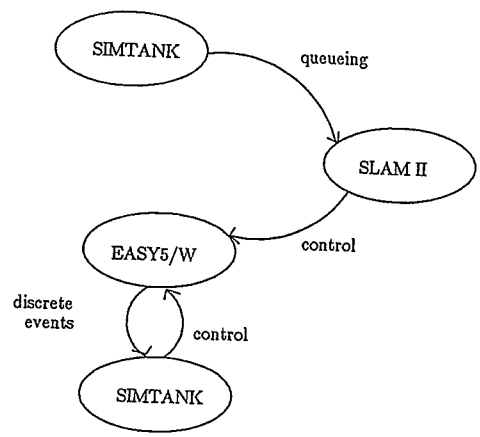

Figure 7: Perspective Switching

7 ACKNOWLEDGMENTS

This research is made possible by a grant from the Florida High Technology Council.

References

[All87] Thomas B. Allen. *War Games The Secret World of The Creators, Players, and Policy Makers Rehearsing World War III Today*, pages 283–305. Mc-graw-Hill, 1987.

[Cel84] Francois E. Cellier. How to enhance the robustness of simulation software. In Ören et al., editor, *Simulation And Model-Based Methodologies: An Integrative View*, pages 519–536, Springer-Verlag, 1984.

[Cla76] James H. Clark. Hierarchical Geometric Models for Visible Surface Algorithms. *Communications of the ACM*, 19(10):547 – 554, October 1976.

[Fei85] Steven Feiner. APEX: An Experiment in the Automated Creation of Pictorial Explanations. *IEEE Computer Graphics and Applications*, 5(11):29 – 37, November 1985.

[Fis86a] Paul A. Fishwick. *Hierarchical Reasoning: Simulating Complex Processes over Multiple Levels of Abstraction.* Technical Report, University of Pennsylvania, 1986. Ph.D. Dissertation.

[Fis86b] Paul A. Fishwick. HIRES: A Multilevel Knowledge-Based Simulation System. *IEEE Software*, 3(2):52 – 53, March 1986. (Research Note).

[Fis87] Paul A. Fishwick. A Taxonomy for Process Abstraction in Simulation Modeling. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 144 – 151, Alexandria, Virginia, October 1987.

[Fis88] Paul A. Fishwick. The Role of Process Abstraction in Simulation. *IEEE Transactions on Systems, Man and Cybernetics*, 18(1):18 – 39, January/February 1988.

[Inc87] Rosencode Associates Inc. ENPORT Reference Manual. 1987.

[RW88] Walter E. Reynolds and Jinner Wolf. *TUTSIM Users Manual.* TUTSIM Products, Palo Alto, California, 1988.

[Ser] Boeing Computer Services. EASY5/W reference manual.

[WF88] Steven M. Walczak and Paul A. Fishwick. A Centralized Methodology for Multi-Level Abstraction in Simulation. *SIMULETTER*, 19(3):–, September 1988.

BIOGRAPHIES

PAUL A. FISHWICK is Assistant Professor of Computer and Information Science at the University of Florida. He has six years of industrial production and research experience working at Newport News Shipbuilding & Dry Dock Company doing CAD/CAM research, and at NASA Langley Research Center performing database machine and engineering data management research. He received the BS in Mathematics from the Pennsylvania State University, MS in Applied Science from the College of William and Mary, and PhD in Computer and Information Science from the University of Pennsylvania in 1986. His current interests deal with paradigms for modeling and simulation, knowledge-based simulation, system science, computer animation, CAD/CAM, machine learning and other areas within artificial intelligence. He is chairman of the IEEE Technical Committee on Simulation and an associate editor for the ACM SIGSIM publication SIMULETTER. He is a member of IEEE, IEEE Computer Society, The Society for Computer Simulation, ACM, AAAI, and IMACS.

Department of Computer and Inf. Sciences
University of Florida
Bldg. CSE, Room 301
Gainesville, FL. 32611
(904)-335-8036
INTERNET: fishwick@bikini.cis.ufl.edu

STEVEN M. WALCZAK is a Ph.D. student at the University of Florida working in the areas of simulation and knowledge-based systems. He has five years experience working for the Department of Defense doing work in expert systems for signal analysis and pattern recognition; and one year experience in the AI laboratory at Harris Corporation. He received the BS in Mathematics from the Pennsylvania State University and an MS in Computer Science from John Hopkins University in 1984.

Department of Computer and Inf. Sciences
University of Florida
Bldg. CSE, Room 301
Gainesville, FL. 32611
(904)-335-8044
INTERNET: smw@beach.cis.ufl.edu