Proceedings of the 1997 Winter Simulation Conference
ed. S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson

THE POWER AND PERFORMANCE OF PROOF ANIMATION

James O. Henriksen

Wolverine Software Corporation
7617 Little River Turnpike
Annandale, Virginia 22003, U.S.A.

ABSTRACT

Proof Animation™ 4.0 is a family of products for animating discrete event simulations. Proof is available in a variety of versions, including an inexpensive, student version, mid-size and unlimited-size commercial versions, a run-time version, and a royalty-free, redistributable demo viewer. Proof is an ASCII-file-driven, post-processing, general-purpose animation system which runs on readily available PC hardware. Its vector-based geometry provides a large animation canvas and the ability to zoom in or out, while maintaining crisp, clear images. Proof includes built-in drawing tools and CAD import/export for ease of creating animation layouts, dynamic bar graphs and plots for displaying statistics, multiple-window display, and a unique presentation-making capability. Proof's open architecture makes it ideally suited for serving as an animation engine for models written in a wide variety of simulation and programming languages. Proof's superior power and performance assure smooth, realistic motion for animations regardless of their size, complexity, or application. Beginning with Release 4.0, Proof is available as a full-fledged Windows 95/NT application.

1 INTRODUCTION

Proof Animation is a general-purpose, ASCII-file-driven, post-processing animator designed for use with the widest possible variety of simulation tools. Every Proof animation requires two files, (1) a layout file, describing static characteristics of an animation, e.g., the background drawing over which objects move, and (2) a trace file, which is a time-ordered sequence of commands which create, destroy, move, and otherwise change objects displayed on the layout, portraying events in a simulation. Both of these files are free-format ASCII files, with well documented ("open") architectures which can be generated easily in a variety of ways. Trace files are almost always written using a simulation language or package, such as SLX (Henriksen 1997), GPSS/H™ (Crain 1997) , Extend™, Slam™, Siman™, Simscript II.5™, etc. Trace files can be written using non-simulation-languages, such as C, and simple trace files can even be prepared using a text editor. Layout files are almost always developed using Proof's built-in drawing tools. Proof also includes a CAD import feature, allowing quick importing of .DXF files. Layout files can also be generated by a program. While this is not done very often, it can and has been done straightforwardly.

Proof's open, simulation-language-independent architecture is a great strength, both technically and commercially. From the user's perspective, Proof provides the opportunity to add high-quality animation to a simulation developed using existing tools, with no requirement to purchase and use simulation tools provided by Wolverine Software. From Wolverine's perspective, Proof provides a stream of sales to users already committed to buying and using simulation tools provided by Wolverine's competitors.

Because language independence is such an important strength of Proof, every effort will be made to ensure that improvements to Proof will preserve this independence. All the improvements to Proof described in this paper will be done in a simulation-language-independent fashion.

2 THE PROOF ANIMATION FAMILY

The Proof Animation product family runs on readily available, inexpensive PC hardware. All versions require a 386 or better CPU, a math coprocessor, and at least a VGA-compatible video card. Running Proof as a Windows 95/NT application requires Direct Draw driver support for the video hardware used. Because these requirements are easily met, animations developed with Proof are very portable.

Proof runs under DOS, Windows 3.x, and OS/2 as a 32-bit extended DOS application. "Native" Windows versions are available for Windows 95/NT. A set of

Windows icons is supplied with each of the Proof Animation products to provide single-click launching. The following products comprise the Proof Animation family:

- PROOF ANIMATION
  Proof Animation is the basic animator. Memory size is fixed and limited. Includes built-in CAD import/export feature. Good for small to mid-sized animations.
- PROOF PROFESSIONAL
  Proof Professional exploits all available extended memory for animating large systems. Includes built-in CAD import/export feature.
- RUN-TIME PROOF PROFESSIONAL
  Run-time Proof Professional runs developed animations or presentations, but has no animation *development* capabilities. It provides a low cost way to run different scenarios with a fixed layout file prepared using Proof Professional or Proof Animation.
- STUDENT PROOF ANIMATION
  The student version of Proof Animation is included with the *Using Proof Animation* text. Size and playing time limitations are imposed; otherwise it is identical to Proof Animation.
- PROOF ANIMATION DEMO MAKER
  Demo versions of animations can be prepared under a licensed copy of Proof Animation or Proof Professional containing the Demo-Maker add-on. Copies of the executable demo files can be reproduced and distributed free of charge and viewed by anyone. No licensed copy of Proof is needed to *view* a demo prepared with the Demo Maker.

