Proceedings of the 1990 Winter Simulation Conference
Osman Balci, Randall P. Sadowski, Richard E. Nance (eds.)

COMPUTER ANIMATION WITH CINEMA

Jacob P. Poorte
Deborah A. Davis

Systems Modeling Corporation
The Park Building
504 Beaver Street
Sewickley, Pennsylvania 15143

ABSTRACT

Cinema is a general purpose animation system designed to animate models developed using the SIMAN simulation language. The Cinema package consists of two separate modules: CINEMA is used to draw the graphical images used in the animation and CSIMAN executes a SIMAN simulation model and associated animation.

Animation typically centers around the presentation of final modeling results. However, the simplicity of developing Cinema animations allows users to exploit animation in more phases of a simulation project. By using Cinema, contributions of animation can be found throughout the entire simulation project from initial model debugging through presentation of the final simulation results.

Recent enhancements to Cinema IV include accumulating conveyors, AGV modeling features, dynamic plots and histograms, symbol fetch from backgrounds, and an improved user interface.

Cinema is available on a wide variety of platforms and operating systems. These include microcomputers under DOS and OS/2, as well as Sun, Apollo, and VAX workstations under the UNIX and VMS operating systems.

1. INTRODUCTION

In recent years animation has become fully accepted as an integral part of the simulation process. One of the reasons for this rapid acceptance is that numeric summary statistics do not necessarily convey information about the dynamic interactions of components of a system. Although summary statistics are a crucial part of evaluating the performance of a simulated system, it is only through animation that the analyst can easily identify the system status under which, for example, bottlenecks occur. Using the general purpose simulation language SIMAN (Pegden, 1990), in conjunction with the Cinema animation system, offers a means of analyzing both the system performance and the dynamics during the simulation of any discrete-event system.

Benefits of animation are not restricted to bottleneck analysis only. Johnson and Poorte (1988) identified four simulation modeling phases where the modeler may benefit from using animation. They are:

1. Debugging and Model Verification
2. Model Validation
3. Bottleneck Analysis
4. Communication and Presentation

1.1 Debugging and Model Verification

Animation provides a tool to visually inspect that the model is behaving as intended when creating the model. For example, logical errors that were mistakenly introduced during the initial model development are usually easily and rapidly 'caught' by watching a running animation. If these errors remain undetected, they may result in incorrect conclusions, which in turn could lead to incorrect decisions involving large associated costs.

1.2 Model Validation

Validation requires thorough knowledge and understanding of the intricacies of the system under consideration. The analyst may have the required awareness but by using simulation only one can never be totally sure that the model is a good representation of the real system. Animation provides the analyst with a powerful tool to ensure that the model is indeed a sufficiently adequate representation of the real system. For example, the impact that a simplification may have on the model is readily seen in a running animation. Assumptions that have a profound effect on the performance of a model may go unnoticed without the use of animation, which, again, could lead to inaccurate results and incorrect conclusions. As a model validation tool, animation allows the modeler to make reasonable simplications and other model assumptions.

1.3 Bottleneck Analysis

As an analysis tool animation can play a major role in bottleneck analysis. Using animation, the modeler has the ability to observe interactions of several simultaneous and interrelated events, thus providing information unavailable in aggregate statistical performance measures. Bottleneck situations are usually simple to locate with animation. Also important are the preconditions under which these and other unfavorable situations occur. From observing an animation these factors become easily apparant. They also may readily point to candidate improvements which would not have been obvious at all from the statistics alone.

1.4 Communication and Presentation

The ultimate objective of any simulation project is to provide credible information to the decision maker. Credibility of a project is strongly associated with the ability of the modeler to effectively communicate during the presentation of the results of a simulation project. Animation gives the analyst a very powerful vehicle to provide the non-expert with insight into a complex model and to explain the proposed solution.

It should be stressed that animation cannot replace standard statistical analysis techniques. However, the above discussion illustrates that with animation the modeler has access to a powerful tool in addition to standard simulation analysis procedures. As such, animation has the ability to effectively change and enhance many phases of the simulation modeling process.

2. DESIGN PHILOSOPHY OF CINEMA

Designing and building realistic simulation models is a complicated and often time-consuming process. Cinema allows the user to readily construct useful graphical displays and to animate SIMAN simulation models with little effort.

Typically, animation centers around the presentation of final modeling results. Yet, the versatility of Cinema allows the user to exploit animation throughout a simulation project from initial model debugging through the presentation of the final modeling results. Certain features of the architecture of the Cinema animation package allows for this versatility.

