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.

ENABLING SMOOTH AND SCALABLE DYNAMIC 3D VISUALIZATION OF DISCRETE-EVENT CONSTRUCTION SIMULATIONS IN OUTDOOR AUGMENTED REALITY

Amir H. Behzadan
Vineet R. Kamat

Department of Civil and Environmental Engineering
University of Michigan
Ann Arbor, M.I. 48109, U.S.A.

ABSTRACT

Visualization is a powerful method for verifying, validating, and communicating the results of a simulated model. Lack of visual understanding about a simulated model is one of the major reasons inhibiting contractors and engineers from using results obtained from discrete-event simulation to plan and design their construction processes and commit real resources on the job site. The fast emerging information technology makes the use of modern visualization applications more appealing to engineers and scientists in different domains. This paper presents the design and implementation of an Augmented Reality (AR) visualization application together with an authoring language that allows the creation of outdoor AR animated scenes of simulated operations while featuring complete user involvement and interaction. The application is based on the concept of scene graphs. It also uses a unique general purpose data transmission method to communicate with hardware components of the system.

1 INTRODUCTION

Discrete-event simulation (DES) has been a commonly used method to model operations in different domains including construction. The fact that DES tools often provide modelers with a large amount of numerical and statistical data without any graphical representation often leads to little confidence in the results of the simulated model (Kamat and Martinez 2002). This usually makes the decision makers and engineers manually verify the results using traditional methods of evaluating the productivity and outcome of operations. The time spent on manually verifying the model, analyzing the numerical data, and fixing any possible errors can be effectively saved by using modern visualization tools to verify and validate the results of the simulated model.

The idea of using Augmented Reality (AR) visualization to create smooth animations of simulated operations is relatively new in the field of construction. Most traditional construction visualization tools use the concept of Virtual Reality (VR) in which the entire animated scene is computer generated and there is no interaction of any real object involved in the animation (Kamat 2003). In contrast, scenes from the real world are used as the background for an AR-based animation while computer generated CAD objects are superimposed on top to create an augmented (i.e. mixed) view of virtual and real objects (Behzadan and Kamat 2006).

A number of researchers have recently used the concept of AR in their work. Webster et al. (1996) presented a system that shows locations of columns behind finished walls, and steel rebars inside columns. Roberts et al. (2002) used AR to overlay locations of subsurface electrical, telephone, gas, and water lines onto real world views. Both applications demonstrated AR's potential in helping maintenance workers avoid buried infrastructure and structural elements as they make changes to buildings and outdoor environments. Webster et al. (1996) also presented an AR system to guide workers through assembly of a space frame. Hammad et al. (2004) augmented contextual information on real views of bridges to help inspectors conduct inspections more effectively. Thomas et al. (1998) explored AR to visualize designs outdoors. Dunston et al. (2002) have also demonstrated the value of mixed reality AR-CAD in collaborative design. To the authors' best knowledge, the use of AR to animate simulated construction models at the operations level of detail and in outdoor environments has not been investigated before.

Creating AR animations from the results of a simulated model can lead to significant time savings by reducing the amount of work required to create CAD models of each and every element in the scene. In addition, it produces a more realistic display of the operations by presenting a mixed view of CAD objects performing a set of simulated tasks while overlaid on real scenes of the surrounding environment.

While AR provides simulation modelers with a powerful platform to perform simulation-based animations, creating realistic AR scenes introduces a unique set of chal-

lenges. Continuous mobile user tracking, accurate registration of CAD objects, and real time acquisition of site data (e.g. site terrain, location of other equipments) are among the most important challenges an AR system developer has to address during the course of creating an animated scene (Kamat and Behzadan 2006). Finding appropriate solutions to each of these problems is essential before the animation can be relied upon as a tool to validate and verify the simulated model, thereby allowing the involved parties to be convinced that the model indeed reflects reality. Having achieved this, the model can be described as a credible representation of the real operations and the results can be conveniently used in real life decision making.

2 SIMULATION-BASED ANIMATION IN AR

The authors have been working on design and implementation of a general purpose visualization system that uses the concepts of scene graphs together with the capabilities of modern tracking technology to create and display augmented views of simulated operations to a mobile user in real time. The result of this ongoing research is presented as an AR visualization application called ARVISCOPE (acronym for the Augmented Reality VIsualization of Simulated Construction OPErations).

2.1 System Description

ARVISCOPE creates and displays augmented views of a simulated system with the passage of time. It is accompanied by an authoring language which allows the animation developer to create an ASCII text file referred to as the "animation trace file" in this paper. Although the trace file is meant to be automatically generated by simulation software to avoid dealing with scene complexities and input errors the user himself can also manually create one using common text editor tools. The trace file is read through by the application and the corresponding animation is created in real time. ARVISCOPE is capable of importing and using a wide variety of CAD file formats.

