

ANIMATION DESIGN WITH CINEMA

Richard A. Kilgore, Vice President
Kevin J. Healy, Vice President
Systems Modeling Corporation
P.O. Box 10074
State College, PA 16805

ABSTRACT

Cinema is a general purpose animation system designed to work with the SIMAN simulation language. A sophisticated yet easy to use graphical interface supports a variety of graphical functions that allow users with little or no programming skills to build realistic and useful animations.

INTRODUCTION

One of the shortcomings of simulation has been that the output from a simulation model typically takes the form of summary statistics or simple graphs. Although these outputs are necessary to measure and draw conclusions about the performance of a simulated system, they provide little insight into the dynamic interactions between the components of the system. Since these stochastic behaviors and complex interactions are often the reason that the simulation study was undertaken, it is essential to also include observations of particular periods during the executing simulation. Failure to perform such observation is neglecting information which the simulation experiment is creating.

Computer animation represents an ideal mechanism for the observation of the dynamic behavior of a simulation model. In recent years, substantial efforts have been devoted to the development of new combined simulation-animation systems as well as animation interfaces to existing simulation languages.

In those applications in which animation has been employed, the benefits of observing the executing simulation model have been substantial. Three main benefits which have been cited are the following:

Improved Model Verification - A properly designed animation is an ideal method for verifying the correct operation of the simulation model. Subtle errors which might not be apparent from standard simulation output become obvious when the operation of the system is displayed graphically. Simple textual "traces" of the execution simulation are an inadequate substitute as the user must manipulate numerous, and often terse, sequential descriptions of events which are happening at the same simulated instant.

Enriched System Understanding - When simulating proposed systems which are not yet observable, the animated display is a knowledge-building and insightful experience which often generates new alternatives for further improvement of system performance. In those systems which support concurrent simulation-animation and dynamic user interaction, the simulation may be stopped or reset to allow the immediate test of a suggested alternative.

Enhanced Communication of the Solution - Animation is an extremely powerful aid in quickly and thoroughly communicating the results of a simulation model to decision makers. The realism and completeness of the animated display is often viewed as representative of the realism and completeness of the underlying model.

There exists, however, a potential to misuse animation. Often, simplifying assumptions can be made or a series of sequential events can be combined in the simulation model without altering the validity or usefulness of the model. When designing the animation, there is a tendency to add unnecessary detail to the model for the sole purpose of producing a more realistic animated display. As animation graphics become more advanced, modelers are frequently called on to make a large number of such model enhancements for the sole purpose of avoiding disturbing questions regarding the unimportant, but observed discrepancies between the animation and the real system.

A more serious misuse of animation has been the suggestion that decisions may be based solely on the output of an animation. An animation is not a substitute for the controlled experimentation and analysis of output from a simulation model. It should be used in conjunction with other analysis aids to gain a detailed understanding of the operation of the model. Simulation is a sampling experiment and it must be recognized that the period of simulated activity being viewed is only one of many realizations of the stochastic behavior of the system. It may not be generally representative.

Another drawback in the past has been the substantial amount of effort and programming skills required to generate even simple animations. Since timeliness is such a crucial necessity for most simulation studies, the addition of complex, graphical programming may detract from the time allocated to data analysis, model development and validation. The validation process may be lengthened if programmed animation is necessary since modeling errors may not be observable if the animated movement shows a correct display. Ideally, the modeler should be able to quickly and easily generate simple animations for use in the development and debugging of a model as well as highly detailed animations for presentation purposes.

These considerations provided the motivation for the development of Cinema, an animation design system which supports graphical display of SIMAN simulation models. A Cinema animation is a dynamic display of graphical objects that change size, shape, color or location on a static background in correspondence with changes in a concurrently executing SIMAN simulation model. The graphical objects may represent the state of various physical components of the system or communicate the value of a system variable, model input parameter, or statistical result.

