Proceedings of the 2001 Winter Simulation Conference
B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, eds.

ENABLING SMOOTH AND SCALABLE DYNAMIC 3D VISUALIZATION OF DISCRETE-EVENT CONSTRUCTION SIMULATIONS

Vineet R. Kamat
Julio C. Martinez

Construction Engineering and Management Program
200 Patton Hall
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061-0105, U.S.A.

ABSTRACT

Visualizing simulated construction operations is an effective means of communicating the logic and the inner working of simulation models in a comprehensive and comprehendible manner. This can facilitate both model verification and validation and help establish credibility of simulation analyses. Due to the inherent working nature of discrete-event simulation systems, visualizing simulated operations in a smooth and continuous manner in 3D virtual worlds presents numerous interesting challenges. This paper describes research being conducted at Virginia Tech to enable smooth and scalable dynamic 3D visualization of discrete-event construction simulations.

1 INTRODUCTION

Discrete-event modeling is an inherently complex activity that is both a science and an art. The modeling of construction operations requires the description, in the language of the simulation modeling system, of mental plans that are often complex and elaborate. Differences between the mental plan and the operation actually modeled in a first attempt are ubiquitous. Verification is the process by which the model creator looks at what has been actually modeled, compares it to what was intended, and updates the model to accurately reflect the intention.

The developer of the computer simulation model, however, may have misconceptions about how the actual operation will take place in the field. Thus, a model may not be an accurate representation of reality despite proper verification by its developer. Such errors cannot be discovered by verification because the model indeed reflects what the model creator intended. The aim of Validation therefore is to determine whether simulation models accurately represent the real-world system under study. This is typically carried out by consulting people who are intimately familiar with the operations of the actual system, but who are not necessarily proficient in simulation. Simulation models are termed as Credible when the models, in their entirety, are accepted as being valid. The results are then used as an aid in making decisions (Law and Kelton 2000).

In the case of both verification and validation, the inner workings of a model and its output need to be communicated to others for discussion and input, and in a way that is both comprehensive and comprehendible (Oloufa and Ikeda 1997). Construction simulation tools typically provide results in the form of numerical or statistical data. However, they do not illustrate the modeled operations graphically in 3D.

This poses significant difficulty in communicating the logic and the inner working of simulation models, especially to persons who are not trained in simulation but are domain experts. Decision makers often do not have the means, the training and/or the time to verify and validate simulation models based solely on numerical output. Potential practitioners are therefore always skeptic about simulation analyses and have little confidence in their results (Ioannou and Martinez 1996).

2 RESEARCH INITIATIVE

The design and analysis of construction operations using simulation is feasible only if the insights gleaned are used in making decisions and increasing understanding (i.e., the models are credible). The logic and the inner working of simulation models can only be verified and validated by communicating the simulated operations in a way that is both comprehensive and comprehendible.

Visualizing simulated operations can be an effective means of achieving this (Law and Kelton 2000, Rohrer 2000, Jain 1999, Henriksen 1998, Tucker et al. 1998, Robinson 1997, Cox 1988, Biles and Wilson 1987). It is a generally accepted fact that visually presented information is understood and grasped more easily than any other form of communication. The need to visualize simulated operations

is more relevant in the context of construction because construction operations analysts (e.g., superintendents) typically do not have the necessary training in simulation to allow them to validate simulation results based on numerical values in tables and charts.

Visualization of simulated construction operations involves being able to "see" graphically on the computer, the operations being carried out in the same way as they would be in the real world. In the case of a simulated world, this includes the same logical and physical relationships that are embedded in the underlying discrete-event simulation models.

3 THE NATURE OF DISCRETE EVENT SIMULATION

In continuous simulation, the state of a model is continuously tracked at every instant in time using differential equations of motion (Law and Kelton 2000). In discrete-event simulation, in contrast, the state of models changes only at discrete, but possibly random, sets of simulated time points (Schriber and Brunner 1999). These time points are typically the start or end of activities, and it is only then that a discrete-event simulation can communicate with other processes, or perform other actions such as input/output.

In order to visualize a simulated operation it is necessary to see, in addition to the physical components of a facility, the equipment, personnel, materials and temporary structures required to build it. Moreover, it is necessary to depict the movements, transformations and interactions between the various simulation entities (e.g. resources). The movements and transformations must be spatially and temporally accurate.