The application has an Object-Oriented Design (OOD) which allows for maximum flexibility in terms of reusability of the methods in other AR-based applications. Each hardware component (as described in subsection 2.2) is accompanied by a software module (i.e. class) which encapsulates different functionalities and methods for data transmission and extraction. Since the design is modular, any hardware/software upgrade or replacement does not affect the integrity of the system.

Also, the methods developed in this work can be easily customized and reused by any potential AR application developer as long as they use the same set of hardware components (Behzadan and Kamat 2007a).

2.2 Hardware Components

Unlike traditional VR-based animation tools in which the final output is displayed on a computer monitor, ARVISCOPE takes advantage of a mobile backpack system which enables a user to walk in the site while watching a continually updating augmented animated scene through the Head Mounted Display (HMD) installed in front of the user's eyes. The video of the surrounding environment is continuously captured and sent to the application to be later shown as the real background for the scene. Figure 1 shows the profile of a mobile user in ARVISCOPE.

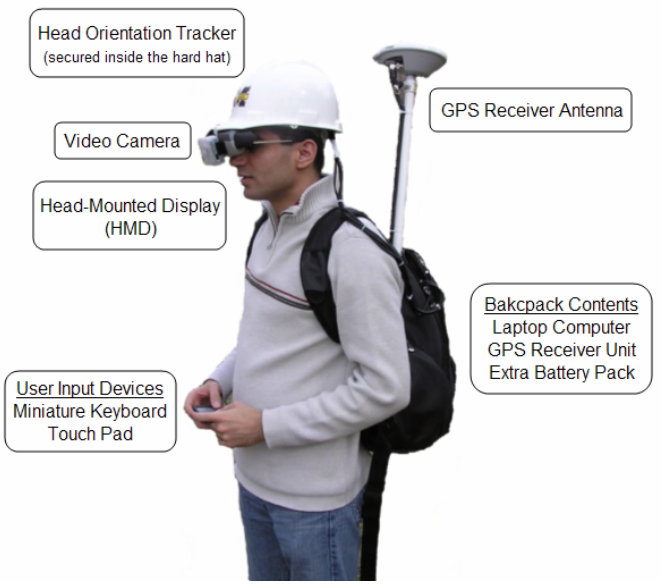

Figure 1: Overview of the hardware components of ARVISCOPE

This hardware configuration also incorporates two important tracking devices namely a GPS receiver and a head orientation tracker. These are the two most important pieces of hardware since they capture user's 6 Degree-of-Freedom (DOF) data in real time and send them to the animation engine to be used for precise registration of CAD objects inside the user's viewing frustum in real time. The computing power of the system is provided by a laptop which is placed and secured inside the backpack. A small touch pad and wrist keyboard have also been provided for any potential user input during the animation while there is no physical access to the laptop (Behzadan and Kamat 2007a).

2.3 Creating AR Animated Scenes

ARVISCOPE visualization system has been designed using OpenSceneGraph which is a C++ open source toolkit. The video capturing capabilities have been adapted from

OSGART, a separate open source library (Looser et al. 2006). ARVISCOPE authoring language is a high level language that allows an external software process (e.g. a running DES model) to author a dynamic animation trace file. Table 1 lists some basic language statements of ARVISCOPE together with a brief explanation of their functionality.

Table 1: Selected ARVISCOPE animation language statements and their functionality

Statement	***Functionality***
SIMTIME	Indicates the simulation time for a group of consequent statements
LOADMODEL	Loads and assigns a CAD file to a class of objects
OBJECT	Creates an instance of a certain class
ROUTE	Defines a route made by a set of points in global space (i.e. longitude, latitude, and altitude values)
POSITION	Places an object at a certain global point or on a route
ORIENT	Changes the local orientation of an object
TRAVEL	Moves an object on a route in a certain duration of time

Figure 2 presents a sample ARVISCOPE trace file which results in a simple animated scene consisting of 3 CAD objects superimposed on a real background. As the trace file is read through, a concrete truck, a concrete form, and a concrete bucket are eventually placed on the scene.

```
SIMTIME 10;

ROUTE ConcDelv
(-83.42779,42.54330,285.00)
(-83.42790,42.54315,284.00)
(-83.42820,42.54290,282.00);

LOADMODEL Truck ConcTruck.ac;
OBJECT RedTruck Truck;
POSITION RedTruck ON ConcDelv;

LOADMODEL Form ConcForm.ac;
OBJECT WoodenForm Form;
POSITION WoodenForm AT
(-83.42820,42.54285,282.00);

LOADMODEL Bucket ConcBucket.ac;
OBJECT OrangeBucket Bucket;
POSITION OrangeBucket AT
(-83.42820,42.54287,282.00);
```

Figure 2: Sample ARVISCOPE trace file

Figure 3 shows how the augmented view is constructed over simulation time and as the trace file is read by ARVISCOPE. Since all the statements of the trace file in Figure 2 are executed simultaneously at simulation time 10, what the user really sees through the HMD is the last snapshot in this figure.

