Proceedings of the 1989 Winter Simulation Conference
E.A. MacNair, K.J. Musselman, P. Heidelberger (eds.)

A GENERAL PURPOSE ANIMATOR

Daniel T. Brunner
James O. Henriksen
Wolverine Software Corporation
4115 Annandale Road
Annandale, VA 22003-2500, USA

ABSTRACT

During the 1980s, the simulation discipline has been revolutionized by the introduction of software for graphical, animated depiction of a running simulation on a display screen. Advocates of simulation animation have given several main reasons for using animation.

There have also been some detractors who felt that animation was not always an appropriate tool. However, it has become clear that animation is here to stay. Discussion is shifting to comparisons among various animation tools and mapping out future directions for animation.

Wolverine Software Corporation waited several years before beginning development of animation software. The company began animation development work in 1987, with the benefit of having observed and otherwise learned about the strengths and weaknesses of then-existing software.

The preliminary results of the development effort are now available in product form. Wolverine has developed a general purpose animator. This software has a wide variety of applications and, thanks to its open architecture, can be used in conjunction with virtually any simulation software and in some non-simulation areas as well. For the IBM PC platform on which it runs, the product redefines several performance characteristics, most notably with its smooth motion and its geometry-based internal data structure.

1. INTRODUCTION

The benefits most often cited by advocates of simulation animation include:

(1) Animation helps those who have created a simulation to "sell" their quantitative conclusions to skeptical upper managers.

(2) Animation, used during the model development cycle, helps the model builder(s) build, verify, and validate the model.

(3) Animation, because it is flashy and fun, helps users and managers generate and maintain interest in exercising the analytical power of simulation, to the presumed benefit of everyone.

Many have urged caution in applying this new technology. A sampling of some of the reasons follows:

(1) Excessive reliance on animation (typically by the upper manager) can dilute or even completely overshadow the reality of the quantitative results.

(2) The time spent in the creation of realistic animations can be long enough to add significantly to the cost of a simulation project, sometimes without quantifiable benefits.

(3) Once animation has helped legitimize simulation in new places, it may no longer be needed, to the potential political detriment of both end users, who may have spent thousands of dollars on hardware and/or software for simulation animation, and vendors, who have invested heavily in software development.

Early-adopting vendors and their products that have influenced animation in the U.S. market include Istel, Inc. (SEE WHY), Minuteman Software (GPSS/PC), Simulation Software Ltd. (PCModel), AutoSimulations, Inc. (AutoMod/AutoGram), Systems Modeling Corp. (Cinema), and Pritsker Corp. (TESS). (Note: Product names used throughout this paper are generally trademarks of the respective companies.)

A few vendors held out against animation, most notably Wolverine Software Corporation. The reasons most often cited were negatives (1) and (3) above.

However, time has passed, animation continues to gain in popularity, and even Wolverine has jumped on the bandwagon. "We are all animationists now."

2. THE PATH TO A NEW TOOL

When Wolverine Software decided in December of 1987 to produce animation software, it had an advantage over its competitors. That advantage was the ability to survey the animation landscape and start fresh on something new.

Existing animation software had strengths in 1987, but offered many areas for improvement:

- Some of it required special hardware.
- Most of it had geometry that was character-based or pixel-based.
- The available animation primitives, while useful, were stylized and sometimes applied only to a particular application area.
- Most could not update the screen fast enough to produce smooth animation of even small simulations.
- Some of it could not run at the desired speed due to sharing one CPU with a running simulation.

- Closed architectures and tight couplings with particular simulation tools limited expansion and customization possibilities.

- Most of it was expensive.

Wolverine's new software was unnamed at press time for the conference proceedings. It is referred to throughout this paper as "Product X". The Product X software addresses the concerns outlined above:

- Product X runs on any 286-or better PC, with a math coprocessor, that has an EGA or VGA display. This allows great portability of animated models for demonstration purposes.

- Product X has a CAD-like data structure for accurate rendering of backgrounds and moving objects at any scale or orientation. Full bitmap accuracy is generated when the data structures are rendered.

- Although its powerful high-level primitives are influenced by demand for solving material handling problems, Product X has a set of extremely low-level and flexible primitives that can be manipulated to address a wide variety of needs. Even the material handling primitives have already been used in areas as diverse as chemical processing and transportation.

- Product X has been carefully crafted at the machine instruction level to produce dramatically high screen refresh rates. On a 386-based VGA PC, animations usually run at 70 updates per second.

