Proceedings of the 1985 Winter Simulation Conference
D. Gantz, G. Blais, S. Solomon (eds.)

GRAPHICAL INTERPRETATION OF OUTPUT ILLUSTRATED BY A SIMAN MANUFACTURING SYSTEM SIMULATION

Lisa A. Pegden
Trevor I. Miles
Gustavo A. Diaz
Systems Modeling Corp.
226 Highland Avenue
State College, PA 6801

This paper discusses the role of graphical analysis of simulation data. Both static graphics and dynamic graphics contribute to the understanding of the simulation model. The advantages and disadvantages of these two major classes of output graphics are described. A SIMAN model of a manufacturing system is used to illustrate the different types of graphical output, three commercially available packages for graphically analyzing the system are discussed.

INTRODUCTION

Computer graphics are becoming increasingly important in the analysis of simulation output. As evidence, several new simulation products have been introduced in the last several years that stress graphical analysis capabilities; See Why, TESS, and Cinema are popular examples.

Simulation and Industrial Engineering conferences give further proof of the advancing role of graphical output analysis. More and more papers discuss new analysis capabilities, and the graphics-oriented sessions are among the best attended.

Computer graphics capabilities are improving rapidly, partly because microcomputers are becoming quite sophisticated. The popularity of the microcomputer has launched the popularity of graphical analysis in simulation in two ways:

1) Inexpensive simulation languages for microcomputers put simulation within nearly everyone's reach, including engineers and managers not content with seeing black and white printouts of summary reports.

2) Microcomputer users are accustomed to having graphics capabilities at their fingertips. The integrated graphics of LOTUS 1-2-3 made it a top-seller and put it on nearly every business computer.

Given the popularity of colorful graphics on computer screens and the expanding role of simulation, "selling the solution" to managerial levels will most likely become a graphical analysis of output.

This paper will discuss the various types of graphical analysis available, and detail their advantages and disadvantages. There are two basic categories of graphics in output analysis, static and dynamic graphics. Static graphics are those in which the picture of the output remains constant. Examples of static graphics include: line graphs, piecharts, plots, and histograms. Static graphic analysis can be real time or postprocessing.

Dynamic graphical analysis involves a changing picture that depicts the system change. This type of graphical analysis is typically referred to as animation. Graphical animations can be real time or postprocessing. They can offer inexpensive character graphics, or costly pixel graphics. PLAYBACK and TESS are examples of dynamic graphics in a postprocessing mode. Cinema and See Why are real time animation packages.

The discussion of the categories of graphical analysis will be illustrated with a SIMAN manufacturing model developed by a client of Systems Modeling Corp. The model will be run through the SIMAN Output Processor in the discussion of static graphics. The PLAYBACK software will be used to illustrate dynamic postprocessing graphics. Finally, an animation of the model will be created using Cinema, and the advantages and disadvantages of real time dynamic analysis will be presented. This paper will also discuss and compare static and dynamic analysis in terms of the information conveyed to the analyst.

1.0 Problem Description

The simulation model discussed here is a subset of a larger model developed by a large Canadian manufacturing concern. The SIMAN simulation language was used to model a proposed Flexible Path Automatic Guided Vehicle System in a large assembly area.

The assembly area consists of a line of serial workstations (the Mainline), and eight parallel work lanes which constitute a work island. Each lane has three stations: prework, operator, and postwork. AGVs travel between the first mainline station and each station in the work island. Following the postwork operation, the AGVs travel along a common exit path to a storage/queuing area. See Figure 1 for a schematic drawing of the simulated system.

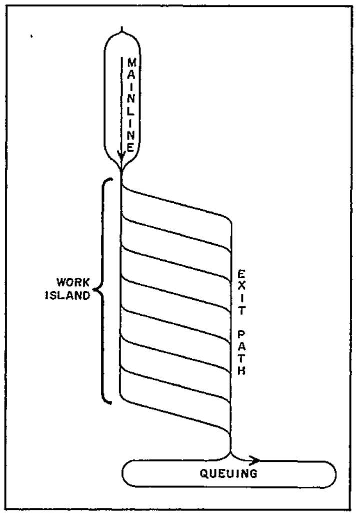

Figure 1: Schematic View of the System

The analysts developed this model to test and debug the AGV system prior to implementation. They were looking for answers to the following questions:

1. Number of AGVs required
2. System throughput
3. Effective breakdown rate
4. Operator utilization

At the time of this writing, the system evaluation is on-going. Graphical animation will be used to present the results of the study.