Three main design objectives for Cinema were:
1) Simplicity
2) Flexibility
3) Effectiveness

The simplicity of use of Cinema permits easy mastery of the task of building an animation; its flexibility allows development of animation of all sorts of systems; and its effectiveness greatly assists in communication of the results of a simulation study.

2.1 Simplicity

A common objective for software architects is that their products be easy to use. Yet, the level of sophistication and consequently complexity of many software products is such that special training is required before users are able to make good use of it.

One key element in the design of Cinema, is its ease of use; no special training is required to readily construct impressive animations. A mouse-controlled user-interface with which users select items from a hierarchy of pop-down menus is the way in which the animation layout is developed. With Cinema, *no programming* is required to build an animation. The user can fully concentrate on developing useful, realistic models and the data to support the analysis, rather than on developing pages of programming statements to drive the animation.

2.2 Flexibility

The Cinema animation package is intimately tied to the SIMAN simulation language. SIMAN is a general purpose language and has been used in a wide variety of environments, such as manufacturing, communication, health care, and defense. Cinema would be used in the same environments, therefore it was absolutely necessary that the same flexibility would be incorporated in the animation. Similar to SIMAN, Cinema is a general purpose modeling system capable of handling a broad spectrum of systems.

Johnson and Poorte (1988) have proposed a hierachical approach to computer animation in simulation modeling. Under their approach, animation is used during all phases of a modeling project. Animations evolve from simple debugging tools, to interactive analysis tools, until the final presentation of the results as a powerful communication vehicle. The associated animations are of increasing complexity. The authors argue that Cinema is unique in that it provides the flexibility necessary for a multi-level approach to computer animation.

2.3 Effectiveness

Effectiveness of an animation system is measured by how valuable the graphical images are during the different phases of a simulation project. Cinema's design assures effective use of animation. It can play the role of a verification and validation tool in the early phases of a project. In the next phase Cinema becomes a powerful analysis tool in which bottlenecks can be detected and many what if question can be asked through the interactive debugger feature. In the final phase of a project, impressive animations can be constructed with Cinema and it becomes a communication tool.

3. BUILDING A CINEMA ANIMATION

The user interacts with Cinema by using a mouse-controlled graphics cursor to pick items from pop-down menus as shown in Figure 1. Context sensitive help is available online simply by pointing to an item and clicking the right mouse button. With the left mouse button the user activates the function pointed

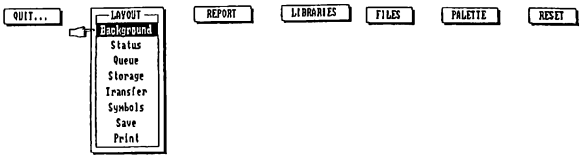

Figure 1. The Cinema Interface

The Cinema layout is a graphical representation of the system being simulated. The layout consists of graphical objects grouped into two conceptual categories: the static component and the dynamic component. The static component of the layout does not change during the execution of the simulation model, while the dynamic component consists of graphical images which do change color, location or shape.

3.1 The Static Component

The static component of a layout represents the physical environment in which the simulated system exists. It would include anything that helps to visually recognize the scene, but which does not change during model execution. For example, in an animation of a manufacturing facility the static component would consist of the walls, aisles, storage bins, and offices.

The static component (called background in Cinema) can be created by using the drawing facilities provided in the CINEMA program. Under the draw menu drawing functions are available for line, box, circle, bar (a box filled with a solid color or a hatch pattern), arc, and freehand sketch. Drawing options can be modified by selecting a different color, line style, and/or line thickness for each drawing function. Text can be added to the background at any location and in any orientation. Text options include any combination of six different sizes, with six different fonts, and 16 active colors. Explode allows the user to zoom in for fine detailing of small portions of the drawing, while Cut Area provides a means of copying or moving rectangular regions of the background.

Line, box, bar, circle, and arc are all performed in "rubberband" mode where the user digitizes, for example, the center of a circle somewhere on the screen and then can choose the radius of the circle by moving the mouse away from the center. As the mouse is moved, the circle will continuously expand or shrink depending on whether the mouse is being moved away from or toward the center of the circle. When the user clicks the left button, a permanent circle is drawn with the current settings for color, width and radius.

