Proceedings of the 2007 Winter Simulation Conference
S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, eds.

A MESSAGE-BASED ARCHITECHTURE TO ENABLE RUNTIME USER INTERACTION ON CONCURRENT SIMULATION-ANIMATIONS OF CONSTRUCTION OPERATIONS

Prasant V. Rekapalli
Julio C. Martinez

School of Civil Engineering
550 Stadium Mall Drive
Purdue University
West Lafayette, IN 47907, U.S.A.

ABSTRACT

This paper describes a preliminary architecture to support user interaction with 3D animations. These interactions are able affect the state of the concurrent discrete-event simulation driving the animation. We first explain how user interaction is conceptually split into user-action and user-intent, and then detail the message-based architecture adopted to support user interaction.

1 INTRODUCTION

Discrete-event simulation (DES) is a powerful tool to model and analyze complex construction operations (Martinez & Ioannou 1999). The aim of any simulation study is to achieve model credibility. Results from credible models stand a better chance of being considered by decision-makers (Law & Kelton 2000). Verification and validation of simulation models is critical to achieving model credibility. Hence, there is a need to develop powerful techniques to verify and, especially, validate DES models.

Visualization is widely recognized as a powerful technique to support the validation of simulation models. Recent advances have made it possible to obtain post-processed animations of simulated construction operations. *Post-processed animation* is an animation that is obtained after the entire simulation model has been processed (Bishop & Balci 1990, Rohrer 2000). Example systems (suitable for construction) for post-processed animation include (2D) PROOF (Henriksen 1998) and (3D) Vitascope (Kamat 2003).

Despite these advances, the process of validation is still quite time consuming and inefficient. This is primarily because post-processed animations show what took place in the particular simulation run that produced it. A single simulation run, however, cannot capture all possible combination of events that can actually take place in the simulation model. Hence, more often than not, it becomes necessary to study animations obtained from several different simulation runs, and in some cases to study a single animation multiple times in order to adequately cover spatially distant areas of the animation. In addition, while immersed in a 3D animation, questions about the behavior of the model may come to mind to the person performing the validation. These questions can often be answered if the state of the model can be instantly changed, and the subsequent behavior observed.

We are currently working to enable visual interactive simulations for construction operations to enhance model validation.

2 VISUAL INTERACTIVE SIMULATION

Interactive simulation-animation or *visual interactive simulation (VIS)* is visual simulation that includes the capability for user interaction on the running model (Hurrion 1989). Simulation models with any form of visualization are termed *visual simulations (VS)* (Bishop & Balci 1990). The interaction can be *model prompted* (i.e. the model initiates the interaction) or *user prompted* (i.e. the user initiates the interaction) (Bishop & Balci 1990).

VIS can significantly improve the process of validation. Rohrer (2000) states "*By changing input and viewing the reaction of the system in an animation, skeptics can get a feel for the level of accuracy and usability of the model.*" (p. 1214). Furthermore, Bishop & Balci (1990) state "*User initiated interaction allows the user to change model parameters and continue execution of the model. This ability to "play" with the model can be crucial for understanding the system.*" (p. 504).

3 VIS FOR CONSTRUCTION: CHALLENGES

Enabling VIS for construction posses some unique challenges. Advances in post-processed animation technology have been, by design, independent of simulation tools. This design philosophy allows users to mix and match tools as suitable to their circumstances, and to take advantage of prior investments in mastering specific tools. In continuing with this philosophy, the current work is designed to be independent of simulation or animation tool. This adds to the challenge, but makes the technology useful to broader audiences.

Post-processed animation, by its very nature, cannot support runtime user interaction. This requires enabling concurrent simulation-animation. *Concurrent simulation-animation* or *concurrent animation* is an animation that can be viewed as the simulation model is being processed (i.e., simulation and animation are synchronized to process simultaneously) (Bishop & Balci 1990, Rohrer 2000). Concurrent animation is the backbone for an interactive simulation–animation system. This challenge has been addressed successfully with the design of a time-advance algorithm that synchronizes the discrete simulation clock and continuous animation clock (Rekapalli & Martinez 2007) in a manner that maintains a constant ratio of simulation time to observation time.

Without concurrent simulation-animation it is not possible to affect the course of a running simulation by interacting with the animation. In addition to concurrency, this requires an architecture that can support the communication of a user's actions from the animation to the simulation.

The architecture described below is designed for use with simulation and animation systems that are user extensible (i.e., independent developers can extend the functionality of the systems). The authors have implemented this architecture using the Stroboscope simulation system (Martinez 1996) and the Vitascope++ animation system (which is an extension of the Vitascope animation system).