2.0 Model Description

The SIMAN block diagram orientation was used to model the described system. SIMAN's station submodel capabilities were used extensively to model the parallel workstations. The 24 workstations in the work island were modeled as three generic stations. The larger SIMAN model, of which this is a subset, had 147 stations in 49 lanes in the work island. This larger system was also modeled using the macro submodel and three generic stations.

Jobs to be processed by the assembly operation were created every ten seconds and sent to the first mainline station. The SIMAN Route block was used to model the AGV transfer between stations. The operands for the Route block are the destination station, and travel time specified as a constant or an expression. Constant travel times were used in this model for the sake of simplicity.

SIMAN's Branch block was used to deterministically select empty work lanes to travel to. Count blocks were used to keep track of individual lane throughput. Discrete time-persistant statistics were collected on utilizations for the prework, operator, and postwork resources, and on the number in the queues preceding the decision point and exit lanes. Figure 2 shows a segment of the SIMAN Block Diagram for the model.

This simulation was developed and run on an IBM PC/XT with an 8087 coprocessor. The model had approximately 130 statements. When simulated for 2.5 hours, execution took 4 minutes and 20 seconds.

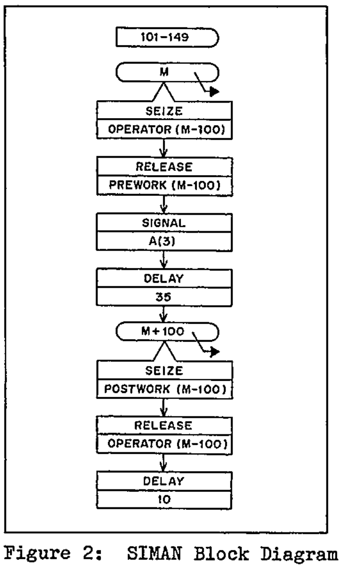

Figure 2: SIMAN Block Diagram

3.0 Analysis of Data Using Static Graphics Methods

Static graphic analysis of simulation output is among the oldest and most widely used means of graphically explaining the results of a model. Nearly every popular simulation language offers built-in report and graphics generators for pictorially displaying output.

Static graphics generators fall under two major headings: real time and postprocessing. With real time graphics, the chart is updated during the simulation. This offers a great advantage in debugging. The simulation run can be stopped when the static graphs indicate a peculiar trend in the data. A real time static plot of "number in queue" that depicts queues filled to the maximum level indicates to the analyst that something is not quite right in the model (or system). Unfortunately, viewing real time static graphics is very time consuming. The simulation execution becomes bogged-down by the graphics generation. In order to view the graphical trends as they occur, the analyst may be glued to his/her screen for a long time, especially during long runs.

Real time static graphics are not very flexible. If the analyst wishes to change the graph format, the simulation must be rerun. Merely specifying new cell widths for a histogram can be a costly, time-consuming procedure.

Using static graphics in a postprocessing mode is a more efficient means of displaying output graphics. With postprocessors, data of interest is stored in a file during the simulation run(s). Using a graphics postprocessor the analyst can access the data, analyze it and re-analyze it without rerunning the simulation.

Postprocessing of output data can be done using language specific features, or the data can be transformed for use with other graphics procedures. Both SIMAN and TESS offer postprocessors for graphical analysis. Transformed output data from simulation runs can be run through graphics procedures using packages like SAS and LOTUS 1-2-3.

Given the efficiency and ease of use of postprocessors, the trend in static graphics analysis seems to be in this direction.

In evaluating the quality of graphics with static output processors, a major division arises between pixel and character graphics. Pixel graphics are much crisper, clearer, and offer greater graphing flexibility. Using pixel graphics, each pixel on the screen can be individually accessed.

The smallest addressable unit when using character graphics is a character cell, which is typically 8x8 or 14x9 pixels. The base unit of all graphics must be letters, numbers, or the standard block character. Non-horizontal lines are difficult to construct. Character graphics do provide less expensive analysis of output than pixel graphics. Less sophisticated hardware is required to generate the graphics.

The SIMAN Output Processor supports both pixel and character graphics. The Output Processor was used to analyze selected data from the AGV model.

The Output Processor program allows the user to take data written to files during a simulation run and subject it to various data analyses, such as: filtering, primary statistical analysis, graphical display, and reformatting and transfer of files.

