Proceedings of the 1992 Winter Simulation Conference
ed. J. J. Swain, D. Goldsman, R. C. Crain, and J. R. Wilson

A FRAMEWORK FOR DESIGNING AN ANIMATED SIMULATION SYSTEM BASED ON MODEL-ANIMATOR-SCHEDULER PARADIGM

James T. Lin
Kuang-Chau Yeh
Liang-Chyau Sheu

Department of Industrial Engineering
National Tsing Hua University
Hsinchu, Taiwan 30043
Republic of China

ABSTRACT

A methodology for designing animated simulation (AS) system has been presented. A Model-Animator-Scheduler paradigm was proposed for unifying different approaches of AS systems into a single structure and serving as a foundation for implementation and further research. A framework which identifies the specification and organization of AS systems was also proposed on the basis of such a methodology. An experimental AS system, SIMGRAPH, has finally been presented here under such a framework.

1 INTRODUCTION

Visual simulation is a powerful tool for system analysts. Proper usage of visual simulation can enhance verification, validation and testing of large or complex dynamic models. Two general families of model data visualization have been previously proposed by Rooks (Rooks 1991), which are: *abstract* and *representative* displays. While abstract displays are only simply visual tools used for data interpretation, representative displays provide a pictorial view of the modeled system in some simplified form(s) of its real appearance. *Animation* is one of the major representative display forms. It refers to graphic displays of information where the information to be imported to the viewer is conveyed through image change. Many simulation tools and packages have provided an animation facility such as SEE-WHY, CINEMA which comes with SIMAN simulation language, and SLAMSYSTEM which is integrated with SLAM II language.

Tools for animation are the most difficult components of visual simulation for design and implementation. They can determine the effectiveness of an entire visual simulation interface. Current literature does not, however, provide a generic view of simulation animation. This is restricted by practical considerations of computer resources or product marketing. Proposing a methodology for developing or realizing animated simulation (AS) systems is therefore the objective of this paper.

Different methodologies have been previously addressed toward design and implementation of an AS system (Bishop and Balci 1990, Brunner 1986, Cox 1987). Three distinct approaches identified by O'Keefe (O'Keefe 1987) for portraying the dynamic behavior of the modeled system are:

Embedded Programming: The simulation programmer codes visual output statements into model directly. It provides great flexibility. The extra development effort can, however, be time consuming. It also refers to the term "simulation-concurrent animation" for the dynamic display being generated through the state of the simulation model.

Automatic Display: A standard type of display is provided, which can be used with little or no effort on the part of the developers. It is very easy to construct. Developers can, however, be constrained by a lack of flexibility.

Post Animation: The simulation automatically produces formatted output which is then decoded by an animator through use of a separately produced static background and rules for icon movement.

These three approaches cover a range of model implementation difficulty and flexibility levels. A developer does not have to choose only one approach to be the principle one. Developing a structure unifying these three approaches is, however, helpful so that the most proper animation scheme for specific application can be easily identified under this general scheme. It will then serve as a foundation for implementation and further research which recognizes and encompasses the broader methodology.

This paper is organized as follows. A fundamental paradigm for representing the structure of AS is first described in Section 2. The specification and organization of the software for implementation of an AS system are described in Section 3. The experimental AS system itself proposed in this paper is described in Section 4. A brief discussion is finally given in Section 5.

2 MODEL-ANIMATOR-SCHEDULER PARADIGM

A paradigm is proposed in this section for representing the unified structure of AS. Characteristics of such a paradigm are also described. Elements of such a structure should first be discussed.

The visual display is composed of two basic parts: (1) a static background picture which is written to the screen once prior to the simulation run, and (2) a dynamic display which moves over the static picture. Two distinct primitives of operating the dynamic display, furthermore, exist. They have been previously referred by Brunner (Brunner 1991) as the *simple primitive* - moving an image block or changing colors, and the *complex primitive* - sophisticated moving on a path. Two modes of animation also provided by GPSS/PC (Cox 1987) are: *Direct Mode* and *Collision Prevention Model* . While simple animation primitives are performed in Direct Mode, some implicit "Move Events" are scheduled to occur when an entity is going to move in later mode to carry out proper moving and collision testing.

