Proceedings of the 1992 Winter Simulation Conference
ed. J. J. Swain, D. Goldsman, R. C. Crain, and J. R. Wilson

PROOF ANIMATION: THE GENERAL PURPOSE ANIMATOR

James O. Henriksen
Nancy J. Earle

Wolverine Software Corporation
4115 Annandale Road
Annandale, Virginia 22003

ABSTRACT

Proof Animation™ is a family of general-purpose, file-driven, vector-based, postprocessing animation software products which runs on readily available, inexpensive PC hardware and includes CAD-like drawing tools and a unique presentation subsystem. Proof Animation features an open architecture, allowing it to serve as an animation "back end" for models written in a wide variety of simulation languages. Proof Animation's mode-based, menu-driven user interface is easy to learn and use, and its superior run-time performance assures smooth, realistic motion in animations with many moving objects.

1 INTRODUCTION

Animation is an extremely powerful tool in *all* stages of a simulation project. However, the capabilities of animation software which are most important to success differ considerably from stage to stage.

At the outset of a simulation project, the most important attribute of animation software is the *feasibility* of its use. Will the animation software work well with the modeling software to be used? Will its demonstrated capabilities to do simple animations be sufficient to handle a full-scale, industrial-strength project? Is the software affordable?

During the development of a simulation/animation, ease-of-use issues are of paramount importance. How easy is it to draw (or import) the animation layout? Does the animation software include constructs which lessen the modeling burden? How easy is it to define and modify the characteristics of moving objects? How easy is it to "home in" (in time and space) on system glitches, to determine whether they represent legitimate quirks of system operation or modeling errors?

As one approaches the end of a project, issues of portability and effectiveness of presentations become very important. Can I take an animation down the hall to my boss's office, or take it to a customer's site, or does it require special hardware that they don't have? Can I assemble a collection of slides and animation clips to summarize the results of the simulation/animation study?

The Proof Animation family of software products provides favorable answers to *all* the questions posed above. In the sections which follow, we describe the family of products, discuss their underlying design philosophy, describe their organization, and give an overview of how they are used.

2 THE PROOF ANIMATION FAMILY

The following products comprise the Proof Animation family:

- PA — The basic animator. Requires a 286 or better processor, a math coprocessor, and an EGA/VGA video card. DOS 640K memory limitations apply.
- SPA — The student version of Proof Animation. Included with the *Using Proof Animation* textbook (Brunner 1992). Size and playing time limitations are imposed; otherwise identical to PA.
- PP — Proof Professional. Requires a 386 or better CPU. Uses 32-bit DOS-extender technology to break the 640K barrier. 1024 X 768 high-resolution version included at no additional cost.

- PADEMO The Demo version of PA. Can only be used to run animations prepared under a licensed copy of PA containing a "Demo-Maker" add-on. Copies of PADEMO can be reproduced and distributed free of charge.

- PPDEMO The Demo version of PP.

- DXF2PA An add-on utility for converting industry-standard .DXF CAD files into Proof Animation layout (.LAY) files.

- PA2DXF An add-on utility for converting Proof Animation layout files into .DXF files.

3 DESIGN PHILOSOPHY

3.1 MS-DOS, PC Platform

Proof Animation was designed to use widely available hardware, in order to maximize portability. PA, the basic version of Proof Animation, requires only a 286 or better CPU, a math coprocessor, and an EGA/VGA-compatible video card. Configurations of this type are very widely available. High-end 486-based PCs, ideal platforms for running Proof Professional (PP), are available for under $2,000. The choice of MS-DOS as a host operating system was made because of its enormous installed base and its simplicity of operation (no multi-tasking). Proof Animation (PA) can also be launched as a full-screen application under both Windows 3.1 and OS/2 Release 2.

