ITcon

www.itcon.org - Journal of Information Technology in Construction - ISSN 1874-4753

INTEGRATED INFORMATION MODELING AND VISUAL SIMULATION OF ENGINEERING OPERATIONS USING DYNAMIC AUGMENTED REALITY SCENE GRAPHS

SUBMITTED: September 2010
REVISED: October 2010
PUBLISHED: February 2011 at http://www.itcon.org/2010/17
EDITOR: Turk Z.

Amir H. Behzadan, Wharton Smith Faculty Fellow and Assistant Professor
Department of Civil, Environmental, and Construction Engineering
University of Central Florida, Orlando, FL 32816, USA
email: abehzada@mail.ucf.edu
http://pegasus.cc.ucf.edu/~abehzada/

Vineet R. Kamat, Associate Professor
Department of Civil and Environmental Engineering
University of Michigan, Ann Arbor, MI 48109, USA
email: vkamat@umich.edu
http://pathfinder.engin.umich.edu

SUMMARY: *This paper describes research that investigated the applicability of 3D Augmented Reality (AR) in visualizing dynamic engineering operations. The research resulted in the design and development of ARVISCOPE, a general purpose high level AR animation scripting language, and ROVER, a mobile computing hardware framework. When used together, ARVISCOPE and ROVER can create 3D AR animations of arbitrary length and complexity at the operations level of detail. ARVISCOPE takes advantage of advanced Global Positioning System (GPS) and 3D orientation tracking technologies to accurately track a user's spatial context while georeferencing superimposed dynamic 3D graphics in an augmented environment. In achieving the research objectives, major technical challenges such as accurate registration, automated occlusion handling, and dynamic scene construction and manipulation were encountered and successfully addressed. This paper, describes the methodology and technical details of how scene graphs enable the process of constructing and updating an augmented scene representing a dynamic engineering operation.*

KEYWORDS: *Animation, Augmented Reality, Construction, Information Delivery, Visualization.*

REFERENCE: *Amir H. Behzadan, Vineet R. Kamat(2011) Integrated Information Modeling and Visual Simulation of Engineering Operations using Dynamic Augmented Reality Scene Graphs, Journal of Information Technology in Construction (ITcon), Vol. 16, pg. 259-278, http://www.itcon.org/2011/17*

1. INTRODUCTION

Visual simulation has emerged as a key planning and analysis tool in engineering processes such as construction and manufacturing because it allows the creation of dynamic scenes using virtual computer generated resources, thereby providing an environment where experimentation can be done without committing real resources or endangering operational safety. 3D Visualization has traditionally been achieved in Virtual Reality (VR). In order to create convincing VR animations, detailed information about an operation and the environment has to be obtained. The data must describe the underlying processes, 3D CAD models of project resources, the facility under construction, and the surrounding terrain (Model Engineering). As the size and complexity of an operation increase, such data collection becomes an arduous, impractical, and often impossible task. This directly translates into loss of financial and human resources that could otherwise be productively used. In an effort to remedy this situation, the research presented in this paper used an alternate approach of visualization called Augmented Reality (AR) to create mixed views of CAD models of construction resources overlaid on real time views of existing jobsite facilities. The application of AR in animating engineering operations has significant potential in reducing the aforementioned Model Engineering and data collection tasks, and at the same time can help in creating visually convincing output that can be effectively communicated. Although some disciplines (e.g. medicine, automotive industry, and military) have been using AR as a frontline technology to overcome visualization challenges in their domain (e.g. Feiner et al. 1997, Thomas et al. 2000, Barfield and Caudell 2001, Gleue and Dähne 2001, Livingston et al. 2002, Suthau et al. 2002, Brown et al. 2003, Piekarski and Thomas 2003), work still needs to be done to develop functional and reliable AR-based visualization tools that can be effectively used in areas such as construction and other engineering disciplines.

An augmented scene presents a combination of virtual and real objects spatially (and logically) arranged and linked in a single view. Real world objects are independent entities that occupy certain space in the 3D environment and can potentially change their position and orientation over time. Virtual objects, however, have to be first calculated and appropriately placed in the scene before they can be manipulated. There are two major approaches to model and place virtual objects in a graphical scene. Such models can be either created as a single unit or assembled from discrete components (Kamat and Martinez 2002). Modeling the scene as a single unit requires the modeler to create a static model with all the individual elements appropriately placed in their correct position and orientation in the scene. The resulting model using this approach is usually suitable for graphical applications in which the main purpose is to render, visualize, and examine large data sets ranging from individual components, to subassemblies, to entire complex layouts (Kamat and Martinez 2002).

An alternative approach which is more effective for animating a graphical scene is to create the entire graphical contents from discrete CAD models. The modeler can then manipulate position, orientation, and scale of individual CAD objects without affecting the integrity of the scene. Another major advantage of this approach is that individual scene components can be created using separate modeling tools (e.g. AutoCAD, Photoshop, AC3D, 3D Studio) or directly inside a Building Information Modeling (BIM) supported environment and later imported to the AR scene. A 3D model created in a BIM environment can be constantly updated during the project life cycle to represent the actual dynamics of the operations (Eastman et al. 2008). This brings significant flexibility and speed to the modeling and animation processes. In this research, discrete component modeling approach was selected as it results in more flexibility, enables BIM integration into the animation environment, and provides higher manipulation control when animating a scene consisting of virtual and real objects.

2. MAIN CONTRIBUTIONS

In order to perform dynamic engineering operations such as construction or manufacturing tasks, several types of physical resources (i.e. equipment, machinery, personnel, material) have to be deployed. As a result, for a 3D visualization to be a realistic representation of the complex dynamics of a typical construction site, it must be able to demonstrate, manipulate, and manage a large amount of CAD objects included in the visual simulation of the operation.

The common problem arising in almost every large scale visualization system is the amount of data that must be loaded and rendered at each frame. As the size and complexity of the animated environment increases, slow rendering speed becomes the bottleneck for real time user interaction with the scene even if fast CPU speed and powerful graphic accelerator cards are used (Zara et al. 2001; Kalra et al. 1998). Overcoming the hardware limitations by reducing the amount of time and CPU work required to handle the CAD objects inside an AR animation was the main motivating factor in this research. The primary contribution of the research presented in this paper is a flexible and dynamic methodology to facilitate the tasks of constructing and managing the contents of the AR animation of a dynamic engineering operation by continuously loading, rendering, and displaying the virtual CAD models and real surrounding environment in which the operation takes place.

This objective was successfully achieved by first arriving at a computer interpretable description of the visible space in front of the user's eyes (i.e. viewing frustum). This helped in limiting the number of CAD objects necessary to be rendered at each animation frame. In particular, only CAD objects located inside this visible space need to be loaded and displayed which significantly decreases the computation load on the graphics card and CPU and at the same time increases the frame rate. The next major step was to establish an efficient object structure to optimize the management of CAD objects inside the user's viewing frustum. This was achieved by applying the concept of scene graphs and introducing a new hierarchical object structure to facilitate CAD object manipulation inside an AR animation.

3. CREATING AN AR PERSPECTIVE VIEWING FRUSTUM

The very first step in creating an animated scene is to define the space in which virtual and real objects coexist. Without having a precise contextual definition of the 3D scene, real and virtual objects in the scene are not able to establish spatial and contextual links with each other and proper registration of CAD objects inside the augmented scene cannot be achieved and maintained throughout the animation. In order to construct the augmented space, the first set of information is the location and dimensions of the visible world. This is achieved by defining a viewing frustum centred on the eyes of the scene observer. Since real world is perceived by human eyes in a perspective manner, the same visual basis has to be applied to the virtual contents of the augmented world in order to ensure proper alignment of the two worlds. As a result, a perspective viewing frustum has to be defined with the observer's eyes always located at the projection point.