- Product X is post-processed. This accomplishes three things. First, it allows very high speed animations, including fast-forward to any point in simulated time. Second, it allows the simulation and the animation to run on different machines. One configuration might be a simulation language running on a "real" computer such as a workstation, VAX, or mainframe, with an Ethernet link to a single-tasking PC (which is well suited for display-intensive applications) running Product X. Finally, post-processing allows development of an Product X "demonstration mode" for creation of highly tuned presentations depicting complete simulation studies.

- Product X has an open architecture. This opens wide areas of future application to vendors and users of other software. The animation processor is detached completely from the underlying simulation tool. Product X is the first animation tool that is adaptable for use by any simulation tool that has a file I/O capability available, including GPSS/H, SIMAN, SLAM II, GPSS/PC, and AutoMod, and for many non-simulation applications as well, including real-time displays and desktop presentations.

- Product X is initially being marketed to people already active in simulation for whom the desired level of animation quality has been unavailable due to the cost of the software.

3. PRODUCT DESCRIPTION

Let us examine each of the seven areas of improvement in more detail, fleshing out our description of Product X as we do so.

3.1 Hardware Requirements

One of the most important decisions taken in late 1987 was the choice of a primary hardware platform. Surveying the hardware landscape during the second half of 1989, we find tens of millions of IBM and compatible PCs, a substantially smaller number of Macintoshes, and a rapidly growing contingent of Unix-based workstations. There are also many organizations that apply the superior instruction-crunching capabilities of IBM mainframes and high-end VAXes to their simulation problems.

The first question in 1987 was which family of hardware to consider: mainframes, workstations, or PCs.

3.1.1. Large Computers. Mainframes (for this discussion, that includes VAXes) were, and still are, out of the question, due to the limited communications bandwidth between the host processor and the display. Dedicated terminals, even high-end graphics terminals, are typically connected to mainframes by links that are too slow to support raster graphics or even higher-level graphics commands at the rates needed for smooth animation.

By making the "terminal" smarter and smarter, one can reduce the need for communication bandwidth. In the limiting case, the display is itself a computer. It made more sense to us to offload the less demanding CPU-intensive part of the animation software as well as the power-hungry graphics. The mainframe is then no longer needed by the animator at all.

3.1.2. Workstations. Graphics workstations are the cream of the crop for standalone graphics applications. However, they too have problems. Despite their image of enormous graphics horsepower, we have observed that the leading edge of workstations graphics tends to cater to visual simulation. How many zillions of colors can it display? How many kinds of shading can it do? What about hidden surface removal? There is no discrete simulation animation software available (in 1989) that portrays photorealistic images, so many of these features are wasted on the expensive workstations that support animation. In fact, such systems can prove slower than one would expect for discrete simulation animation due to their orientation toward producing beautiful but often static images.

Another problem with Unix-based workstations is their multitasking, multiuser environment. Good animation software makes frequent demands for chunks of CPU time, and cannot wait for even 1/60 second for someone else to load a large file. Even in standalone mode this seemed likely to create problems, because Unix itself generates many little processes that nibble away at the CPU periodically.

Finally, and most importantly, we had (in 1987) watched a fine crop of Motorola-based workstations, already GPSS/H-capable since 1985, attract little interest from potential customers. Everyone wanted software for a PC, it seemed, and Sun and Apollo were not well known among prospective discrete event simulation users. (As of 1989, this has of course changed dramatically, and simulation software is becoming more popular on certain workstations,

including the Motorola-based models and the RISC-based Silicon Graphics 4D (MIPS) and Sun-4 (SPARC) machines.

3.1.3. Personal Computers. In personal computers, there were (and are) IBM PCs and Macintoshes. PCs were the obvious choice, but we did consider Macintosh.

Why not Macintosh? There were several reasons. Macintoshes, aside from comprising not more than 10% of the installed personal computer base, had a specific problem in 1987. Graphics-intensive applications demanded cycles from the main CPU chip for all calculations and writes to video memory (which was just a dedicated piece of not-terribly-fast main memory). Worse, the only path to the main/video memory was through ROM-based I/O routines written in 680x0 assembly language. This meant that the only way to build a graphics co-processor that would run with all Macintoshes would have been to try to create one based on a second Motorola 680x0 CPU. To the best of our knowledge, no one built such a device. (In the summer of 1989, we have finally seen the introduction by one manufacturer of a NuBus-based graphics accelerator, which works if one has that manufacturer's display adapter card. All calls to the ROM-based screen graphics routines are trapped and routed to the card.) Apple Computer, in 1987, provided us with information that convinced us that that high-speed simple graphics would not come quickly to Macintosh. In 1989 we see Macintosh behaving more and more like a workstation, with image beautification taking precedence over frame rates, and the threat of disruptive multitasking on the horizon.