3.2 General-Purpose Orientation

Proof Animation was designed to be general-purpose in two ways. First, it was designed to be independent of any particular simulation language. While for obvious reasons, we'd prefer that you use Wolverine GPSS/H as your simulation language, we are pleased to have provided animation software to users who develop models in SIMAN, SIMSCRIPT, and SLAM. Animation software from most vendors is tightly coupled to their simulation software. The advantage most often touted for this approach is that developing animations is made easier by the animator's direct, automatic access to the events which occur in a simulation. This is true for small, simple animations, in which there is a one-to-one relationship between simulation events and animation events, *e.g.*, initiating or stopping the motion of an object. In more complex simulations, however, the relationships between these two kinds of events are too complicated to fall within the scope of simple, built-in rules. "Programmable" logic is needed to handle these situations.

The second way in which Proof Animation is general purpose is in the design of its command set. Commands such as CREATE, DESTROY, PLACE, MOVE, and SET COLOR are easily learned and easily used. They provide exactly the kind of flexibility necessary to implement programmable animation logic.

Purveyors of tightly coupled simulation/animation software would have you believe that their approach is the *only* way to add animation to a simulation. It's not. A mix-and-match strategy for acquisition of software allows you to select the functionality you want, at prices you are willing to pay. Sole sources of any goods or services tend to be expensive.

3.3 ASCII File-Driven Architecture

Two input files are required for every animation run under Proof: a layout (.LAY) file, and a trace (.ATF) file. The layout file describes the geometry of the fixed background over which objects move, and it provides definitions for such things as shapes and colors of objects, and paths along which objects move. The trace file contains a time-ordered sequence of commands such as CREATE, DESTROY, PLACE, MOVE, etc. It is the life's blood of an animation.

Trace files are free-format, and the syntax of trace file commands is designed for ease of generation. Any language which can produce formatted ASCII output can easily write a trace file. In SIMAN, the WRITE statement is used; in SIMSCRIPT, the PRINT statement is used; and in SLAM, FORTRAN-coded event routines are used. In Wolverine GPSS/H, PUTPIC and BPUTPIC are especially well suited for writing trace file commands.

Ordinarily, layout files are produced at least in part by using Proof Animation's CAD-like drawing tools; however, because we publish the .LAY file command set specifications, it is possible to write programs to generate layout files. For example, some of our users have written front ends which query a user for system design parameters and use these values to design a physical layout which is written into a .LAY file.

3.4 Importing .DXF and .PCX Files

Sometimes a CAD drawing already exists for a system which is to be animated. Directly using the CAD drawing offers two advantages: (1) the effort of redrawing an entire layout can be avoided, and (2) credibility with

end-users of the animation is enhanced, because they are used to viewing the CAD drawing of the system. Proof Animation's optional add-on DXF2PA and PA2DXF utilities provide the capability of converting industry-standard .DXF CAD files into Proof Animation layout files, *and vice versa*. Note that as of this writing, Proof Animation is the only animation software we know of which allows translation in *both* directions.

In addition to being able to import and export CAD files, Proof Animation can read and write bit-mapped screen images. It is very straightforward to save Proof Animation screen images in industry-standard .PCX files and incorporate them into presentations as *slides*. Third party packages for producing very high-quality charts, graphs and slides can also be used. There are many such packages available, and virtually all of them can export images as industry-standard .PCX files.

3.5 Maximizing Interoperability

Taken collectively, the design philosophies presented in the three preceding sections can be summarized as maximizing *interoperability* with other software. We live in a mix-and-match era of software acquisition. Clearly, no single vendor can hope to be the best possible source of software to fulfill *all* the requirements of an animation (simulation, animation, CAD, and presentation graphics). Proof Animation's use of an open, published, flexible, file-driven architecture, and its ability to read and write industry-standard file formats, enable users to implement a mix-and-match approach.

3.6 Post-Processor Operation

Proof Animation is a "post-processing" animation package. As we have seen above, Proof Animation requires two input files, the layout file and the trace file. These files must be produced prior to invoking Proof Animation; they cannot be written and read concurrently. The post-processing approach offers two advantages: (1) it allows total exploitation of PC hardware resources for doing the animation, and (2) it allows the "playing" of an animation many times and in many ways without incurring the overhead of rerunning the simulation which drives the animation. The ability to jump back and forth in time during playback, coupled with the ability to speed up and slow down viewing speed, makes it easy to investigate unusual system behavior.