Additional animation events should be calculated and scheduled for performing complex operations, i.e. an automated guided vehicle (AGV) or a moving belt conveyor. The dynamic display is to be updated either within an event (direct animating without changing time), or following a time beat (where one or more animation events will be executed). The animation operations and associated events should be managed by a certain coordination mechanism.

In summary, three basic divisions are proposed on the basis of the interactions within an animated simulation system:

•*Model*: This term here is specific to a combination of data and programs on a computer representing a target system. It includes description of processes or entities in which analysts are interested.

•*Animator*: It contains all facilities which perform animation on a computer display and associated tasks, like moving objects and drawing static background, etc.

•*Scheduler*: It coordinates the time-varying operations occurring between the model and animator. Some operations similar to moving entities or monitoring variables of a model whose timing need to be pre-calculated and vary at every time beat cannot be directly assigned to the animator. They should be scheduled by the scheduler before they are actually performed.

These three divisions interact with each other as model execution progresses. Each division needs certain information in serving their proper tasks. The three divisions and their relationships are shown in Figure 1. The information occurring among these three divisions are classified as follows:

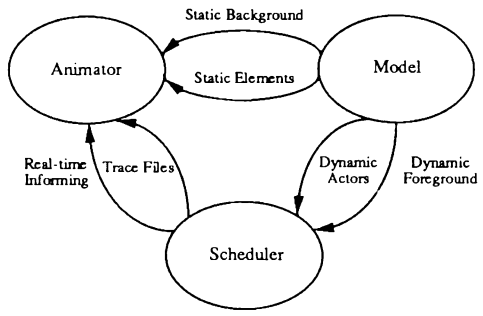


Figure 1 Model-Animator-Scheduler paradigm

Model-Animator:

Animator requires two families of description from simulation model, which are:

•*Static Background*: Static background is the image of the target system without changing over the simulation duration. It shows the geometric or abstract outlook of the target system, e.g., the layout of a factory.

•*Static Elements*: Static elements are the static objects of the target system. They are similar to static background. They can, however, be used to reflect the differences between experiments, e.g., the locations of machines under the same layout.

Many systems do not distinguish between these two families of description. A similar concept applied in Proof system (Brunner 1991) is, however, the *information definition* in a layout file.

Model-Scheduler:

Two classes of information should be designated to the scheduler by the model for the time-varying properties:

•***Dynamic Actors***: Dynamic actors are those movable objects or entities of the target system. Since their locations may change at every time instance according to the form of events, these events should be sent to the scheduler in association with specific actors.

•***Dynamic Foreground***: Dynamic foreground is the summary of time-varying system status or variables. Their locations are fixed, but the containing values are to be updated by certain events.

Similar components are also found in CINEMA/SIMAN, SLAMSYSTEM and GPSS/PC. The prior is referred to as *entities* or *transactions*, and the later is referred to as the summary of *resources*, *queues* and *variables*.

Scheduler-Animator:

Two different forms of event notices exist from scheduler to animator:

•***Real-time Informing***: The Scheduler directly sends notices at the time of occurrence to the animator. They are usually in the form of function calls.

•***Trace Files***: The scheduler stores these notices in some specific formatted storage, e.g., a data file. It then transfers this storage to the animator in performing animation after the simulation has been completed.

The concurrent animation is, for example, provided by TESS through interacting with SLAM II user-written FORTRAN code. The trace files are also interacted with through usage of the data collection procedures.

The *Model-Animator-Scheduler paradigm* can fully represent the structure of an AS scheme. First, embedded programming is the case which occurs where the model and the animator are tightly coupled. They share the same copy of program code and data as a means of communication with each other. The scheduler is combined with the time advancing mechanism of simulator, since the animated events and model events are not separable. The diagram under the integration of model and animator is shown in Figure 2.

Second, automatic display is the case which occurs where the model passes the animated notices to the animator through a dummy scheduler. Restated, the notices produced by the model are directly sent to the animator. The image update, e.g., movement of an AGV, is irreverent to the simulator clock. The diagram under automatic display is shown in Figure 3.