Figure 1 shows the parameters of a perspective viewing frustum used in this research. In this Figure, the volume of space inside the truncated pyramid confined by near plane (marked by points P_1 through P_4) and far plane (marked by points P_5 through P_8) is called the perspective viewing frustum. Objects inside this limit are visible to the user and hence have to be displayed and animated. Other objects are either culled (if completely located outside the frustum) or clipped (if partially located inside the frustum) (Shreiner et al. 2004). The relative distances of near and far planes to the observer can change depending on the amount of visible 3D space to the observer. Since the observer and objects are constantly moving in the scene, the origin (i.e. projection point of the frustum) also changes position which results in the contents of the perspective viewing frustum to be continuously changing over time.

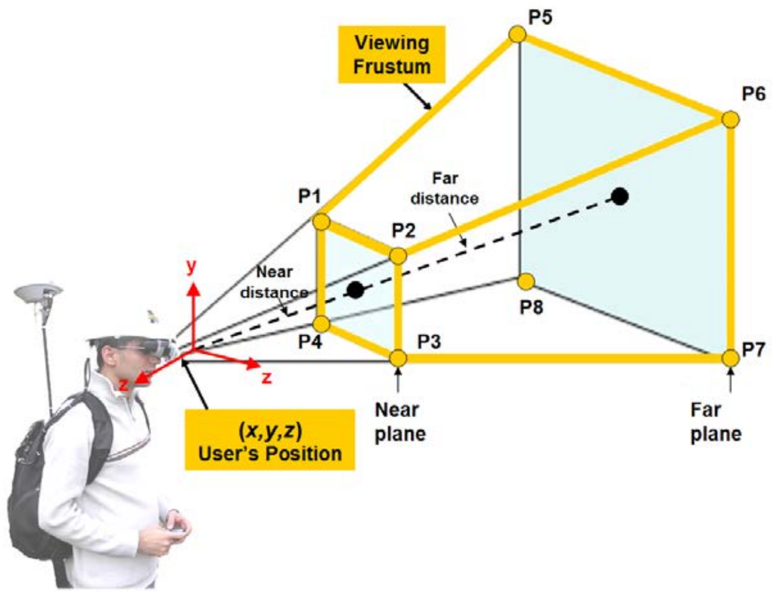


FIG. 1: Definition of a Perspective Viewing Frustum

In addition, the extent of the frustum can change by increasing or decreasing the horizontal and/or vertical field of view angles as shown in Figure 2. In this Figure, the Vertical Field of View (VFOV) angle is set to be 45 degrees and the Horizontal Field of View (HFOV) angle is set to be 60 degrees. The near and far planes in these Figures are also set to be 1 and 500 meters from the user respectively.

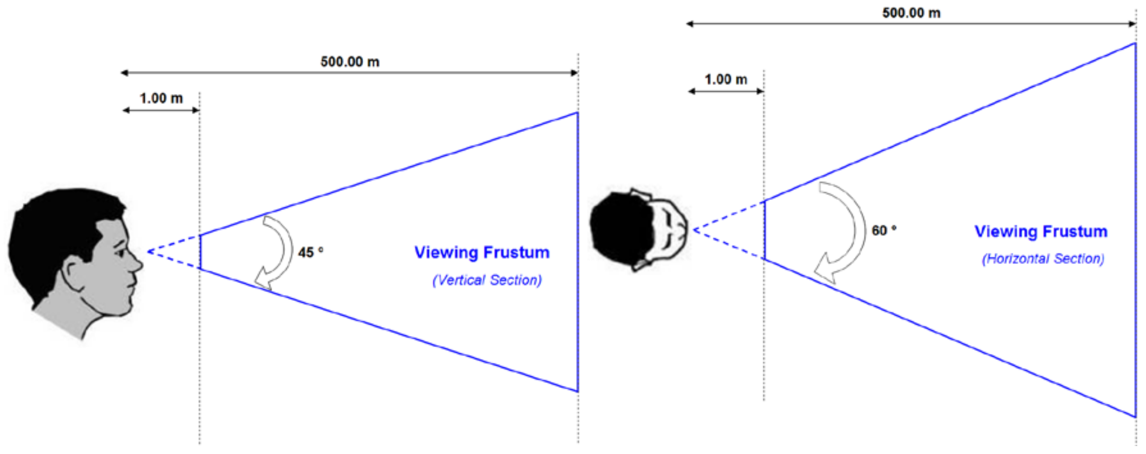


FIG. 2: Side and Top View of a Perspective Viewing Frustum

A virtual object is visible to the user as long as it is located inside the perspective viewing frustum. The real world contents of the augmented scene (which are captured by a video camera connected to the frustum projection point) are also visible to the user as long as they are inside the perspective viewing frustum of the camera. In order to achieve best results, different parameters of the virtual and real perspective viewing frustums (near plane, far plane, VFOV angle, and HFOV angle) had to be set to allow for proper alignment of the two worlds in one single augmented view.

4. DEVELOPING THE AUGMENTED SCENE GRAPH

The concept of scene graphs can be effectively used to facilitate the creation and management of assembled animated scenes. In general, a scene graph is a data structure used to organize and manage the contents of hierarchically organized scene data (Kamat and Martinez 2002; Nadeau 2000). In the realm of computer graphics, scene graphs are means to create and manipulate hierarchical organization of shapes, groups of shapes, and clusters of groups that collectively define the content of a scene (Nadeau 2000). Scene graphs address low level problems that generally arise in scene composition and management. In computer animation, scene graphs are widely used as structures that arrange the logical and spatial representation of a graphical scene (Kamat and Martinez 2002; Behzadan and Kamat 2007-1). Scene graphs allow the scene modeler to focus on what 3D objects to render rather than how to render them. It eliminates several low level programming complexities involved in rendering process of 3D objects in a scene (Kamat and Martinez 2002; Dollner and Hinrichs 1997).

There are several computer graphics implementations that build on the concept of scene graphs. Some of them (e.g. Java3D (Selman 2008)) are in form of an Application Programming Interface (API), designed to be integrated into standard programming languages such as C/C++ or Java. Alternatively, scene graphs may be in the form of a text or binary file. Such files usually adopt an authoring (or markup) language format and are generally synonymous with web applications. Examples of two such implementations are the Virtual Reality Modeling Language (VRML) and Extensible 3D (X3D). VRML is an international standard similar to Hyper Text Markup Language (HTML) (Ames et al. 1997; Lipman and Reed 2000), whereas X3D works within the constructs of the Extensible Markup Language (XML) file format (Benoit 2000; Walsh and Bourges-Sevenier 2001). In order to take advantage of scene graphs in creating graphical contents of an augmented scene, an open source, cross platform graphics toolkit called OpenSceneGraph was used in this research. Based on the concept of scene graphs, OpenSceneGraph provides an object oriented framework on top of OpenGL freeing the developer from implementing and optimizing low level graphics calls. It also provides many additional utilities for rapid development of graphics applications (OpenSceneGraph 2008).

