Enabling 3D Visualization of Simulated Construction Operations

Vineet R. Kamat

Thesis submitted to the Faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Master of Science
In
Civil Engineering

Julio C. Martinez, Chair
Jesus M. de la Garza
Michael C. Vorster

October 2000
Blacksburg, Virginia

Keywords: 3D Visualization, Animation, Simulation, Construction Operations, Computer Graphics, and Scene Graphs

Enabling 3D Visualization of Simulated Construction Operations

Vineet R. Kamat

(ABSTRACT)

Simulation modeling and visualization can substantially help in designing complex construction operations and in making optimal decisions where traditional methods prove ineffective or are unfeasible. However, there has been limited use of simulation in planning construction operations due to the unavailability of appropriate visual communication tools that can provide users with a more realistic and comprehensible feedback from simulation analyses. Visualizing simulated construction operations in 3D can significantly help in establishing the credibility of simulation models. In addition, 3D visualization can provide valuable insight into the subtleties of construction operations that are otherwise non-quantifiable and presentable.

New software development technologies emerge at incredible rates that allow engineers and scientists to create novel, domain-specific applications. This study capitalized on a computer graphics technology based on the concept of the "Scene Graph" to design and implement a general-purpose 3D Visualization System that is Simulation and CAD-software independent. This system, the "Dynamic Construction Visualizer", enables realistic visualization of modeled construction operations and the resulting products in 3D and can be used in conjunction with a wide variety of simulation tools. This thesis describes the "Dynamic Construction Visualizer" as well as the "Scene Graph" architecture and the Frame Updating algorithms used in its design.

ACKNOWLEDGEMENTS

The support of the National Science Foundation (Grant CMS-9733267) to this study is gratefully acknowledged.

I would also like to thank all individuals who have assisted and supported me in this endeavor. The guidance, indulgence, and encouragement of my committee members, Drs. Julio C. Martinez, Jesus M. de la Garza, and Michael C. Vorster is deeply appreciated.

Sincere thanks to Dr. Julio Martinez for being my mentor and constant source of inspiration throughout the journey. Thank you for believing I could succeed and for giving me the opportunity to do so.

Special thanks are due to my cousin, Dr. Manoj Dharwadkar for encouraging me to pursue graduate studies in the United States and for guiding me through all the steps of the process. Thanks for helping me in making the right choice of selecting Virginia Tech to pursue my career objectives.

Finally, I would like to thank my parents and my wife for their unending love, support, and encouragement during my study at Virginia Tech and throughout my life.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

Chapter 1
INTRODUCTION

1.1 PROBLEM STATEMENT

1.1.1 Background

Construction operations range from the relatively simple to the most complex. They involve multiple pieces of equipment, labor trades, and materials that can interact in complex ways. Traditional methods used to design them prove ineffective in many cases where simulation modeling and visualization can be of substantial help.

Numerous advances have been made in the area of construction operations modeling (Martinez & Ioannou, 1999). However, the Visualization/Animation aspect has mainly focused on the design of the construction product (3D CAD) or on the product as it evolves through construction (4D CAD). Very little attention has been given to visualizing the construction operations that lead to the end product, which includes temporary structures and materials, equipment and labor as they create the product. Modeling and subsequently visualizing various construction scenarios can provide significant insights that can be very helpful in effectively planning and scheduling construction operations.

Significant advances have been made in the field of simulation analyses and process visualization of manufacturing systems. Prominent among the commercially available systems are Deneb Robotics' Quest™ and AutoSimulations' AutoMod™. These 3D-enabled simulation systems, although potentially capable of visually depicting construction operations, are intricately linked to their own simulation engines, which are in turn, specifically designed for modeling situations in the manufacturing sector. In addition, the prohibitive cost of such systems makes their use in the construction industry unlikely.

Construction planners are therefore not able to simultaneously obtain both, 1) Advanced construction operations modeling capabilities and 2) 3D visualization. To my knowledge, there presently exists only one process visualization tool that is independent of any particular simulation system. This tool, PROOF™, has been successfully used in the past to visualize many scenarios in the fields of construction engineering and mining (e.g., Martinez, 1998; Sturgul & Seibt, 1999). PROOF™, although effective, inherently lacks in the real world 3D capabilities that are indispensable for the realistic visualization of some complex construction operations.

1.1.2 The Needs of a Construction Process Visualization System

Effective planning and scheduling of construction operations is critical for the success of any construction project, irrespective of its magnitude. As such, the planner's judgment, imagination, intuition, and experience are relied upon heavily to ensure success. This often compels the planner to develop a mental model of the actual construction process and the sequence of various activities and operations involved therein. This can be a demanding task in today's technically advanced construction scenario.

The current state-of–the-art in the area of construction process modeling is capable of providing a valuable construction management tool that is well suited to the analysis of construction operations, majority of which are resource driven. Simulation allows the planner and/or analyst to experiment with and evaluate different scenarios during the planning phase. Despite this fact, the analysis of construction operations using simulation techniques has been generally neglected (Huang & Halpin, 1994). This is largely due to the fact that simulation tools provide the user with a large amount of numerical and statistical data, but are not designed to illustrate the modeled operations graphically. The potential practitioners, i.e. construction planners and analysts, who are typically well versed with the actual construction operations, thus perceive the use of simulation as being a black box not worthy of trust.