The Cinema animation design system does not generate a simulation model. This is accomplished using, with minor exceptions, the standard SIMAN simulation language. The Cinema program is then used to construct an animation layout which is a graphical depiction of the simulated system being modeled. The SIMAN simulation model can then be executed with or without the Cinema layout to generate a graphical animation of the dynamic behavior of the system.

The development of a Cinema animation requires no programming skills because of the *mouse/menu* user interface which is based on pioneering developments made at the Xerox Palo Alto Research Center (PARC) in the mid-seventies [1].

The mouse is a hand-held pointing device that controls the motion of a cursor on a graphics screen. By moving the mouse across a desk top, the cursor is guided to symbols on the screen which represent the commands of the Cinema program. By pressing a button on the mouse, you can either select the commands you want to perform, or request a help-screen on the function of the command. The mouse is also used as an instrument for drawing and positioning animation components on the screen.

The two screen symbols used to represent commands in the Cinema program are called *headings* and *menus*. The commands are organized in a hierarchy of levels. At each level, the commands are represented by a set of headings arranged across the top of the screen. When you position the cursor on a heading and press a button on the mouse, a menu "pulls-down", revealing a list of secondary choices. A copy of an actual screen image in Figure 1 illustrates the operation of the mouse in conjunction with the layout menu.



Figure 1: Pull-down Menu

HARDWARE REQUIREMENTS

The Cinema system is designed to run on a variety of microcomputer and engineering workstations. An IBM PC-AT or a 100% compatible microcomputer such as the Compaq 386 or IBM PS/2 Model 60 with at least 640K bytes of memory is required to run two versions of Cinema under the DOS operating system. The Cinema/EGA implementation supports 640x350 resolution in 16 colors on standard Enhanced Graphics Adapters with 256K bytes of video memory. Superior quality, high-resolution animations are available through Cinema/HGA which requires a proprietary, high-resolution (1024x768, non-interlaced) color graphics display board and a high-resolution monitor.

Cinema/SUN and Cinema/GPX are exceptionally powerful Cinema implementations on the SUN 3C and 4C engineering workstations under the Unix operating system and the VAXstation II/GPX workstation under the VMS operating system. These recently developed additions to the Cinema family provides an ideal combination of the fast, large-model development capabilities of minicomputers, with the sophisticated graphical capabilities and user-friendly development environment of the personal desktop microcomputer.

THE ANIMATION LAYOUT

The Cinema animation layout is a user-defined combination of graphical objects that form a representation of the system being modeled. The objects that comprise a layout are one of two types referred to as *static* and *dynamic*. The static objects within a layout form a background which will not change during the animation. In a simulation of a manufacturing system, this might correspond to a diagram of the walls, aisles and fixtures of the facility being modeled.

The dynamic objects are superimposed on the static background and change size, shape, color, or location in correspondence with changes in the state of the system during the execution of a simulation. Workpieces, workers, machines, and transporters would be represented as dynamic objects within a layout. Dynamic objects also include the changing digital and graphical representation of the values of system variables and summary statistics. An example of a Cinema animation layout is shown in Figure 2.