An alternate method for generating a background is to start with a DXF-formatted file. DXF files are generated by many of the computer-aided design (CAD) packages available, such as AutoCAD, PC-CAD, CADVANCE, etc. Many of these packages have sophisticated drawing functions, especially for three-dimensional images, allowing the user to develop geometrically accurate and more complex backgrounds. DXF-formatted files are ASCII files, therefore any DXF drawing generated on any hardware platform can be input into Cinema. A utility program called DXF2LAY is available for this purpose. DXF2LAY converts the drawing into the static component of a layout. An example of a layout generated from a DXF file is illustrated in Figure 2.

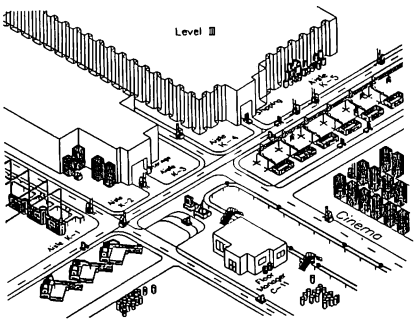

Figure 2. Cinema Layout Generated with DXF2LAY

DXF-formatted files represent vector-oriented graphic images (also called display lists), whereas Cinema represents a raster-oriented graphics system. Each system has its own advantages and disadvantages. By using the DXF interface the user obtains the best of both worlds. When using for example AutoCAD the user has access to powerful 3-D drawing commands, whereas CINEMA contains functions that cannot be performed by CAD packages, such as area fills and pixel-based editing. It is found that impressive displays have been generated by using AutoCAD to develop the "technical part" of the layout followed by CINEMA which is then used to fine tune the "aesthetic part" of the layout.

3.2 The Dynamic Component

The dynamic component of the layout consists of graphical images which change shape, color, size, or location in response to a change of status in the SIMAN model. The dynamic objects in a Cinema layout are directly tied to SIMAN modeling constructs. As the state of the constructs change during execution, the associated dynamic objects will change to reflect this change of the SIMAN construct.

The user may choose to animate as much or as little of the SIMAN model as is useful and necessary to convey pertinent information. If certain sections of the model have been omitted from the animation, the model will continue to run unhindered.

A large number of dynamic objects are available; each one is discussed in some detail below.

Entities - In a SIMAN simulation model, entities represent the items that are being processed or flow through the system. An entity is represented in a Cinema layout as a moving and/or changing symbol. The symbol could be a sketch of a workpiece or a packet of information moving down a communications line. As the entity moves from work area to work area within the SIMAN simulation model, its corresponding symbol is automatically moved across the static background in the Cinema layout.

Entity symbols are created by drawing the icon on a blown-up grid ('fat bits'), using the mouse-controlled cursor. The same drawing functions available in background are provided through pull-down menus. Each box in the grid corresponds to one picture element (pixel) of the actual symbol image, and therefore allows great flexibility for detailing. As the symbol is created on this grid, it is simultaneously displayed in actual size in the upper left corner of the screen, as shown in Figure 3.

The entity icons are created and stored in an entity symbols library. Symbol libraries are maintained separately from the animation layout and may be saved (stored on disk) and recalled at will. This allows a user to save time by reusing symbols that were previously created.

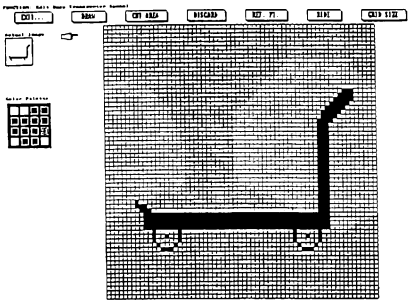

Figure 3. Creating an Entity Symbol

To establish the association between a graphical entity symbol and a specific entity within the SIMAN model, the user must identify one of the general purpose entity attributes in the SIMAN model. The modeler is responsible for using that attribute to assign an entity number to entities in the SIMAN model. In Cinema, a specific symbol from an entity library is then associated with a particular entity number. If the designated entity attribute changes values in the SIMAN simulation model, the corresponding entity symbol changes in the animation. Consider, for example, a simulation of an automotive assembly plant. The designated attribute of an entity arriving to an assembly station might be set to a value that corresponds to a symbol of a car body without doors. After leaving the workstation, the associated symbol could be changed to a car body with doors simply by assigning a new entity number to the designated attribute in the SIMAN model.