The capability to pictorially visualize the planned construction operations in 3-D can greatly facilitate the usability of construction process simulation as an operations

management tool in the construction industry. 3-D visualization of the planned construction operations will provide easier accessibility for the planners by graphically conveying the physical configuration of the system of interest in a very realistic but virtual format. This will allow users to easily and clearly comprehend the dynamic and interrelated behavior of the system model after each simulation run. 3-D visualization will thus establish the credibility of the analyses by facilitating the validation and verification of complex simulation models. This will provide an opportunity to convince all the parties involved that the model indeed reflects the reality, thus favoring the reaching of a common ground.

1.1.3 Benefits

Planning, Scheduling, and Controlling complex construction projects can benefit greatly by integrating traditional project management techniques with construction operations simulation and visualization. 3-D Visualization of the planned and simulated construction operations can greatly enhance the process of constructing facilities as follows:

- Visualization of future construction operations allows decision makers to make prudent decisions without relying on their intuition, instincts, and opinions.
- Visualization facilitates greater awareness and understanding about the project among the management and the site personnel by providing them with a virtual experience of how and what should be constructed.
- Improved and easier detection of errors in scheduling, flaws in design, and constructability problems are facilitated before the actual commencement of the construction operations.
- Construction personnel, management and workers alike, can comprehend the construction process better allowing them to visualize precisely how activities relate to one another, thus reducing conflicting interpretations and communication problems.
- Sequencing of interrelated activities in an operation can be better planned by organizing the resources for maximum productivity.
- Problems with equipment positioning and manpower congestion in certain areas can be visualized prior to the actual operation, thus preventing accidents and

safety problems such as collision between two machines and losses in productivity.

- Visualization of construction operations can serve as a tool in the evaluation of construction claims by providing an improved method for documenting construction activities.
- Quick adjustments and revisions, if any, to the construction schedule are possible by allowing the project management personnel to precisely determine the state of the project by visualizing and comparing the planned and actual construction progress.
- Construction operation designers and planners can take full advantage of advanced process simulation capabilities by first simulating and then visualizing numerous possible "what if" scenarios, thus eliminating costly and risky trial and error techniques.
- Construction activities can easily be viewed on the computer screen using 3-D CAD models, thus eliminating the cumbersome and costly physical models that might be needed to reconstruct certain scenarios.
- 3-D visualization can be used very effectively as a medium for educational training in construction process design and execution.
- Construction simulation model developers can use visualization capabilities for debugging their model and verifying that analytical models indeed behave correctly.

1.2 PURPOSE, SCOPE, AND OBJECTIVES

1.2.1 Purpose

Several CAD systems have been developed and employed in the past for the purpose of visualizing constructed facilities. These systems have mainly focused on the design of the finished product (3D CAD) or on the product as it evolves through construction (4D CAD). 3D CAD systems provide a virtual tour of the completed constructed facility. These systems are essentially used by architectural and design firms in conceiving the design and conveying the details of the design to their clients. In addition, real estate developers utilize 3D CAD systems to woo potential customers by allowing them to

"walk through" their area of interest in the facility. On the other hand, numerous 4D CAD systems were developed with the aim of providing project management tools to construction managers and schedulers. These systems essentially break up the 3D CAD model of the facility into manageable parts such that each part thereof can be suitably assigned or linked to a distinct activity in the construction schedule. The result is the visual simulation of the schedule whereby the parties concerned are able to "see" the facility on the computer screen as it gets built with the passage of time and the completion of the respective activities in the schedule. These types of systems have been used in the past in the resolution of construction contract claims (Morad & Beliveau, 1991, Battikha & Davidson, 1996).

In all these systems, very little attention has been given to visualizing the actual construction operations that lead to the completion of the constructed facility. Visualizing the construction operations involves being able to view on the computer screen, the interaction of the various resources as they build the facility with the passage of time. These resources include, but are not limited to, temporary structures, materials, equipment, and labor as they create the product. Such a system would essentially be a 4D CAD system, but with a construction operations level-of-detail rather than a schedule level-of-detail.

1.2.2 Scope of the Research

This study researched previously conducted studies and the present state-of-the-art in construction product and process visualization. Previously developed visualization applications were examined and their applicability/feasibility was studied. This was accomplished by carrying out an extensive literature survey of the pertinent material.

This study also laid the foundation for the development of a general-purpose Visualization/Animation system that has been specifically designed keeping in mind the unique requirements for the realistic visualization of construction operations. This system has been designed to work in conjunction with a process modeling simulation system, but at the same time, is entirely independent of the simulation software being used to model

the scenario of interest. As such, this system can be used in conjunction with the user's simulation software of choice. The designed 3D visualization system is a Trace File driven system thus allowing its seamless integration with any process modeling software capable of generating text output during simulation runs. A Trace File is an ASCII text file consisting of sequential animation command statements, which are then processed by the system to visually simulate the modeled operations in 3D virtual space. This is being accomplished by using 3D computer models of the involved resources and the constructed facility at various levels of completion. The results are spatially and chronologically accurate, user-controlled 3D animations of the modeled construction operations.

Simulation software tools can generate the required input Trace File automatically. STROBOSCOPE (Martinez, 1996) has been used for all the testing carried out on the system. This system enables visualization of both the construction processes, and the resulting product, in 3D. The tool enables the easy creation of realistic 3D animations using 3D CAD models from supported data file formats. The system was tested using VRML models of all the entities involved. VRML was chosen as a supported file format since it is steadily emerging as the de facto standard for extremely lightweight and portable computer graphics applications. In addition, practically every CAD modeling program can export their data files in VRML format, thus making the designed visualization system independent of any CAD modeling software as well.