Third, post animation is the case which occurs where the scheduler is simply an event filer. It receives all events being produced by the model and stores them in a formatted data file through the order of time sequence. The diagram under post animation is shown in Figure 4.

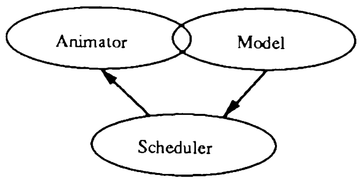


Figure 2 Embedded Programming

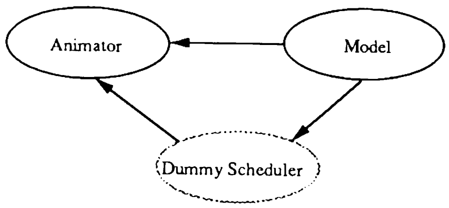


Figure 3 Automatic Display

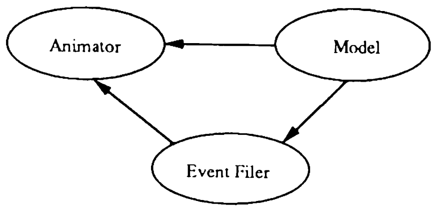


Figure 4 Post Animation

3 A MODEL SPECIFICATION AND ORGANIZATION

A specification and a software organization for implementation are required for accomplishing the previous framework. Several design environments should be discussed in advance before designing such an AS system. These environments are to determine the implementation architecture of the system, which are:

•***Computer/hardware environment***: What kind of computer and associated display hardware is to be used?

•***Simulation environment***: Which general purpose simulation language (e.g. SLAM II or SIMAN) or high-level programming language (e.g., FORTRAN or C) is to be chosen?

•***Display type***: What kind of display scheme should be applied on a given specific hardware? Possible types include bitmap-based, pixel-oriented, geometric-based, and vector-oriented display.

•***Animation type***: What kind of animation scheme is to be applied under such an AS system? It may be block-based which uses local bit-block-transfer method, or frame-based which updates the whole screen on each time change.

Given particular factors of these environments, the basic essence of an AS system specification under the framework is based on a *model organization specification* of an animated simulation model. Additionally, two specific modules are necessary in manipulating and maintaining the elements in such a model organization, which are: (1) *function modules* corresponding to model-animator-scheduler and (2) *utility modules* with respect to model organization. The elements and requirements of these three components are discussed below.

Model organization specification

Summarizing the previous information and forms of animated events, a formal model organization specification of an animated simulation model is derived. It includes a *simulation model,* a *static background, static elements,* a *dynamic foreground, dynamic actors* and the *trace file structure.* These are the five basic elements of the model organization specification. Figure 5 shows the diagram of such a specification. The generation and maintenance of data containing this organization are discussed in more detail later.

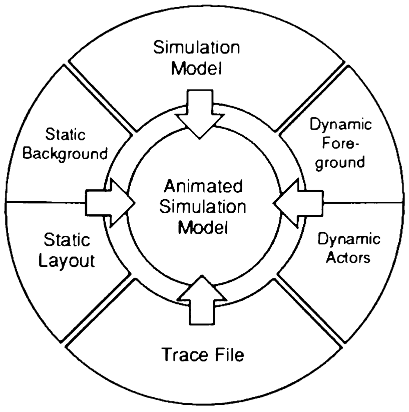


Figure 5 Specification of an animated simulation model

Function modules

Corresponding to the three basic divisions mentioned previously, function modules can be classified as follows:

•***Graphic and animation commands***: Under a certain simulation environment, the graphic and animation commands should properly collaborate with model description statements. For instance, these animation commands under an event-scheduling simulation system can be in the form of a procedure-call or function-call placed in event handling routines. These functions are to parse command, assign graphic tasks to graphic display functions and send animation events to the scheduler. These functions should be grouped as an independent module.

•***Static and dynamic graphic display functions***: The tasks of an animator include two major function groups: graphic drawing and animation performing. The design of these two function groups is dependent types of display, type of animation and graphic hardware.