Each scene graph is a collection of nodes in a graph structure connected together via child-parent relationship. A node is an object that can be part of or entirely comprise a scene graph. Each node is a collection of one or more *fields* (values) and *methods* that together perform a specific function. It encapsulates the semantics of what is to be drawn. A method applied to a parent node will affect all its child nodes. In order to build a larger scene graph, smaller scene subgraphs can be created separately and attached together via links to the highest level node called the *root node* (Kamat and Martinez 2002; Behzadan and Kamat 2007-1). A *link* connecting two separate nodes describes the relationships that exist between them in a meaningful way. The nodes are ideally arranged in a hierarchical manner that corresponds spatially and semantically to the modeled scene (Woolford 2003). Figure 3 shows a sample scene graph. In this Figure, the node Jobsite is the root node. Scene subgraphs are created and attached to the root node to complete the scene structure by encapsulating the entire jobsite. For example, scene subgraphs Dozer, Truck, and Terrain are all 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.

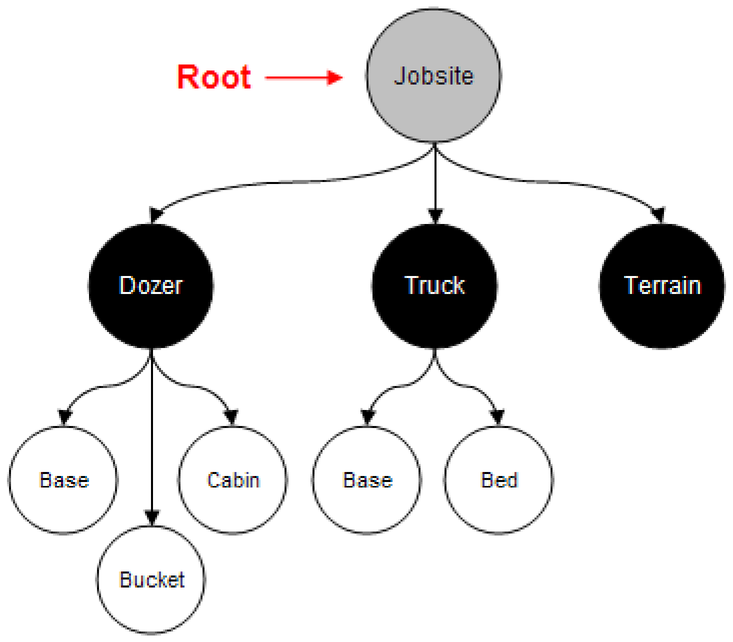


FIG. 3: A Sample Scene Graph Hierarchy

The traditional scene graph, as shown in Figure 3, has been previously used by researchers to create Virtual Reality (VR) based animations (Kamat and Martinez 2002). Although this type of scene graph correctly represents the graphical contents of a dynamic scene, the real world and the observer (as part of the real world) are not properly represented. Since this research was focused on AR visualization, a new hierarchical representation of the animated scene had to be conceptualized and developed in order to encapsulate both groups of real and virtual objects in one scene graph. In a mobile AR visualization system, the augmented graphics have to be precisely registered and displayed in accurate positions relative to the observer's eyes. As a result, the developed AR-based scene graph must be able to store, retrieve, and display graphical data from a viewpoint that represents the observer's eyes. Since the observer can constantly change position and/or head orientation in the animation, this viewpoint has to change accordingly which will cause the entire animated contents of the scene to change. In order to consider the latest changes in the observer's global position and head orientation angles, a mobile root node was used in the adopted AR-based scene graph in this research. The designed AR-based scene graph is shown in Figure 4 in which the same imaginary jobsite is represented in the presence of the real world. The root node in the scene graph shown in this Figure represents a moving viewpoint connected to the observer's eyes and moves as the observer changes position and/or head orientation in the augmented animation. In this

Figure, a new *video node* is also a part of the scene graph. This node is not a regular node in the AR-based scene graph as it does not store any geometrical data to represent a single CAD model or a group of CAD objects. Instead, it is capable of storing and displaying the latest video texture captured by the video camera in the background. The AR application combines this texture data with the original virtual world data through the *wrapper node* to construct the augmented representation of the scene.

As described earlier, there is a major difference between the two scene graphs shown in Figures 3 and 4. In a purely VR-based scene graph (Figure 3), 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 graph (Figure 4), the root node itself moves and/or rotates as a change occurs in the observer's position and head orientation. While CAD objects may change their position and orientation due to them being individually manipulated in the scene, any change in observer's position and orientation will affect the entire scene graph and updates the augmented view immediately (Behzadan and Kamat 2007-2). There are two coordinate systems involved in creating a scene graph: world (global) coordinate system, and object (local) coordinate system. Since the position of the root node in the scene graph presented in Figure 3 is always fixed, objects can be placed at a relative distance from the root assuming the root coordinates to always be at the origin of the global coordinate system during the animation. However, in the AR-based scene graph shown in Figure 4, the position and orientation of the root node can potentially change over time due to any observer's movement in the augmented scene. As a result, the global position of the root node (i.e. the observer) 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 2007-2).

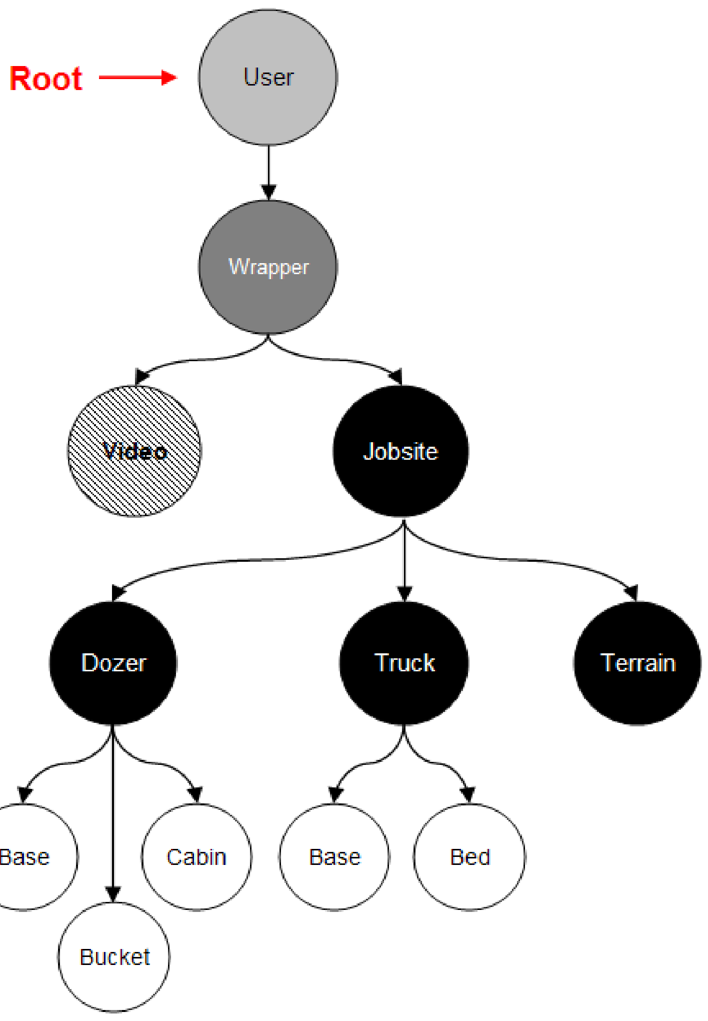


FIG. 4: The Structure of the Developed AR-based Scene Graph

The authors have previously developed a reusable framework for communication with standard GPS devices and head orientation trackers (Behzadan and Kamat 2007-3) which enables real time retrieval of positional and orientation data from the tracking devices connected to the mobile user such as Global Positioning System (GPS) receivers, Three Degree-of-Freedom (3DOF) head orientation trackers, and motion sensors (Kamat and Behzadan 2006; Behzadan et al. 2008). 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 in this research is capable of establishing real time communication with every type of GPS receivers as long as it outputs data streams (containing longitude, latitude, and altitude) following the National Marine Electronics Association (NMEA) standard (Bennett 2006). In addition, 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).