Figure 3: Construction of the augmented scene over the simulation time in ARVISCOPE

3 THE AUGMENTED VIEW SCENE GRAPH

3.1 Overview and Characteristics

The fact that in a dynamic animated scene, the animation engine should be capable of manipulating (i.e. positioning,

orienting, and scaling) each CAD object individually is the main reason why a scene has to be assembled from discrete components as opposed to be modeled as a single unit (Behzadan and Kamat 2007a, Kamat and Martinez 2002).

The concept of scene graphs can be effectively used to facilitate the creation of assembled scenes. A scene graph is in fact a hierarchical data structure of objects often referred to as *nodes* that can be arranged and manipulated individually or as a group inside a scene (Kamat and Martinez 2002). Each node encapsulates the semantics of what is to be drawn. All the elements in the scene graph are connected either directly or indirectly (i.e. through other nodes) to the *root node*. Figure 4 shows an illustrative scene graph structure in a sample VR-based scene.

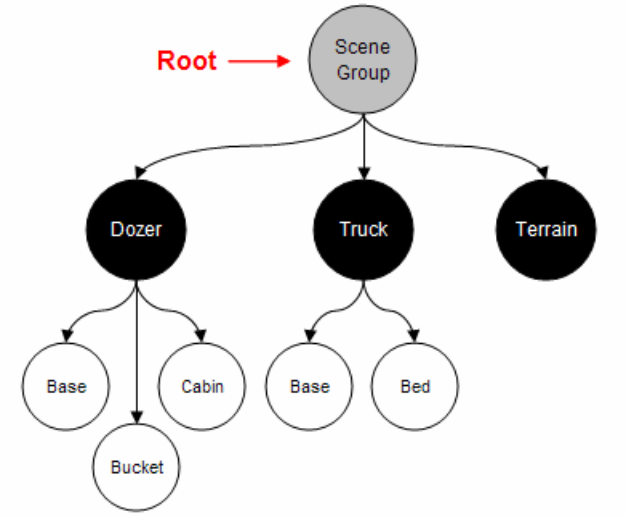

Figure 4: Scene graph of a sample animated scene in VR

In this figure, the node "Jobsite" is the root node. Scene sub-graphs are created and attached to the root node to complete the scene structure by encapsulating the entire job site. In Figure 4, scene sub-graphs "Dozer", "Truck", and "Terrain" are children of the root node. Nodes "Dozer" and "Truck" have also their own children nodes connected to them at the lowest level of the hierarchy. These lowest level nodes contain the geometrical description of the individual components of their parent nodes. By definition. "Dozer" and "Truck" nodes that group together a number of geometrical nodes are called *group nodes*. The low level nodes containing the geometrical description of the components are called *leaf nodes*. A single CAD file can be assigned to each leaf node which contains the geometric shape, properties, and dimensions of the node.

In a purely VR-based scene, the root node has a fixed position and orientation in the scene and the CAD objects connected to this node through the scene graph only change position and orientation when they move or rotate. However, in an AR-based scene, the root node itself moves and/or rotates as a change occurs in the user's position and orientation. While CAD objects may change their position and orientation due to them being individually manipulated in the scene based on the simulation logic, any change in user's position and orientation will affect the entire scene graph and updates the augmented view immediately (Behzadan and Kamat 2007b). Figure 5 shows an illustrative scene graph structure in a sample AR-based scene.

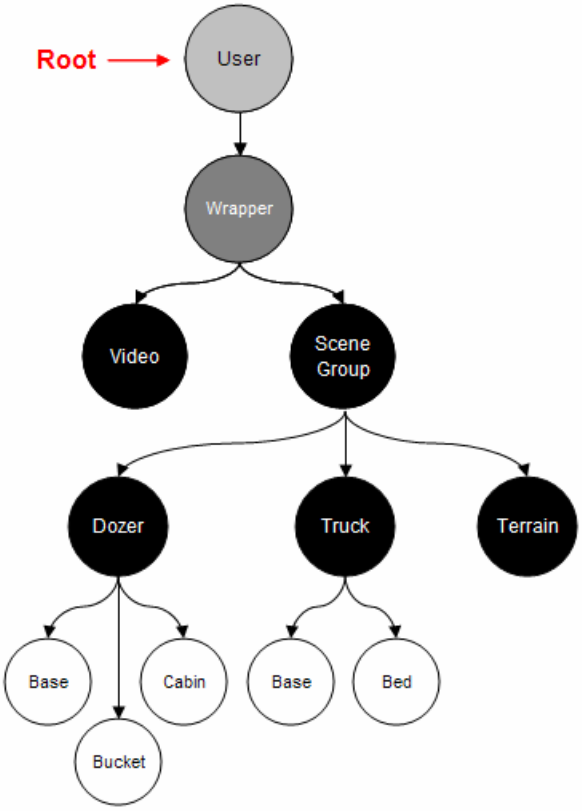

Figure 5: Scene graph of a sample animated scene in AR

In this figure, a new *video node* is also a part of the AR scene graph. Real visual data from the surrounding environment is captured and stored in this node and the AR application uses this data to construct the real background of the animation.