•***Scheduler***: Two approaches in implementing a scheduler are: (1) It combines a time-advancing mechanism with the animation events for updating the simulator clock as well as other events, and (2) It is independent from the simulator and the animation events do not affect the simulator clock. These two approaches are, however, not exclusive. A hybrid design is possible.

Utility modules

In addition to function modules, utilities for generating and maintaining image outlooks of an animation model are necessary. These utilities correspond to the five elements of the formal model organization of an AS model, which are:

•***Static background generator***: This generator should provide functions of drawing and editing a background outlook, or provide utilities to transfer the output of other graphic packages or CAD programs into a specific formatted file that the animator can display.

•***Static elements editor***: This editor should provide functions of defining elements like static texts output, static icons display, or color assignment, etc.

•***Actor/icon editor***: This editor should provide editing functions and libraries to maintain associated actor images.

•***Dynamic foreground generator***: This generator should provide functions to define elements like the paths of actors moving, display of resources/queues and associated icons, or type and location of statistical charts display, etc.

•*Trace file player*: The player should be able to interpret outputs from the scheduler and perform animation.

These generators and editors generate data or files as the front inputs of an executable AS model. Trace file player, which is an option for post animation, receives the outputs from AS model. Figure 6 shows the construction and execution cycle of a model, and the total organization of such an AS system.

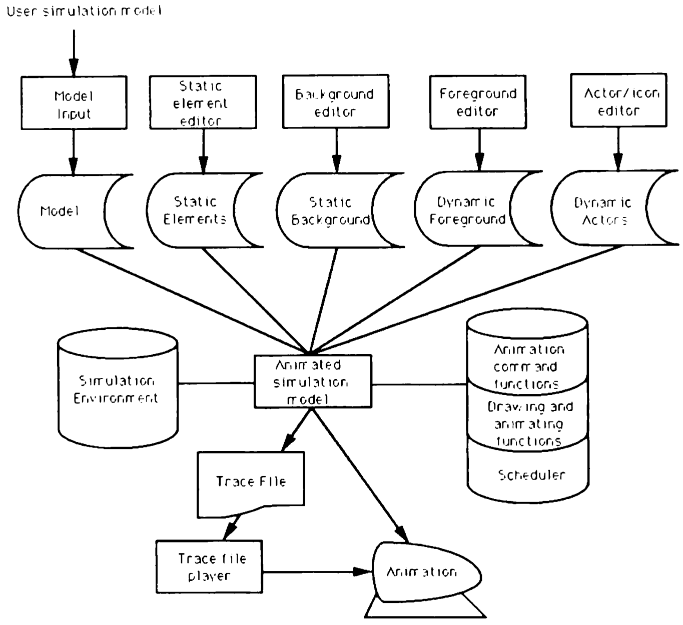


Figure 6 Specification and organization of an AS system

4 PASIMT/SIMGRAPH: AN ANIMATED SIMULATION MODEL CONSTRUCTION

A construction of an AS model, SIMGRAPH, is illustrated in order to have more concrete idea of a proposed AS system. It has been previously developed by the Industrial Engineering Department at National Tsing-Hua University (NTHU), Taiwan R.O.C. SIMGRAPH is an animator which is cooperative with PASIMT (Lin 1990), a discrete-event simulation package also developed by NTHU. It is bitmap-based and designed for IBM PC or compatible with a VGA/EGA display. PASIMT is a collection of procedures and functions that allow discrete event simulation programs to be easily developed in TURBO PASCAL. The package, which implements the event view, has procedures for creating and deleting entities, managing lists or queues, event scheduling and sequencing, system tracing and data collection. PASIMT makes it easy to add animation functions on the basis of Model-Animator-Scheduler paradigm, Since PASIMT has unique data structures and a time advancing mechanism.

Consider a *tanker-port example* (Pritsker 1986). The model is formulated through a group of event handling routines, since PASIMT is event-oriented. Animation commands are in a form of callable Pascal procedures mixed with PASIMT codes.

SIMGRAPH uses the block-based animation type on a bit map display for considerations of animation performance. The utilities include:

•*Background generator*: SIMGRAPH provides a utility, SCRNDUMP.EXE, to capture screen image and save into a binary data file as the input of SIMFILM.EXE to generate background file. The background image can be produced by graphic packages or CAD programs whose screen output will be captured by SCRNDUMP. The screens on operating SIMFILM are shown in Figure 7.