The core of the research as far as design was concerned, was the development of a simple yet robust set of animation commands, and the capability to process sequences of these commands adhering to spatial and chronological accuracy. In addition, the designed system was required to provide the user with a capability to navigate effortlessly in 3-D virtual space, so that he/she did not get "lost" in the virtual world.

The effectiveness of the designed system has been tested by applying it to various simulation case studies. The case studies brought to the fore the effectiveness and

advantages of using 3D operations visualization as a valuable construction process design, planning, and control tool.

This research also brought to light the necessary hardware and software capabilities that are required to make such a system practical and feasible. As such, this study has provided the direction for future research in the area.

1.2.3 Specific Objectives

The specific objectives that have been realized by this research are as follows:

- The current scenario in the area of construction operations modeling and visualization was comprehensively studied.
- A 3D Visualization/Animation language to visualize modeled construction operations was designed.
- The 3D Visualization/Animation language was implemented, creating the first version of a general-purpose Visualization/Animation system that is specifically designed for the construction industry. This system is capable of depicting construction operations realistically; is easy to use and is suitable for its intended users; and runs on standard Desktop PCs, eliminating the need for specialized Graphics Workstations.
- Created a system that allows construction operations modelers to capitalize on their previous knowledge by allowing them to take full advantage of the designed system by merely extending the techniques acquired in using simulation software.
- Created the prototype of a useful tool that can be easily used, thus facilitating the transformation of construction process simulation and visualization into an indispensable process-level planning tool.

1.3 RESEARCH METHODOLOGY AND RESULTS OBTAINED

The research methodology mainly comprised the following areas:

1) Literature Review
2) Animation Language Design
3) Implementation

The lessons learnt from the literature review demonstrated the visualization capabilities possessed by currently available systems. It also exposed the shortcomings and limitations of the existing systems. In addition, this study brought to light the qualities necessary for a system with realistic Construction Operations Visualization capabilities and with a capacity to seamlessly integrate with process modeling software. A first version of such a system was designed in the study and applied to classic simulation case studies, which were used to demonstrate the capabilities of the designed system.

1.3.1 Software Development Platform and Tools

The Microsoft Windows operating system was used as the development platform for the implementation part of this research. This family of operating systems was chosen because of its stability, portability, and popularity. Visual C++ 6.0 was used as the software development environment.

The following computer graphics programming libraries were used to accomplish the graphical requirements of the designed system. These libraries are distributed by Silicon Graphics Inc., which is the pioneer and market leader in the area of high performance computer graphics hardware and software.

Cosmo3D

Cosmo3D is a C++ toolkit that brings 3D graphics programming to desktop applications. It speeds up and facilitates the process of creating complex graphics applications. It allows application developers to use a higher-level interface than the lower-level OpenGL language that it is based on. The OpenGL graphics programming language is the industry standard for a software interface to computer graphics hardware. Cosmo3D provides scene-graph architecture support, which allows developers to arrange visible objects in a hierarchical structure. This feature was essential in depicting the realistic motion of complex construction equipment such as dumptrucks and forklifts, as well as that of complex hierarchical assemblies such as cranes and backhoes. Cosmo3D thus facilitates the development of complex graphic applications.

OpenGL Optimizer

OpenGL Optimizer is another C++ toolkit for CAD applications. It enables interactive, robust visualization of large and complex models. It allows developers to optimize the performance of their applications. This tool is therefore used primarily to improve the performance of applications developed using Cosmo3D.

1.3.2 Research Products

The products of this research were the following:

- A comprehensive study about the current applications and the present state-of-the-art in the area of construction process and product visualization.
- A first version of a general-purpose Visualization/Animation system that was specifically designed for the realistic visualization of construction operations.

1.3.3 Applicability of this Research

This study will provide an impetus to the use of simulation modeling as an indispensable operation-level planning tool in the construction industry. Due to its capability of running on standard desktop PCs and laptops, the designed system has potential applications not only in the planning office, but also on the actual construction site.

Due to the flexibility of the command set and its independence from any particular simulation modeling software, the designed system has numerous potential applications in fields other than construction, such as in the manufacturing and service industries.

1.4 CONCLUSION

The sole motivation for using simulation as a project management tool in construction is to obtain insights into the consequences of using different techniques and strategies and thus helping the planner in making the most advantageous decisions. The state-of-the-art construction operations modeling systems provide users with a plethora of information such as statistical production charts, resource usability, and breakdown times in the system being modeled. There were however, no supporting tools that could illustrate the modeled operations in 3D.

3D visualization of the geometrical characteristics of modeled systems, and the dynamic nature of all the entities involved in construction operations can be very helpful in ensuring the success of construction projects. The designed system will provide such visualization support to numerous simulation languages and packages, thus enabling construction planners and designers to get a more realistic and comprehensible feedback from simulation analyses.

1.5 THESIS OUTLINE

This thesis describes the "Dynamic Construction Visualizer (DCV)" system as well as the "Scene Graph" architecture and the Frame Updating algorithms used in its design. All chapters have been written so that they can be read and understood by a wide audience, including people without any prior exposure to programming and 3D graphics. Chapter 2 explains the design objectives of the DCV and explains with an example, the steps necessary to visualize modeled constructions operations in 3D. Chapter 3 provides detailed explanation about the architecture and the algorithms used in the implementation of the DCV language.