3.2 Creating the Scene Graph

There are two coordinate systems involved in creating every scene graph: world (global) coordinate system, and object (local) coordinate system. Since the position of the root node in a VR-based scene graph is always fixed, objects can be placed at a relative distance from the root assuming the root coordinates to always be the origin of the global coordinate system during the animation. However, in an AR-based scene graph, the position and orientation of the root node can potentially change over time due to any user's movement in the augmented scene. As a result, the global position of the root (i.e. the user) has to be acquired

in every frame using accurate tracking devices and the entire scene has to be reconstructed based on the new positional and orientation values of the root node. (Behzadan and Kamat 2007b). These values are mainly obtained from the tracking devices connected to the mobile user. Examples are Global Positioning System (GPS) receivers, head orientation trackers, and motion sensors. Authors have previously developed a reusable framework for real time communication with standard GPS devices and head orientation trackers (Behzadan and Kamat 2007b, Kamat and Behzadan 2006). The term "standard device", in the context of this paper, is defined as a tracking device which complies with a certain data transmission protocol. For example, the framework developed by the authors is capable of establishing real time communication with every type of GPS receivers as long as they parse out data streams (containing longitude, latitude, and altitude) following the National Marine Electronics Association (NMEA) standards (Bennett 2006). Also, the developed framework can communicate with a wide range of head orientation tracking devices that send binary data packets containing the 3D head orientation angles (i.e. yaw, pitch, and roll angles).

Knowing the global coordinates of the user and those of the CAD objects, a modified version of the Vincenty algorithm is applied to calculate the relative distance between the user and each of the CAD objects in x, y, and z directions (Behzadan and Kamat 2007b). The scene is created by appropriately placing these geometrical objects (each created and defined in its own local space) at appropriate positions and orientations in the world space relative to the user. This is accomplished using *transformation nodes*. Transformation nodes allow scene graph developers to manipulate the location (translation), rotation, and scale of their child nodes. They are basically group type nodes that translate the local coordinates of their child nodes into the coordinates of their parent nodes. Transformation nodes are an implicit part of the scene graph. For example, in Figures 4 and 5, each of the group nodes "Truck", "Dozer", and "Terrain" has an implicit transformation node in order to be able to be placed and oriented at desired positions relative to the node "SceneGroup". Moreover, the node "SceneGroup" in Figure 5, has an implicit transformation node so it can be positioned and oriented relative to the mobile user at each instant of time.

3.3 Visualizing the Scene Graph

The main difference between a VR-based and an AR-based scene graph from the visualization point of view is the mechanism through which different views of the scene are displayed to the viewer (i.e. the user). In VR, this is done by changing viewpoints which are basically a set of hidden cameras in the scene. The user can switch between cameras in order to view the scene from different perspectives. Cameras are not parts of the scene graph and hence do not have a nodal representation. They are external mechanisms for visualizing the encapsulated data in the scene graph.

In contrast, in an AR-based scene graph the term viewpoint has a completely different definition. The scene is viewed by the mobile user as he or she walks in the site. As such, there is only one moving viewpoint attached to the user's eye. This means the only viewpoint in an AR animation which involves a mobile user is a 6 DOF viewpoint attached to the eye of the moving user. The user can change her global position (i.e. longitude, latitude, and altitude) while she is rotating her head in a 3D global space (i.e. yaw, pitch, and roll angles) (Kamat and Behzadan 2006). As shown in Figure 5, this single viewpoint has a nodal representation and in fact, serves as the root of the scene graph.

3.4 Animating the Scene Graph

Animating a scene graph is achieved by manipulating the values of the transformation nodes. Position, orientation, and scale of each component in the scene can be manipulated relative to its parental node. Updating these values at each frame leads to a dynamically changing scene. Since this is done continuously over time, a smooth animation is displayed to and viewed by the user.

To illustrate this fact, consider the dozer in the scene graph of Figure 5. As described earlier, the relative position of the dozer can be changed due to two main reasons: 1) the dozer moves in the scene from one point to another in a certain duration of time following simulation logic, or 2) the user's global position changes and a new GPS data stream is captured by the application. Figure 6 shows the situation in which the user is fixed in the scene and a CAD object (a dozer in this figure) changes its position. In fact, this situation simplifies the AR-based scene graph and makes it very similar to the VR-based scene graph in which the root node is fixed in the global space. As the dozer moves along the x axis, its global coordinates change. Knowing the global position of the user in terms of longitude, latitude, and altitude, the relative distance between the user and the dozer is calculated at each frame. The values of the dozer transformation node is updated based on the calculated relative distance and the position of the dozer is modified accordingly inside the user's viewing frustum. As a result, the user views a moving dozer inside the augmented viewing frustum.