The AR scene is created by appropriately placing individual CAD objects each defined by a geometrical model. Each geometrical model is created in its own local coordinate frame, stored as a leaf node in the AR-based scene graph, and placed at an appropriate position and orientation inside the coordinate frame of its parent node. This is accomplished using *transformation nodes*. As shown in Figure 5, transformation nodes allow scene graph developers to manipulate the location (translation), orientation (rotation), and size (scale) of different nodes. They are basically group type nodes that translate the local coordinates of their child nodes into the coordinates of their parent nodes (Kamat and Martinez 2002; Woolford 2003).

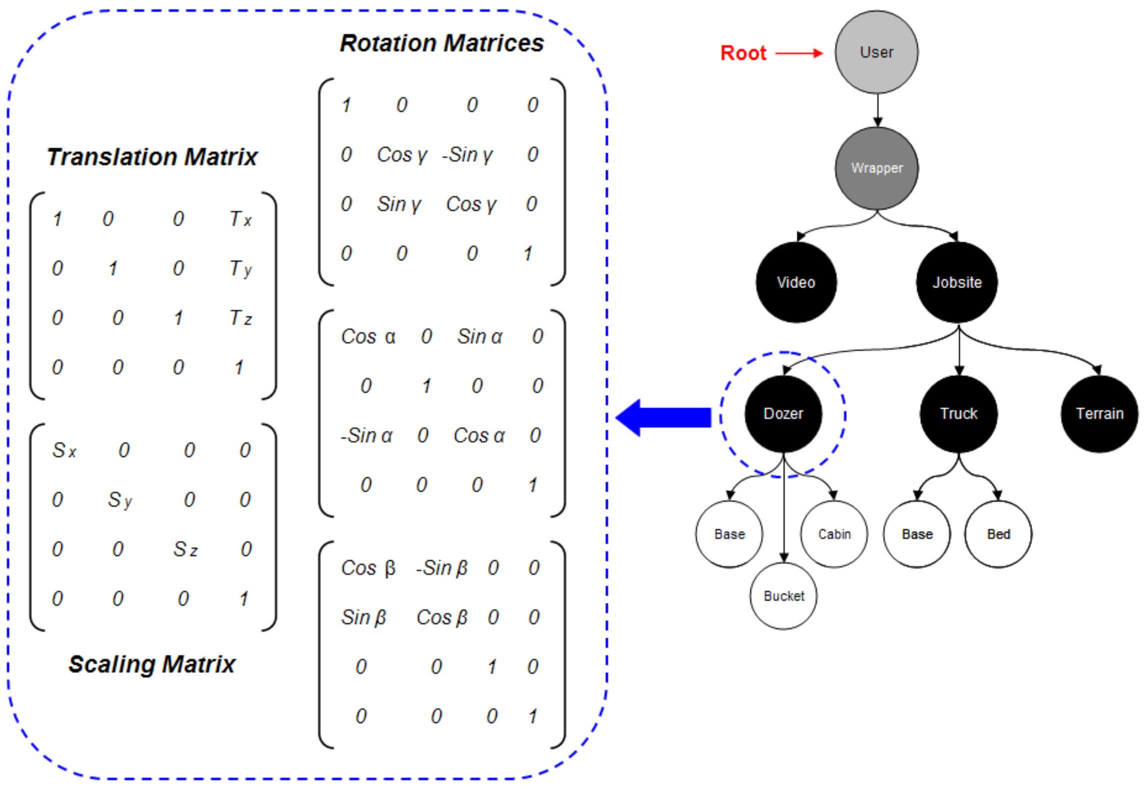


FIG. 5: Transformation Matrices Connected to a Scene Graph Node

Transformation nodes are an implicit part of the scene graph. For example, in Figures 3 and 4, each of the group nodes Truck, Dozer, and Terrain has an implicit transformation node which helps in placing and orienting the corresponding geometrical model at a desired position relative to the node Jobsite. Moreover, the node Jobsite in Figure 4 has an implicit transformation node so it can be positioned and oriented relative to the mobile observer at each instant of time. The overall transformation of a single CAD object relative to the root node is obtained by multiplying the individual matrices of Figure 5 as follows:

$$T_{object}^{user} = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & Cos\gamma & -Sin\gamma & 0 \\ 0 & Sin\gamma & Cos\gamma & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Cos\alpha & 0 & Sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -Sin\alpha & 0 & Cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Cos\beta & -Sin\beta & 0 & 0 \\ Sin\beta & Cos\beta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The transformation data of the scene (model) relative to the root node (world) is stored in a special matrix called *model view matrix*. In an AR-based scene graph, as shown in Figure 4, the root node always represents the observer's eyes. Hence, there is a one to one correspondence between the observer's eyes and the origin of the animated scene. As a result, the identity matrix is often used as the initial model view matrix. In order to draw a scene graph, its nodes are traversed downward starting from the root node and the following operations are performed for each link out of the root node:

- The current model view matrix is saved by pushing it onto the matrix stack.
- The model view matrix on the right is multiplied by the transformation matrix associated with the link.
- The drawing procedure is called recursively, pretending that the endpoint of the link is the root.
- The original model view matrix is restored by popping it from the matrix stack.

From the visualization point of view, the main difference between traditional VR-based and modified AR-based scene graphs is the mechanism through which different views of the scene are displayed to the observer. In a purely VR environment, 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 part 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 mobile observers as they walk in the animation. As such, there is only one moving viewpoint attached to an observer's eyes. This means that the only viewpoint in an AR animation which involves a mobile root node is a Six Degree-of-Freedom (6DOF) viewpoint attached to the scene observer's eyes. Users can change their global position (i.e. longitude, latitude, and altitude) while simultaneously rotating their head in 3D global space (i.e. yaw, pitch, and roll angles) (Kamat and Behzadan 2006). As shown in Figure 4, this single viewpoint has a nodal representation and in fact, serves as the root of the scene graph.

5. ANIMATING AND UPDATING THE AUGMENTED 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 parent node. In this research, the process of updating the transformation matrix of an object is done automatically by creating a link between a simulation model and the animation (Section 6). However, the presented AR scene graph has the potential of managing objects that have been previously created in a BIM environment.

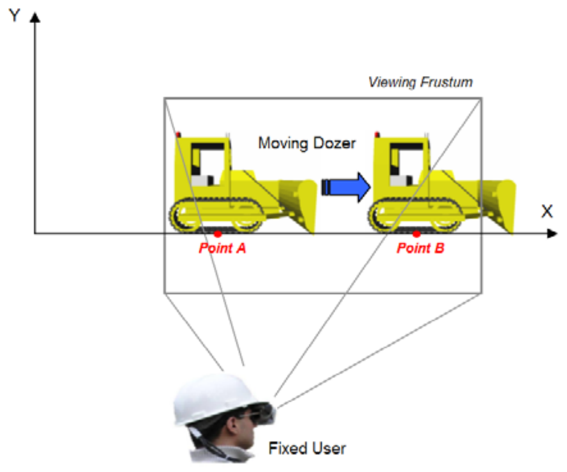
A BIM-based CAD object can be essentially described as a 3D geometry with embedded information that can describe the geometrical parameters and constraints, materials, specifications, timing requirements, code requirements, assembly procedures, prices, manufacturers, vendors and any other related data associated with how the object is actually used. When linked to the AR scene graph, the embedded spatial and timing information in 3D BIM-based CAD objects can be effectively used to update the transformation values at each frame and create a dynamically changing scene. Since this is done continuously over time, a smooth animation is displayed to and viewed by the observer. Figures 6 and 7 show how a virtual CAD model of a dozer (previously introduced in Figure 4) can be manipulated inside the perspective viewing frustum. As described earlier, the relative position of the dozer can be changed due to three main reasons:

1) Dozer moves in the scene from one point to another, or

2) Observer's global position changes and new positional data is captured from the GPS receiver.