•*Static elements / layout editor*: SIMGRAPH generates a static element file from a formatted text file that the user can directly input and edit on a text editor. The definitions of static elements include *static text string location*, *string output*, and *color assignment*.

•*Actors/icon editor*: ICED.EXE provides the functions of editing and the libraries for the maintenance of actor outlooks. The screens on editing the image of a tanker under ICED are shown in Figure 8.

•*Foreground generator*: SIMBUILD.EXE provides an interactive foreground generating facility. The definitions of dynamic foregrounds includes: (1) *paths of actors* through specification of a path line on screen, (2) *displays of system status* through specification of the locations of queues, resources and icons, and (3) *statistics display* through specification of locations of bar-charts, pie-charts, and trace plots. The screens on adding paths under SIMBUILD are shown in Figure 9. The output of the tanker-port system example is shown in Figure 10.

The total software organization of SIMGRAPH/PASIMT is summarized in Figure 11. It is a typical implementation under the previously proposed specification. Users construct basic elements of an animated simulation model through usage of SIMFILM, SIMBUILD, ICED and a text editor. PASIMT and SIMGRAPH libraries are linked to the executable simulation program after the user model program

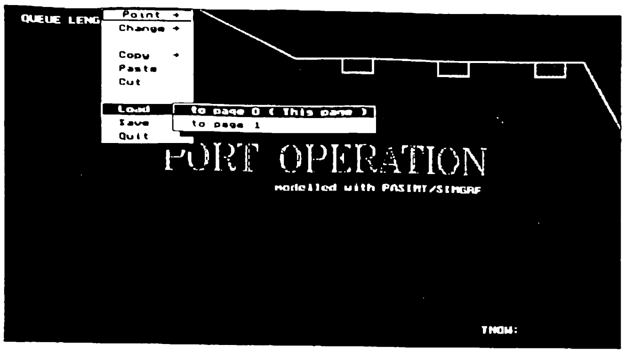


Figure 7 SIMFILM operation

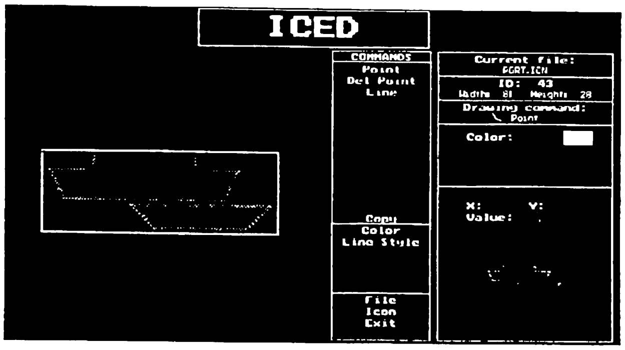


Figure 8 ICED operation

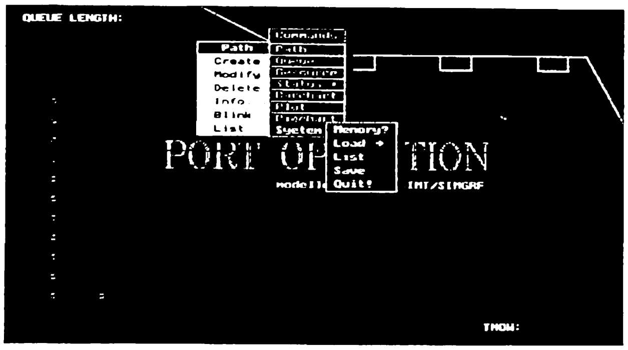
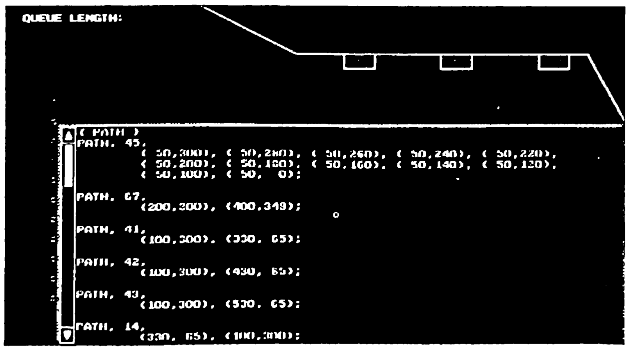