In another situation, the CAD object is fixed and the user changes position over time. This has been shown in Figure 7. As the user moves to the left, global coordinates are continuously acquired from the GPS device. Knowing the global coordinates of the CAD object (a dozer in this figure), the relative distance between the user and the dozer is calculated at each frame. Again, the values of the dozer transformation node are updated based on the calculated relative distance and the position of the dozer is modified

accordingly inside the user's viewing frustum. As a result, the user movement to the left is interpreted as a translation of the dozer to the right. This is essentially equivalent to the case in which the user is fixed and the dozer moves to the right inside the viewing frustum.

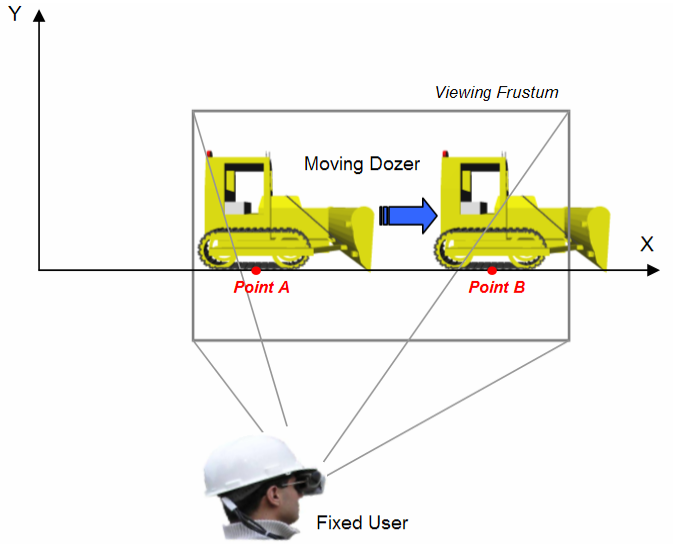

Figure 6: Animating a moving CAD object in an AR-based scene graph while the user is fixed

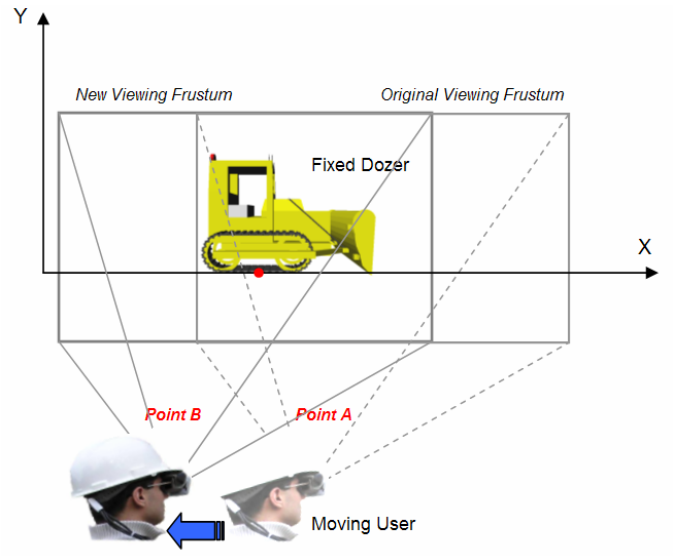

Figure 7: Animating a fixed CAD object in an AR-based scene graph while the user is moving

The same procedure is applicable to the case in which the head orientation of the user changes. Based on the new values obtained from the orientation tracker device, the orientation matrix of the node "SceneGroup" is continuously updated and as a result, the entire scene rotates in response to the user's head orientation change.

It is obvious that the values of a transformation node can be manipulated as a result of simultaneous user and CAD object movements. In this case, each movement is handled separately and the resulting values are combined to produce the final animated scene graph. In order to achieve a realistic animation, every CAD object has to move smoothly between two points rather than jump from one point to another. This is achieved by updating the viewing frame continuously at high refresh rates.

4 CONSTRUCTION OF A SAMPLE AUGMENTED SCENE GRAPH

Considering the trace file represented in Figure 2, the construction of an augmented scene graph which serves as the basis for an AR-based animation starts with reading the statements in the trace file. Sine the trace file of Figure 2 starts with a `SIMTIME` statement with a time value of 10, manipulation of virtual objects over the real background starts after 10 simulation time units are passed. At this point, a route `ConcDelv` is defined by specifying the start, end, and an intermediate point in terms of longitude, latitude, and altitude values (i.e. global coordinate system). Since a route does not have a nodal representation it is not a part of the scene graph. In general, routes are only used to compute translation fields of the transformational nodes that place objects along the route on the scene graph.

The scene graph starts to take real shape as ARVISCOPE reads the first set of `LOADMODEL`, `OBJECT`, and `POSITION` statements in the trace file. The `LOADMODEL` statement defines a leaf node, `Truck`, which contains a geometry from the CAD file `ConcTruck.ac` in AC format. `OBJECT` statement instantiates a transformation node `RedTruck` and adds a child object conforming to the geometry defined by class `Truck` to it. The `POSITION` statement places the created object in the scene by adding the `RedTruck` transformation node to a parent node called "SceneGroup". By default, the object is placed at the beginning of the route the coordinates of which has previously defined in the trace file. The critical point in placing CAD objects inside the user's viewing frustum is that they have to be placed relative to where the user is standing in the global space. As a result, the global coordinates of each CAD object have to be converted to relative displacement values between that object and the user in terms of Δx, Δy, and Δz. This is continuously done in ARVISCOPE using the modified Vincenty algorithm.