Resources - Resources are used in SIMAN to model limited items in a system, such as machines and workers. Entities compete for the limited number of units of a resource and incur queueing delays when enough units are not available. In a SIMAN model, each resource assumes one of four possible states: idle, busy, inactive, or preempted. The resource is in a busy state when it has been allocated to an entity at a *SEIZE* block in SIMAN. The resource remains busy until it is released by the entity at a *RELEASE* block. Once released, its status changes to idle (making it available to be re-seized by other entities). Units of a resource can be removed or made unavailable by using an *ALTER* block. This changes the status of those units of the resources to inactive until they are placed back in service using the *ALTER* block again. When a resource is allocated to an entity at a *PREEMPT* block in SIMAN its state is preempted until it is released by the entity at a *RELEASE* block.

Resource status changes within a SIMAN model are displayed in a Cinema animation by using resource symbols that represent the resource in each of the possible states. Like entity symbols, resource symbols are created by drawing icons on the enlarged grid, with the mouse cursor. They are stored in a resource symbol library file that can be recalled and used with other models.

A resource is added to the layout by selecting an idle symbol from the library and positioning it on the static background, using the mouse. At that time the user must also associate a SIMAN resource with that symbol (as entered in the experiment frame of the SIMAN model). When the status of a resource changes in the SIMAN model, the associated Cinema resource symbol (busy, idle, inactive, or preempted) is displayed at the designated location on the screen.

Queues - A Cinema queue is a dynamic representation of the entities residing in an associated SIMAN *QUEUE* block. These entities might represent workpieces awaiting the availability of a machine, a set-up operator, a space on a conveyor, etc.

A queue can be added to the layout at any location, and in any length and orientation. The mouse-controlled graphics cursor is used to digitize a first point corresponding to the head of the queue. As the graphics cursor is moved to locate a second point corresponding to the tail of the queue, a line segment is rubberbanded between the head and current tail point to show the length and orientation of the queue. Pressing the left button on the mouse digitizes the second point and adds the queue to the layout. This graphical representation is then associated with a specific queue number in the corresponding SIMAN simulation model.

When an entity enters a queue in the SIMAN model, the entity's symbol is displayed along the corresponding queue symbol at the proper location relative to the other members of the queue. When an entity exits the queue in the model, its associated symbol is removed and all following symbols are moved forward one position. When the queue in the SIMAN model contains more entities than can be displayed along the fixed length of the graphical queue symbol, subsequent arrivals to the queue are not displayed. The entities that are not displayed will eventually become visible as they move forward into the visible portion of the queue as preceding entities exit. Notice that the queue symbols themselves do not appear during the animation; only visible are the symbols that are in the queues.

Stations - Distinct workcenters within a system are modeled using the SIMAN *STATION* construct. In Cinema a station symbol is used to designate start and end points for the movement of entities between the workcenters in the SIMAN model. The station symbols, like the queue symbols, do not appear during the execution of the animation; they are only used to govern the placement of the entity symbols as they move through the system. Station symbols are added to a layout by simply entering the station number and clicking the left mouse button to digitize the desired spot.

Routes - The Transfer menu, under the CINEMA Layout menu, is used to define the paths of travel between the stations of the SIMAN model. One method for modeling the movement of entities between *STATIONS* is to use the *ROUTE* block. When an entity reaches a *ROUTE* block in the SIMAN model, its associated entity symbol is continually redisplayed at new points along a predefined Cinema route path to produce the effect of movement.

A route can be placed anywhere on the layout and consists of one or more rubberbanded line segments that are connected to form the path. To add a route to a layout, the user must begin the path inside the starting station symbol, draw the intermediate path segments and end inside the destination symbol.

Movement that involves SIMAN material handling constructs is designated in much the same way as the routes. If entities are transferred by a *CONVEYOR*, the user will digitize paths called segments, and if the entities are moved by *TRANSPORTERS*, the paths will be called distances. All route, segment and distance paths are transparent during model execution, yet the entity symbols will trace these paths as they move.

Variables - While a simulation is executing, SIMAN automatically maintains the value of many status variables which define the system state. Examples of these status variables are: the value of simulated time, the number of entities in a queue, the number of parts processed, the number of busy machines in a workcell, the cumulative utilization of a worker, etc. A representation of any SIMAN status variable can be incorporated into an animation layout using one of four dynamic features in Cinema.

A digital display of the numeric value of a status variable can be added to a layout using a feature called dynamic variables. The user enters the variable and defines the format for display, range of values, size, and color using pull-down menus and then positions the variable anywhere on the screen using the mouse.