4 COMPONENTS OF USER INTERACTION

The rationale for our message-based architecture is based on the concept that user interaction can be viewed as two separate components: (a) user-action, and (b) user-intent.

4.1 User-Action

User-action refers to the actions performed by the user on the concurrent animation using the available input devices. For example, double-clicking an animation object may indicate the user-action "animation-object-selected".

A user-action can be achieved by various graphical user interface (GUI) actions or by combinations of them. For example, the "animation-object-selected" user-action can be generated by performing a single or double mouse-click using the left, middle, or right mouse-key, with or without holding down a specific key on the keyboard. The action can also be generated by user interaction with hardware other than a mouse, such as a joystick or wand.

Raw GUI actions are captured by what we call *animation-event-handlers*. Independent developers can design new animation-event-handlers and register them with the animation system to support different types of user-actions. The model developer can choose which animation-event-handlers to use for a particular model.

4.2 User-Intent

User-intent refers to the intention of the user when performing a particular action. For example, performing the user-action "animation-object-selected" may result in the breakdown of the equipment that is represented by the selected animation object. This may be achieved in the simulation by preempting the activity that is currently engaging the equipment resource.

User-intent can also be comprised of several model-specific actions that result in changes to the state of the simulation.. The actions that can be performed are determined by functionality that the simulation system provides to allow model developers to affect the state of a running simulation.

5 MESSAGE-BASED ARCHITECTURE

Since the concurrent animation acts as the front-end to the simulation model, it is responsible for capturing user-actions. Affecting the state of the simulation to reflect the user-intent, however, is the responsibility of the simulation system. Hence, supporting user interaction requires (a) a mechanism to determine user-intent when a user-action is performed, and (b) a protocol for communication from the animation to the simulation that allows the interpretation and processing of user-intent.

Message-based architectures are used very effectively in the graphical user interfaces of operating systems (e.g., Microsoft Windows). This architecture was chosen because it provides the best structure for extensibility by independent developers. This is important because it does not burden the authors with designing a comprehensive set of mechanisms that allow users to interact with running simulations in a variety of ways to produce a variety of end results. This architecture also allows for simulation and animation systems to remain independent of each other.

The central piece to this architecture is the *message-interface*, which has four main components: (a) event-message, (b) message-data, (c) data-packager, and (d) data-unpackager.

5.1 Event-Message

The *event-message* determines what is communicated from animation to simulation for it to react accordingly. After exploring different protocols governing the content of an event-message, we decided that event-messages should represent user-actions. This charges the simulation to both interpret and process user-intent.

This protocol provides the most flexible and extensible mechanism to support user interaction because (a) as demonstrated earlier, a user-action can be generated in a wide variety of ways, and (b) a wide variety of user-intents can be associated with the same user-action. For example, the different user-intents associated with the "animation-object-selected" user-action might include (but not limited to): (a) breakdown the associated equipment, (b) change the duration of the associated activity, or (c) change the properties of the associated resource represented by the animation object.

5.2 Message-Data

An event-message, when passed from the animation to the simulation, is accompanied by *message-data* that contains the details of the user-action performed. For the "animation-object-selected" user-action, the accompanying message-data might include the name of the animation object that was selected, and other attributes that can be used to identify the uniqueness of user's raw GUI actions that generated the event-message.

5.3 Data-Packager

The *data-packager* is responsible for packing the individual data components together so that they can be passed along with the event-message. This data-packager is to be provided as an independent module that can be used by any animation-event-handler.

5.4 Data-Unpackager

The *data-unpackager* unpackages the message-data such that its individual components are available to the model developer from within the simulation environment. As described earlier, the power of this architecture is the ability to associate different user-intents to a particular user-action. With the data-unpackager, the model developer can, based on the details of the user-action (contained in the message-data), specify a different user-intent.

6 PROCESSING USER-INTENT

This section describes the functionality added to Stroboscope for model developers to define user-intent. The REGEVENT and ONEVENT statements were added.

"REGEVENT" allows the model developer to associate a user-intent with a particular event-message using the following syntax:

```
REGEVENT <name> <message> <event_expression>
```

Where:

- <name>: is the user-defined name of the event.
- <message>: is the event-message associated with the event.
- <event_expression>: an expression that evaluates to a number.