Figure 8 shows the augmented scene graph after the first set of `LOADMODEL`, `OBJECT`, and `POSITION` statements are read. The subsequent three statements create and place a concrete form in a fixed position. Figure 9 shows the augmented scene graph after the second set of `LOADMODEL`, `OBJECT`, and `POSITION` statements are read.

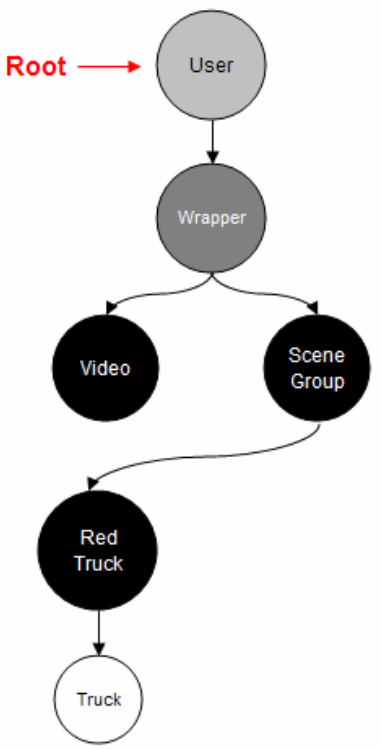

Figure 8: The AR scene graph after the concrete truck is placed on the path

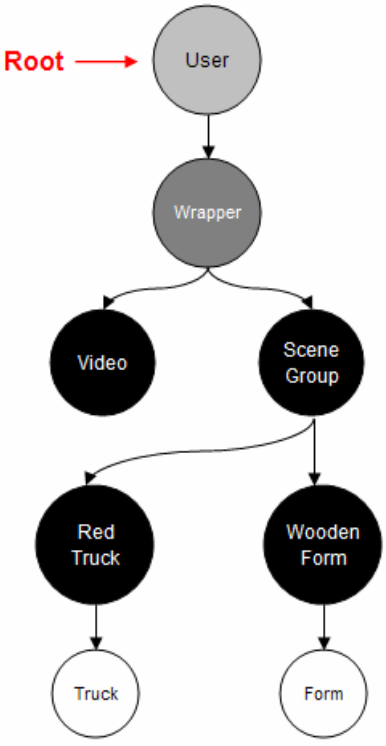

Figure 9: The AR scene graph after the concrete form is placed in the scene

Finally, the last three statements create and place a concrete bucket in a fixed position. Figure 10 shows the augmented scene graph after the last set of `LOADMODEL`, `OBJECT`, and `POSITION` statements are read.

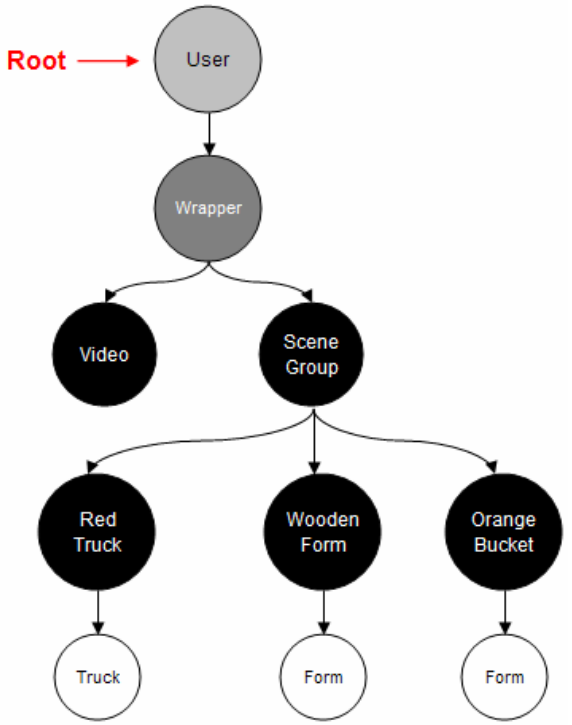

Figure 10: The AR scene graph after the concrete bucket is placed in the scene

It is worth mentioning that the root node of the scene graph is the mobile user which is created immediately before the animation starts. All CAD objects in the scene are placed and any future manipulation on their transformation values are done relative to the "SceneGroup" node which itself is a grandchild of the root node. The global position of the user is first obtained when the data communication with the GPS receiver is established as the application starts. The initial head orientation values of the user are also obtained from the orientation tracker as soon as it is initialized at the beginning of the application.