A second way to display status variables is with a feature called levels. This analog representation of a variable's value provides a means of displaying dynamic graphs. Three different level shapes are included in CINEMA: a box, a circle, and a dial. During execution of an animation, box or circle level fills and empties in response to changes in the value of the associated status variable. The dial is a circular level with a sweep hand that rotates either clockwise or counter-clockwise. Dials are typically used to represent simulated time with a clock.

A feature called global symbols represents a third way to display the value of a status variable. Like entity and resource symbols, global symbols are drawn on the enlarged grid and maintained in a separate global symbol library. To incorporate global symbols in a layout, the user associates selected symbols from a library with designated values of a specified system status variable. For example, a symbol saying "STARVED" could be displayed when the number in a queue for a machine is zero, and a second symbol saying "BLOCKED" could be displayed when the queue is full to capacity.

Dynamic colors represents a fourth way to display the value of a status variable. The user first associates one of the sixteen Cinema color indices with a specific status variable. The user then defines one or more different colors for that index (in the form of different combinations of red, green, blue intensities) and associates each new color with a designated value of the status variable. For example, in the simulation of a steel making facility, a status variable that represents the temperature of a furnace could be tied to the color index used to draw the furnace in an animation layout. Specific values of the variable (furnace temperature) could be associated with different combinations of red, green, and blue intensities that produce colors ranging from gray to red. As the temperature of the furnace increases in the simulation model, we would see the furnace gradually change from gray to red.

3.3 Recent Cinema Enhancements

With the recent release of SIMAN IV (Pegden and Sturrock, 1990), extensive enhancements have been made to the SIMAN simulation language. Significant additions include the ability to model accumulating conveyors, powerful AGV modeling constructs, user-defined variables and attributes, and flexible I/O statements. Cinema is fully compatible with these new SIMAN constructs.

In Cinema, two objects support the new AGV modeling features: intersections and links. Intersections, which are represented by a diamond shape, closely resemble the functionality of a station. Both stations and intersections represent locations at which some kind of transfer takes place. The difference lies in that stations are conceptual objects, whereas intersections represent physical zones in which a single AGV can stop, wait, make turns, and travel.

Links are analogous to routes, segments, and distances, in that they represent paths along which entities or transporters travel. Links, however, are used specifically to model AGV movements. Individual links are interconnected into AGV networks through intersections. The intersections are the nodes in these networks and represent the endpoints of one or more links.

The intersection and link constructs allow the user to build AGV systems ranging from simple loops to very complex network configurations. The SIMAN model uses standard SIMAN transporter blocks to simulate control of the vehicles. When building an AGV network in Cinema the user need only be concerned with the position and relative distances of intersections and links within the network. No other information is required for the animation.

In addition to the new SIMAN features, Cinema also incorporates enhancements not directly related to SIMAN changes. These include: dynamic plots and histograms, enhanced drawing

facilities, the ability to fetch an area from the background into a symbol library, and an improved user interface.

A new Cinema feature is the ability to dynamically display plots and histograms representing system status variables. As the system status changes during a simulation, these graphs are updated automatically to reflect these changes. The user has complete control over the appearance of these plots including size, color, overlayed plots, plot labels, and refresh rates.

With histograms, time-persistent information is displayed dynamically during an animation run. The same flexibility of the histogram feature in SIMAN's Output Processor has been incorporated in Cinema. Thus, the user has control over factors such as size of graph, width of the cells, number of cells, colors of cells and borders, presence or absence of cumulative lines, graph labels, etc.

In previous versions of Cinema, the selection of variables that could be displayed as a variable, level, or global symbol, were restricted to standard SIMAN variables. Presently, any user-defined variable and even complicated expressions can be used to be displayed at run time. As an example, it is possible to have a level display the total number of entities in three queues by entering NQ(1) + NQ(2) + NQ(3) as the expression to be displayed. Plots and histograms also accept expressions and user-defined variables.

Another enhancement to Cinema is the capability fetch a symbol from the static background into one of the symbol libraries. This is particularly useful to users who use the DXF2LAY utility to build Cinema layouts starting with DXF-formatted drawings. Now, users can essentially draw all their symbols using a CAD package, such as AutoCAD, convert the DXF-file into a Cinema layout and fetch the symbols from the background into a symbol library.

Cinema's user interface has been improved in several ways. For example, pop-down menus stay up when the hand curser moves away from the target area. Prompts have been made more informative and the addition of a drawing snapmode greatly enhances the production of geometrically correct layouts.