Figure 9 SIMBUILD operation



```
( PATH )
PATH, 45,
       ( 50,300), ( 50,280), ( 50,260), ( 50,240), ( 50,220),
       ( 50,200), ( 50,180), ( 50,160), ( 50,140), ( 50,120),
       ( 50,100), ( 50,  0);

PATH, 67,
       (200,300), (400,349);

PATH, 41,
       (100,300), (330, 65);

PATH, 42,
       (100,300), (430, 65);

PATH, 43,
       (100,300), (530, 65);

PATH, 14,
       (330, 65), (100,300);
```

Figure 10 Dynamic foreground output from SIMBUILD

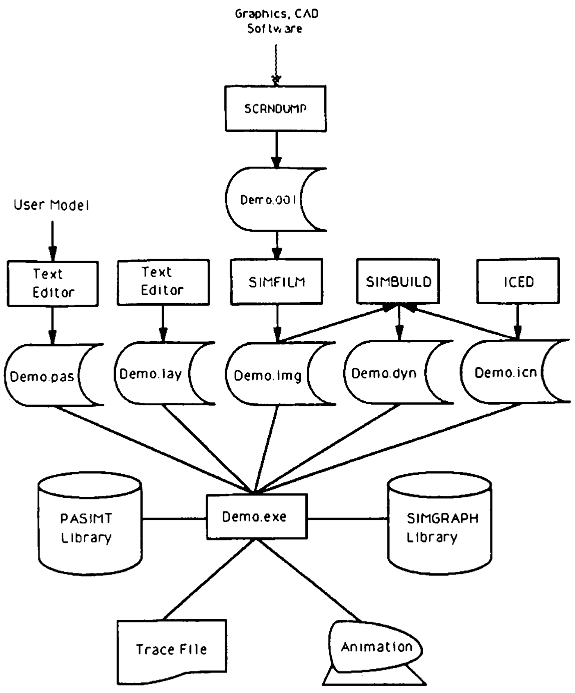


Figure 11 PASIMT/SIMGRF software organization

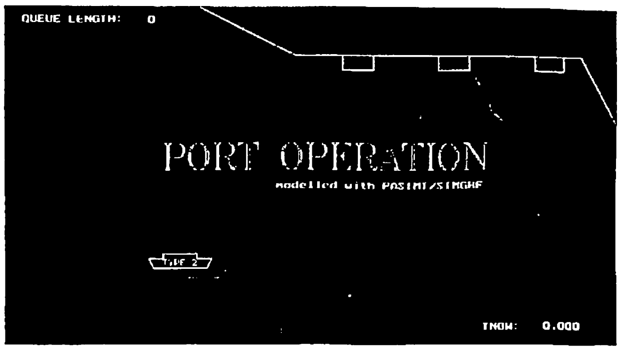


Figure 12.a Execution of tanker-port example

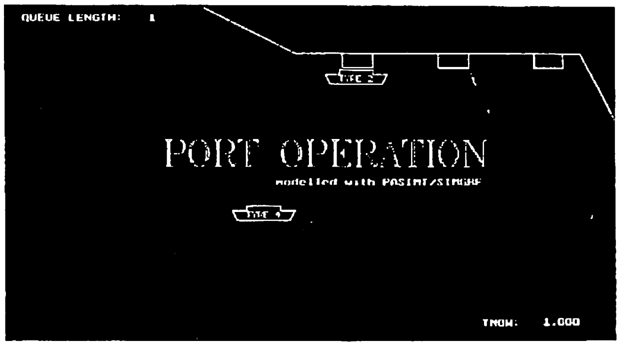


Figure 12.b Execution of tanker-port example

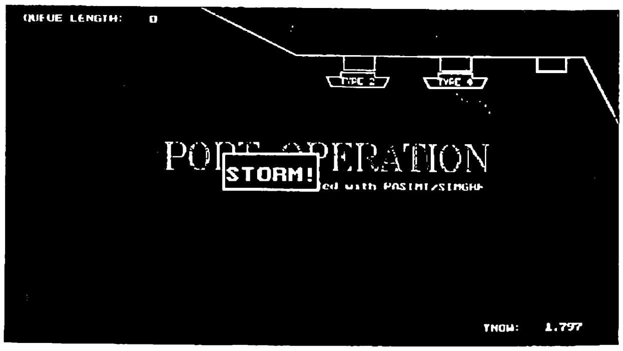


Figure 12.c Execution of tanker-port example

compiled. The simulation program reads those data files provided by each utility and first performs initialization tasks. It then proceeds on to the model execution and animation. The execution sequences of the tanker-port system example are shown in Figure 12.

5 CONCLUSION

A design framework has been proposed on the basis of a Model-Animator-Scheduler paradigm. Several different approaches have been unified into a single methodology during this research. Advantages of such a design framework have included:

•**Modularization**: From the modularized nature of the structure, such an AS system would inherent the benefit of modularization and extendibility. Since each module is functionally independent, different computer display types and animation technology, e.g., geometric display and frame-based animation, could be applied without changing other modules.

•**Flexibility**: Applications of such an AS system could have both flexible and easy-to-use properties. This design framework has fallen into a spectrum bounded by these two levels of flexibility. A developer could easily choose a proper scheme from the framework.

•**Applicability**: An AS system of this framework could provide a wide range of applications for different specific domains. Different applications could possibly require different variant schemes of AS. This structure has become adaptive for them.

