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

PROOF: THE GENERAL PURPOSE ANIMATOR

Nancy J. Earle
Daniel T. Brunner
James O. Henriksen

Wolverine Software Corporation
4115 Annandale Road
Annandale, Virginia 22003-2500, USA

ABSTRACT

Proof™, developed by Wolverine Software Corporation, is a high quality, PC-based animation and presentation package. Proof runs on any standard 286 or better machine with a math coprocessor and EGA or VGA graphics card. Because it is a post-processing animator, Proof provides smooth motion and portability. An open architecture makes Proof compatible with most popular simulation packages.

1. INTRODUCTION

Animation is no longer an afterthought to simulation projects. As competition for project funding becomes more intense, presentation of results becomes more important. During all phases of the project, animation is a critical tool in *presenting* simulation results to management and clients.

Animation plays an important part in *design* as well as presentation. It provides a common ground for communication between the designers and the modeler. In the past, the design team and the modelers have worked separately. This methodology has a built-in pitfall in that as the design evolves, the animation must change. Extensive animation changes used to require considerable time. A design team usually cannot stop development while waiting for animation updates.

With this in mind, Wolverine Software Corporation developed an animation software package called Proof, which helps link design, simulation and presentation. Proof has been designed as a high quality animation and presentation package, with a focus on smooth motion, portability, and ease of use.

2. OVERVIEW OF FEATURES

Proof is a PC-based product. The minimum hardware is a 286 machine with a math coprocessor and either an EGA or VGA display. This common hardware allows Proof to be portable, which is especially advantageous when projects are presented off-site.

Proof is a post-processing animation package. To run Proof, two files must exist: the layout file and the trace file. This means that the simulation model must produce a trace file which drives the animation; therefore, the size of the simulation model is irrelevant. The model can be run on a mainframe or workstation, and the resulting trace file can be downloaded onto the PC. In animations that run concurrently, the user is limited to the constraints of the particular hardware type on which the simulation is running.

Post-processing has other benefits. First, animations can be viewed at high speeds while maintaining smooth motion. An added benefit is the ability to fast-forward to a particular simulated time. This eliminates waiting for the system to "warm up." Finally, post-processing allows Proof to run in "demonstration mode." Professional-looking static slides can be created and interwoven with animation segments and shown on the computer screen.

Proof has an open architecture. This, along with the post-processor feature, makes Proof compatible with many popular simulation software packages. Actually, simulation-specific software is not necessarily needed to drive the animation. If the software used has file I/O that is capable of producing an ASCII file, it can produce a Proof trace file.

Proof has a CAD-like geometric data structure. Changing the orientation or scale of an object, static or moving, will not affect the quality of its appearance on screen.

Although it is not a full-featured CAD program, Proof has a mouse driven, CAD-like drawing mode. Through a series of pull-down menus, the layout and object definitions can be created using a series of simple and complex primitives. The layout file can also be created by reading existing CAD layouts into Proof. This CAD link is achieved through Proof's open architecture.

Proof's drawing mode has been developed to easily handle a wide variety of systems. This means that diverse applications, such as network communications and health care systems, can be handled as easily as complex material handling systems.

Proof can generate sounds, adding a new dimension to animation. Often, animations show many simultaneous events, making it difficult to draw attention to a specific event. Sound can signal machine downtimes, shift changes, or any event that the modeler would like to monitor. This can be a beneficial addition to model debugging, and an aid in system design and presentation.

3. PROOF FEATURES - A DETAILED LOOK

3.1 Hardware Requirements

The decision was made early to operate Proof 1.0 on IBM or compatible PCs, primarily because of the large installed base of color-capable systems.

MS-DOS was chosen because it behaves as a single-tasking environment (even under some third-party multitasking software, such as DESQview). This enables Proof to take total control of the CPU. The result is an animation with smooth motion. Also, new simulation packages, such as GPSS/H 386, have been developed that take advantage of a 386- or 486-based machine's faster 32-bit architecture and large memory address space while still running under MS-DOS. This means that the MS-DOS PC can handle large simulation models. Using Proof, the user can do any simulation and the animation on one machine.