4. RUNNING AN ANIMATION

As discussed previously, the Cinema system consists of two components. So far we have discussed the capabilities of the layout generation program, CINEMA. We now turn our attention to CSIMAN which is used to run the simulation and render the animation.

CSIMAN is a special version of SIMAN which includes the ability to update the dynamic component of the layout interactively during execution as well as control the speed of execution.

The user interface to CSIMAN is identical to that of CINEMA with different menu items. Since Cinema is a real time animation system, as opposed to a post processed system, the CSIMAN menu includes facilities to temporarily stop a run and use the SIMAN interactive debugger features. The debugger allows a user to change variables, look at the entities in a particular queue, look at status variables not displayed on the animation, and monitor practically any variable in the system. A user can turn on the SIMAN trace function so that the animation can be viewed simultaneously with the model execution statements. This feature is very useful for validation and debugging. A stepping facility halts the model execution after each event is processed. Execution is resumed when the user presses the space bar and continues until the time the next event is processed.

The user may associate any number of layouts with a single SIMAN model. These layouts may be recalled from the CSIMAN menus or directly from the keyboard.

Another CSIMAN feature is to save a snapshot. This saves a picture of the layout at a specific instant in simulated time as well as saving the value of all the system variables. The snapshot may be recalled at a later time so that the simulation can progress from the time that the snapshot was saved. This feature is particularly useful in demonstrating and presenting critical situations and comparing variations of the system at the same moment in time.

The speed of the animation is controlled by both menus and the keyboard. The user can choose a scale factor which is an association between the simulation time and real time. The smaller the scale factor, the slower the animation will progress. The dynamic objects are updated as often as possible between changes of the simulation clock. During animation the simulation clock will progress smoothly. Consequently, if the state of the simulation does not change over a long period of time, the graphical image will remain static for a corresponding period of time. The scale factor can be dynamically increased and decreased by using the "<" and ">" keys. If the time of interest is later on in the simulation, the user may disable the animation and skip ahead to a predetermined time. The simulation will be performed as fast as possible, without updating the animation, so as to greatly reduce the time required to get to the desired simulation time.

In Cinema/HGA the user may also zoom and pan during the execution of the simulation. This feature has been found very useful for demonstration and presentation purposes.

5. HARDWARE REQUIREMENTS

Since the first release of Cinema, many new platforms have been added. The primary platform has been the IBM PC-AT (or compatible) and the DOS operating system. There are currently three graphic cards available for the PC/DOS configuration, namely the EGA card (or VGA card in EGA mode), a high-resolution card purchased through Systems Modeling, or the IBM 8514/A card. The EGA has a pixel resolution of 640x350 while the high-resolution cards (HGA and 8514) have a pixel resolution of 1024x768. Cinema also is available for IBM PC-AT or compatibles computers running under the OS/2 operating system. This version allows simulation of systems virtually unlimited in size.

Cinema is available on the Digital Equipment Corporation (DEC) VAXstation series of workstations, which includes MicroVAX workstations with the GPX upgrade; the Sun Microsystems Sun-2, Sun-4, and Sun 386i series; and the Apollo DN3000 series.

6. SUMMARY

Cinema is a general purpose animation system which allows for the easy animation of any SIMAN simulation model. Because Cinema utilizes user constructed icons, any type of system ranging from manufacturing, to distribution, to health care, transportation, and communications may be animated.

Three main design objectives of Cinema are: (1) simplicity, (2) flexibility, and (3) effectiveness. The mouse oriented, menu-driven user interface allows for the rapid development of animations without the need for programming. Cinema animations aid the analyst in the process of debugging and verification of the SIMAN model. Other benefits of Cinema can be found during model validation and analysis phases, as well as in the presentation of the final results.

REFERENCES

Johnson, M.E. and J.P. Poorte (1988), "A Hierarchical Approach to Computer Animation in Simulation Modeling," *Simulation 50*, 1, 30-36.

Pegden, C.D. R.E. Shannon, and R.P. Sadowski (1990), *Introduction to Simulation Using SIMAN*, McGraw-Hill, New York, NY.

Sturrock, D.T. and C.D. Pegden (1990), "Introduction to SIMAN," In *Proceedings of the 1990 Winter Simulation Conference*, O. Balci, R.P. Sadowski, and R.E. Nance, Eds. IEEE, Piscataway, NJ, 109-114