Also, there was the overriding marketing factor. People really seemed to want software that would run on an IBM PC, although this attitude is beginning to erode somewhat in 1989. PCs and compatibles are somewhat handicapped by primitive and memory-constrained MS-DOS operating system and a slow bus. (As we will discuss later, MS-DOS does have one significant advantage for animation.) However, PCs are cheap and ubiquitous, and we wanted to stress portability in our final design. There is certainly no other machine that one is practically guaranteed to find on site at any location.

So, the decision was made. Product X would run on the PC. But the story does not end there.

3.1.4. Graphics Hardware. PC graphics hardware has evolved steadily over the years. Our biggest problem during the development cycle (and it cropped up more than once) was, "EGA or VGA?" What about dedicated graphics accelerator cards (then as now, very expensive if significant performance gains were included)? At least one competitor took the latter approach, requiring them to provide an add-in board (one not commonly known or generally available) along with their software.

In late 1987, VGA was new and just catching on. Third-party VGA cards for non-IBM PS/2 machines were not yet available. EGA had been around for over two years, the cards were stable, and there was a substantial installed base. From a marketing standpoint, we wanted people to be able to take their animations into anyone's office and run them.

The overriding reason to stick with EGA, however, was technical. Standard EGA cards have 256K of video memory. One standard EGA pixel needs four bits (16 colors at once), and four times 640 times 350 (the EGA screen dimensions, in pixels) is 896,000 bits, or 112K bytes. This lets us 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 was and remains very important to us.

We considered VGA, but standard VGA also has 256K of memory. At 640 by 480, one screenful of information in 16 color mode takes up over 150K. Double buffering in VGA mode is impossible with a standard VGA card. Also, standard VGA cards offer EGA emulation modes (some better than others, as we have discovered and dealt with). This means that standard VGA card users can run standard Product X in EGA mode.

Recently, we have been evaluating "extended VGA" cards, which offer 512K of video memory, usually as an option. In fact, we are considering supporting one or more such 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 standards, all of which offer resolution at or above 1024 by 768.

3.1.5. Operating Environment. One final note is the choice of operating environment. In 1987, standard DOS was the only real option. Now that OS/2 and Unix have gained a little more support, we could consider them. Unfortunately, we would lose the single-tasking environment, that by providing total control over the CPU, is so good for animation (this is the hidden strength of MS-DOS alluded to earlier). If memory proves to be a problem (it has not so far, as Product X processes its occasionally voluminous input information from disk in a serial fashion), we will consider taking Product X to a DOS-Extended environment.

3.2 Graphics Data Structure

It was obvious from the beginning that we needed a CAD-like data structure, in which all graphics data were stored using coordinates instead of pixels. In this we were probably influenced by AutoSimulations' AutoMod, which could rotate, pan, and zoom on its high-powered workstation display. No one had done anything with as much capability as AutoGram on a PC. Only one product (PCModel/GAF, later renamed CADmotion, from SimSoft) had a vector-based orientation at all. Of the other PC-based animators, SEE WHY/Witness was character-based, as was GPSS/PC. Cinema, the well-known PC simulation animation software from Systems Modeling, was pixel-based.

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.

3.2.1. Drawing Capabilities. Product X will have a drawing environment that resembles a simplified CAD program. At this writing, the drawing environment is under development, and cannot be described in detail.

In the absence of a drawing environment, early commercial users of Product X have had to create manually the ASCII layout file that contains the geometry information. This has been done using a set of low-level drawing primitives such as LINE, ARC, and TEXT. Although this takes some extra time, and results in simplified looking objects and layouts, the process has not been an obstacle to the creation of some impressive animations. The published layout file format, which will continue to exist once the

drawing environment is available, is an important feature (see Section 3.6).

The drawing primitives and the drawing environment will share a common characteristic: the same techniques are used for drawing background information, moveable Objects, and Paths (see Section 3.3.3 for more about Paths). It will also be possible to scale and manipulate drawn entities. This manipulation will include creation and maintenance of object libraries.