3 MAXIMUM PERFORMANCE DESIGN

3.1 The General-Purpose Approach

One of the ways in which the performance of Proof Animation is maximized is its general-purpose design. A general-purpose animation product can be used without being tied to a specific simulation or programming language. While built to work easily with Wolverine's GPSS/H simulation software, Proof Animation also provides affordable and powerful animation software to users who develop models in other simulation and programming languages.

Most animation software from other vendors is directly coupled to their simulation software. In other words, one cannot use *their* animation software without also using *their* simulation software. In some cases, the simulation and animation software are sold only as a pair, so both must be purchased regardless of the needs of the user. The suggested advantage of the coupled approach is that because the animator has direct access to the simulation events, development of the animation is supposedly simplified. However, the real advantage is felt by the vendors. They sell more software since their users do not have an option to pick and choose based on their needs. Moreover, a user of the coupled software has little or no control over what information is passed to the animation; therefore, he or she may actually have to alter the modeling approach used in the simulation to achieve the desired appearance and level of detail for the animation.

Another disadvantage of the tightly coupled simulation/animation package is cost. Sole sources tend to be expensive. Vendors of these tightly coupled packages often claim that their approach is the *only* way to add animation to a simulation. Proof Animation has proved that wrong. The number of success stories using Proof Animation with other software continues to grow. Furthermore, a benefit of the mix-and-match strategy for software purchases is that the selection can be based on optimal functionality and price.

3.1.1 ASCII Input Files

Proof Animation can be used with other software because of two major design features. One is that Proof Animation is driven by ASCII files. Therefore, any software capable of writing ASCII text files can be used with Proof Animation.

Proof Animation requires two ASCII input files to run an animation: the layout file and the trace file. The layout file describes the geometric details of the background over which objects move, provides geometric definitions and properties for such objects, and defines logical paths along which the objects move.

Ordinarily, layout files are produced at least in part by using Proof Animation's drawing tools; however, the layout file command set specifications are published so programs can easily be written to generate layout files. For example, some users have written front ends for their simulation models that allow different system design parameters to be specified for each run. Based on these parameters, different geometric configurations are written and incorporated into a layout file. The new layout appears on screen when Proof Animation is invoked.

The trace file contains a time-ordered sequence of commands such as CREATE, DESTROY, PLACE, PLOT, MOVE, SET SPEED, SET COLOR and many more. This file provides Proof Animation with information on when, where, and what to create, destroy, place, plot, etc. Trace files are free-format, and the commands are easily learned and used. They provide exactly the kind of flexibility necessary to easily be integrated with the simulation model logic. Any language that can produce formatted ASCII output can write a trace file.

3.1.2 Post-Processor Animation

The second of the two design features that make Proof Animation compatible with other software is that it is a post-processing animation package. Post-processing means that it runs *after* the simulation has executed. Both the layout and trace files must exist before invoking Proof Animation. They cannot be written and read concurrently.

Two great advantages result from the post-processing approach. First, PC hardware is not shared between the simulation and the animation. This leaves the entire CPU for running the animation. Second, it provides the abilities to jump back and forth in time during the animation playback, to speed up or slow down the viewing speed, or show all or a specific portion of an animation. These features make it easy to investigate unusual system behavior or highlight points of interest.

3.2 Vector-Based Geometry

In the Proof Animation product family all layout file information is based on vector geometry. Vector-based descriptions are automatically mapped into the screen's pixels to build the image. One of the advantages of this approach is that layouts can be much larger than a single screen. With the ability to zoom in or out and pan, larger layouts are easily navigated to show the overall layout or zoomed in to whatever level of detail is necessary. Vector-based geometry also provides the ability to have moving objects realistically *rotate* around corners instead of the sliding effect to which other animation packages are limited.

Another advantage of vector-based geometry is that many CAD packages are capable of producing standard vector-based .DXF files. In many cases, a CAD drawing already exists for the system to be animated. If that is the case, the effort of redrawing an entire layout can be avoided. Proof Animation's built-in CAD Import/Export feature provides the capability to convert industry-standard .DXF files into Proof Animation layout files, *and vice versa*. Credibility of the study is enhanced when viewers see a familiar CAD drawing of the system integrated into the animation. These advantages maximize the power of the animation by giving a user total flexibility on the detail and complexity of the drawing.