ARVISCOPE handles the scene graph in a completely scalable manner. All CAD objects in the scene that use the same geometrical representation can be conveniently created by making instances of a single CAD file associated to a class of objects. As a result, the CAD file needs to be loaded only once by the application which saves a lot of memory and running time. Hence, even very complex scenes such as an entire steel frame structure consisting of hundreds of beams and columns can be depicted by loading only a few CAD models of beams and columns and placing them repeatedly at appropriate locations using multiple transformation nodes. This is due to the capability of

the transformation nodes to scale and rotate objects of the same geometry.

5 TIME TRACKING IN ARVISCOPE

ARVISCOPE measures time in terms of *animated time units*. One animated time unit can equal whatever duration is most suitable for the animation (e.g. a second, a minute, or a day) as long as it matches the time unit in the simulation model that is driving the animation. The primary time-tracking ARVISCOPE statement is `SIMTIME`. The syntax of the `SIMTIME` command is as follows:

```
SIMTIME timevalue;
```

The `SIMTIME` statement waits for the animation clock to reach the new value specified. ARVISCOPE then executes the statements that follow it until next `SIMTIME` statement is reached. After verifying that the *`timevalue`* is greater than or equal to the current animated time, ARVISCOPE suspends the reading of any more lines from the trace file until the animation time specified by the `SIMTIME` statement has been reached or exceeded. When that happens, ARVISCOPE reads and processes the next line(s) in the trace file until another `SIMTIME` statement is encountered. This procedure repeats until the end of the trace file is reached. Figure 11 shows how a sample trace file is being read and processed by ARVISCOPE.

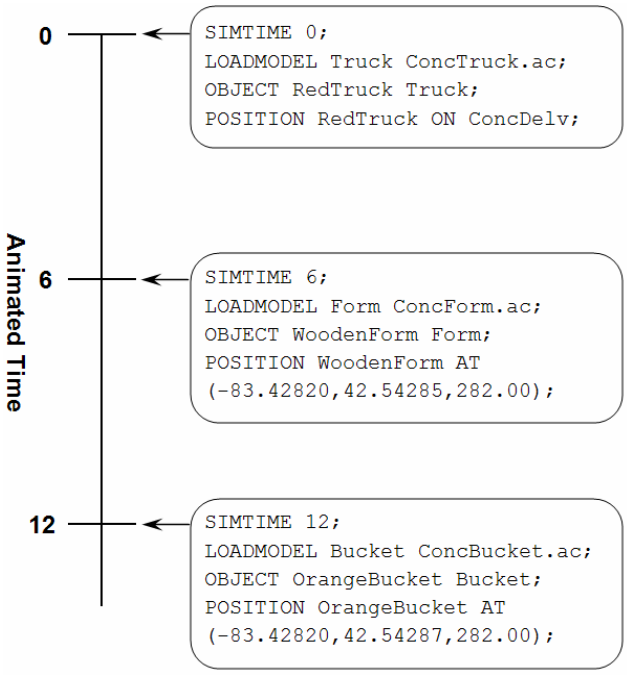

Figure 11: Processing of statements in a trace file containing various `SIMTIME` statements

By default, the augmented animation runs at a 1:1 time ratio in ARVISCOPE. This essentially means that one real clock second corresponds to one simulation time unit. Thus, if a truck is set to move along a path for 2 minutes, its moving duration has to set to 120 units inside the animation trace file. This is mainly due to the fact that the main objective of this application is to have the user interact with the scene by walking inside the augmented animation. Hence, a too low or too high animation speed keeps the user from being involved at normal rate of walking or head rotation. The exception, however, is the case in which the user is only an observer of the animation with no real interaction with the scene. In this case, the animation speed can be set to be higher or lower than the normal rate to animate very slow or very fast operations respectively.

6 CONCLUSION

The authors introduced an AR-based application called ARVISCOPE as a scalable visualization tool which is capable of creating smooth animations of simulated construction models in AR. ARVISCOPE takes advantage of four critical components to achieve this capability: 1) the results of a DES model which can be used to automatically generate the animation trace file, 2) a scene graph structure of CAD objects in order to provide required functionality to manipulate them inside the user's viewing frustum in a hierarchical manner, 3) robust and precise position and orientation tracking methods to obtain real time 6 DOF of the user and appropriately register CAD objects inside the augmented dynamic scene, and 4) an efficient time keeping technique to keep track of the animation and simulation time passage.

The main challenge in creating a smooth animation of DES models is the fact that DES systems can communicate with other processes only at discrete simulated time points which are typically start and finish points of activities. Transforming such a discrete system to a continuous and smooth set of animated scenes when combined with the requirements of AR-based visualization introduces unique set of challenges. These are including but not limited to continuous mobile user tracking using GPS and head orientation devices, accurate registration of CAD objects inside the user's viewing frustum, and real time acquisition of site data (e.g. site terrain, location of other equipment) to update the contents of the AR scene.

Using the concept of scene graphs inside ARVISCOPE allows the effective creation, organization, manipulation, and maintenance of the graphical data of CAD objects inside an AR scene which is the primary requirement to depict a dynamic animation. A sample animation was described and illustrated in detail to display the capabilities of the application and the mechanism it uses to handle a dynamically changing AR animated scene.