Data treatments using the SIMAN Output Processor can be grouped into three categories:
(a) Graphical presentations - useful in visually interpreting the results of a simulation run and detecting transient states for terminating systems.
(b) Statistical methods to perform preliminary statistical inference.
(c) Data transfer and transformations capabilities that allow the user to upload or download files between microcomputers and mainframes. Free-formatted ASCII data files can also be converted to a SIMAN - formatted data set.

The eighteen commands in the SIMAN Output Processor are summarized in Table 1.

The SIMAN Output Processor combines graphical output with statistical methods for a thorough treatment of data. The figures below illustrate various data treatments.

Figure 3 shows a plot of the mainline processing time and the time spent waiting for an initial AGV. From the figure it can be seen that both operations exhibit an extended start-up or transient time. The observations obtained during this transient time will bias any statistical analysis performed on the data. The FILTERS command was used to discard the biased observations from time 0 to time 1400. The effect of filtering the data is shown in Figure 4. The bias of the transient observations is also demonstrated using the MEANSTEST command, see Figure 5. This graph compares the means of two sets of data and determines if they are statistically different. Figure 5 shows that the 95% confidence range of the difference between the means of the filtered and original data does not span 800. Thus it is determined that there is a significant difference between the means from the two sets of data.

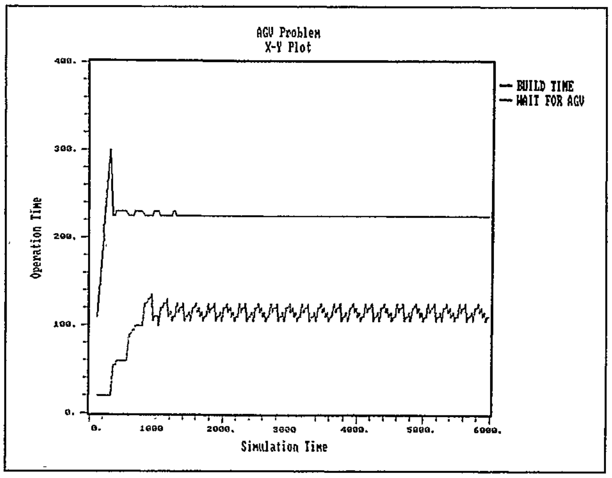

Figure 3: Plot of Build Time and Waiting Time

The BARCHART command in the Output Processor can be used to give an area representation of the time a job waits for an AGV after departing from the last mainline station. This is depicted in Figure 6. The HISTOGRAM command gives frequency information for the AGV Wait Time as shown in Figure 7.

From these figures, we see the most obvious advantage of static graphic analysis. Hardcopies of the analysis are readily available. The output graphics can be

Table 1: Summary of Output Processor Commands

Command Name	Description
BEGIN	Directs a copy of the output results to a specified file
Graphical Capabilities Commands:	
PLOT	Plots data from specified files on X-Y axes
BARCHART	Plots data in a barchart format for a specified observational variable
HISTOGRAM	Calculates frequency information from data in a specified file
TABLE	Generates a table of values of observational or time-persistent variables
Statistical Methods Commands:	
CORRELOGRAM	Calculates correlation between observations from a specified data file
FILTER	Generates a filtered data set for a specified file containing either observational data, discrete time-persistent or continuouos time-persistent data using truncation and batching
INTERVALS	Constructs confidence intervals of data from specified files
STDINTERVALS	Calculates standardized time series confidence intervals of data from specified files
SDINTERVALS	Constructs standard deviation confidence intervals of data from specified files
MEANTEST	Performs a comparison of means between two data sets from specified files
VARTEST	Performs a comparison (ratio) of estimated variances from specified files
ONEWAY	Performs a one-way analysis of variance of data from a specified file
Data Transfer Commands:	
DIFFILE	Outputs the data in a text file DIF format which can be read by other .DIF-format supporting programs (microversion)
EXPORT	Outputs SIMAN-formatted files as ASCII-formatted files
IMPORT	Outputs ASCII-formatted files as SIMAN-formatted files
EZPREP	Outputs SIMAN-formatted files as Tektronix PLOT 10 Easy Graphing Software formatted files
LOAD	Outputs non-SIMAN ASCII formatted files as SIMAN formatted files