3) Observer's head orientation changes and new head orientation data is captured from the head orientation tracker.

Figure 6 shows the situation in which the observer is looking at the augmented scene from a fixed position and the CAD model of the virtual dozer changes position. This situation leads to a simplified AR-based scene graph which is quite similar to the traditional VR-based scene graph of Figure 3 in which the root node is fixed in the global space. As the dozer moves 5 meters along the positive X axis, its global coordinates change as well. Knowing the global position of the user in terms of longitude, latitude, and altitude (coming through the GPS device), the relative distance between the user and the dozer is calculated at each frame using Modified Transformation Method (MTM), a georeferencing method developed by the authors and described in (Behzadan and Kamat 2007-2). 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 observer's viewing frustum. As a result, a moving dozer is displayed inside the augmented viewing frustum. As shown in Figure 6 and recalling the transformation matrices assigned to each CAD object from Figure 5, the dozer's movement along the X axis from point *A* to point *B* will directly affect the translation component of its transformation matrix. In this Figure, T_{dozer}^{user} denotes the transformation matrix of the dozer CAD model inside the local coordinate frame of the user.

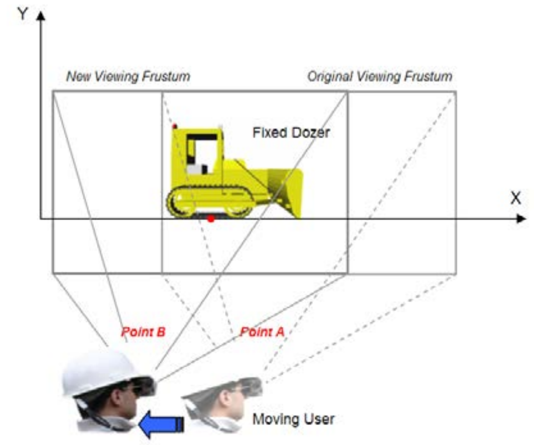


$$T_{dozer}^{user} = \begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & -8 \\ 0 & 0 & 0 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 0 & 0 & 8 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & -8 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

FIG. 6: Animating a Moving CAD Object in an AR Scene with a Fixed User

In another situation shown in Figure 7, the CAD model of the virtual dozer is fixed and the observer changes position over time. As the observers moves 5 meters to the left, global coordinates are continuously acquired from the GPS device. Knowing the global coordinates of the CAD object, the relative distance between the observer and the dozer is calculated at each frame using the MTM method (Behzadan and Kamat 2007-2, Behzadan 2008).

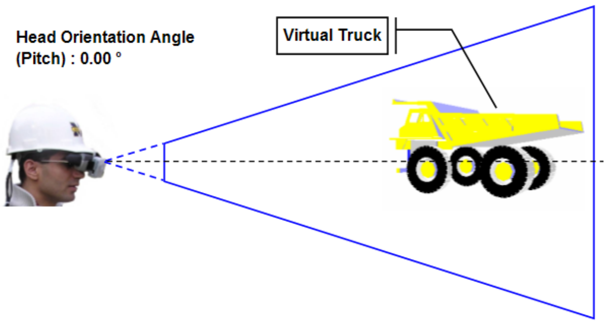
Consequently, the transformation values of the dozer node are updated based on the calculated relative distance and the position of the dozer is modified accordingly inside the viewing frustum. As a result, the observer's move to the left is interpreted as a translation of the dozer to the right. This is essentially equivalent to the case in which the observer is fixed and the dozer moves to the right inside the viewing frustum. As shown in Figure 7 and recalling the transformation matrices assigned to each CAD object from Figure 5, observer's move along the negative X axis from point A to point B by 5 meters will change the translation component of the transformation matrix assigned to the dozer CAD model along positive X axis by 5 meters. In this Figure, T_{dozer}^{user} denotes the transformation matrix of the dozer CAD model inside the local coordinate frame of the user.



$$T_{dozer}^{user} = \begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & -8 \\ 0 & 0 & 0 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 0 & 0 & 8 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & -8 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

FIG. 7: Animating a Fixed CAD Object in an AR Scene with a Moving User

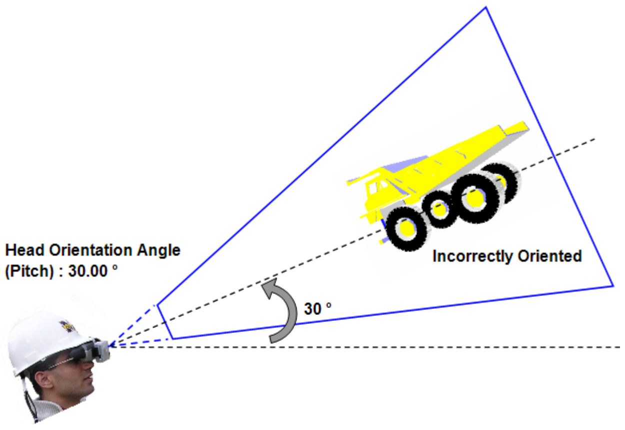
A similar procedure is applied to the case in which the observer's head orientation changes. Figure 8 shows an augmented viewing frustum in which a CAD model of a virtual truck is displayed to the observer. As shown in this Figure and recalling the transformation matrices introduced in Figure 5, the initial orientation of the CAD object relative to the observer's direction of look can be expressed in form of the identity matrix. T_{truck}^{user}



$$T_{truck}^{user} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

FIG. 8: Initial Orientation Configuration of a CAD Object

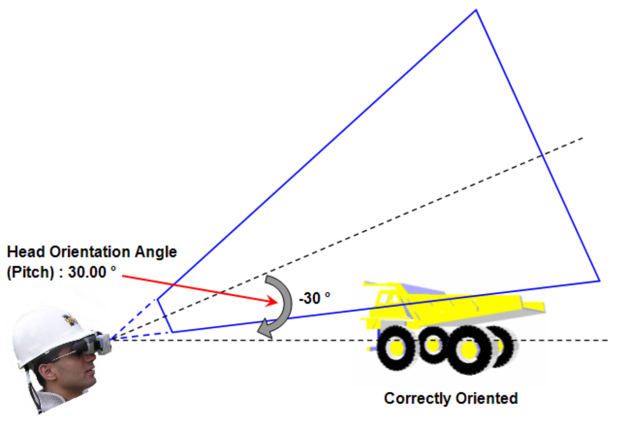
As shown in Figure 9, if the observer's head rotates upward by 30°, the head orientation tracker determines a change in the pitch angle and the new pitch angle is sent to the AR application. Recalling Figure 4, since the observer's eyes are at the root node of the augmented scene graph, the entire scene will be rotated 30° which results in a disoriented scene.



$$T_{truck}^{user} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

FIG. 9: Incorrectly Oriented Augmented Scene due to User's Head Rotation

This leads to a distorted and unacceptable visual output especially since the virtual contents of the scene have remained exactly the same although the observer's view of the real world (which is captured by the video camera) has been completely changed. In order to fix this problem, the T_{truck}^{user} transformation matrix needs to be updated. This is shown in Figure 10 in which the orientation of the virtual truck is modified by -30°.



$$T_{truck}^{user} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & Cos\ -30° & -Sin\ 30° & 0 \\ 0 & Sin\ 30° & Cos\ 30° & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

FIG. 10: Correctly Oriented Augmented Scene due to User's Head Rotation

The new transformation matrix will be used immediately at the next scene graph traversal (i.e. next animation frame) to update the position and orientation of the CAD object inside the observer's viewing frustum. The same steps are necessary in case the observer's head orientation changes about the other two major axis (i.e. roll and yaw angles). The values of a transformation node can be manipulated as a result of simultaneous observer and CAD object movements (i.e. change in position and orientation). In this case, each movement is handled separately using the procedures described above 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.