Instead of vector geometry, animation packages may use a pixel-oriented approach for drawing. With the ability to manipulate individual pixels, one can produce detailed images. However, this level of detail is time consuming to draw and can often be lost because of the scale at which the animations are viewed. Some other disadvantages of pixel-orientation are: (1) pixel-oriented images cannot be rotated; (2) layouts are often confined to single-screen images. Some animators offer multi-screen operation; however, the individual screens are disjoint and independent, unlike Proof Animation's single, continuous canvas; (3) zooming in on pixel images magnifies the jagged edges inherent in all such images. When a zoom-in is performed in Proof Animation, the vector-based image maintains its crisp and clear appearance. Lines continue to look like lines. If a zoom-in is performed on a pixel-based animation layout, the effect of the jagged edged image makes a line look more like a stairway.

3.3 Smooth Motion

The maximum performance design of Proof Animation achieves very smooth motion. Proof Animation maintains this smooth motion by updating or refreshing the screen 60-70 times per second. Other software can often sustain refresh rates of only 5-10 updates per second. The ultimate purpose of an animation is to achieve a realistic depiction of the system being studied, allowing the audience to gain confidence in the results of the simulation study. Objects that move smoothly across the screen are more realistic than those that jump across the screen.

3.4 Color and Resolution Options

Beginning with Release 3.0, all versions of Proof, including the student version, provide for operation in 256-color mode in a variety of screen resolutions.

The “normal” operating mode for Proof is 256-color mode, and the “normal” resolution is 640x480. Video hardware which supports this mode and resolution is very commonplace. Proof can also run at 800x600 and 1024x768 resolutions, provided enough video memory is available. These resolutions are now even available on some *laptop* machines. For older hardware with less video memory, Proof can run using 640x400 resolution. For very old hardware, Proof can still be run in its original 640x350, 16-color mode.

256-color animations are *much* more attractive than 16-color animations. The expanded color palette contains 24 foreground colors and 8 background colors. The background colors consist of 7 layout colors and 1 backdrop color. The background colors do not interfere with the foreground colors and therefore give a user much more flexibility when drawing the static background portion of the layout. With more colors from which to choose, the background can be drawn with more detail without sacrificing the color integrity. In addition, in 256-color mode, objects can be multi-colored. (In 16-color mode, objects must be mono-chrome.)

4 NAVIGATING PROOF'S MODES AND MENUS

Proof Animation is organized into seven menu-driven *modes*. Each mode is a collection of closely related functions. Switching among these functions is very easy. Usually, a single mouse click is all that is required. Switching among *modes* is also easily done, but it implies major changes of context. For example, running an animation and drawing a layout are vastly different activities. Each mode has one or two main menu bars at the top of the screen. Clicking on main menu items invokes the options for the lower level tools. The seven modes are summarized as follows:

- RUN MODE
  This is the mode in which animations are viewed. It provides menu tools for starting and stopping an animation, changing views, controlling viewing speed, jumping ahead and back in time, and more.
- DEBUG MODE
  This mode provides tools for stepping through an animation by individual events or time commands and examining the resulting movement. Information pertaining to an individual object can be obtained by clicking on the object with the mouse.
- DRAW MODE
  This mode contains the drawing tools used for creating the layout background. Tools are provided for drawing static elements such as lines, arcs, text, fills, etc. Dynamic elements such as messages, bars, plots, and layout objects are defined in Draw Mode.
- CLASS MODE
  This mode is used for defining object classes. A class serves as a template for creating both the dynamic objects that move around a layout and layout objects that generally remain stationary. The template determines an object's size and shape and other initial properties such as default speed and color. An animation will usually contain multiple object classes. For example, an animation of a hospital might contain classes that represent doctors, nurses, equipment, and patients.
- PATH MODE
  This mode is used for defining fixed paths. A path is a perfect mechanism for describing for guided, directional movement such as conveyors. The geometry or route of a path is easily defined by clicking on existing lines and arcs of a layout. Tools are also provided for defining path speeds, circularity, and accumulation status. Accumulating paths provide automatic queuing for objects that pile up at the end of the path.
- PRESENTATION MODE
  This mode is used for running scripted presentations. Scripts can include static bitmap slides and snippets of animation, separated by special effect segues such as screen fades and dissolves.
- SETUP MODE
  This mode is used for examining and altering infrequently changed configuration data. For example, the color palette can be customized or the mouse speed changed in this mode.