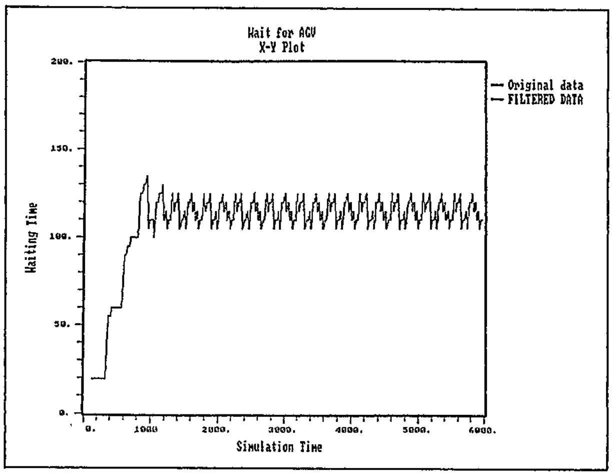

Figure 4: Plot of Filtered Data

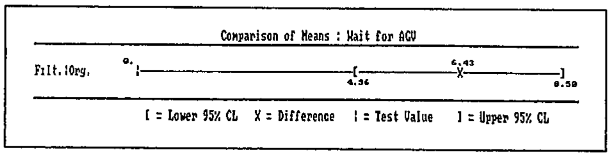

Figure 5: The MEANTEST Command

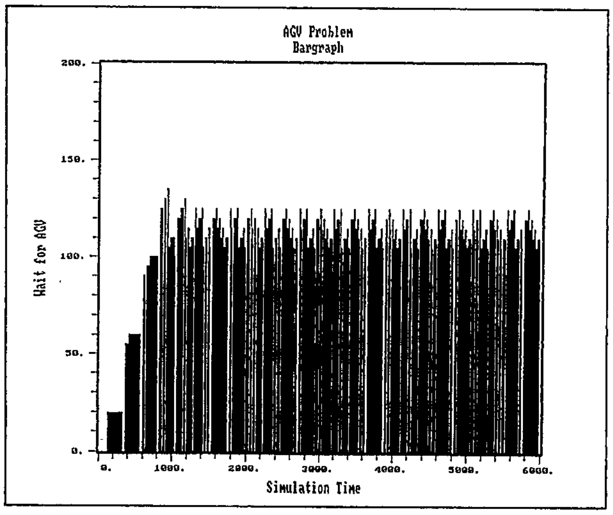

Figure 6: Bargraph of Wait Time

easily added to a report or copied onto a slide. Static graphs are easy to understand. They depict data in common readable formats, such as histograms and plots. Using a postprocessor such as SIMAN's, which is interactive and help-oriented, static graphics are quick and easy to generate.

Perhaps the greatest advantage of static graphs of output is that they depict trends over time. The occurances of a 40 hour work week can be viewed on a single chart. Herein also lies the greatest disadvantage of static graphs, they do not show the entire picture. Static graphs portray what happened in the system, but give no insight as to why things happened. The missing pieces limit our awareness and ability to foresee typical system interactions.

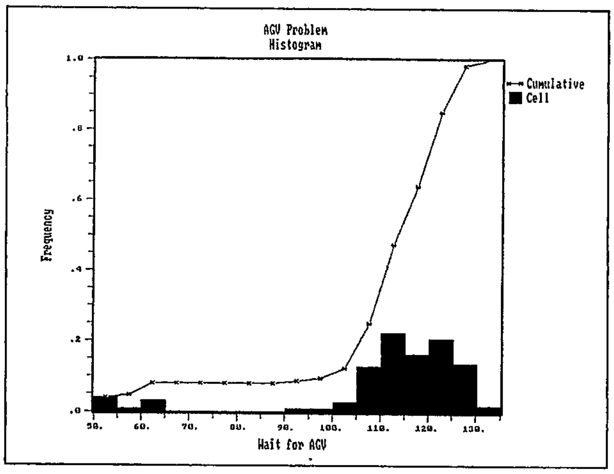

Figure 7: Histogram of AGV Wait Time

Dynamic graphical analysis helps explain why the system behaves as it does. Sections 4.0 and 5.0 explain two different methods of dynamic analysis.

4.0 Dynamic Analysis Using a Postprocessor

Commonly referred to as computer animation, dynamic analysis of output is the newest art form in simulation analysis. Animation graphics vary widely, from simple graphics limited to bar graphs changing, height to sophisticated wireframe schematics able to be rotated.

PLAYBACK is a general purpose animation package using character graphics in a postprocessing mode. The dynamic graphics consist of bars that change height and boxes that change color in representation of the changing status of a variable. While the animation runs, a digital clock shows the current simulated time. The PLAYBACK animation allows the analyst to view dynamic interactions between system components, such as resources and queues.