In order to depict smooth motion, visual elements must be shown at the right position and orientation several times per second. Issues such as trajectories in 3D space, speed and acceleration need to be considered. Due to the nature of discrete-event simulation and the dynamic and complex nature of construction, this was a challenging proposition. The basic question, then, was how to achieve realistic and dynamic continuous motion, based on information that is available only when discrete events occur in a simulation.

4 DESCRIBING DYNAMIC 3D VISUALIZATION THROUGH PARAMETRIC TEXT STATEMENTS

The primary research activity in the presented work was to determine the amount and the nature of information that would be needed to accurately and unambiguously describe, both spatially and temporally, a complex construction operation so that it could be recreated in a virtual world. This includes the movements and transformations of the people, machines, materials and the evolving constructed facility.

Secondly, it had to be determined whether it was possible to obtain that information from discrete-event simulation models. Without it, simulation models would not be able to communicate the required information and author visualizations.

The specific nature of the communicated information cannot be determined a-priori by a general-purpose simulation system, as it is dependent on the specific operation that is being modeled. Thus, this communication has to be achieved by end-user programming of the simulation system. The communication therefore should be based on a language that is both expressive (to achieve realistic visualization) and simple (so that it can be generated by end-user programming).

During a simulation run, the simulation clock tracks the passage of simulated time (as distinct from wall-clock time). The clock advances in discrete steps (typically of unequal size) during the run (Schriber and Brunner 1999). After all possible actions have been taken at a given simulated time, the clock is advanced to the time of the next earliest event. Then the appropriate actions are carried out at this new simulated time, etc. The execution of a run therefore takes the form of a two-phase loop: "carry out all possible actions at the current simulated time," followed by "advance the simulated clock". These two phases are repeated over and over again until a simulation run-ending condition is encountered. Due to this nature, two things occurring at the same simulation time are, in effect, processed serially by discrete-event simulation systems.

This phenomenon suggested that designing a straight-line (Appel 1997) dynamic 3D visualization language would be the most appropriate form of the above-described communication. Such a language needed to be simple enough to be generated by end-user programmable software such as a discrete-event simulation system. At the same time, the language should be semantically rich in order to be able to realistically visualize complex construction operations based on the limited amount of discrete information that can be communicated by a running simulation model. By necessity, programs in such a language would be arbitrarily long.

4.1 Designing the Dynamic 3D Visualization Language

Table 1 lists the key animation commands designed and implemented in the language and provides a concise explanation of their functionality. Each parametric animation statement can span multiple lines with arguments separated by white space. Arguments that include white space must be enclosed in single quotations. A statement ends with a semicolon. Comments can be placed in trace files by making the first non white space character after a statement a "/". The comment continues until the end of the line. The designed language commands can be broadly classified into four categories, i.e. System, Scene-Building, Property-Setting, and Motion-Depicting Commands.

TIME is the primary system command. The TIME statement keeps track of the simulation time during visualizations by indicating the instant at which all subsequent commands until the next TIME statement take place. Every discrete event that is recorded by a simulation model in a trace file will have a preceding TIME statement to indicate the simulation time at which the event took place.

Table 1: Key Animation Language Commands and their Functionality

Statement	Functionality
TIME	Indicates the simulation time at which all subsequent commands take place.
CLASS	Associates a class of simulation entities with their geometric description contained in a CAD file.
CREATE	Creates specific simulation objects by instantiating predefined classes.
PLACE	Places simulation objects at particular locations or at the beginning of resource movement paths.
SET CLASS… FORECLEARANCE	Specifies the minimum distance to be maintained between two following objects of the same class.
MOVE	Simulation objects begin moving on resource movement paths at the time specified by a preceding TIME statement.
ROTATE	Simulation objects begin rotating along specified planes at the time specified by a preceding TIME statement.

The Scene-Building commands set up the visualization environment and manage the initial and dynamic creation and destruction of simulation entities. For instance, CLASS commands identify the CAD files that contain the geometric representation of simulation entities. CREATE commands subsequently create specific simulation objects at various times in the simulation by instantiating predefined classes.

The Property-Setting commands allow the specification and manipulation of certain physical properties for simulation entities at both a class and object level. For instance, the SET CLASS FORECLEARANCE command enables the authoring simulation model to specify the minimum distance to be maintained between two following objects of the same class during visualization (e.g. trucks traveling on the same path). The SET OBJECT FORECLEARANCE similarly permits this property to be specified on a per-object basis if necessary.