5 CREATING ANIMATIONS AND PRESENTATIONS

5.1 Drawing the Layout

The first step in developing an animation is to draw a layout. If a .DXF formatted CAD drawing of the system is available, a user can begin by importing the drawing into a Proof Animation layout file. This is done using the built-in CAD import/export utility. Once imported, the drawing can be examined by layer or by line style. Specific layers and line styles can be deleted from the drawing as desired. When you save the resulting drawing, it is saved as a Proof Animation layout file. The original .DXF file remains intact.

If a user does not have a CAD drawing or prefers to draw using a computer, the drawing tools provided in Draw Mode are easy to use. Although it is mouse-oriented, Draw Mode also allows keyboard input, so if a user needs to draw a line of a specific length at exactly a certain angle, he or she can enter these specifications *numerically*, and the geometry will appear on the screen. To help in drawing scaled, accurate layouts, a visible grid is turned on automatically when Draw Mode is entered. For additional aid in drawing, Proof Animation has the Snap-to-Grid option. This option is also *on* as the default setting. Snap-to-Grid limits the drawing of layout elements from grid point to grid point, thus eliminating the chance of small gaps between the endpoints of *seemingly* connected lines. Other snap options which help draw accurate layouts are Snap-to-Endpoint which *magnetically* attracts the mouse cursor to the ends of lines and arcs, and Snap-to-Tangent which quickly finds points of tangency between lines and arcs. All of these options can be turned on or off by the user during the drawing session.

5.2 Defining Object Classes

Once the background of the animation is drawn, the second step in developing an animation is usually to define one or more object classes. This is done in Class Mode. Objects and object classes are among the most important constructs in Proof Animation. A class provides the geometric description of the individual objects that move throughout the animation. The class definition also includes the initial properties such as physical clearances, color, and speed of the individual objects. Each animation will usually have a collection of object classes.

It is helpful to think of an object class as the template from which the individual objects are made. An individual object is based on the single geometric description of a particular object class. There can be an arbitrary number of *objects*, such as widgets, in the system at once, but there need be only *one* widget object class.

Motion and color-changing commands in the trace file operate on *objects*. The drawn background components, produced in Draw Mode, cannot be moved or changed. If dynamic changes in background elements are required, the appropriate components must first be defined as object classes and can then be created and positioned directly in Draw Mode. Objects that are created and placed in the layout while the user is drawing the background are called *layout objects*. Layout objects enable a user to scale and position the objects into the layout while having the background components visible as reference points. While the animation is running, layout objects can be manipulated using trace file commands. For example, if an idle machine is shown as green and a busy machine as red, the machine must first be defined as an object class. Objects from that class can be created and placed as part of the layout file, and their color can be changed while the animation is executing.

5.3 Defining Paths

Proof Animation provides two kinds of motion: absolute and guided. Absolute motion, specified by the MOVE trace file command, causes an object to be moved between two points. Guided motion always occurs along a fixed route, called a path. If objects will follow guided motion, such as travel on conveyors or along guide wires, the next step in the animation development is to use Path Mode to define one or more paths.

Paths are comprised of lines and arcs that represent the route that the objects will follow. This underlying geometry must first be drawn using Draw Mode or imported from CAD. The logical path segments are then defined *on top* of the existing lines and arcs. A single line or arc can be part of one or more paths. Once defined, paths are saved as part of the layout file.

Using paths is very simple because Proof Animation does all the work. The most commonly used trace file path command is PLACE *object*ID ON *path*. Once an object is placed on a path, it will follow that path until it visually comes to rest at the end of the path or until it is PLACEd elsewhere or DESTROYed. All objects traveling on the same path can be stopped simultaneously and resume movement at a later time. Paths provide outstanding animation power in response to a single trace file command.

Accumulating paths provide even greater power for animating paths on which queuing can take place. On accumulating paths, Proof Animation reflects physical reality by *visually* queuing objects when bottlenecks occur. This often makes a simulation model of the system much simpler to construct, because such queuing need not always be explicitly represented in the model code. Most systems contain some accumulation. This property can be used to represent certain types of conveyors, cars at a traffic signal, bank lines, and more. Paths play an especially important part in transportation, product flow, and material handling animations.