3.7 Vector-Based Descriptions

In Proof Animation, all layout information is stored in vector form. Vector-based descriptions are automatically mapped into pixels (dots) to build a screen image. The advantages of this approach include the following: (1) layouts can be much larger than a single screen; (2) it is possible to pan, rotate, and zoom in and out on demand; and (3) objects can be made to *rotate* (rather than *slide*) as they go around corners.

Many animation packages use a pixel-oriented approach for drawing. This offers one great advantage: by being able to manipulate individual pixels, one can produce detailed, artsy images. However, this advantage is outweighed by the following potentially crippling disadvantages: (1) pixel-oriented images cannot be rotated; (2) enlarging pixel images magnifies the "jaggies" inherent in all such images, but tolerable with normally sized images; (3) 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.")

3.8 Emphasis on Performance

A great deal of emphasis has been placed on Proof Animation's run-time performance. High performance enables Proof Animation to achieve very smooth motion (60-70 updates per second). Other software can often sustain refresh rates of only 5-10 updates per second. The ultimate purpose of an animation is to convince someone that you know what you're doing. Objects which move smoothly across the screen are more realistic than those that don't. Everyone knows that forklifts are supposed to roll across the factory floor, not jump.

4 THE ORGANIZATION OF PROOF ANIMATION

Proof Animation is organized as a collection of menu-driven *modes*. A 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 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 lower level tools.

Proof Animation's modes are summarized as follows:

Run Mode is the mode in which animations are viewed. It provides menu tools for starting and stopping an animation, controlling viewing speed, jumping ahead and back in time, etc.

Debug Mode is a variant of run mode. It provides tools for stepping through individual events in an animation, examining moving objects, etc.

Draw Mode is used for constructing layouts. Tools are provided for drawing lines, arcs, text, messages (text which can be altered at run-time), bars (for run-time display of statistics), area fills, and named, preinitialized objects.

Path Mode is used for defining fixed paths along which objects can move. The geometry of a path is quickly defined by clicking on lines and arcs of a layout. Tools are also provided for defining path speeds, circularity, and accumulation status. (Accumulating paths provide automatic queueing for objects which "pile up" at the end of the path.)

Class Mode is used for defining object classes. An object class serves as a template for creating dynamic objects which move around a layout and static objects which generally remain stationary. The template determines an object's size and shape and other properties such as default speed. An animation of a freeway toll booth might, for example, contain object classes for cars, trucks, buses, and toll booths.

Presentation Mode is used for running scripted presentations. Scripts can include static slides and snippets of animation, separated by special effects, *e.g.*, screen fades, dissolves, etc.

Setup Mode is used for examining and altering infrequently changed "configuration" data, *e.g.*, altering the color palette.

5 OVERVIEW OF PROOF ANIMATION USAGE

5.1 Drawing a Layout

The first step in developing an animation using Proof is to draw a layout. If you're fortunate enough to have a CAD drawing of the system you wish to animate, you can begin by importing the CAD drawing (using the DXF2PA add-on utility). The drawing tools provided in Draw Mode are easy to learn and use. While heavily mouse-oriented, Draw Mode also provides for keyboard input, so if you need to draw a line of length 12.5 at a 45-degree angle, you can enter these specifications *numerically*, rather than settling for a mouse-drawn approximation.

5.2 Defining Object Classes

The second step in developing an animation is to define one or more object classes, in Class Mode. Objects and object classes are among the most important constructs in Proof. An object class is a geometric description of some type of object, such as an automobile. A traffic model might include object classes for Automobiles, Trucks, Buses, Campers, and Motorcycles. In addition to shape information, an object class contains a few other properties such as physical clearances, color, and an optional speed.

Although Proof Animation does not purport to implement a true "object oriented" framework, it is meaningful to call an object an "instance" 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; fast cars and slow cars. Each of these cars is an object, based on the single geometric description of an automobile. There can be an arbitrary number of "Automobile" objects in the system at once, but there need be only one "Automobile" object class.