Because PLAYBACK uses character graphics, it can be run on inexpensive equipment. Character graphics require a small amount of display memory, therefore memory can be allotted for multiple pages (screens) to be accessed and updated during the animation. Animations based on character graphics also have a low information transfer rate. This means the software may be accessed through a remote terminal.

Animation packages using character graphics cannot show a picture with much detail. Circles, triangles and slanted lines are unavailable. This makes for crude depictions and less interesting output. Software that offers user-definable character sets can partially alleviate these shortcomings.

Like static graphics, animation can be real time or postprocessing. As a postprocessor, PLAYBACK operates on output data previously saved in a data file. The same data that is saved for static analysis is merely portrayed in a different format.

By dynamically postprocessing output data, the analyst does not burden the simulation with the graphics. Also, the analysis is flexible. The user can jump forward or backward in simulated time during the animation without preplanning. Output can be displayed for the beginning of the run, and an instant later for the end of the simulation.

Postprocessing lends speed and flexiblity, but does not allow you to "tweak" the system. Analysis of any changes to the system require new output files to be generated by re-running the simulation. Once changes are made to the model and new data files are created, they can be accessed at anytime for a dynamic view of the system.

PLAYBACK allows the analyst to animate up to six variables on up to four pages at once. The program is menu-driven and interactive, and easy to learn and use. Sixteen different colors and four different time advance options are available. Queues are typically represented as bars that change height to reflect the changing number in the queue. Resources, conveyors, etc. are represented as boxes that change color to reflect busy or idle status. Although the analyst does not see a detailed picture of his/her system, PLAYBACK does show the cyclical effects between queues and resources, and the overall flow of the simulated process.

In this evaluation, PLAYBACK was used to animate six variables on two pages. Figure 8 shows a printout of

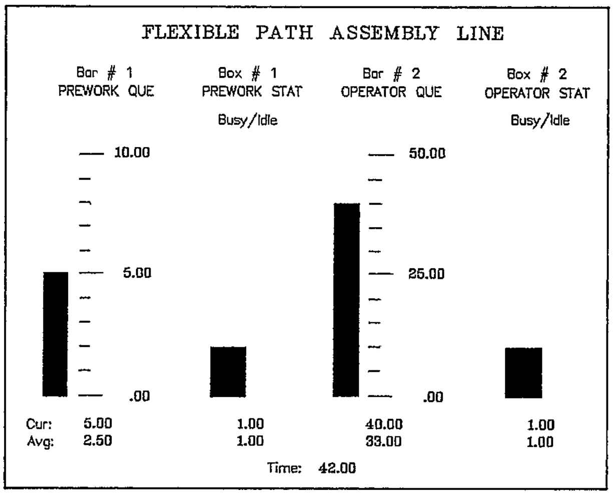

Figure 8: PLAYBACK Animation, Page 1

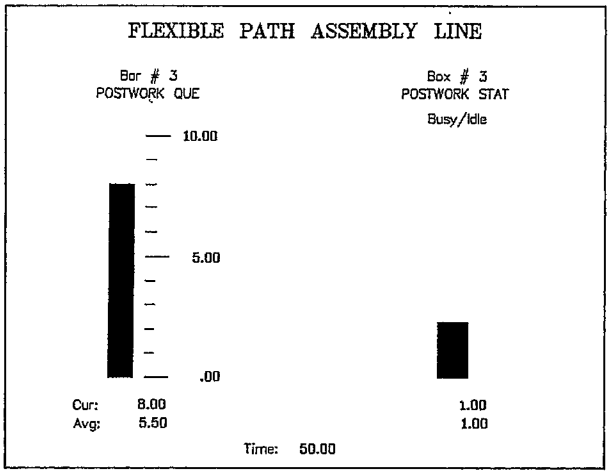

Figure 9: PLAYBACK Animation, Page 2

page 1 of the animation after 42 time units have elapsed. The Prework queue has five entities in it, and both the Prework and Operator resources are busy. On page 2 of the animation layout, the Postwork queue and the Postwork resource have been animated. Viewing the PLAYBACK animation for several hundred time units yields insight as to why the queues grow as large as they do, and as to the cyclical nature of the queue/ resource relationship. In order to recognize the startup bias in the simulation model, the analyst must remember the beginning of the animation and compare it to the remainder of the animation.