Scene graphs internally calculate and use bounding volume of each leaf node to determine which CAD objects are visible to the viewer (Woolford 2003). A bounding volume is an imaginary simple 3D primitive (e.g. box, sphere) around a more complex 3D object (Assarsson and Moller 2000). Bounding volumes are calculated from the leaf nodes upwards to the root node. Following this process, the bounding volume of a parent node in a scene graph is obtained by merging the individual bounding volumes of its immediate child nodes. Having calculated and stored all the bounding volumes in a scene graph, the graphics pipeline can then determine which CAD objects or groups of CAD objects are partially or completely outside the viewing frustum of the user and hence have to be culled. The bounding volume of a particular node encapsulates all the geometries described by all nodes that descend from it. As a result, if this bounding volume is outside the viewing frustum the entire subgraph is disregarded (culled) from the rendering stage. In scene graph terminology this is often referred to as pruning (Woolford 2003).

6. AUGMENTED REALITY SCENE GRAPH EXAMPLE

The potential of the presented scene graph structure in creating a robust and scalable platform for animating engineering operations in AR has been successfully validated using the ARVISCOPE visualization system. ARVISCOPE is a high level scripting language, developed by the authors, that allows an external software process such as a running discrete event simulation (DES) model to author a dynamic animation trace file (Behzadan 2008). It has been designed using OpenSceneGraph, a C++ open source toolkit and has video capturing capabilities adapted from OSGART, a separate open source library (Looser et al. 2006). 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 11 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 sequentially 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);
```

FIG. 11: Sample ARVISCOPE Trace File

The construction of an augmented scene starts with extending the hierarchical structure of the corresponding AR-based scene graph by sequentially uploading and placing virtual CAD objects in the augmented view. CAD objects are updated and/or placed in the scene as an animation script (i.e. trace file) is read and interpreted by ARVISCOPE. Figure 12 shows how the structure of an AR-based scene graph is related to the actual contents of the augmented scene. Since leaf nodes contain geometry information, for every class of CAD objects, one leaf node has to be created. One or more CAD objects can then share the same leaf node if they have the same geometry. The scene graph of Figure 12 starts to take real shape as the node `RedTruck` is added to the scene graph and linked to its corresponding geometry through the leaf node `Truck`. In order to place the red concrete truck in the scene, the `RedTruck` transformation node is added to the parent node called `Jobsite`. The same procedure is applied to the scene graph again to create a new leaf node (i.e. `Form`), add an object conforming to this geometry (i.e. `WoodenForm`), and finally add the transformation node of the object to the parent node `Jobsite`. The last virtual object to be placed in the scene is a concrete bucket. Again, a geometry node (i.e. `Bucket`) is added to the scene graph. An object node (i.e. `OrangeBucket`) is added to the scene graph, and connected to the to the `Jobsite` node through a transformation node. In addition, note that the node `Jobsite` itself is a grandchild of the root node.

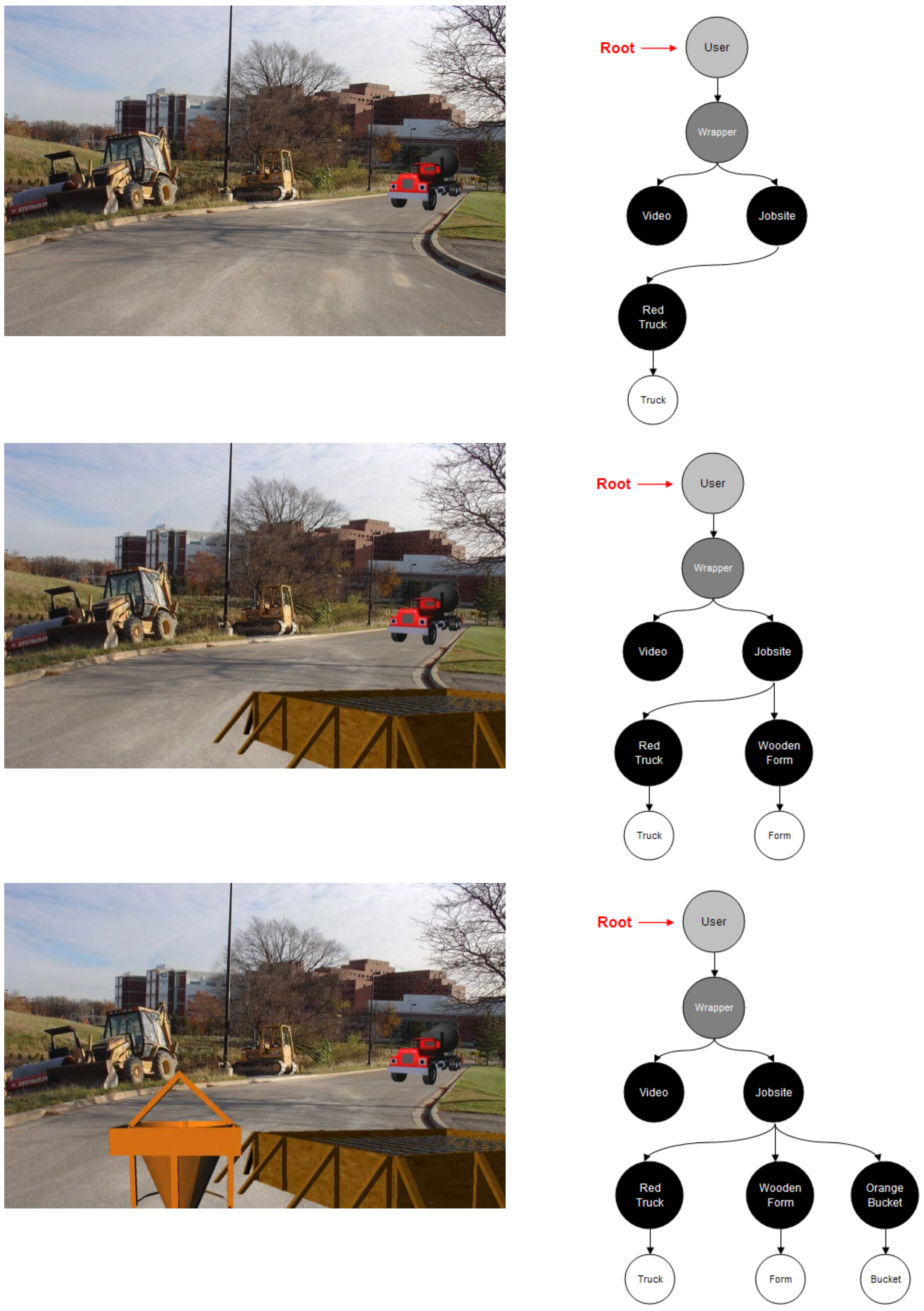


FIG. 12: Relation between the Scene Graph and the Augmented Scene

7. SCALABLE GEOMETRIC INSTANCING

Geometric instancing in the context of this paper is defined as the ability of the visualization engine to construct complex scenes potentially consisting of a large number of CAD objects from a limited number of virtual models and to maintain performance levels as the size of the operation increases (Kamat and Martinez 2002). The common problem arising in almost every large scale visualization system is the amount of data that must be loaded and rendered at each frame. As the size and complexity of the animated environment increases, slow rendering speed becomes the bottleneck for real time user interaction with the scene even if fast CPUs and powerful graphic accelerator cards are used (Zara et al. 2001; Kalra et al. 1998).

Scene graphs support geometric instancing when constructing the graphical contents of an augmented scene. The application of scene graphs allow the modeler to conveniently create all CAD objects in the scene that use the same geometrical representation by making instances of a single CAD file associated with 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 and speeds up the process of scene rendering. Figure 13 shows the AR-based scene graph of a scene in which three identical dump trucks are placed in the augmented view. Although there are four CAD objects present in the scene (i.e. one backhoe and three dump trucks), only two distinct CAD models need to be loaded (i.e. one for truck geometry and one for excavator geometry). In theory, an infinite number of identical dump trucks can be created and placed in the scene using the same geometrical representation of the truck already introduced to the scene graph as a leaf node. Using geometric instancing, even more 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 steel sections, and placing them repeatedly at appropriate locations using multiple transformation nodes. This is due to the capability of the transformation nodes to position, rotate, and scale the same geometry.