Although Product X is primarily two dimensional (see Section 3.2.2), it does support the concept of layers. Initial versions of Product X 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 (see Section 3.1.4), 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).

3.2.2. Viewing Modes. A CAD-based architecture allows unprecedented (among PC animators) control over the viewing environment. The geometric data structure allows complete panning, zooming, rotating, and changes of viewpoint. The best way for us to illustrate the viewing power of the CAD-based architecture is with illustrations.

Figure 1 is a screen snapshot of a running animation. (Menus have been omitted.) This simple AGV layout was created and animated as part of a benchmark study (McKay and Rooks 1989). For more information on the simulation model on which this animation is based, see reference (Crain and Brunner 1989). Figures 2 through 5 are different views of the animation, frozen at the same instant in animated time.

Panning allows the user to use the display screen as a "window" to a much larger "canvas" of animation activity. The Product X coordinate system allows animations of virtually unlimited size. Figures 4 and 5 have been panned.

Zooming allows the user to view the running animation with a microscope or a "macroscope." The viewer can always shrink the layout down so that it is all visible on one screen. Or, the viewer can grow the images up until main memory for storing ready-made video bitmaps becomes scarce. (This is usually quite a large zoom factor). Figures 2, 4, and 5 have been zoomed in or out.

Rotating allows the stationary observer to "turn around" at the overhead viewing point in orthogonal mode (see below), and to "walk around the catwalk" in isometric mode. Rotation always occurs about the screen center at the then-current pan location and zoom factor. There is no illustration of rotation, which is similar to isometric mode (see below) but without the foreshortened effect.

Changes of viewpoint — we call it "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 a flavor of "z" (depth) to the viewing process. Many layouts give a convincing illusion of more than two dimensions in this mode. Figures 3 and 5 show isometric mode.

3.3. 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 Product X as "animation primitives."

There are really two things to look at here. The first is the flexible, low-level primitives that allow Product X to present moving images of just about anything. The second is the higher level primitives that allow Product X to animate certain sophisticated things with surprising ease.

This discussion is necessarily brief, and omits some existing and many potential features.

3.3.1. Objects and Object Classes. Of course, the animation primitives are closely interrelated with the graphics data structure of Product X. The most important concepts are the Object Class and the Object.

An Object Class is a geometric description of some type of object. It could be 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, on the other hand, is 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; 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 needs to be only one "Car" Object Class.

Note: One could correctly infer from the above paragraph that Objects can have only one color. This is true, but as of this writing, Compound Objects are under development and should be part of the first release of Product X.

All of the motion and color changing primitives in Product X 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 that is desired, then the components can be made into Objects. (See discussion of AGV example model.)

3.3.2. Simple Object Manipulation Primitives. The simple things one can do with an Object include: CREATE or DESTROY (causing it to exist or not), PLACE or ERASE (making it visible or not); MOVE (causing smooth motion from point A to point B); SET COLOR; and SET TEXT (there is a single variable text field present in each object, for a tag or value to be displayed; there could also be an arbitrary amount of non-varying text).

3.3.3. Paths. The more complicated things one can do with an Object involve Paths. Actually, using Paths is very simple, because Product X 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