5.4 Writing the Trace File

The next step in the animation development is producing the animation trace file. Trace files consist of very readable ASCII commands. Trace files are time ordered. That means the specific animation events take place between TIME commands. Consider the following portion of a trace file:

```
TIME  34.6
CREATE  PLANE 1
PLACE 1 ON RUNWAY3
SET  1 SPEED 75
TIME  52.8
```

It is very easy to visualize the results of these commands. At time 34.6, an object with an id number of 1 is created with geometry and properties inherited from class PLANE. This object will appear on screen at the beginning of a path named RUNWAY3 and begin moving along the path. The speed at which object 1 will move is set to 75 units of distance per unit of simulated time. These units are user-determined, e.g., feet and seconds. Proof Animation will continue reading trace file commands until it reads the TIME 52.8 command, signaling the end of the events that are to begin at time 34.6. It is very easy to produce simple trace files with any ASCII editor.

For most applications, it is impractical to create trace files by hand. Using a simulation model or program to generate the trace file is usually the *only* viable approach. In order to produce a trace file, output statements are inserted into the simulation model to write the appropriately formatted commands. The Proof Animation trace file command set has been designed to be easily generated. Any language with the ability to write a formatted ASCII file is capable of producing a trace file.

5.5 Building a Presentation

As an optional final step, a professional looking presentation can be built using Proof Animation. Presentation Mode lets users display scripted sequences consisting of bit-mapped screen images or slides, full animations, and/or segments selected from full animations. These presentation elements can be linked together using fades, dissolves, and other special effects, to produce a polished presentation. This is done by writing a simple ASCII presentation script file. Complete presentations can be viewed without ever exiting Proof Animation. This eliminates the awkwardness of switching back and forth between the computer and other display media during a presentation.

Slides can be created directly in Proof Animation or any software package capable of exporting industry-standard .PCX image files. There are many such packages available, and virtually all of them can produce very high-quality charts, graphs, and slides. Proof Animation can both read and write these .PCX screen images. It is very straightforward to save Proof Animation screen images as .PCX files and incorporate them into presentations as slides.

Presentations can be developed so that slides and animations appear on the screen for a defined amount of time. The viewer does not have to interact with the computer for the presentation to continue. Presentations can also be developed to continue once a key or mouse button is pressed, giving the viewer or presenter ample time to comment on what is currently on the screen.

When developing the presentation, a user can choose to highlight areas of interest within the animation by using different views or sound to draw the viewer's attention to particular aspects of the animation. Presentations can incorporate selectable menus defined by the presentation developer. These menus can be set up by topic, giving the viewer or presenter complete control and flexibility of what to show.

6 THE PROOF PROFESSIONAL ADVANTAGE

Proof Professional offers obvious advantages because it is limited only by the total memory available on a computer. No artificial memory limits are imposed; therefore, large-scale animations can be run on the PC.

Proof Professional runs surprisingly well on 386-based machines with a math coprocessor and VGA display; but when run on high-end PCs with fast video hardware, Proof Professional's performance is *stunning***.**

7 RECENT ADDITIONS TO PROOF

A number of improvements have been made to Proof to provide enhanced realism for vehicular motion. When an object representing a wheeled vehicle moves along an animation path, there are now four options for representing its motion. An object is by default assumed to be non-directional. If so, it will not rotate to point in the direction in which it is moving; i.e., it will *slide* around corners, rather than turning them. For most applications, this is unacceptable (See Figure 1). If an object is declared to be directional, and it will always point in the direction it is moving. When such objects move along arcs, they are drawn tangent to the arc. For long objects, this can produce unsightly fishtailing (See Figure 1). A second point of attachment to a path can be declared for an object. If so, the second point is called a rear guide point (RGP). At all times, both the object's hot point (normal point of path attachment) and its RGP are kept in contact with a path (See Figure 1). When real-world wheeled vehicles turn a corner, the rear wheels do not follow the same path as the front wheels. Rather, they are *pulled* along a minimum-distance trajectory, always traveling a shorter distance than the front wheels (See Figure 1). If a vehicle turns a corner and resumes linear motion, its rear wheels only *asymptotically* approach the path followed by the front wheels. Proof now offers an "RGP Pulled" option which supports very realistic wheeled vehicle motion.