All motion and color-changing commands in Proof Animation operate on objects. Most layouts are drawn directly on the screen, and the background geometry components cannot move or change color. If such changes are required, the appropriate components must be defined in an object class, and objects from the class must be created and prepositioned in Draw Mode. Objects so defined and created are called *layout objects*.

5.3 Defining Paths

The next step in developing an animation is almost always to define one or more paths, using Path Mode. Proof Animation provides two kinds of motion: absolute motion and guided motion. Absolute motion, specified by the MOVE trace file command, allows moving an object between any two arbitrary points A and B. Guided motion always occurs along a fixed route, called a path. The geometry of a path is defined by clicking on previously drawn (or imported) lines and/or arcs. 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 path command is PLACE [object] 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). Paths provide outstanding power in response to a single trace event command.

Accumulating paths provide even greater power for animating paths on which queueing can take place. On

accumulating paths, Proof Animation reflects physical reality by *visually* enqueuing objects when blockages occur. This often makes a simulation model of the system much simpler to construct, because such queueing need not always be explicitly represented in the model. 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.

5.4 Producing a Trace File

In order to produce a Trace File, one must insert output statements into a simulation model, to produce appropriately formatted commands. The Proof Animation Trace File command set has been designed to be easily generated. Any language which can do formatted writes to an ASCII file is capable of building a Trace File.

5.5 Building a Presentation

As an optional, final step, one can build a Proof presentation. Proof Animation's Presentation Mode lets users create scripted sequences of slides and animated segments selected from full animations. These presentation elements can be linked together, using fades, dissolves, and other special effects, to produce a polished presentation. Complete presentations can be viewed without ever leaving Proof Animation. Thus, awkward switching back and forth between overhead transparencies and computer displays during a presentation is eliminated. The presentation developer can choose to highlight areas of interest (in space or time) within the animation, and thus draw the viewer's attention to particular aspects of the simulation. Self-directed presentations can be developed by incorporating viewer-selectable menus.

6 SUMMARY

Animation is a powerful addition to any simulation effort. An animation benefits the modeler in verification, validation, and presentation of results, and helps with the overall system design process.

Simulation and animation technology is improving. Wolverine Software Corporation is contributing to this improvement by providing an innovative animation package called Proof Animation. This general purpose animator boasts many important features. Among these features are the ability to create presentations, an open architecture (for compatibility with a variety of software), a CAD-like structure, smooth motion, and powerful drawing features.

REFERENCES

Brunner, D.T. 1992. *Using Proof Animation.* Annandale, Virginia: Wolverine Software Corporation.

Brunner, D.T. and N.J. Earle. 1991. *Proof Animation CAD Translator User's Guide.* Annandale, Virginia: Wolverine Software Corporation.

Brunner, D.T. and J.O. Henriksen. 1989. A General Purpose Animator. In *Proceedings of the 1989 Winter Simulation Conference*, eds. E.A. MacNair, K.J. Musselman, and P. Heidelberger, 249-253. Institute of Electrical and Electronics Engineers, Piscataway, New Jersey.

Earle, N.J., D.T. Brunner and J.O. Henriksen. 1990. Proof: The General Purpose Animator. In *Proceedings of the 1990 Winter Simulation Conference*, eds. O. Balci, R.P. Sadowski, and R. Nance, 106-108. Institute of Electrical and Electronics Engineers, Piscataway, New Jersey.

AUTHOR BIOGRAPHIES

JAMES O. HENRIKSEN is the president of Wolverine Software Corporation. He is a frequent contributor to the literature on simulation and has presented many papers at the Winter Simulation Conference. 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.

NANCY J. EARLE is a Senior Industrial Engineer at Wolverine Software Corporation. She received B.S. (1982) and M.S. (1984) degrees in Industrial Engineering from Purdue University, where her concentration was in simulation. She joined Wolverine in 1989. Her responsibilities include consulting, training, technical support, and product development support. Previously, she worked for Corning, Incorporated as a simulation analyst in manufacturing. While there, she developed and taught short courses in simulation. Ms. Earle is a member of SCS and is the Exhibits Chair for the 1992 Winter Simulation Conference.