Chapters 2 and 3 are intended to be "ready-to-submit", stand-alone publications. These chapters may therefore contain certain repeated information. The appendices contain supplemental information not found in either of the chapters. Appendix A contains summary information about the various DCV language statements. Appendix B presents an Earthmoving simulation model. This model has been instrumented to automatically generate a trace file written in the DCV language during simulation runs. Appendix C presents a portion of the trace file that is automatically generated by the instrumented simulation model in Appendix B. The DCV implementation utilizes the trace files to depict the modeled operations graphically in 3D.

1.6 REFERENCES

Battikha, M.G., and Davidson, C.H. (1996) “Cause and Effect 3-D Model for Measuring Performance in Construction Acceleration: A Decision Support System”, *Building Research and Information, Vol. 24, No.6.*

Huang R., and Halpin, D.W. (1994). “ Visual Construction Operation Simulation: The DISCO Approach”, *Journal of Microcomputers in Civil Engineering, 9(6),* 175-184.

Martinez, J.C. (1996). “STROBOSCOPE: State and Resource Based Simulation of Construction Processes”, Ph.D Dissertation, University of Michigan, Ann Arbor, MI.

Martinez, J. C. (1998). "Earthmover - Simulation Tool for Earthwork Planning", *Proceedings of the 1998 Winter Simulation Conference,* Society for Computer Simulation, San Diego, CA, 1263-1271.

Martinez, J.C., and Ioannou, P.G. (1999). “General Purpose Systems for Effective Construction Simulation”, *Journal of Construction Engineering and Management,* 125(4), ASCE, 265-276.

Morad, A.A., and Beliveau Y.J. (1991) “Knowledge-Based Planning System”, *Journal of Construction Engineering and Management, Vol. 117, No.1,* ASCE.

Silicon Graphics, Inc. (1998) “*Cosmo 3D Programmer’s Guide*”, Mountain View, CA.

Silicon Graphics, Inc. (1998) “OpenGL Optimizer Programmer’s Guide: An Open API for Large-Model Visualization”, Mountain View, CA.

Sturgul, J.R., and Seibt, F. (1999). “A Simulation and Animation Model of the Transport of Coal to the Port", *Proceedings of the 8th Mine Planning and Equipment Selection Symposium,* Balkema, Rotterdam, Holland.

Wolverine Software Corporation (1995). *Using Proof Animation,* 2nd Edn. Annandale, VA.

Chapter 2
VISUALIZING SIMULATED CONSTRUCTION OPERATIONS USING THE DYNAMIC CONSTRUCTION VISUALIZER

2.1 INTRODUCTION

One of the distinct characteristics of the construction industry is the uniqueness of each constructed facility. Products are built by performing numerous construction operations that involve complex interactions between multiple pieces of equipment, labor trades and materials. Experienced construction planners have elaborate thoughts about how they will carry out operations. They mentally evaluate and visualize alternatives that can be quite complex. There is a limit, however, to the number of issues that can be effectively considered and mentally carried out by humans. Simulation modeling and visualization, if capable of representing all pertinent aspects, can go a long way in overcoming this limitation.

2.1.1 Construction Simulation

Construction operations have been modeled for many years using discrete-event simulation (Martinez and Ioannou 1999, Tucker et al. 1998). Simulation is a powerful objective function evaluator that is well suited for the design of construction processes. The state-of-the-art construction simulation systems allow the modeling of complex construction operations in great detail and with utmost flexibility (Martinez and Ioannou 1999). Simulation is thus capable of providing the project manager with detailed information about planned operations such as resource utilization, resource idleness, operation bottlenecks, production rates, and the resulting cost before commencing the actual implementation in the field (Liu and Ioannou 1993). Despite this fact, simulation has not been used to its maximum potential in planning construction operations (Huang and Halpin 1994, Tucker et al. 1998).

This can be largely attributed to the prevalent skepticism among the potential practitioners to trust simulation analyses. Construction planners and analysts, who are typically well versed with the actual construction operations, are reluctant to base their decisions solely on the statistical text and graphical chart output provided by most simulation systems (Ioannou and Martinez 1996). This constitutes what has become known as the "black-box effect" and is a major impediment in validating and verifying simulation models. The resulting lack of credibility hinders the widespread use of simulation in the construction industry.

In addition, typical simulation models do not consider the transformation of space that results from the evolution of the constructed product. The efficient usage of working space has been the subject of many studies that do not involve simulation (e.g. Riley and Tommelein 1996, Akinci and Fischer 1998, Riley 1998). However, the requirement and/or the limitation of working space needed to accomplish the modeled construction operations is not considered in most simulation models. For instance, maneuverability problems at loading and dumping areas in earthmoving operations, the restricted visibility of the crane operator in steel girder erection, overcrowding in particular work zones due to simultaneous execution of different trades in building construction, and a host of other safety problems such as potential collision between two machines cannot be directly deciphered from simulation analyses unless the models are explicitly designed to do this. In addition, the requirements of working space and its usage over the period of construction are unique for every construction project. This prevents analysts from making generalized assumptions regarding space requirements and its management over the period of construction.