"ONEVENT" allows the model developer to associate the actions that need to be performed on the state of the simulation to reflect the user-intent using the following syntax:

```
ONEVENT <name> STATEMENT <statement>;
REGEVENT <name> EXPRESSION <expression>;
```

Where:

- <name>: is the name of the event that has been previously registered using "REGEVENT".
- STATEMENT: is a keyword that indicates that following action is a Stroboscope statement that has to be executed.
- <statement>: the actual statement that has to be executed.
- EXPRESSION: a keyword that indicates that following action is an expression that has to be evaluated by Stroboscope.
- <expression>: the actual expression that has to be evaluated.

When Stroboscope receives an event-message, it retrieves all the events that were registered with that event-message (i.e., using "REGEVENT"). For each of these events, the associated <event_expression> is evaluated, and if found to be non-zero - the statements/expressions registered with that event (i.e., using "ONEVENT") are processed. If the <event_expression> of more than one event evaluate to non-zero, the user is provided with a GUI interface to choose which event to process.

7 CONCLUSIONS AND FUTURE WORK

We described how user interactions with concurrent simulation-animations of construction operations can be conceptually split into user-action and user-intent. This conceptual view allows for a clear delineation of responsibilities between the simulation and animation systems.

We also described how a message-based architecture can be used to tie together independently developed DES and 3D animation tools. This architecture also lends to independent development of pluggable interfaces that can be used by model developers to enable a variety of user interactions that have a variety of end results.

Future work is required to enable data integration between simulation and animation systems. This is needed to identify the counterpart simulation entities to animation entities, which is crucial for having the concurrent animation act as the front-end to the simulation model.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the support of the National Science Foundation (NSF) (Award # CMMI-0732560). 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 NSF.

REFERENCES

Bishop, J. L., and Balci, O. (1990), “General purpose visual simulation system: A functional description”, *1990 Winter Simulation Conference*, IEEE, Piscataway, NJ, 504-512.

Henriksen, J. O. (1998), “Windows-based animation with PROOF^{TM}”, *1998 Winter Simulation Conference*, IEEE, Piscataway, NJ, 241-247.

Hurrion, R. D. (1989), “Graphics and interaction”, In *Computer Modeling for Discrete Simulation*, M. Pidd, Ed., John Wiley & Sons, New York, NY, 101-119.

Kamat, V. R. (2003), “VITASCOPE: Extensible and scalable 3D visualization of simulated construction operations”, *PhD Dissertation*, Virginia Polytechnic Institute and State University, Blacksburg, VA.

Law, A. M., and Kelton, W. D. (2000), “Simulation Modeling and Analysis”, 3rd Edition, McGraw-Hill, 2000.

Martinez, J. C. (1996), “STROBOSCOPE: State and resource based simulation of construction operations”, *PhD Dissertation*, University of Michigan, Ann Arbor, MI.

Martinez, J. C., and Ioannou, P. G. (1999), “General-purpose systems for effective construction simulation”, *Journal of Construction Engineering and Management*, Vol. 125, No. 4, ASCE, Reston, VA, 265-276.

Rekapalli, P. V., and Martinez, J. C. (2007), “Time advance synchronization in concurrent discrete-event simulation and animation of construction operations”, *Eleventh International Conference on Civil, Structural, and Environmental Engineering Computing*, Civil-Comp Ltd., Stirling, U.K. (Accepted).

Rohrer, M. W. (2000), “Seeing is believing: The importance of visualization in manufacturing simulation”, *2000 Winter Simulation Conference*, IEEE, Piscataway, NJ, 1211-1216.

AUTHOR BIOGRAPHIES

PRASANT REKAPALLI is a doctoral candidate in the School of Civil Engineering at Purdue University. He received his MS in Civil Engineering (Construction Engineering and Management) at Virginia Tech in 2004; and a Bachelor of Technology (B.Tech) degree in Civil Engineering (minored in Engineering Project Management) at the Indian Institute of Technology – Madras (Chennai, India) in 2002. His research interests include simulation and visualization of construction operations, and virtual reality. He is currently involved is developing a discrete-event simulation based virtual reality environment for construction operations for his doctoral dissertation. His email and web addresses are `<prekapal@purdue.edu>` and `<http://web.ics.purdue.edu/~prekapal/>`.

JULIO MARTINEZ is an associate professor in the School of Civil Engineering at Purdue University. He received his PhD 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 MS in Civil Engineering at 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 as part of his doctoral dissertation research. In addition to discrete-event simulation, his research interests include construction process modeling, decision support systems for construction, scheduling of complex and risky projects, and construction management information systems. His email and web addresses are `<julio@purdue.edu>` and `<http://www.ezstrobe.com/>`.