This animation took only 1 hour to construct, but yields interesting information not able to be gleaned from a static graph. However, PLAYBACK's character graphics present only a skeleton of the actual system, whereas pixel graphics can give a detailed view of the system as it operates.

5.0 Real Time Dynamic Analysis

As computer graphics become more sophisticated, simulation with animation becomes more desirable. The opportunity to see a realistic "cartoon" of a proposed system in action is often worth much more than reports stuffed with static analysis.

The Cinema System is a simulation/animation workstation built around the IBM PC/AT. It is a real time processor - as the simulation executes so does the animation. Cinema allows you to build a SIMAN simulation model, and tie it to a detailed picture of the simulated system. As entities are processed in the model, they are processed on the animation layout. See the Cinema Tutorial by Kevin J. Healy in this proceedings for a more detailed discussion of Cinema.

The graphic analysis performed with Cinema is based on pixel graphics. With Cinema's resolution of 832x624, a half million pixels can be addressed to create a detailed animation layout. Using pixel graphics the analyst has access to circles and non-horizontal lines, and can develop an animation that pictorially resembles his/her system.

Because of the sophistication required to generate pixel graphics, the associated hardware tends to be more expensive. Animation systems based on high-quality graphics may have price tags that prohibit their use in some situations. Also, addressing every pixel on the screen requires a high data transfer rate. This prevents accessing animation software from a terminal over a serial line. Engineering workstations built around super microcomputers or minicomputers are required to run the software. Examples of workstation controllers are: APOLLO, SUN, IBM PC/AT, and MICROVAX. Some analysts accustomed to simulation using a mainframe computer may find that their models go beyond the capabilities of the workstation.

When pixel graphics are combined with a real time processor, the dynamic analysis also becomes a valuable method for debugging the model. While the animation is running it can be interrupted and the model embellished. When the animation is restarted, the changes are immediately reflected in the picture. Variables can be reset, machines can be broken-down, and workers can be preempted.

A major disadvantage of real time analysis is the time involved in viewing the animation. A system simulated for 10,000 time units would require a great time committment to view the entire animation. While the animation executes, the analyst sees "instants" in time. He must remember previous views to construct an opinion on the system status over time.

Real time animation does not offer the flexibility of modular postprocessors. If the analyst wishes to step backwards and forwards through the animation, he must plan for it in advance. Cinema allows "snapshots" of the system to be saved and later recalled.

The figures following show snapshots of three different animation layouts for the AGV system model. Figure 10 shows a simple animation developed by the client. It took approximately three hours for an uninitiated user to develop this layout for the simulation model.

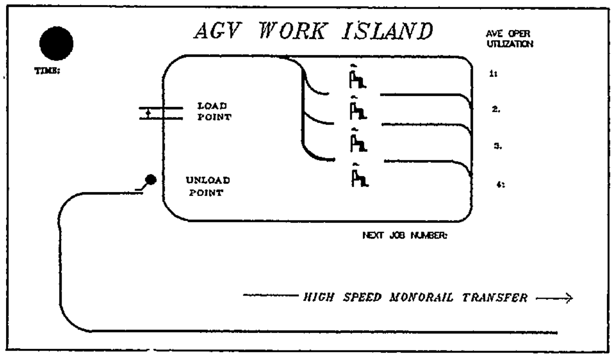

Figure 10: Simple Cinema Animation

Figures 11 and 12 show animations developed by the author, and illustrate two different methods for depicting a system. Figure 11 shows an animation layout that resembles the schematic view of the system. Very little detail is shown here. Resources are modeled as circles, parts as rectangles. This animation took only 2 hours to develop; and shows enough detail to yield insight as to the "whys" of the system. Anyone familiar with the actual system would recognize the animation and understand the results of the simulation model.

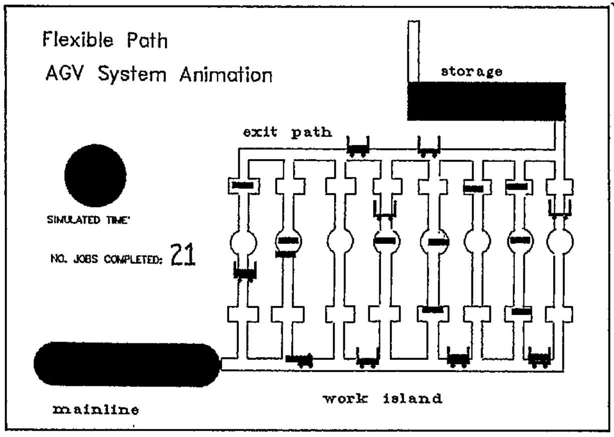

Figure 11: Cinema Animation of Schematic View

Figure 12 is a more descriptive animation of the system. The resources resemble their actual counterparts, and the layout has more detail. This layout required 4 hours to develop. The figure shown here is a "zoomed-in" view of the layout.