The Motion-Depicting commands form the core components of the designed animation language. These commands depict the dynamic state of the simulation models. For instance, the MOVE command moves instantiated simulation objects on movement paths to describe the motion of resources and other simulation entities on the construction site. ROTATE is another primary Motion-Depicting command that allows the depiction of rotation of simulation entities during visualization.

5 THE DYNAMIC CONSTRUCTION VISUALIZER

The Dynamic Construction Visualizer (DCV) is a general-purpose 3D visualization/animation system that implements the above-described dynamic 3D visualization language. The DCV allows simulation model developers to visualize modeled operations with chronological and spatial accuracy in 3D virtual space. The system is independent of any particular simulation-modeling program or CAD modeling software.

Files written in the designed 3D visualization language (hereinafter referred to as the DCV language) unambiguously describe the physical configuration of modeled operations with the passage of time. The DCV language allows the construction and manipulation of complex 3D scenes. Simulated operations are visualized by processing sequential, time-ordered animation commands written in the DCV language. The animation commands are contained in an ASCII text file hereinafter referred to as the trace file.

DCV trace files are meant to be generated by simulation software. Any simulation software capable of writing custom text output during a simulation run can generate the trace files automatically. These include most of the programmable generic and special-purpose simulation languages as well as high-level programming languages such as BASIC, FORTRAN, C and C++. Non-language based simulation software may also be adapted to generate trace files during a simulation run (Henriksen 1998).

The DCV uses 3D models of all pertinent resources and system entities to depict the simulated operations and the evolving product in 3D. The DCV system does not possess any built-in 3D model building capability. Instead, required 3D models of system entities can be imported from a wide variety of 3D CAD modeling software. The DCV provides direct support for the VRML file format. Geometry files from practically every 3D modeling program (e.g. AutoCAD™, MicroStation™, 3D Studio™) can be easily exported or converted into VRML format.

6 ANIMATING SIMULATED OPERATIONS

Understanding the working of the DCV animation clock is fundamental to understanding the animation capabilities of the DCV. The DCV measures time in floating point ani-

mated time units. One time unit can equal whatever duration is most suitable for the animation (e.g. a microsecond, a minute, or a day) as long as it matches the time unit in the simulation model that is driving the animation.

DCV animations can run at any desired animation speed. The animation speed, also known as the viewing ratio, represents the number of animated time units per second of viewing time. For instance, if the simulation model (and the animation) uses seconds as a unit of time, and the viewing ratio is 6, then the DCV animation is running at a rate of six animated seconds per viewing second. Consequently, a modeled activity requiring one minute for completion in reality would be accomplished in 10 (i.e. 60/6) seconds in the animation. In the DCV application, the user can change the viewing ratio of an animation at any time depending on the animation speed desired.

The primary time-tracking DCV command is TIME. The syntax of the TIME command is:

```
TIME timevalue;
```

The TIME command waits for the animation clock to reach the new value specified. The DCV then executes the commands that follow it until another TIME command is reached. When a TIME statement is encountered in a trace file, the DCV initially verifies that the timevalue is greater than or equal to the current animated time. If not, the animation terminates with an error. After ascertaining that the TIME command specifies a future time, the DCV suspends the reading of any more lines from the trace file until the animation time specified by the TIME command has been reached or exceeded. When that happens, the DCV reads and processes the next line(s) in the trace file until another TIME statement is encountered. Statements are read and processed in this manner until the end of the trace file is reached or the viewer interrupts the animation. The reading and processing of the trace file statements is practically instantaneous. All the while, the DCV continues to display the animation as it progresses at a constant, user-specified viewing ratio.

Figure 1 presents a sample DCV trace file. The implications of visualizing this trace file in the DCV are easily interpretable. Immediately before the onset of the animation, two paths, "LoadToDump" and "DumpToLoad", and three classes, "Terrain", "Excavator", and "Truck" are defined and their representations are stored into memory. Further reading of statements from the trace file is suspended until the animation time equals 6. Thus, a person viewing the animation sees a motionless excavator in the terrain for six animation time units. At this point, a truck, "Truck1", is created and placed in the scene at the beginning of path "LoadToDump". Six animation time units later (at time 12), "Truck1" starts moving along the path "LoadToDump" at a speed that will require 120 animation time units to reach the end of the path. At the same time (12), another truck, "Truck2", is created and placed in the scene at the beginning of path "DumpToLoad". At animation time 18, "Truck2" starts moving along the "DumpToLoad" and will require 90 time units to reach its destination. At the moment "Truck2" starts moving, the already in motion "Truck1" will have completed about 1/20th of its journey.