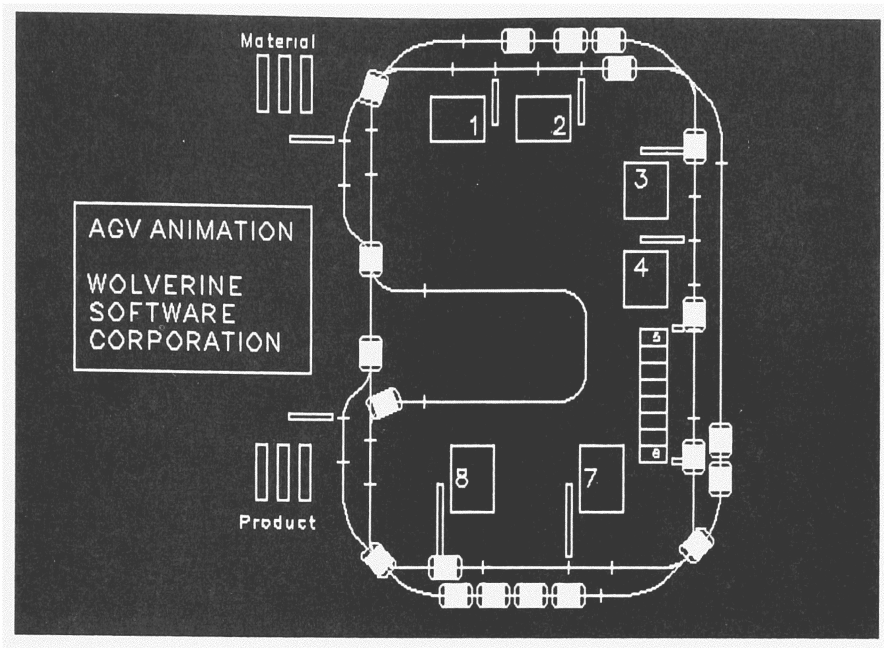

Figure 1. AGV Layout at Default Orientation and Scale

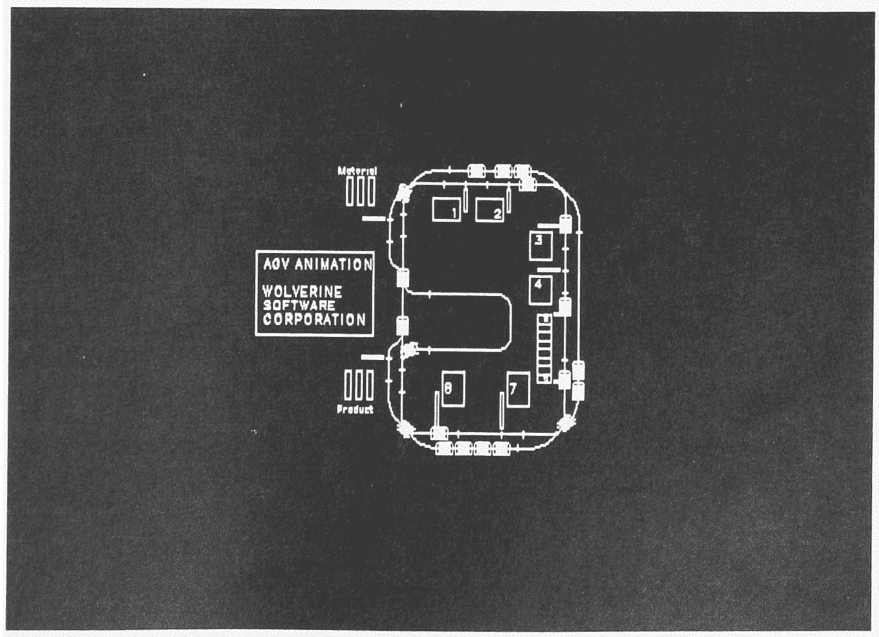

Figure 2. AGV Layout, Zoomed Out

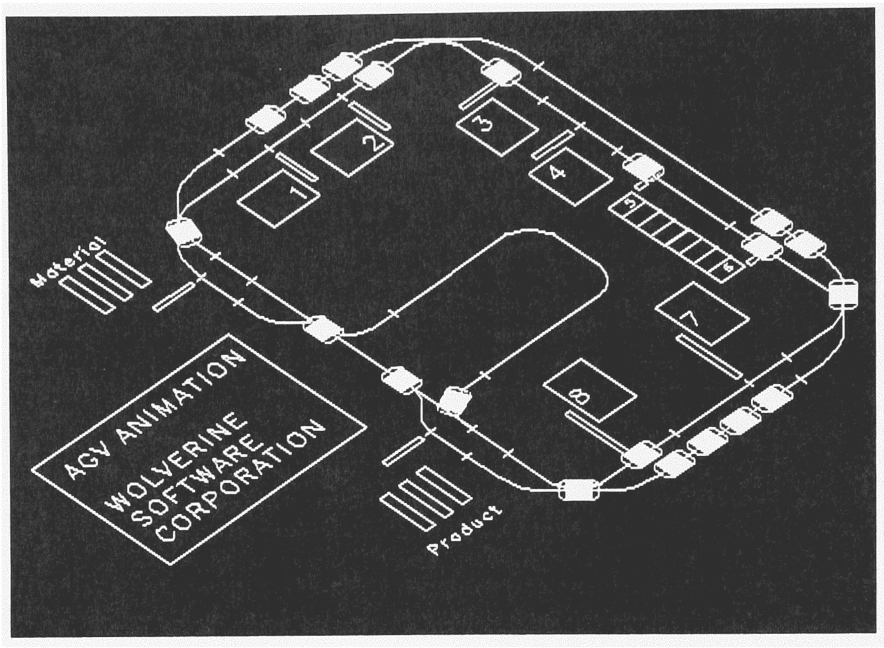

Figure 3. AGV Layout in Isometric Viewing Mode

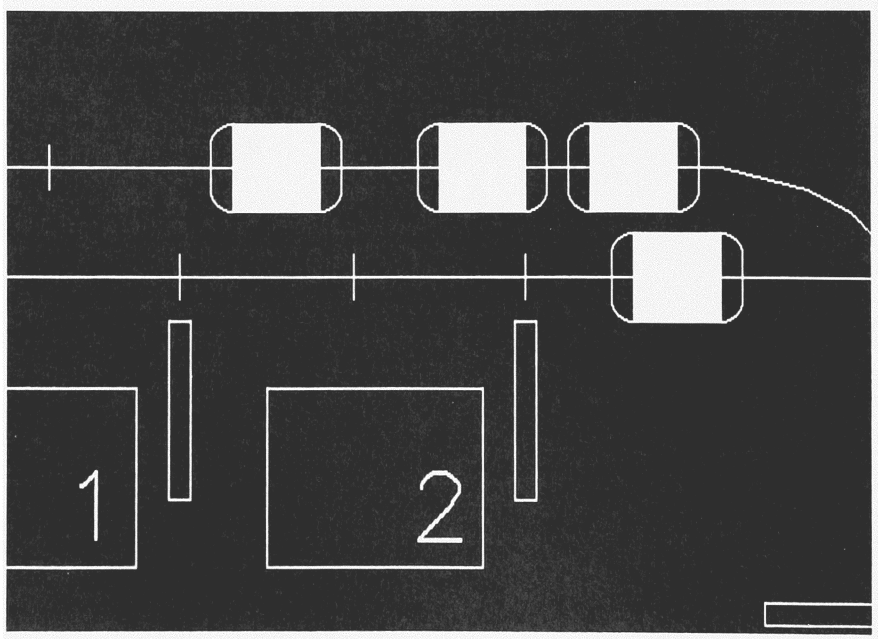

Figure 4. AGV Layout, Zoomed In and Panned

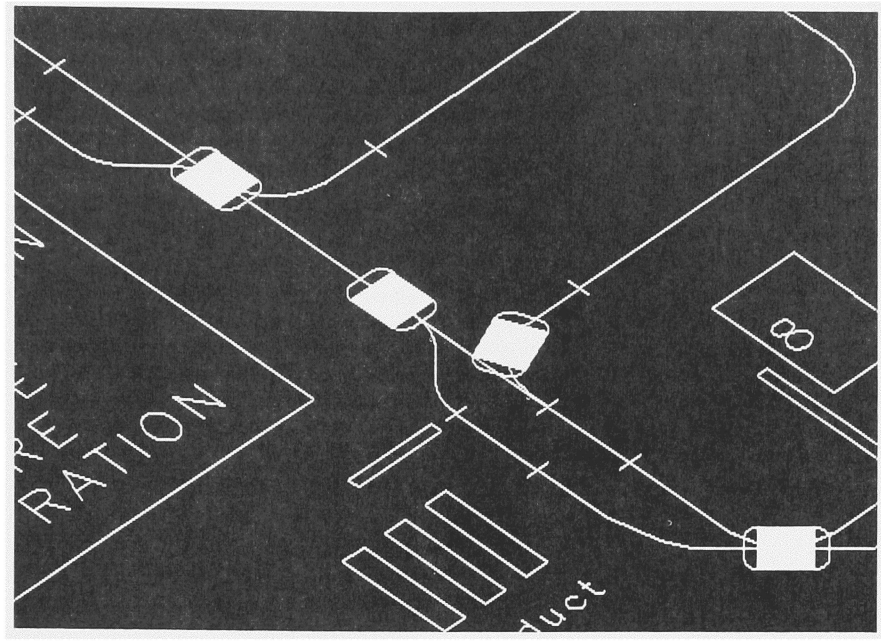

Figure 5. AGV Layout in Isometric Viewing Mode, Zoomed In and Panned

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, Product X reflects physical reality when two Objects would otherwise appear to collide and move through one another. (See Figure.) This often makes a simulation model of the system much simpler to construct. A surprising number of systems behave in this manner, from conveyors to human queues to toll plaza lanes.

3.3.4. Business Graphics. Under development at this writing, the business graphics features of Product X will include time-varying representations (i.e. "dancing bar graphs") as well as static displays.