All graphics software is numerically intensive. This is why Proof requires a math coprocessor, a separate chip that must be installed in an 80286- or 80386-based PC running Proof. Potential buyers of high-end PCs should note that the 80486 chip has its math coprocessing capabilities built-in.

The graphics hardware of the PC suits the needs of Proof. There were a few choices, the most obvious being EGA, VGA, or both. Proof supports both at a resolution of 640 x 350.

In Proof, one pixel needs four bits (16 colors at once), and four times 640 times 350 (the screen dimensions, in pixels) is 896,000 bits, or 112K bytes. Given the 256K of video memory found on nearly all EGA cards, Proof can *double buffer* — that is, improve the appearance of the animation by keeping two copies of the screen in video memory: displaying one while updating the other. Double buffering remains a very important quality.

We considered adding the VGA-standard 640 x 480 mode to Proof's VGA support, but a standard VGA card has only 256K of memory. At 640 by 480 by 16 colors, one screenful of information takes up over 150K, making double buffering impossible.

Recently, we have been evaluating "extended VGA" cards, which offer 512K of video memory (usually as an option). We are considering supporting one or more manufacturers' 512K VGA cards at resolutions up to 800 by 600 pixels of resolution. Also, we are carefully studying the rapidly developing 8514/A, TIGA, and "Super VGA" display options, all of which offer resolution at or above 1024 by 768. However, these will not be available in Release 1.0 of Proof.

3.2 Post-\ ocessing

Post prc essing means that the simulation runs first, then the information ɭ ovided by the results drives the animation. Some animation pack ıges run concurrently. What are the characteristics of each type?

Concurrent animation has one major benefit. It is possible with most concurrently running simulation/animation software to make certain limited types of changes to the system and watch the impact. However, many types of changes (such as scheduling changes) are difficult or impossible to animate without advance work by the modeler.

Concurrent animation has one major pitfall. It is completely dependent on the execution of the simulation model. This can be very tedious when the software is under consistent use, especially if the underlying simulation software is not particularly fast at executing.

We have summarized the difference between Proof and concurrent packages in Table 1 below:

Table 1.

	Existing Concurrent Animators	Proof
Ability to change the simulation and immediately see results	Fair	Limited
Simulation performance (execution speed)	Variable	Excellent
Animation performance (speed, smooth motion)	Poor	Excellent
Ability to "fast forward" and "rewind" the animation	Limited	Good
Ability to run models and animations on separate computers	None	Excellent

Post-processing adds to the portability of Proof. The user need only take the Proof software on site, and the target machine need not be equipped to handle the simulation software.

Finally, the post-processing approach permits the implementation of a demonstration mode. In this mode, the analyst can create complete "bullet-proof" yet interactive presentations that can be viewed by others. The "viewer" could be anyone with or without any simulation experience.

3.3 Demonstration Mode

Proof has a fully equipped demonstration mode. This provides the ability to create slides made up of words, objects, or even frozen and dynamic views of the animation. These slides can be linked together to produce a polished presentation. Eliminated are the awkward transitions from overhead transparencies to computer screen within a presentation. The user can choose to show only areas of interest within the animation, and thus draw the viewer's attention to particular aspects of the simulation.

3.4 Open Architecture

Some animation software is integrated into a simulation language or package. In order to use the animation, the user must go through the process of building a simulation model using the integrated tool. Others are post-processing, but the specifications of the trace file are not available to the user.

Proof has an open architecture; therefore, it can be used in a wide variety of contexts. The most dramatic impact of Proof's open architecture will initially be the quick adoption of Proof as the animation engine of choice for people using simulation software other than Wolverine Software's own GPSS/H.

It is also possible to build graphical depictions of systems that have not been simulated, or to build a new simulation/animation package around the Proof graphics engine. Proof can also be used by non-engineers as presentation software, even competing with established PC software packages. These capabilities open some wide doors for Proof and its users.

The architecture of Proof consists of a very simple, record-oriented animation language with English-like commands. A typical animation has a layout file and an animation trace file. The trace file is used for recording the time-dependent information that controls the animation activity. The layout file is used to hold static geometry information and initialization commands. Both files are populated with printable ASCII characters that form commands and data for the animation engine. Most of the commands can be used in either file.