Figure 1 also displays graphically the processing of commands in the presented trace file. The real-timeline displayed below the animated-timeline assumes a viewing ratio of 6.

7 CONCLUSION

The presented research involved the design of a dynamic visualization language that enables the realistic, continuous 3D depiction of construction operations simulated using discrete-event simulation tools. The implementation of the language (The DCV) enables running simulation models to record discrete sequential events in trace files using parametric text statements conforming to the DCV language. The statements contained in the files are then processed sequentially and the simulated operations are depicted in a virtual world using 3D CAD models of all the involved entities.

The presented DCV language is capable of depicting several simulated construction operations dynamically in 3D. Simulation modelers and decision makers can, as a result, effectively visualize many simulated construction operations and better understand what was simulated. The current DCV language and its implementation are capable of describing operations that involve objects moving at constant speeds, that do not require any sort of run-time CAD model deformation, that do not involve unstructured or flowable entities (e.g. concrete), and in which resources (equipment and crew) always move on well defined paths.

Current and future research is/will be focused on enriching the DCV language lexicon and augmenting the implementation to enable the realistic visualization of the entire gamut of complex construction operations and the resulting products.

ACKNOWLEDGMENTS

The research presented here has been supported by the National Science Foundation (Grant CMS-9733267). Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

Appel, A.W. 1997. *Modern Compiler Implementation in C: Basic Techniques.* New York, NY: Cambridge University Press.

Biles, W.E., and S.T. Wilson. 1987. Animated Graphics and Computer Simulation. *Proceedings of the 1987 Winter Simulation Conference,* ed. H. Grant, W. D.

```
PATH LoadToDump (3,2,1) (0,1,5) (-5,0,3);
PATH DumpToLoad (-4,0,3) (1,1,5) (2,2,1);
CLASS Terrain Terrain.wrl;
CLASS Excavator EX1100.wrl;
CLASS Truck A30C.wrl;

TIME 0;
CREATE ExTerrain Terrain;
PLACE ExTerrain AT (0,0,0);
CREATE Excvtr1 Excavator;
PLACE Excvtr1 AT (5,2,1);

TIME 6;
CREATE Truck1 Truck;
PLACE Truck1 ON LoadToDump;

TIME 12;
MOVE Truck1 LoadToDump 120;
CREATE Truck2 Truck;
PLACE Truck2 ON DumpToLoad;

TIME 18;
MOVE Truck2 DumpToLoad 90;
```

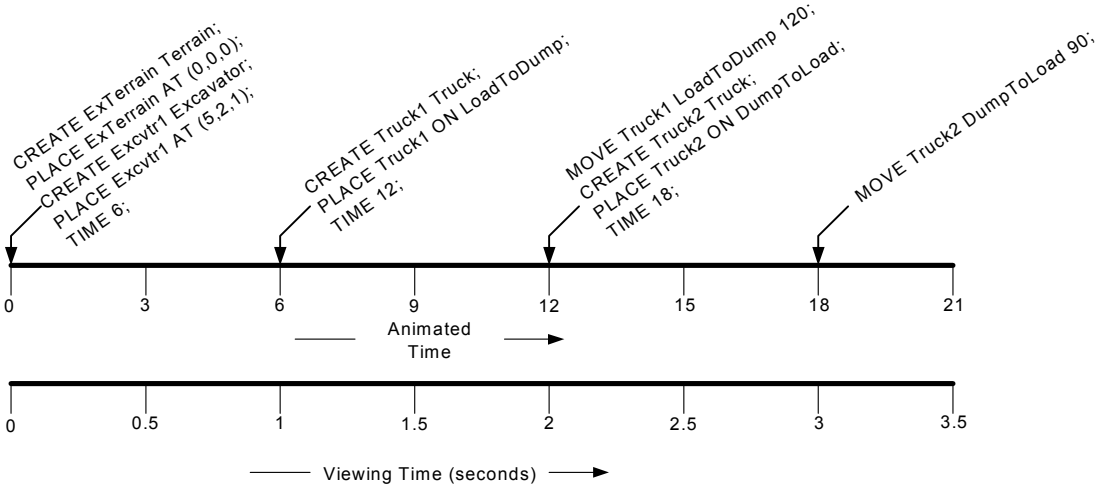

Figure 1: Processing of Commands in a Trace File

Kelton, and A. Thesen, 472-477. San Diego, CA: Society for Computer Simulation.