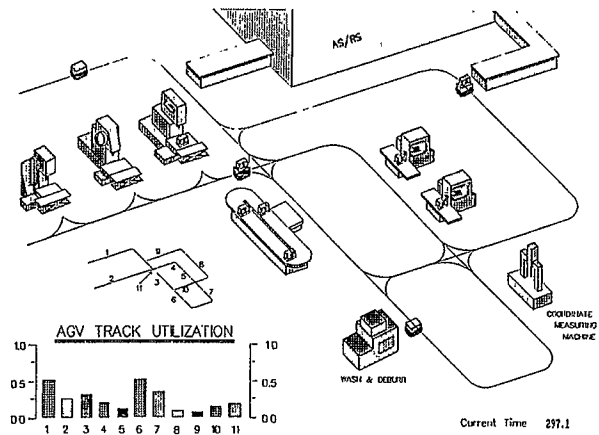


Figure 2: Animation Layout

The Static Component

The static component of the layout is constructed using a set of elementary computer-aided drawing functions such as line, box, bar, circle and arc. The mouse is used to choose the function from the Cinema headings and menus. The user may also use the mouse to set attributes of the drawing function such as the color, width, style and orientation of the line. The basic functions are drawn in a *rubberband* mode, which allows the user to view the size and orientation of the object before it is actually added to the layout. For example, a box is drawn by using the mouse to position the cursor on a screen location that represents one corner of the box before pressing a button on the mouse. You then see the box expand and contract as the cursor is moved to locate the opposite corner of the box. After positioning the opposite corner, you press the mouse button a second time to add the box to the layout.

Other features include a *sketch* function which allows the construction of any free-form curve by simply moving the cursor along the desired path while holding down a button on the mouse. A *fill* function paints any enclosed region with either solids or patterns. *Text* can be added to the static background in a choice of fonts, sizes, colors and orientations that are selected using the pull-down menus. Editing features for manipulating images on the screen include an *erase* function which erases all graphics and text in the path of the cursor as long as a button on the mouse is depressed. Portions of the screen may be moved or copied to other areas with the *move* and *copy* functions.

Through the use of the *explode* function, more detailed drawing operations can be performed using many of the previously described drawing functions on individual picture elements (pixels). Examples of the various drawing functions are illustrated in Figure 3.

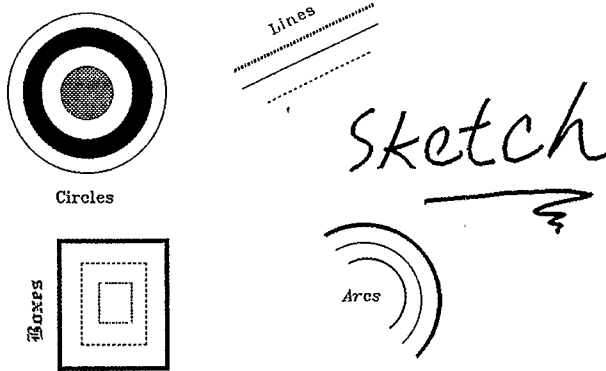


Figure 3: Examples of Drawing Functions

The AutoCAD DXF Interface

The DXF/Cinema Interface is a feature which makes it possible to convert an AutoCAD™ Data Exchange File (DXF) into a Cinema layout file and use it as the background for an animation layout. It is a separate utility which is included with the Cinema software. In addition to AutoCad, the DXF/Cinema Interface can be used with other graphics or CAD systems that generate drawing files which are compatible with the AutoCAD DXF definition.

This is particularly convenient in manufacturing simulation projects where layouts already exist as CAD drawings. The combination of the vector-based graphics of a CAD package with the pixel-based graphics of Cinema present expanded drawing opportunities. The example Cinema layout in Figure 4 illustrates an example of a static background translated from a DXF file.

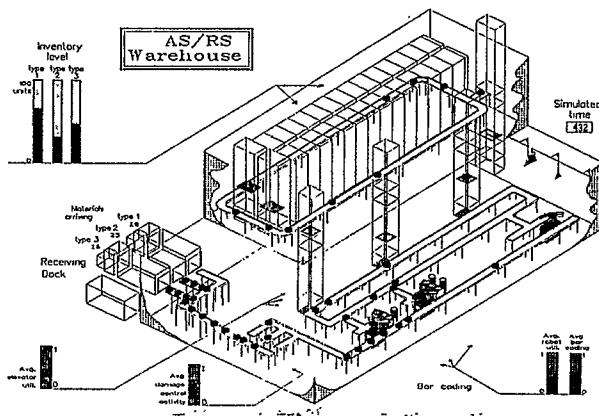


Figure 4: Layout Generated by DXF/Cinema Interface

The Dynamic Component

While a SIMAN simulation model is executing, it is continually updating an internal representation of the state of the system. The state of the system is defined by the current values of status variables such as the number and location of entities within the system, the values assigned to attributes of the entities, the status of the resources within the system, and the current simulated time.