Here are a few examples of the easy-to-use commands:

Drawing Commands	Animation Commands
LINE	SET...COLOR...
ARC	MOVE
POLYGON	PLACE..ON..AT..
DEFINE OBJECT	CREATE
DEFINE PATH	TIME
	DESTROY

Normally, the commands will be used repeatedly to comprise the animation trace file. This process will usually be automated. For simulation, this means that the model will write commands such as SET COLOR into the file. For non-simulation applications such as presentation graphics, the process can be similarly automated.

3.5 Graphics Data Structure

There were two choices for the graphics data structure: pixel-based or vector-based. A CAD-like, vector-based structure is used in Proof, since this allows the user to rotate an object, pan, and zoom in or out without losing the integrity of the object. In contrast, zooming in with a pixel-based system makes the object's edges appear jagged.

The power of a CAD-like data structure provides benefits in two areas. The first is the versatility of the available drawing tools. The second is the flexibility with which the display can be manipulated. Proof's CAD-like architecture allows total control of the viewing environment. This is unprecedented among PC simulation animators. The geometric data structure allows complete panning, zooming, rotating, and changes of viewpoint.

Panning allows the user to use the display screen as a "window" to a much larger "canvas" of animation activity. The size of the animation layout is virtually unlimited because of the Proof coordinate system.

Zooming allows the user to view the running animation with a microscope or a "macroscope." The viewer can either shrink the layout down so that it is all visible on one screen, or enlarge the images until main memory for storing ready-made video bitmaps becomes scarce. (This is usually quite a large zoom factor.)

Rotating pivots the viewing point about an axis. The axis is always located at the current screen center.

Changes of viewpoint — "isometric" vs. "orthogonal" viewing mode — allow the viewer to shift the vantage point from directly over the layout so that it becomes above and off to one corner. This is done without perspective, and only in two dimensions, but still adds depth to the viewing process. Many layouts give a convincing illusion of more than two dimensions in this mode.

Complementing the graphics data structure is a CAD-like drawing mode for creating the layout file. Although Proof is primarily two dimensional, it does support the concept of layers. Initial versions of Proof support two layers: the layout/background layer and the object layer. Objects can move freely over the layout and background without disrupting it on the screen. Additional layers will probably not be available during animation runs until better graphics hardware becomes available, although more layers might be added to the graphics database before that time. (Layers, in a CAD database, allow for selective display and editing of different subsystems while in draw mode.) Using a series of menus and a mouse, the static layout, dynamic objects, and paths can be created.

Proof is the first animation package that enables a two-way CAD interface via the DXF file format. This separate utility makes it possible to read in an existing DXF file to create the background portion of the Proof layout file. This saves time in completing the animation. Then, if changes are made in the animation layout, the utility can produce an updated DXF file (comprised only of the subset of CAD primitives available in Proof). The project design team can now rely on the simulation and animation as a timely and dependable design check. If changes are needed, they can first be tested with the simulation. Once a final design is achieved, the updated animation will produce the final layout in a file that can then be read into nearly any PC CAD system.

3.6 Animation Primitives

Any animation software needs basic commands or features that permit dynamically changing an object's shape, color, or location. We refer to these features in Proof as "animation primitives."

There are two types of primitives: simple and complex. Simple primitives are flexible and low-level. They allow Proof to present moving images of just about anything. Complex or higher level primitives allow Proof to animate certain sophisticated events with surprising ease.

Animation primitives are closely interrelated with the graphics data structure of Proof. The most important concepts are the Object Class and the Object.

An Object Class is a geometric description of some type of object, such as an Automated Guided Vehicle (AGV) in a material handling animation. An automobile traffic model might have five different Object Classes: Cars, Trucks, Buses, Campers, and Motorcycles.

An Object is a subset of an Object Class. Expanding on the traffic model mentioned above, one could have northbound and southbound cars; cars making continuous turning movements; red, green, or beige cars; large cars and small cars. Each of these cars is an Object, based on the single geometric description of a car. There can be an arbitrary number of "Car" Objects in the system at once, but there need be only one "Car" Object Class.

All of the motion and color changing primitives in Proof operate on Objects. Note that we have not discussed background drawing (e.g. plant floor layout). Most layouts will be drawn directly on the screen and their components cannot move or change color. However, if movement or color change is desired, then the components can be made into Objects.