Cox, S.W. 1988. GPSS/PC™ Graphics and Animation. *Proceedings of the 1988 Winter Simulation Conference*, ed. P. L. Haigh, J. C. Comfort, and M. A. Abrams, 129-135. San Diego, CA: Society for Computer Simulation.

Henriksen, J.O. 1998. Windows-Based Animation with PROOF™. *Proceedings of the 1998 Winter Simulation Conference*, ed. J. S. Carson, M. S. Manivannan, D. J. Medeiros, and E. F. Watson, 241-247. San Diego, CA: Society for Computer Simulation.

Ioannou, P.G., and J. Martinez. 1996. Animation of Complex Construction Simulation Models. *Proceedings of the 3rd Congress on Computing in Civil Engineering*, 620-626. Reston, VA: American Society of Civil Engineers.

Jain, S. 1999. Simulation in the Next Millennium, *Proceedings of the 1999 Winter Simulation Conference*, ed. D. T. Sturrock, G. W. Evans, P. A. Farrington, and H. B. Nemhard, 1478-1484. San Diego, CA: Society for Computer Simulation.

Law, A.M., and W.D. Kelton. 2000. *Simulation Modeling and Analysis*, 3rd Ed. New York, NY: McGraw-Hill.

Oloufa, A.A., and M. Ikeda. 1997. Library-Based Simulation Modeling in Construction. *Proceedings of the 4th Congress on Computing in Civil Engineering*, 198-205. Reston, VA: American Society of Civil Engineers.

Robinson, S. 1997. Simulation Model Verification and Validation: Increasing the User's Confidence. *Proceedings of the 1997 Winter Simulation Conference*, ed. D. H. Withers, B. L. Nelson, S. Andradottir, and K. J. Healy, 53-59. San Diego, CA: Society for Computer Simulation.

Rohrer, M.W. 2000. Seeing is Believing: The Importance of Visualization in Manufacturing Simulation. *Proceedings of the 2000 Winter Simulation Conference*, ed. P. A. Fishwick, K. Kang, J. A. Joines, and R. R. Barton, 1211-1216. San Diego, CA: Society for Computer Simulation.

Schriber, T.J., and D.T. Brunner. 1999. Inside Discrete-Event Simulation Software: How it Works and Why it Matters. *Proceedings of the 1999 Winter Simulation Conference*, ed. D. T. Sturrock, G. W. Evans, P. A. Farrington, and H. B. Nemhard, 72-80. San Diego, CA: Society for Computer Simulation.

Tucker, S.N., P.J. Lawrence, and M. Rahilly. 1998. Discrete-event Simulation in Analysis of Construction Processes. *CIDAC Simulation Paper*, 1-14. Melbourne, Australia.

AUTHOR BIOGRAPHIES

VINEET R. KAMAT is a PhD student and Research Assistant in the Via Department of Civil Engineering at Virginia Tech. He received his M.S. in Civil Engineering at Virginia Tech in 2000; and a B.E. in Civil Engineering at Goa University, India in 1998. He designed and implemented the Dynamic Construction Visualizer with J. Martinez as part of his graduate research. His research interests include discrete-event simulation and visualization of construction operations. His email and web addresses are <vkamat@vt.edu> and <http://filebox.vt.edu/users/vkamat>.

JULIO C. MARTINEZ is an Assistant Professor in the Via Department of Civil Engineering at Virginia Tech. He received his Ph.D. in Civil Engineering at the University of Michigan in 1996; an MSE in Construction Engineering and Management from the University of Michigan in 1993; an M.S. in Civil Engineering from the University of Nebraska in 1987; and a Civil Engineer's degree from Universidad Catolica Madre y Maestra (Santiago, Dominican Republic) in 1986. He designed and implemented the STROBOSCOPE simulation language with P. Ioannou and is currently V. Kamat's research advisor. In addition to discrete event simulation, his research interests include construction process modeling and decision support systems for construction. His email and web addresses are <julio@vt.edu> and <http://strobos.ce.vt.edu>.