ACKNOWLEDGMENTS

The presented work has been supported by the National Science Foundation (NSF) through grant CMS-0448762. The authors gratefully recognize this support from the NSF. Any opinions, findings, conclusions, and recommendations expressed by the authors in this paper do not necessarily reflect the views of the NSF.

REFERENCES

Behzadan, A. H., and V. R. Kamat. 2007a. General Purpose Modular Hardware and Software Framework for Mobile Outdoor Augmented Reality Applications in Engineering, *Advanced Engineering Informatics*, New York, NY: Elsevier Science. (In Review)

Behzadan, A. H., and V. R. Kamat. 2007b. Georeferenced Registration of Construction Graphics in Mobile outdoor Augmented Reality, *ASCE Journal of Computing in Civil Engineering*, Reston, VA. (In Press)

Behzadan, A. H., and V. R. Kamat. 2006. Animation of Construction Activities in Outdoor Augmented Reality, In *Proceedings of the 11th International Conference on Computing and Decision Making in Civil and Building Engineering (ICCCBE-XI)*, Montreal, QB, Canada.

Bennett, P. (2006). *National Marine Electronics Association - FAQ*. Edge of Space Science. Available via <http://www.eoss.org/pubs/nmeafaq.htm> [accessed December 5, 2006].

Dunston P., X. Wang, M. Billinghusrt, and B. Hampson. 2002. Mixed Reality benefits for design perception. In *Proceedings of 19th International Symposium on Automation and Robotics Construction (ISARC 2002)*,191-196, Gaithersburg, MD: NIST.

Hammad A., J. H. Garrett, and H. Karimi. 2004. Location-based computing for infrastructure field tasks. Telegeoinformatics: Location-based computing and services, 287-314. CRC Press.

Kamat, V. R. 2003. VITASCOPE: Extensible and Scalable 3D Visualization of Simulated Construction Operations. PhD dissertation, Department of Civil and Environmental Eng., Virginia Tech, Blacksburg, VA.

Kamat, V. R., and A. H. Behzadan. 2006. GPS and 3DOF Tracking for Georeferenced Registeration of Construction Graphics in Outdoor Augmented Reality, In *Proceedings of the 13th EG-ICE Workshop of Intelligent Computing in Engineering and Architecture*, Ascona, Switzerland.

Kamat, V. R., and J. C. Martinez. 2002. Scene Graph and frame update algorithms for smooth and scalable 3D visualization of simulated construction operations, *Journal of Computer-Aided Civil and Infrastructure Engineering* 17(4): 228-245, Malden, VA:

Looser J., R. Grasset, H. Seichter, and P. Lamb (2006). OSGAT: ARToolkit for OpenSceneGraph. Available *at* < http://www.artoolworks.com/ community/osgart/> [accessed April 10, 2007].

Roberts G. W., A. Evans, A. Dodson, B. Denby, S. Cooper, and R. Hollands. 2002. The use of Augmented Reality, GPS, and INS for subsurface data visualization. *International Congress*, Washington, D.C.

Thomas B., W. Piekarski, and B. Gunther. 1998. Using Augmented Reality to visualize architectural designs in an outdoor environment. *DCNet'98 Online Conference,* Available via <http://www.arch.usyd.edu.au/kcdc/journal/vol2/dcnet/sub8> [accessed: March 10, 2003].

Webster A., S. Feiner, B. MacIntyre, W. Massie, and T. Krueger. 1996. Augmented reality in architectural construction, inspection and renovation. In *Proceedings of 3rd Congress on Computing in Civil Engineering*, 913-919. Reston, VA: ASCE.

AUTHOR BIOGRAPHIES

AMIR H. BEHZADAN is a Ph.D. Candidate in the Department of Civil and Environmental Engineering at the University of Michigan. He got his Master's degree in Construction Engineering and Management from the same university in 2005. He also holds a BE degree in Civil Engineering from Sharif University of Technology (Tehran, Iran). His current research interests are Augmented Reality visualization of construction operations, discrete-event simulation and Information Technology. He is an associate member of ASCE Construction Institute and a student member of CMAA and PMI. His e-mail address is <abehzada@umich.edu> and his web address is <http://www-personal.umich.edu/~ abehzada>.

VINEET R. KAMAT is an Assistant Professor in the Department of Civil and Environmental Engineering at the University of Michigan. He received an MS and Ph.D. in Civil Engineering at Virginia Tech in 2000 and 2003, respectively; and a B.E. in Civil Engineering from Goa University (Goa, India) in 1998. His primary research interests include virtual and augmented reality, simulation, information technology, and their applications in Civil Engineering. He designed and implemented the VITASCOPE visualization system with J. Martinez as part of his doctoral research and is currently supervising the design and implementation of the ARVISCOPE augmented reality system as part of A. Behzadan's doctoral research. His email and web addresses are <vkamat@umich.edu> and <http://pathfinder.engin. umich.edu>