3.7 Making Things Happen

The simple things the user can do with an Object include: CREATE or DESTROY, PLACE (making it visible), SET COLOR and SET VELOCITY, and MOVE (causing the Object to move smoothly from points A to B.) Object movement can also be achieved via a Path.

The more complicated things one can do with an Object involve Paths. Actually, using Paths is very simple, because Proof does all the work. That is why we refer to Paths as a higher level primitive. The most commonly used Path command is PLACE [object] ON [Path].

A Path is a data structure composed of an arbitrary number of line and/or arc segments. Once an Object is placed on a Path, it will follow the Path until it comes to rest at the end. Paths provide outstanding power in response to a single command.

A variant is the Accumulating Path, which offers even more power. On an Accumulating Path, Proof reflects physical reality by allowing objects to queue if there is a blockage. This often makes a simulation model of the system much simpler to construct. A surprising number of systems behave in this manner, from certain types of conveyors to supermarket checkout lanes.

Paths play an especially important part in transportation and material handling animations. In an AGV system, for instance, the AGV travels from control point to control point. On the simulation side, routing and distance data are needed in the model. A design change, such as adding another control point between two existing points, usually requires an updated version of each matrix. Without Proof, these matrices can be very time consuming to create and edit.

Proof has made life simpler by automatically maintaining distance information for each Path and by allowing the person creating the animation to specify routings over multiple Paths. This enables Proof to write travel sequences and the distances between consecutive points to data files that can be read by the simulation software when the model executes. Now, when a change is made on the Paths in the layout, the means are provided for automatically updating the simulation model. This feature, still under development, may not be available in Proof 1.0.

3.8 Smooth Motion

Smooth motion was a primary design goal for Proof, and it has been achieved with stunning success. In most media, it is necessary to create and recreate static images rapidly in order to create the illusion of motion. This is, of course, the principle behind motion pictures and television as well as cartoon animation.

In the case of a computer and raster-based CRT screen, or the equivalent raster-based video game, the image is created as a set of discrete pixels represented in video memory. For these applications, the pixel representation must be either recreated continuously at different locations, or saved and "blitted" to different locations on the screen. This process must be repeated many times per second, or the motion will appear jerky.

How fast is fast enough? Motion pictures run at something like 20 frames per second, and standard television at about 30 frames. Simulation animation software available in the 1980s was plagued by slow frame rates. Due to the discreteness of the pixels, computer displays of artificially created objects can require even higher frame rates than television, or the motion will appear to buzz or jerk. The frame rates on much of the available software have been on the order of 10 frames per second or less, while Proof has starting rates of 60 to 70 frames per second

What happens when Proof cannot keep up? With other animation software, the apparent speed of objects moving across the screen generally diminishes in such circumstances. With Proof, a constant (though user-adjustable) ratio of animated time to "wall clock" time is always maintained, even when the model cycles between being congested and not-so-busy. This ratio is maintained by reducing the frame rate and increasing the increment by which each object travels. Proof performs this adjustment continuously. With Proof's high starting frame rates, the effect of reducing the frame rate remains visually acceptable.

3.9 Reasonable Cost

Wolverine Software Corporation is committed to driving the performance of PC animation software up while driving the price down. That is why Proof is more affordable than other existing products.

4. SUMMARY

Animation is a powerful addition to any simulation effort. An animation benefits the modeler in verification, validation, and presentation of results (especially to an audience not familiar with simulation).

Simulation and animation technology is improving. Wolverine Software Corporation is contributing to this improvement by providing an innovative animation package called Proof. This general purpose animator boasts many important features. Among these features are portability, smooth motion, an open architecture, a CAD-like structure and drawing mode, the ability to create presentations, and affordability.

REFERENCES

Brunner, D. (1990), *Alpha Proof Documentation*, Wolverine Software Corporation, Annandale, Virginia.

Brunner, D. and J.O. Henriksen, "A General Purpose Animator," In *Proceedings of the 1989 Winter Simulation Conference*, E.A. MacNair, K.J. Musselman, and P. Heidelberger, Eds. IEEE, Piscataway, NJ, 249-253.