In the meantime, it is simple enough to construct all manner of graphs, including animated ones, using the lower level drawing primitives.

3.4 Smooth Motion

Smooth motion was a primary design goal for Product X, and it has been achieved with stunning success.

In most media, it is necessary to create and re-create 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 on the fly at different locations, or saved and "blitted" to different locations on the screen. This process must be repeated quickly 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 has been plagued by slow frame rates. Compounding this problem is the fact that, due to the relatively few colors available and 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.

As of mid-1989, the capabilities of Product X for handling large animations have not been thoroughly tested. However, on small- to mid-sized animations, frame rates of 60 (EGA) or 70 (VGA) frames per second are routinely achieved on a 386 or fast 286 based machine. Furthermore, these rates are most likely to be affected by the total number of pixels in motion, and not by the number of objects being moved. Thus, full-floor views of material handling systems can be run at arbitrarily fast speeds.

What happens when Product X cannot keep up? With other animation software, the apparent speed of objects moving across the screen generally diminishes in such circumstances. With Product X, a constant (though user-adjustable) ratio of animated time to

"wall clock" time is always maintained, even when the model cycles between congested and not-so-busy. This is accomplished by reducing the frame rate and increasing the increment by which each object travels. Product X performs this adjustment "on the fly." With starting frame rates of 60 or 70 frames per second, the effect of reducing the frame rate remains visually acceptable.

3.5 Post-Processing

Product X offers "post-processed" animation. Some other software provides "concurrent" animation. What is the difference?

3.5.1. Post-Processed vs. Concurrent Simulation Animation. Concurrent animation has one major benefit. It is possible with most simulation/animation software that runs concurrently to make certain limited types of changes to the system (e.g. machine breakdowns) and "watch that queue build up." Many types of changes are difficult to animate without advance work by the modeler (e.g. added machines, especially in an unexpected place), and many of those are impossible to simulate without advance work as well (e.g. a sudden utterance of "say – didn't we decide to make that subassembly based on periodic inspection, instead of weekly?").

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

We have summarized this situation in the table below.

	Existing Concurrent Animators	Product X
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 large models on a faster machine and download the results for animation	None	Excellent

Note that the decision to detach the Product X engine from any simulation tool (see Section 3.6) does not permanently affect the 1989 decision to offer only post-processing. It may be possible someday for those desiring concurrent animation to use the detached Product X engine. All that is required is the specification of a simple run-time programmer's interface, and some means of allowing both processes to run at the same time. Even on one CPU running DOS, this could be accomplished through a stylized form of cooperative multitasking that the authors believe would not be difficult to implement. Market demand will dictate these future directions.

3.5.2. Animation Portability. Another important benefit of post-processing is that, coupled with the ubiquity of the "lowest common denominator" hardware platform and the hardware independence of the CAD-based data structure (any Product X animation will run on any Product X equipped machine), Product X animations are extremely portable. An analyst or a sales engineer can easily take an animation "on the road" or across the hall. The target machine does not need to be equipped to run any simulation software.

Finally, the post-processing approach permits the implementation of a demonstration mode. In this mode, the analyst can create complete "bullet-proof" presentations that can be viewed by others. The "viewer" in this case could be anyone with absolutely no simulation experience, from a top manager to a line operator. This mode is under development as of mid-1989.

3.5.3. Demonstration Mode. The potential for a demonstration mode in a general purpose animator is vast. Features could range from viewer control over speed and viewing orientation all the way to explanatory windows with arrows and highlighting that stay on the screen for a predefined length of time and call the viewer's attention to particular aspects of the simulation. There could even be dialog boxes that allow the viewer to choose various critical times or even to choose alternative experimental system configurations for closer inspection.

3.6 Open Architecture

One of the fundamental characteristics of Product X is its open architecture. What does this mean?

Some animation software is integrated into a simulation language or package. In order to use the animation, one must go through the process of building a simulation model using the integrated tool.

Other animation software is post-processed and file driven. Unfortunately, in the most notable of these cases, the specification for the animation trace files is not provided by the manufacturer. Users have decoded the file format and driven the animation from other software, but without an official sanction, this practice is not likely to be widespread.

Product X has an open architecture. It is the intent of its creators that Product X be used in a wide variety of contexts. The most dramatic impact of Product X's open architecture will initially be the quick adoption of Product X as the animation engine of choice for people using simulation software other than Wolverine Software's own GPSS/H.