Each dynamic object in an animation layout is associated with a specific element of the system status as represented by the SIMAN model. For example, one object might be associated with the value of a variable such as the simulated time while another might be associated with the status of a SIMAN resource. Dynamic objects are automatically updated as the state of the system changes during the simulation. Following is a discussion of the dynamic objects which can be incorporated into a Cinema layout and the association between these objects and specific modeling constructs within the SIMAN language.

Entity Symbols

A SIMAN process model is a step-by-step description of the processes (creation, queuing, processing delays, etc.) an entity encounters as it flows through a system. In an animation, an entity is represented as a moving and/or changing symbol. The symbol could be a sketch of a workpiece or a partially assembled car. As the entity moves from workstation to workstation within the simulation model, its corresponding symbol is automatically moved across the static background.

Entity symbols are created by coloring boxes on a screen sized grid. Each box in the grid corresponds to one pixel of the actual symbol image. As the symbol is created on the grid, it is displayed in actual size in the upper left corner of the screen. An example of a symbol drawing grid is shown in Figure 5.

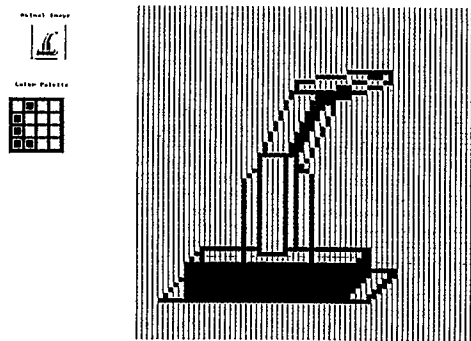
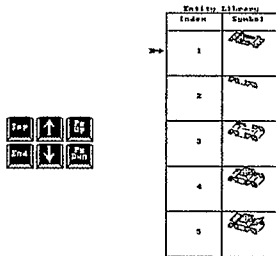


Figure 5: Symbol Drawing Grid

The symbols are stored in an *entity symbol library*. Symbol libraries are maintained separately from the animation layout and may be saved (stored on disk) and recalled for later use.

A general purpose entity attribute must be reserved in the SIMAN model to establish the association between an entity symbol and a specific entity within the SIMAN model. This attribute is used to store a number that is associated with a specific Cinema symbol from an entity symbol library. When the designated entity attribute is assigned a value in the model, the corresponding entity symbol is displayed in the animation.

The symbol can be changed by assigning a new value to the entity attribute. 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 value to the designated entity attribute. An example entity library is shown below in Figure 6.



| Index | Symbol |
|-------|--------|
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |

Figure 6: Example Entity Symbol Library

Resource Symbols

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 queuing delays when shortages occur. In a SIMAN model, a resource assumes one of four possible states: busy, idle, inactive, or preempted. The resource is in a busy state when it has been allocated to an entity at a SEIZE block. The resource remains busy until it is released by the entity at a RELEASE block, which changes its status to idle. Units of a resource can be removed from use with an ALTER block. This changes the status of those units of the resource 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 its state is preempted until it is released by the entity at a RELEASE block.

Resource symbols are used to display changes in the status of a resource. A resource is represented by four distinct symbols called the idle, busy, inactive, and preempted symbols. Like entity symbols, resource symbols are created by coloring boxes on a screen sized grid and stored in a resource symbol library. 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. The SIMAN resource number must then be associated with the symbol (This is one of the reasons why the SIMAN model should be built before the creation and positioning of dynamic symbols).

Transporter Symbols

In SIMAN, the term transporter denotes a general class of movable devices that may be allocated to entities. Examples of devices that might be modeled as transporters include lift trucks, automated guided vehicles and cranes.