An experimental AS system, SIMGRAPH, has also been presented here under such a framework. It can fully represent a typical implementation of a general purpose AS system, eventhough it is simply an experimental work which demonstrates the framework. Further research of the detailed development and organization of SIMGRAPH has still been underway.

REFERENCES

Bishop, J. L. and O. Balci. 1990. General Purpose Visual Simulation System: A Functional Description. *Proceedings of the 1990 Winter Simulation Conference*. 504-512.

Brunner, D. T. and J. O. Henriksen. 1986. A General Purpose Animator. *Proceedings of the 1986 Winter Simulation Conference*. 155-163.

Brunner, D. T. and J. O. Henriksen. 1991. Proof Animation: A General Purpose Animator. *Proceedings of the 1991 Winter Simulation Conference*. 90-94.

Cox, S. 1987. Interactive graphics in GPSS/PC. *Simulation* 49:3 117-122.

Lin, J. T. 1990. PASIMT: A Discrete Event Simulation Tool Kit in PASCAL. *Journal of Management Science* (Chinese Management Association), Vol 7, No. 2, Dec. 213-231.

O'Keefe, R. M. 1987. What is Visual Interactive Simulation ? (And Is There a Methodology for Doing It Right ?). *Proceedings of the 1987 Winter Simulation Conference*. 461-464.

Pritsker, A. A. B. 1986. *Introduction to Simulation and SLAMII*. System Publishing Corporation.

Rooks, M. 1991. A Unified Framework for Visual Interactive Simulation. *Proceedings of the 1991 Winter Simulation Conference*. 1146-1155.

AUTHOR BIOGRAPHIES

JAMES T. LIN is an Associate Professor in the Department of Industrial Engineering at National Tsing-Hua University (NTHU), Taiwan R.O.C. He received his Ph.D. degree in Industrial Engineering at Lehigh University in 1986. His current research interests include simulation modeling methodology, performance evaluation of manufacturing systems, and modeling of Automated Guided Vehicle System. He is a member of IIE, SCS, and IEEE.

KUANG-CHAU YEH is a Ph.D. student at the Industrial Engineering Department at National Tsing-Hua University (NTHU), Taiwan R.O.C. He received his M.S. degree in Industrial Engineering from NTHU in 1989. His research interests include the application of AI and simulation.

LIANG-CHYAU SHEU is a Ph.D. student at the Industrial Engineering Department at National Tsing-Hua University (NTHU), Taiwan R.O.C. He received B.S. and M.S. degrees in Industrial Engineering from NTHU in 1988 and 1990. His research interests are focused on feature-based design, and simulation modeling.