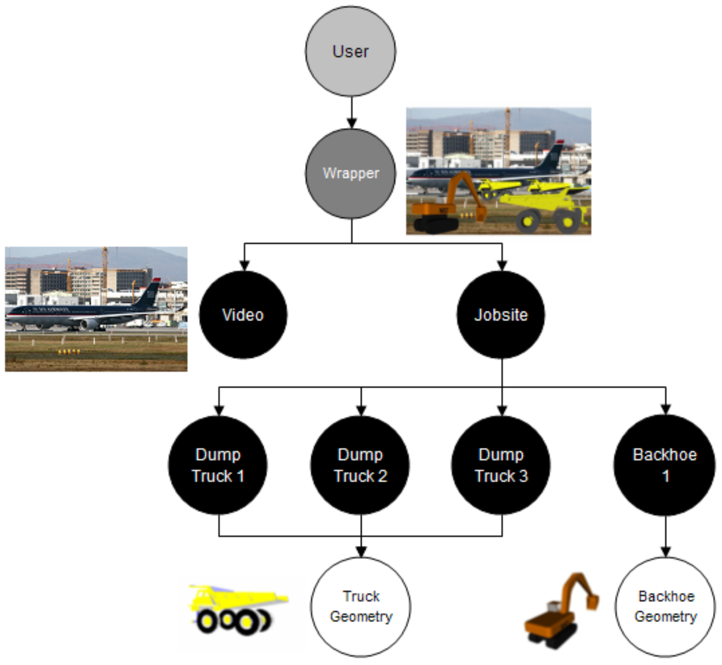


FIG. 13: Using Geometric Instancing in the AR-Based Scene Graph

Figure 14 shows how the scene graph of a large steel structure constructed and populated from a limited number of CAD models. In practice, the beam and column objects shown in Figure 14 can be first created in a BIM environment and linked to the project schedule to create 4D objects. A 4D object consists of a 3D geometry and an extra dimension representing time. Depending on the project requirements, the time dimension can be interpreted as a tag representing the time of an event that involved a 4D object. For the beams and columns shown in Figure 14, the time at which each object is scheduled to be installed can be effectively used as a time tag. The resulting objects will then be imported and used inside the AR animation.

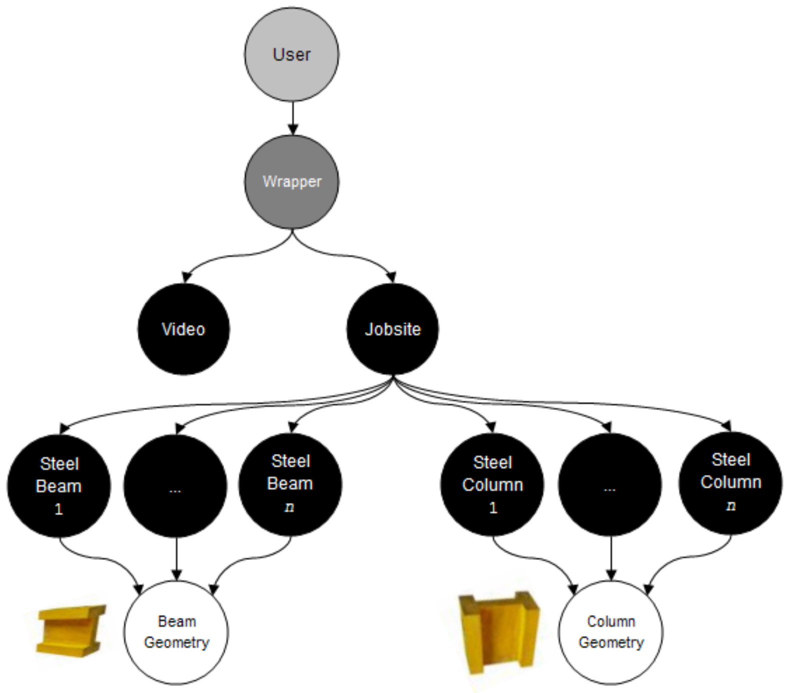


FIG. 14: Creating Large Number of Elements from a Few CAD Models

8. LARGE-SCALE RECONSTRUCTION OF ENGINEERING OPERATIONS IN AR

Creating real time dynamic animations of architecture/engineering/construction (AEC) operations can help decision-makers gain a better insight of planned or ongoing activities by examining the spatial and chronological details of the animated processes from different perspectives and at different operation times. In Construction, AR can be of great potential in visualizing planned activities prior to the actual commitment of resources on the jobsite to determine possible underutilization of certain pieces of equipment such as excavators or cranes. At the same time, by reconstructing a full-scale 3D replica of a project in AR, possible space conflicts can be resolved before real objects are assembled on the project location. The authors conducted a series of outdoor experiments to validate the functionality and robustness of the developed AR-based scene graph technique in precisely reconstructing an actual engineering environment in which virtual 3D models are operating based on a simulation script. In each experiment, the 3D augmented animation was first simulated in a DES tool and the animation trace file and the visualization were done in ARVISCOPE. In addition to assist in project constructability analysis, results of such experiments can be used to capture timing data of different pieces of equipment, and design more time efficient processes that yield to minimum resource idle times and improved productivity. Videos of selected validation experiments can be accessed online at the following URLs:

http://video.google.com/videoplay?docid=-2470337473329635846 (structural steel erection)

http://video.google.com/videoplay?docid=-3793758707076518917 (earthmoving)

http://video.google.com/videoplay?docid=-2361248290439389834 (offshore concrete delivery)

http://video.google.com/videoplay?docid=-2465769261700956727 (traffic intersection)

9. SUMMARY AND CONCLUSIONS

Developing and implementing effective CAD object management methods is a crucial step in creating realistic visual output in AR animations of engineering processes. Virtual objects should be designed and displayed as independent entities (similar to real objects) in an AR scene so that their position, orientation, and size can be easily manipulated during the course of an animation. This is essential in creating the illusion that both groups of objects (real and virtual) coexist in a single scene and operate in a seamless manner. The high variability involved in a dynamic engineering operation requires the graphical engine of the AR application to create, load, and handle a large amount of 3D objects in each animation frame. As the size of the operation increases, CAD object manipulation becomes a memory intensive task that can easily reduce the animation speed and the ability to interact with the animated process in real time.

Establishing a hierarchical database of all CAD objects operating in an augmented scene is a promising approach to optimize the process of manipulation of the virtual contents of an AR animation. Scene graphs have proven to be effective means in organizing such databases and are widely supported by several industrial standards and commercial graphical libraries (Kalra et al. 1998; Shreiner et al. 2004). Updating and maintaining the graphical content of a scene graph, however, requires the development of algorithms and methods specific to the platform on which the AR application is based. Such algorithms must be designed taking several parameters into consideration.

One such important parameter is geometric instancing which is defined in this paper as the ability of the visualization engine to construct complex scenes potentially consisting of a large number of CAD objects from a limited number of virtual models while maintaining performance levels. This is extremely important since most of the time, graphical hardware and CPU processing speed are the primary bottlenecks in creating and displaying animations at acceptable frame rates. Hence, an efficient approach is to create a complex scene consisting of a large number of virtual objects from as few individual CAD models as possible. Scene graphs are very effective tools to achieve this goal since all CAD objects in the scene that use the same geometrical representation can be conveniently created by creating instances of a single CAD file associated with 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 and speeds up the process of scene rendering.

10. ACKNOWLEDGMENTS