It would also be possible to build graphical animated depictions of systems that have not been simulated, or to build a new simulation/animation package around the Product X graphics engine. Or, Product X could be used by non-engineers as presentation software, even competing with established PC software packages such as Show Partner FX. These capabilities open some pretty wide doors for Product X.

Although Product X is being released initially as a post-processed, standalone engine, there is little reason for that to be its only form. Wolverine Software, as of mid-1989, is considering opening up a real-time programming interface to Product X that would allow it to be used for concurrent simulation animation or for non-simulation real-time applications.

As of mid-1989, the open architecture of Product X 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. Both files are populated with printable ASCII characters that form commands and data for the animation engine. The layout file is used to hold static geometry information and initialization commands. The trace file is used for recording the time-dependent information that controls the animation activity. Despite this distinction, 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	ERASE
DESTROY	
TIME	

This is just a sampling of available commands. It is not the purpose of this paper to publish the specifications of the open architecture of Product X. For more information about obtaining the specifications, contact the authors.

Product X has its own built-in drawing capability. This makes it easier for Product X users to populate the CAD-like database with LINE, ARC, POLY, and similar commands simply by pointing and clicking a mouse. However, the existence and publication of the database specifications makes it possible for anyone to create program-generated layouts or to create a translator from any other source of CAD data. Wolverine Software will produce such translators as market demand dictates.

The animation capabilities of Product X can be exercised in similar ways. Normally, the animation primitives will be used over and over in populating the animation trace file. This process will virtually always be automated. For simulation, this means that the model will write commands such as SET COLOR into the file. For an example of using a language-based simulation model to populate the animation trace file with animation commands, see reference (Crain and Brunner 1989). For non-simulation applications such as presentation graphics, the process can be similarly automated.

3.7. Reasonable Cost

At this writing, we do not know what Product X will cost the end user. What we do know is that Wolverine Software Corporation is committed to driving the performance of PC animation software up while driving the price down. We expect that Product X will be much more affordable than existing products.

4. SUMMARY

Animation of discrete event simulation has been a powerful reason for the rapidly expanding use of simulation technology. The 1990's should see further expansion in the use of simulation and animation, thanks to continued technological innovation in both areas. The new "Product X" general purpose animation software from Wolverine represents an important step forward for animation technology. Important features of Product X include availability on standard hardware platforms, a CAD-like graphics data structure, flexible animation primitives, smooth animated motion, a post-processing orientation, an open architecture, and reasonable cost.

ACKNOWLEDGEMENTS

The authors are grateful for the thoughtful input of Nancy Earle, Roy Pafenberg, and Elizabeth Tucker of Wolverine Software, and for the efforts of our adventuresome software alpha testers.

REFERENCES

Crain, R. C., and Brunner, D.T. (1989). Extended Features of GPSS/H. In *Proceedings of the 1989 Winter Simulation Conference* (E. A. MacNair, K. Musselman, and P. Heidelberger, eds.).

McKay, K., and Rooks, M. (1989). WATMIMS JIT/Kanban Benchmark – Summary and Recommendations. In *Proceedings of the 1989 Winter Simulation Conference* (E. A. MacNair, K. Musselman, and P. Heidelberger, eds.).

AUTHORS' BIOGRAPHIES

DANIEL T. BRUNNER received a B.S. in Electrical Engineering from Purdue University in 1980, and an M.B.A. from The University of Michigan in 1986. He has been with Wolverine Software Corporation since 1986. Mr. Brunner is a member of SCS, and served as Publicity Chair for the 1988 Winter Simulation Conference.

JAMES O. HENRIKSEN is the president of Wolverine Software Corporation, which he founded in 1976 to develop and market GPSS/H, a state-of-the-art version of the GPSS language. Since its introduction in 1977, GPSS/H has gained wide acceptance in both industry and academia. From 1980-1985, Mr. Henriksen served as an Adjunct Professor in the Computer Science Department of the Virginia Polytechnic Institute and State University, where he taught courses in simulation and compiler construction at the university's Northern Virginia Graduate Center. Mr. Henriksen is a member of ACM, SIGSIM, SCS, the IEEE Computer Society, ORSA, and SME. 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 presently serves as the ACM representative on the Board of Directors of the Winter Simulation Conference.

Daniel T. Brunner
James O. Henriksen
Wolverine Software Corporation
4115 Annandale Road
Annandale, VA 22003-2500, U.S.A.
(703) 750-3910
email: wolverine_software@um.cc.umich.edu