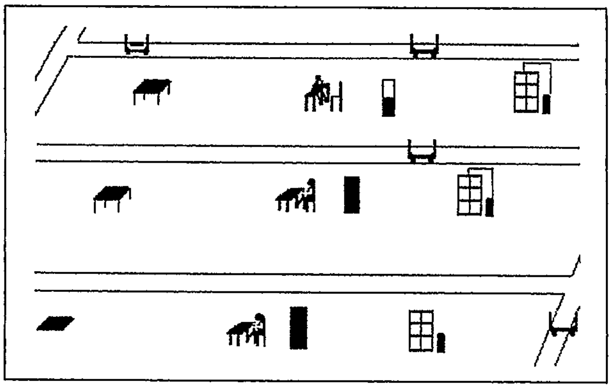

Figure 12: Zoomed-in view of a Cinema Animation

As is evidenced by the preceding figures, the high-quality graphics capabilities of a pixel-oriented animation system create endless possibilities for output presentation. See the Appendix for a detailed description of the development of the animation layout in Figure 12.

The animation depicted in the snapshots above gives insight to the dynamic interactions of the system components. It helps explain why the static analysis yielded the results it did. However, a single showing of the system cartoon cannot substitute for statistical analysis, it can be likened to touring a manufacturing facility on any given day. Obviously management would not make decisions based on what was seen on one short tour, he/she may form ideas on problems to look for later. Likewise, static graphic analysis is still needed to yield a view of system status over time.

Graphic animation is a powerful tool for selling the solutions the simulation generates. It provides a basis for communication between the simulation analyst and the system expert. Using high-quality graphics to describe the simulation make the description a quick and easy glance. Management can be assured that the model accurately portrays the actual system. Dynamic graphical analysis also helps expose modeling errors, and faulty assumptions.

6.0 Concluding Remarks

As simulation becomes a more important, required part of the decision-making process, all types of graphical analysis will become more important. The high costs and hidden variables involved in implementing the "factory of the future" make it important to "simulate before you automate". An animation that depicts the real system using striking graphics is often requested, and will probably become expected in the future. When management does not understand simulation, graphics can provide the lines of communication that enable the analyst to explain new systems.

Animation is invaluable in verifying that the simulation model is correct, and in selling the solution. However, it is important not to become too caught-up with the pretty pictures of the animation. Graphics should not replace traditional statistical analysis. It is ridiculous to view an animation for an hour and profess in-depth knowledge of the system operation. The animation will help assure that the model is accurate, but statistical analysis must be used to verify the results and more inferences from them. Dynamic analysis often has an initial bias problem. The early part of the run is often too favorable, system bottlenecks take time to appear.

Both static graphics and dynamic graphics should be used to augment the statistical analysis. Static graphs are simple to understand and generate, and can be easily used in reports. Dynamic graphics give insight as to the whys of the system, and are visually interesting. High-quality pixel graphics yield striking pictures, but can have a striking price tag to match. Real time animation systems allow the analyst to play with the system as the simulation executes.

As simulation use grows and graphics capabilities improve, analysts will look to graphical analysis to present their ideas and solutions, or managers will look elsewhere for their solutions.

APPENDIX
CREATING THE ANIMATION LAYOUT SHOWN IN FIGURE 12

The animation in Figure 12 took approximately 4 hours to develop. Much of the time was spent deciding upon the level of detail to include in the "cartoon". The main area of interest in the simulation model is the part processing in the work island. For the sake of simplicity and time, it was decided to animate the work island only. The mainline processing operation was not included in the animation layout. The snapshot in Figure 12 is only a partial view of the animation.

The first step in creating an animation "layout" is to develop the static background. This is the part of the layout that does not change during the animation. Here, the drawing process was initiated by drawing the AGV pathway using "rubberband" lines. After selecting the LINE function on the DRAW menu, a line is constructed by first digitizing a point on the layout using the mouse. A rubberband line is shown between the digitized point and the current cursor position. Moving the mouse changes the length and direction of the line. When the desired line is achieved, the mouse button is pressed to fix the location on the screen. The line portion of the static background was drawn using this method.