The presented work has been supported by the National Science Foundation (NSF) through grant CMS-0448762. The authors gratefully acknowledge NSF’s support. The authors would also like to thank Professor Klaus-Peter Beier at the University of Michigan for his valuable insight and advice, as well as Ph.D. student Ms. Sara Jabbarizadeh for her assistance in conducting some of the experiments. Any opinions, findings, conclusions, and recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the NSF, or the individuals mentioned here.

11. REFERENCES

Ames A, Nadeau D, Moreland J (1997) VRML 2.0 sourcebook. John Wiley and Sons, New York, NY

Assarsson U, Moller T (2000) Optimized view frustum culling algorithms for bounding boxes. Journal of Graphics Tools 5(1):9-22

Barfield W, Caudell T (2001) Fundamentals of wearable computers and augmented reality. Lawrence Erlbaum Associates, London, UK

Behzadan A H (2008) ARVISCOPE: Georeferenced visualization of dynamic construction processes in three-dimensional outdoor augmented reality. Dissertation, University of Michigan

Behzadan A H, Kamat V R (2007-1) Enabling smooth and scalable dynamic 3D visualization of discrete-event construction simulations in outdoor augmented reality. In: Proceedings of the Winter Simulation Conference, Washington, DC, pp 2168-2176`

Behzadan A H, Kamat V R (2007-2) Georeferenced registration of construction graphics in mobile outdoor augmented reality. Journal of Computing in Civil Engineering 21(4):247-258

Behzadan A H, Kamat V R (2007-3) Reusable modular software interfaces for outdoor augmented reality applications in engineering. In: Proceedings of the International Workshop on Computing in Civil Engineering, Pittsburgh, PA, pp 825-837

Behzadan A H, Timm B W, Kamat V R (2008) General purpose modular hardware and software framework for mobile outdoor augmented reality applications in engineering. Journal of Advanced Engineering Informatics 22(1):90-105

Bennett P (2006) National marine electronics association. In: FAQ: Edge of space science. Available via http://www.eoss.org/pubs/nmeafaq.htm. Accessed 5 Dec 2006

Benoit M (2000) Applied XML solutions. SAMS Publishing, Indianapolis, IN

Brown D, Julier S, Baillot Y, Livingston M (2003) An event-based data distribution mechanism for collaborative mobile augmented reality and virtual environments. In: Proceedings of the IEEE Virtual Reality Conference, Los Angeles, CA, pp 23-29

Dollner J, Hinrichs K (1997) Object-oriented 3D modeling, animation, and interaction. Journal of Visualization and Computer Animation 8(1):33-64

Eastman C, Teicholz P, Sacks R, Liston K (2008) BIM handbook: A guide to building information modeling for owners, managers, designers, engineers, and contractors. John Wiley and Sons, New York, NY

Feiner S, MacIntyre B, Hollerer T, Webster A (1997) A touring machine: Prototyping 3D mobile augmented reality systems for exploring the urban environment. In: Proceedings of the International Symposium on Wearable Computing (ISWC'97), Cambridge, MA, pp 74-81

Gleue T, Dähne P (2001) Design and implementation of a mobile device for outdoor augmented reality in the Archeoguide project. In: Proceedings of the 2001 Conference on Virtual Reality, Archeology, and Cultural Heritage, Glyfada, Greece, pp 161-168

Kalra P, Magnenat-Thalmann N, Moccozet L, Sannier G, Aubel A, Thalmann D (1998) Real-time animation of realistic virtual humans. Journal of Computer Graphics and Applications 18(5):42-56

Kamat V R, Behzadan A H (2006) GPS and 3DOF angular tracking for georeferenced registration of construction graphics in outdoor augmented reality. In: Smith I F C (ed) Lecture notes in computer science (intelligent computing in engineering and architecture). Springer, New York, pp 368-375

Kamat V R, Martinez J C (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

Lipman R, Reed K (2000) Using VRML in construction industry applications. In; Proceedings of the Web3D-VRML2000 5th Symposium on Virtual Reality Modeling Language. Monterey, CA, pp 119-124

Livingston M, Rosenblum L, Julier S, Brown D, Baillot Y (2002) An augmented reality system for military operations in urban terrain. In: Proceedings of the Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC '02), Orlando, FL, pp 1-8

Nadeau D R (2000) Volume scene graphs. In: Proceedings of the Symposium on Volume Visualization. Salt Lake City, UT, pp 49-56

OpenSceneGraph Website (2008) Introduction to OpenSceneGraph. Available via http://www.openscenegraph.org/projects/osg/wiki/About/Introduction. Accessed 20 Feb 2008

Piekarski W, Thomas B H (2003) An object-oriented software architecture for 3D mixed reality applications. In: Proceedings of the 2nd International Symposium on Mixed and Augmented Reality (ISMAR), Tokyo, Japan, pp 247-256

Selman D (2006) Java 3D programming. Manning Publications, Greenwich, CT

Shreiner D, Woo M, Neider J, Davis T (2004) OpenGL programming guide. Addison Wesley, Reading, MA

Suthau T, Vetter M, Hassenpflug P, Meinzer H, Hellwich O (2002). A concept work for augmented reality visualization based on a medical application in liver surgery. Technical University Berlin, Commission V, WG V/3

Thomas B, Close B, Donoghue J, Squires J, Bondi P, Morris M, Piekarski W (2000) ARQuake: An outdoor/indoor first person augmented reality application. In: Proceedings of the 4th International Symposium on Wearable Computers (ISWC2000), Atlanta, GA, pp 149-146

Walsh A E, Bourges-Sevenier M (2001) Core Web3D. Prentice Hall, Upper Saddle River, NJ

Woolford D (2003) Understanding and using scene graphs. Available via http://www.itee.uq.edu.au/~comp4201/scene_graphs_dsaw.pdf. Accessed 20 Feb 2008

Zara J, Chromy P, Cizek J, Ghais K, Holub M, Mikes S, Rajnoch J (2001) A scalable approach to visualization of large virtual cities. In: Proceedings of the 5th International Conference on Information Visualization. London, UK, pp 639-644