In a SIMAN model, a transporter assumes one of three possible states: busy, idle, or inactive. The transporter is in a busy state when it has been allocated to an entity at a REQUEST or ALLOCATE block. The transporter remains busy until it is freed by the entity at a FREE block, which changes its status to idle. A transporter can be removed from use with a HALT block. This changes the status of the transporter to inactive until it is placed back in service using the ACTIVATE block.

Transporter symbols are used to display changes in the status of a transporter. A transporter is represented by three distinct symbols called the idle, busy, and inactive symbols. Like entity and resource symbols, transporter symbols are created by coloring boxes on a screen sized grid and stored in a *transporter symbol library*.

A transporter is added to the layout by associating a SIMAN transporter number with an idle symbol from the transporter symbol library. When the status of a transporter changes in the simulation model, the associated transporter symbol (busy, idle, or inactive) is displayed to reflect the status change.

Transfers

Distinct workstations within a system are modeled in SIMAN as station submodels. The movement of entities and material handling devices between station submodels is accomplished through the use of three transfer blocks, ROUTE, TRANSPORT and CONVEY. Entities transferred from these blocks later arrive at STATION blocks within the SIMAN model. In Cinema, the tasks involved in defining the paths on the screen that entities follow when they are transferred between stations are grouped into the TRANSFER section.

The first step is to position *station symbols* on the layout. These symbols represent the beginning and end of any route, transporter, or conveyor path. Each station symbol is numbered to match the corresponding station number in the SIMAN model. A route, transporter, or conveyor path is then defined as a series of connected line segments between two station symbols. When an entity is transferred between two stations in the model, its entity symbol is periodically displayed along the corresponding path on the screen. When an entity uses a transporter to move between stations, both the entity symbol and the busy transporter symbol are shown moving along the path.

Queues

Queues are the result of delays that entities incur while waiting for a prescribed change in the state of the system to occur. For example, entities must wait in line when a resource, transporter, or space on a conveyor is needed but is unavailable. In SIMAN, queues are modeled by the QUEUE block. One of the operands of the QUEUE block is the number of an internal file in which the entities reside while queued.

In Cinema, a *queue symbol* is used to display the entities that reside in a file associated with a SIMAN QUEUE block. The queue symbol is a line segment of any length and orientation that can be added at any location of the layout. The first point defined on the line segment represents the head of the queue and the second point represents the tail. The queue symbol is then associated with a specific file number in the corresponding SIMAN model.

When an entity enters a queue in the simulation model, the entity's symbol is displayed along the corresponding queue line 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 model contains more entities than can be displayed along the length of the queue line, subsequent arrivals to the queue are not displayed. These entities eventually become visible as they move forward into the display portion of the queue when preceding entities exit.

Status Variables

The dynamic features described so far are tied to specific modeling constructs in the SIMAN language such as entities, resources, queues, transporters and conveyors. Cinema provides four additional dynamic features that can be used to represent the value of any status variable maintained by SIMAN during the execution of a simulation.

A digital display of the value of a status variable can be added to a layout using a feature called *dynamic variables*. You select the variable and define the display format (number of significant figures and places to the right of the decimal), size, and color using pull-down menus. You then position the variable anywhere on the screen using the mouse. When the variable changes values during a simulation run, the value will be output at that location on the screen using the defined size, color and format.

A collection of three different analog indicators, called *levels*, can also be used to display the value of any status variable. Two of the indicators, one shaped like a box and the other a circle, fill and empty in response to changes in the value of the associated status variable. The third, called a dial, is a circular level with a sweep hand that rotates either clockwise or counter-clockwise. The shape, fill and empty colors, fill direction, size and location of a level are specified using pull-down menus and the mouse.

A feature called global symbols are used to associate specific values of a status variable with different user-drawn symbols. When the variable changes values during a simulation run, the corresponding symbol is displayed on the screen. Like entity, resource, and transporter symbols, global symbols are created by coloring boxes on a screen sized grid and stored in a global symbol library. You then copy symbols from the library onto the layout and associate them with specific values of a status variable.