8 IMPROVEMENTS UNDERWAY

As of this writing, the "native" Windows 95/NT version of Proof (Release 4.0) is under development. Rehosting Proof to be a true Windows application will greatly facilitate its use with other software; however, our plans extend well beyond rehosting. We plan to implement an interface which will allow Proof to communicate with simulation software packages by a means other than files. Although our design is not fully formulated as of this writing, in all probability, we will provide a DLL (Dynamic Link Library) version of Proof for use under Windows. DLLs provide a widely used standard for which developer and user support is readily available.

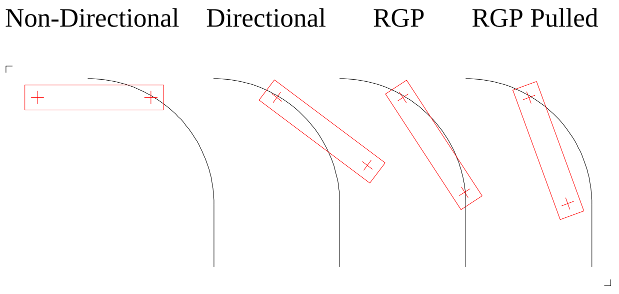

Figure 1: Wheeled Vehicle Motion

The ability to call DLLs has already been added to Wolverine's SLX simulation system. We anticipate that other vendors will develop conforming "client" applications which use Proof as an "animation server." This approach will preserve Proof's simulation language independence, but offer tighter integration between Proof and simulation tools than is possible by means of a file-driven interface. Establishing a non-file-based interface for information presently stored in layout and trace files will be a fairly straightforward first step. A second step will be to implement a new "command channel" to allow another application to exert dynamic control over Proof, e.g., "run this animation now," and "restart the current animation at time 100, using a new view."

Over the longer term, we hope to implement a user-callable subroutine interface for Proof. For example, trace file commands such as *create, place on*, etc. could be implemented as user-callable subroutines. A user-callable subroutine interface to Proof Animation is very appealing, because it would provide the basis for concurrent animation in a simulation-language-independent manner. In principle, building a callable interface is straightforward, because it only requires establishing public interfaces for primitive operations which already take place. However, the devil lies in the details. Providing greater user access into what have up until now been regarded as Proof internals will require careful re-examination of security and error-checking issues. Until a thorough re-examination of these issues is performed, it's difficult to assess how big an effort will be required to develop the user-callable interface. Accordingly, no definite schedule has been set for this development effort.

9 SUMMARY

Wolverine Software's Proof Animation has set the standard for maximum power and performance. Some of Proof Animation's many unique features include the ability to show statistics using bar graphs and plots, create presentations, built-in CAD import/export, drawing tools, smooth motion, and a unique multi-windowing display.

Proof Animation is not tied to a specific application. There are features that make it an ideal choice for the animation of systems like computer networks, health care, transportation, reengineering, manufacturing, and more while maintaining ease of use.

An animation benefits a user in every phase of the study: verification, validation, presentation of results, and the overall system design process. Proof Animation's unmatched features make it the perfect tool for each of these phases regardless of the application.

REFERENCES

Crain, R.C. 1997. Simulation using GPSS/H. In *Proceedings of the 1997 Winter Simulation Conference*, eds. S. Andradóttir, K.J. Healy, D. H. Withers, and B. L. Nelson.

Earle, N.J., and J.O. Henriksen. 1994. Proof animation: Reaching new heights in animation. *Proceedings of the 1994 Winter Simulation Conference*, eds. J. Tew, S. Manivannan, D. Sadowski, and A. Seila, 509-516. Institute of Electrical and Electronics Engineers, Piscataway, New Jersey.

Henriksen, J.O. 1997. An introduction to SLX. In *Proceedings of the 1997 Winter Simulation Conference*, eds. S. Andradóttir, K.J. Healy, D. H. Withers, and B.L. Nelson.

Wolverine Software. 1995. *Using Proof Animation (Second Edition)*. Annandale, Virginia: Wolverine Software Corporation.

AUTHOR BIOGRAPHY

JAMES O. HENRIKSEN is the President of Wolverine Software Corporation. He is a frequent contributor to the literature on simulation. Mr. Henriksen served as the Business Chairman of the 1981 Winter Simulation Conference and as the General Chairman of the 1986 Winter Simulation Conference. He has also served on the Board of Directors of the conference as the ACM/SIGSIM representative.