Text was added to the static backgroung using Cinema's TEXT function. The desired size and location for the text was selected using the mouse and menus, and the actual text was entered at the keyboard.

Once the static background is complete, the dynamic symbols and constructs are developed. Dynamic symbols and constructs are items that change shape, color, or position during the animation. Examples of dynamic objects are resources, entities, variable symbols and routes. The first dynamic objects created for this animation were resources. Symbols for the PREWORK, OPERATOR, and POSTWORK resources were drawn on a grid and stored in a Resource Symbol Library. The symbols were created on a 64x64 pixel grid using drawing functions similar to those available when creating the static background. A busy and idle symbol was created for each of the three resources. In addition, a preempted and inactive symbol can be designated for the symbols in the library.

When the resource symbols are stored in the library, their position determines the resource number they will be identified with. When the resource with this number changes in the SIMAN model, the corresponding symbol will change on the layout. When the simulated resource becomes idle, the idle symbol will be shown on the layout. After the symbol library is created, the symbols are positioned on the layout in the desired location using the mouse. For this animation there were three resources in five lanes. Because the lanes are parallel and indentical, the three resource symbols were copied five times for positioning on the five worklanes. The PREWORK resource is symbolized by a table with test equipment. The OPERATOR resource is depicted with a worker seated at a bench. When the OPERATOR is idle, the symbol is a worker seated on his bench. The POSTWORK symbol depicts a storage unit with automatic testing. The busy symbol shows testing in progress with a colored bar.

Next, the entitiy symbols were created on the grid and stored in the Entity Symbol Library. Once saved in the library, the symbols must be copied into the current symbol list. The symbol's position in the list ties it to a value of a special entity animation attribute. Each time an entity's the animation attribute is re-assigned or changed in the SIMAN model, the entity symbol corresponding to the new value is shown on the layout. The entity symbol list for this animation consists of a workpiece symbol which represents the jobs being processed in the model, and a symbol which represents the AGVs that travel from station to station in the model. Different symbols could have been defined to represent the entities in various states, such as in prework, operation, or postwork. Merely assigning a new attribute value causes a new symbol to appear.

After specifying the entity symbols, route paths were designated. The routes specify the paths along which the entities will travel in going from station to station. The route paths do not show up when the animation is running. In this animation, routes were drawn along the AGV path in the background. This enables the AGV entity to be routed to the various work island stations, and appear like it is traveling on the AGV path.

Variables were added to the layout to show the number in the queue at each PREWORK station. Cinema Variables show the current value of a simulation variable in analog format. Variables are added by choosing the VARIABLES function and selecting the specific system variable to depict. The system prompts for the variable index and format. The format is represented by asterisks separated by a decimal point in the appropriate location. This format was placed on the layout beside the PREWORK resources using the mouse. While the animation is running, the current values in the PREWORK queues are shown at these locations.

Levels are an alternate method for depicting simulation variables. Levels are boxes, circles, or dials that fill and empty in relation to a variable's value. For example, the "time in system" variable is typically represented by a dial. In the AGV animation, levels were used to illustrate the OPERATOR resource utilization. After choosing the NR(I) variable and entering the correct indices and minimum and maximum values, the level characteristics were selected. A rectangular box was specified and positioned on the layout beside the OPERATORs. This serves to further illustrate the busy/idle status of these resources.

Once the static and dynamic portions of the animation layout are created, the simulation/animation is ready to be run. The Time Advance criteria are selected from the menus, and the necessary files are read into the run processor. The AGV animation was stopped during exectution to record the snapshot in Figure 12.

Lisa A. Pegden is Vice President of Systems Modeling Corporation. She received a Bachelor's degree in Industrial and Management Systems Engineering from the Pennsylvania State University. Her interests lie in the area of manufacturing system simulation, and methods for "selling the solution."

Trevor Miles is a Software Engineer with Systems Modeling Corporation. He is interested in statistical analysis of simulation output and graphical display of data. Trevor modified the SIMAN Output processor to display pixel graphics. Trevor is currently a Ph.D. student in Industrial Engineering at the Pennsylvania State University.

Gustavo Diaz is a Software Engineer at Systems Modeling Corporation. He is a Ph.D. student in Industrial Engineering at the Pennsylvania State University. His current interests are statistical analysis and random number generation in simulation. Gustavo was instrumental in revamping the SIMAN Output Processor, and in implementing improved random variate generators.

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