The fourth feature, called dynamic palette, can be used to associate the color of an object with specific values of a status variable. As the status variable changes values during the simulation run, the object changes color on the screen.

Each of the sixteen colors in Cinema are formed by combining different amounts of the three primary colors red, green and blue. The amounts of each to combine are determined by numeric intensity values which range from 0 to 3 for the 64-color palette supported by Cinema/EGA and 0 to 15 for the 4096-color palette supported by Cinema/HGA. A dynamic color is incorporated into the layout by first selecting one of the 16 color indices and associating that index with a status variable. Up to ten colors may then be defined for that index in the form of different red, green and blue intensities. Each color is then associated with a value of the status variable. As the status variable changes values during the simulation run, any objects drawn in that color index will change color.

For example, a simulation of a steel making facility could define several different colors ranging from gray to red to represent the temperature of a furnace. As the temperature of the furnace increases, the furnace would gradually change from gray to red.

RUNTIME FEATURES

The Cinema system includes a special version of the SIMAN simulation language. This special version performs the additional function of dynamically updating the graphics screen when a simulation is executed with an animation layout. A mouse/menu interface has also been added to facilitate the management of a simulation/animation session. Other features in this special version as well as features in the general release version of SIMAN allow interaction with a simulation/animation model while it is executing.

The speed of an animation is controlled by specifying values for two parameters that are used to scale simulated time to real screen update time. A *skip ahead* option allows the animation to be disabled for a specified period of simulated time so that the model executes without the imposed time delays. While the simulation is executing, Cinema/HGA has *zoom* capabilities by through the '+' and '-' keys. The ability to *pan* to different areas of the magnified display is then available through the use of the four arrow keys.

The execution of a simulation animation can also be interrupted to recall other layout files which might represent different sections or views of the system that was modeled. There is no limit to the number of layouts that can be built to correspond to a single simulation model. In addition, a snapshot of the system status and graphics can be saved at any time for later recall. Recalling a snapshot immediately restores the state of the system and graphics screen to the point in time when the snapshot was saved.

In conjunction with the SIMAN interactive debugging facility, the simulation may be interrupted at any point by pressing the escape key. Alternatively, the simulation may be interrupted at predefined times or at certain break points in the model. The interactive debugger provides an interactive command language which allows the user to show current the value of input parameters, system variables and output statistics. The ability to change parameters and capacities in combination with the animated display provides a powerful tool for determining the short term effects of changes in operating policies.

SUMMARY

The development of the Cinema system and recent enhancements represent a significant contribution toward making animation a practical part of every simulation analysis. With minimal instruction and practice, a user with little or no programming skills can easily construct and execute a highly detailed animation. The mouse/menu interface between Cinema and the user makes this task simple and efficient. The flexibility of the features of the layout design encourages the user to construct a layout that provides the appropriate level of animated detail to help in the analysis of those items of interest to the simulation study. The promise of better validated models and improved communication of model results to decision makers will continue to establish animation as an indispensable ingredient in successful simulation development.

REFERENCES

1. Lampon, B.W., "Bravo Manual" in Alto Users Handbook, Xerox Palo Alto Research Center,

AUTHOR'S BIOGRAPHY

Kevin J. Healy is Vice President, Development at Systems Modeling Corporation. He holds a Bachelor of Science in Industrial Engineering and a Master of Science in Industrial Engineering and Operations Research from the Pennsylvania State University.

Richard A. Kilgore is Vice President, Products at Systems Modeling Corporation. He holds a B.B.A. and M.B.A. from Ohio University and is presently completing his Ph.D. in the Department of Management Science of the College of Business Administration at the Pennsylvania State University.

Systems Modeling Corp.
P.O. Box 10074
State College, PA 16805
(814) 238-5919