The foregoing problems justify the need for support tools that allow decision makers as well as simulation model developers to obtain a more realistic and comprehensible feedback from simulation studies. Visualization of simulated construction operations can be of substantial help in describing the intricacies of simulation models and in obtaining valuable insight into the subtleties of construction operations that are otherwise non-quantifiable and presentable.

2.2 VISUALIZATION OF CONSTRUCTION OPERATIONS

There are several ways to visualize modeled construction operations. The choice depends on the level-of-detail and realism, and on the information and insights to be gleaned from the process. The main purposes of visualization are to establish credibility by enabling model validation and verification, and to provide non-quantifiable and otherwise presentable insights about the modeled system.

2.2.1 Schematic Visualization

Schematic models capable of dynamically displaying associated model information have been used in the past to illustrate simulated construction operations (Huang and Halpin 1994). This technique uses schematic model diagrams and dynamically overlays them during simulation runs with statistical data such as the simulation time, percentage of time resources are idle, and number of resource units idle.

Graphical iconic animation overlaying schematic models has also been used to provide an animated environment in construction simulation systems (Liu and Ioannou 1993, Shi and Zhang 1999). This methodology animates the resource flows during a simulation run by moving icons on a schematic model background. The icons representing model resources change position appropriately on the model background during a simulation run to reflect their flow in the modeled system.

Schematic visualization can be of some help to simulation model developers in debugging models during the development stage. Debugging is an essential step in model verification that determines whether the model contains any modeling flaws or whether it performs as intended by the developer. On the other hand, the aim of validation is to determine whether the simulation model accurately represents the real-world system under study (Law and Kelton 1991). Validation is carried out by consulting people who are intimately familiar with the operations of the actual system. Typically, these domain experts are not simulation modelers and do not understand simulation models unless they are effectively communicated. Schematic visualization has limited capability to communicate simulation models to persons who do not understand the particular

simulation system. In addition, insights into problems that may arise due to shortage of and conflicts in working space are difficult to interpret using dynamic schematic models. Therefore, although schematic visualization is somewhat good at verification, it has little applicability in model validation.

"True" visualization of construction operations involves being able to see the graphical depiction of the modeled operations being carried out in the same way as they would be in the real world. This type of visualization facilitates both model verification and validation. In addition, it also provides subtle visual details about the modeled operations that can be critical in making decisions. Modeled construction operations can be graphically visualized either in two (2D) or three (3D) dimensions. In addition to previously mentioned selection criteria (i.e. the level-of-detail and realism, and the quality of information required), the choice between 2D and 3D visualization also depends on the time and human resources that are available to produce the required visualization components such as the CAD models and the augmented simulation model. The available computing resources might also influence the selection criteria if they are insufficient for obtaining satisfactory performance during 3D visualization.

2.2.2 2D System Animation

2D system visualization tools describe and depict smoothly moving animations of complex modeled systems whose states are constantly changing. PROOF ™ from Wolverine Software is an ASCII text file driven visualization system that supports two-dimensional drawing and animation (Henriksen 1999, Wolverine Software Corporation 1995). Changes in the state of the modeled system are illustrated graphically and usually "to-scale" in one plane of motion by changing the position, shape, and/or color of icons representing resources. PROOF ™ is independent of any particular simulation system. To create an animation, model developers use the software's built-in drawing capabilities to create layouts and specify resource movement paths. The model developer instruments the simulation software to generate the required sequence of animation commands while the simulation runs. PROOF ™ then processes the animation commands to depict the dynamic state of the modeled system with chronological accuracy. PROOF™ has been

effectively used to visually communicate modeled construction (Ioannou and Martinez 1996, Martinez 1998) and mining operations (Sturgul and Seibt 1999) as well as operations in other disciplines.

2D visualization of construction operations is effective in establishing the credibility of many simulation models. However, it inherently lacks in the real-world 3D capabilities that are indispensable for the realistic visualization of many complex construction operations. In addition, due to the degrees of freedom exercised in accomplishing most construction operations, it is not possible to accurately depict entire system configurations in two-dimensional space. For instance, modeled system characteristics such as the gradients of haul routes and other topographical characteristics in earthmoving operations, the vertical movement of resources in elevators and lifts in addition to the activities performed on the ground during building construction, and the accurate motion of complex construction equipment having multiple degrees of freedom cannot be visually comprehended by animating the simulation model in two dimensions.

2.2.3 3D System Animation

Simulation systems enabled to produce 3D graphical output display real-time 3D animation of modeled operations during simulation runs. Examples include Deneb Robotics' Quest ™ and AutoSimulations' AutoMod ™. Some construction operations can be modeled and visualized in 3D using these simulation systems (Tucker et al. 1998). However, these 3D visualization-enabled systems are tied to their own simulation engines based on process interaction. This makes them quite effective for modeling manufacturing systems but not a clear choice for construction. See (Martinez and Ioannou 1999) for a discussion on simulation strategies (Process Interaction vs. Activity Scanning) and their impact on construction operations modeling.

Furthermore, due to the tight coupling between the simulation engines and the built-in visualizers, the use of these systems to model and animate construction operations requires a radical change in the frame of thought of construction model developers (Aloufa 1993, Tucker et al. 1998). In past modeling efforts, the authors decided to use the

tools they were familiar with (STROBOSCOPE & PROOF™) rather than an unfamiliar 3D-enabled simulation system.

Typical manufacturing systems are characterized by sequential process and assembly lines that are fixed in location and geometry. In contrast, construction operations are carried out in a more complex manner. They involve the transformation of space and the evolution of a product. 3D enabled manufacturing simulation systems are thus unable to handle effectively the additional complications introduced by the dramatic changes in the geometry of the construction site as the work progresses (Tucker et al. 1998).

2.2.4 4D CAD

4D CAD focuses on the visualization of the constructed product over the period of its construction by animating the transformation of space over time. This is accomplished by linking a 3D CAD model representing the complete product design and a construction schedule (McKinney et al. 1996). 4D CAD systems provide construction planners significant insights necessary to analyze the project and subsequently plan the required operations at a schedule level-of-detail. However, 4D CAD systems do not enable the visualization of the construction operations that lead to the end product. Visualizing the construction operations involves being able to view the interaction of the various resources as they build the product. These resources include, but are not limited to, temporary structures, materials, equipment, and labor as they create the product. This limitation restricts the use of 4D CAD in analyzing a system at the operation level-of-detail and in planning and designing the required construction processes.

Given the foregoing state of affairs, we wondered about the possible ways to improve the visualization of construction operations and the resulting products. We experienced the need for a generic 3D visualization system that would be capable of realistically depicting modeled construction operations as well as the evolving products in 3D virtual space.

2.2.5 Generic 3D Visualization Tool

The ideal solution should enable the visual communication of the static and dynamic intricacies of simulation models in addition to establishing the credibility of the analysis in its entirety. In addition, the solution should allow modelers to continue using their current tools by being independent of any particular simulation software. This will allow simulation model developers to obtain 3D visualization capabilities without compromising the power, performance, functional suitability, and familiarity of use of their preferred simulation-modeling program.

The unavailability of such a generic visualization tool motivated the authors to embark upon this research endeavor. The first result of this continuing research is a general-purpose 3D visualization system that allows simulation model developers to visualize modeled operations with chronological and spatial accuracy in 3D virtual space. This system, the Dynamic Construction Visualizer (DCV), is independent of any particular simulation-modeling program as well as any CAD modeling software. The remainder of this paper describes the methodology and elucidates the procedure of visualizing simulated construction operations using the DCV.

2.3 THE DYNAMIC CONSTRUCTION VISUALIZER

2.3.1 System Description

The DCV is implemented as a Microsoft Windows application that can process text files written in the DCV language. This language is designed such that it unambiguously describes the visual configuration of modeled systems with the passage of time.

The DCV is implemented as a “post-simulation” visualization engine that possesses the following characteristics:

- Maintains an independent simulation clock, the speed of which can be controlled by the viewer depending upon the animation speed desired.
- Allows the user to navigate easily in the 3D virtual space and place himself/herself at any desired vantage point.

- Allows the user to jump ahead or back to any desired location in the simulation by specifying a future or past time value.
- Permits the viewer to start and pause the animation at any time to make static observations in the modeled system as desired.

2.3.2 Design Approach

The fundamental purpose of this research was to design a visualization support language that could allow simulation model developers to unambiguously convey the essence of their simulation models. The implementation of this language would enable the verification and validation of simulated construction operations, and would help establish credibility. In addition, it would provide valuable visual insights into the modeled system that are difficult to be quantified and depicted numerically or by any other form of visualization.

The primary design consideration in the development of the DCV was its independence from any particular simulation modeling software. The "post-processed" visualization methodology adopted by PROOF™ provided a beneficial launch pad to the authors in approaching the design of the DCV. The text file-driven approach incorporated in PROOF™ suggested the possibility of developing an independent 3D animation language that could unambiguously depict the static and dynamic characteristics of simulated systems.

The DCV language was designed keeping in mind the unique requirements of visualizing complex construction operations. Using the DCV, modeled operations are visualized in 3D virtual space by processing sequential, time-ordered animation commands written in the DCV language. The animation commands are contained in an ASCII text file hereinafter referred to as the animation trace file. Although the syntax of the DCV language is not very complex, the manual creation of trace files is not envisioned. DCV trace files are meant to be generated by simulation software. Any simulation software capable of writing text output during a simulation run can generate the trace files

automatically. These include most of the programmable generic and special-purpose simulation languages as well as high-level programming languages such as BASIC, FORTRAN, C and C++. Non-language based simulation software may also be adapted to generate trace files during a simulation run (Wolverine Software Corporation 1995).

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

Table 2-1 Summary of Selected DCV Animation Language Commands

STATEMENT	FUCTIONALITY
TIME	Indicates the simulation time at which all subsequent commands take place.
CLASS	Associates a class of simulation entities with their geometric description contained in a CAD file.
CREATE	Creates specific simulation objects by instantiating predefined classes.
PLACE	Places simulation objects at particular locations or at the beginning of resource movement paths.
MOVE	Moves simulation objects on resource movement paths.

Table 2-1 lists a few key commands incorporated in the DCV animation language and provides a concise explanation of their functionality. Figure 2-1 presents a portion of a sample trace file written in the DCV language.

```
PATH LoadToDump (5.2,2,0) (2,2.4,3) (-1.2,2,4);
CLASS Truck A30C.wrl;
TIME 0.00;
CREATE Truck1 Truck;
PLACE Truck1 ON LoadToDump;
TIME 12.00;
MOVE Truck1 LoadToDump 24.00;
```

Figure 2-1 Portion of a Sample Trace File written in the DCV Language

The results of processing the above commands can be explained by referring to table 2-1 and perusing the trace file segment in figure 2-1. A path, “LoadToDump”, is defined by specifying the beginning, ending, and all intermediate 3D coordinates of the points constituting the path. The example defines a path consisting of two segments. Next, a class, “Truck” is defined by the CAD file A30C.wrl. The file A30C.wrl contains the 3D model of a truck in VRML format. As soon as the animation commences (i.e. at time 0.00), one truck conforming to the geometry defined by class “Truck” is created and placed at the beginning of path “LoadToDump”. Further processing of the trace file stream is suspended until simulation time 12.00 is reached. At this time “Truck1” will start moving on the path “LoadToDump” and will require 24.00 simulation time units to complete the journey. During the animation, the ratio of simulated time to viewing time, known as the viewing ratio, is maintained at a user-specified constant value. For example, with a viewing ratio of 6, the animation will show “Truck1” at the beginning of path “LoadToDump” for 2 seconds and then moving along the path for 4 seconds.

Another characteristic that guided the development of the DCV was the capability of the visualization engine to function on standard desktop and laptop PCs running Microsoft Windows operating systems. This ability was determined to be of significance because we wanted the DCV to be accessible to a wide audience. Although the DCV will yield satisfactory results on standard machines without the need for any special hardware, it will take advantage of extra computing power and special graphics accelerators for increased performance.

The next section illustrates, with the help of an example, the steps necessary to visualize a simulated operation using the Dynamic Construction Visualizer.

2.4 VISUALIZING A SIMULATED OPERATION

Earthwork operations involve the excavation, transportation and placement or disposal of materials. In typical earthmoving operations, small improvements in planning result in substantial cost and time savings because they involve repetitive work cycles, expensive fleets and large volumes of work. The study and improvement of earthwork operations has been the focus of many studies (Martinez 1998, Smith et al. 2000) and thus represents a classic example for demonstrating a new technology such as the DCV.

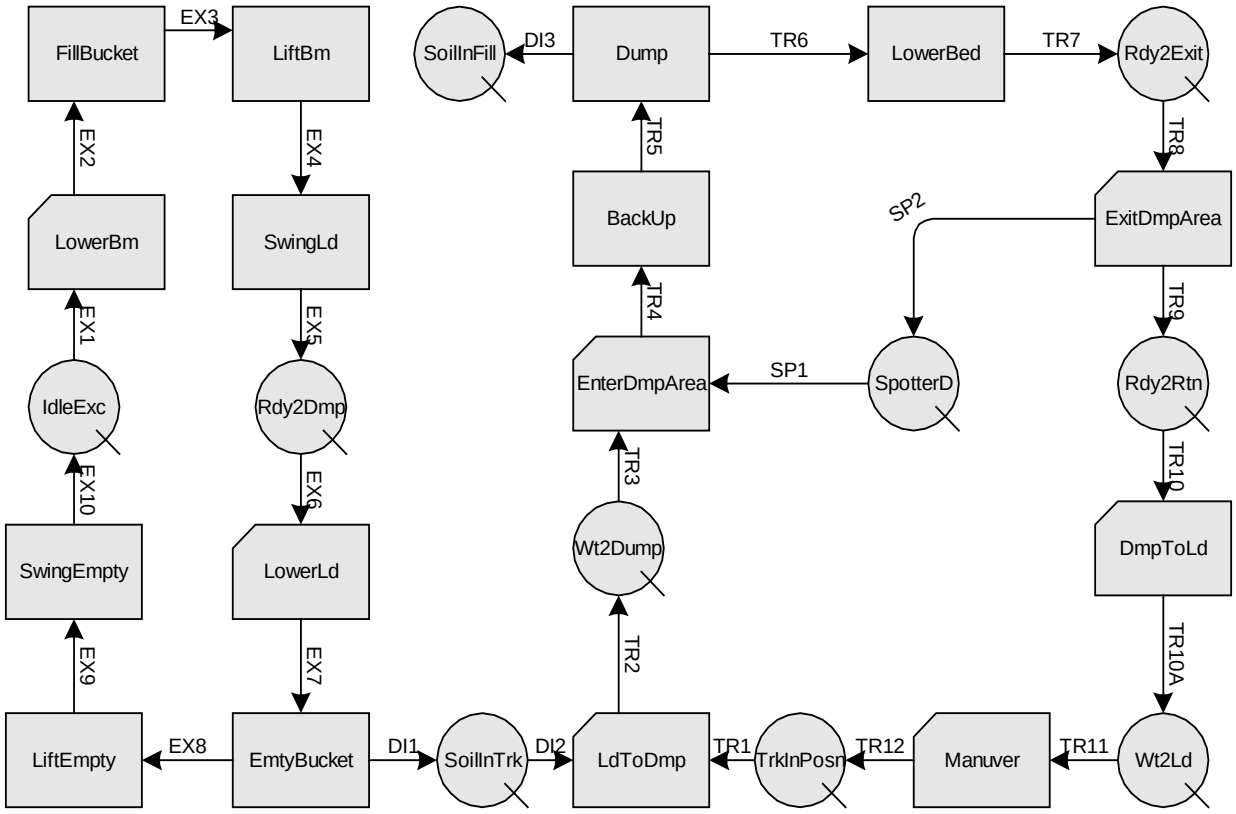


Figure 2-2 Schematic Model Diagram for an Earthmoving Operation

2.4.1 Modeling The Operation

Figure 2-2 shows the schematic model diagram of an Earthmoving operation in STROBOSCOPE format. STROBOSCOPE (Martinez 1996) is a programmable and

extensible simulation system designed for modeling complex construction operations in detail and for the development of special-purpose simulation tools. The illustrated operation in figure 2-2 is essentially self-explanatory and consists of typical Load-Haul-Dump-Return cycles involving the transport of dirt from the source to the destination. The intention of this paper is to describe the process of visualizing modeled operations and not to describe the process of modeling itself or to design earthwork operations using discrete-event simulation. As such, the productivity data and the system parameters of the model such as the volume of work, required productivity, available equipment and steps required to maximize productivity are excluded from the discussion.

Simulation models need to be instrumented to generate DCV animation commands during a simulation run. The statement below, for example, tells STROBOSCOPE to add two lines of text to the DCV trace file every time a truck starts to enter the dump area.

```
ONSTART EnterDmpArea PRINT ATF
"TIME %.2f\059
MOVE Truck%.0f EnterDmpArea %.2f\059\n"
SimTime EnterDmpArea.Truck.ResNum EnterDmpArea.Duration;
```

These two lines will be written to the DCV trace file numerous times, each of which will look similar to:

```
TIME 353.86;
MOVE Truck1 EnterDmpArea 50.21;
```

The time, truck number, and the duration to enter the dump area will of course be different each time. The DCV trace file will contain other lines of text that will be written out when other parts of the modeled operation take place. Thus, the time-ordered sequence of animation statements written out by all the activities in the model during a simulation run constitutes the trace file required to visualize the modeled operations. Figure 2-3 shows a larger portion of the DCV trace file that could be generated during a simulation run.

```
TIME 542.53;
MOVE Truck2 BackUp 24.14;
TIME 547.49;
ROTATE ExcBucket VERT 90 4.38;
TIME 551.87;
ROTATE Stick VERT -20 4.25;
ROTATE Boom VERT 35 4.25;
TIME 551.87;
MOVE Truck4 LdToDmp 115.57;
TIME 551.87;
MOVE Truck5 Maneuver 21.01;
TIME 556.12;
ROTATE ExcCabin HOR -90 9.13;
TIME 565.25;
ROTATE Boom VERT -27 4.34;
ROTATE Stick VERT -10 4.34;
TIME 566.67;
ROTATE TruckBed2 VERT 55 9.81;
TIME 569.59;
ROTATE ExcBucket VERT -90 7.78;
TIME 576.48;
ROTATE TruckBed2 VERT -55 5.47;
TIME 577.37;
ROTATE Boom VERT 27 4.62;
ROTATE Stick VERT 10 4.62;
TIME 581.95;
MOVE Truck2 ExitDmpArea 17.67;
TIME 581.99;
ROTATE ExcCabin HOR 90 12.54;
TIME 599.62;
MOVE Truck2 DmpToLd 109.01;
```

Figure 2-3 Larger Portion of a Trace File written in the DCV Language

The size of the generated trace files depends on the amount of detail modeled and the length of the simulation. The size of typical trace files will vary from a few hundred lines for simple models to several thousand lines for detailed and complex models that simulate operations over long periods of time. The trace file for visualizing 8 hours of the discussed earthwork operation is, for example, 14700 lines long. The STROBOSCOPE model required 18 animation-specific commands to do this. There is no limit on the size of the trace files that can be processed and is therefore not a constraining issue.

2.4.2 Developing / Selecting 3D Computer Models of Simulation Objects

3D computer models in VRML format of all the system entities and the involved resources need to be imported to visualize the simulated operations. For the earthmoving operation discussed here, the 3D models included the terrain, the dumptrucks, and the excavator. As stated previously, the required models can be created with any 3D modeling software capable of exporting their data files in VRML format. A library of 3D models of commonly used construction equipment and other machines can be created and the models reused in various visualizations. In addition, it is possible to purchase such libraries from CAD model vendors (Lipman and Reed 2000).

To create and/or modify the 3D models required to visualize this particular operation, the authors used an easy-to-use and inexpensive 3D modeler named AC3D (Colebourne 2000). The models were created or modified adhering to a level-of-detail that would optimize the 3D rendering process without sacrificing the acceptable realism required to visually communicate the intricacies of the simulated operations.

2.4.3 Visualization Results and Performance Attained

The trace file generated by the simulation model is processed sequentially to visualize the modeled operations. In this particular earthmoving operation visualization, the user is able to observe the accumulating trucks waiting to be loaded, the trucks maneuvering to get into position under the excavator, the excavator digging the earth and loading the trucks until they are full, the trucks traveling to the dumpsite, accumulating occasionally to enter the dump spot, backing up and tipping their load, and then returning to the loading site to begin another cycle. The viewer is able to navigate to any desired position within the 3D virtual jobsite using keyboard keys to steer. Figures 2-4 and 2-5 present animation snapshots displaying the users' view from vantage points near the loading area and the dumpsite respectively. The snapshots presented are not successive computer frames observed during visualization. The discretely captured frames are displayed in a filmstrip format merely to depict a sense of motion. Portions of the Earthmoving visualization explained above are available as video clips accompanying this document from the Virginia Tech Electronic Thesis and Dissertation web site (http://etd.vt.edu).

Figure 2-4 Animation Snapshots depicting the User's View from a Vantage Point near the Loading Area

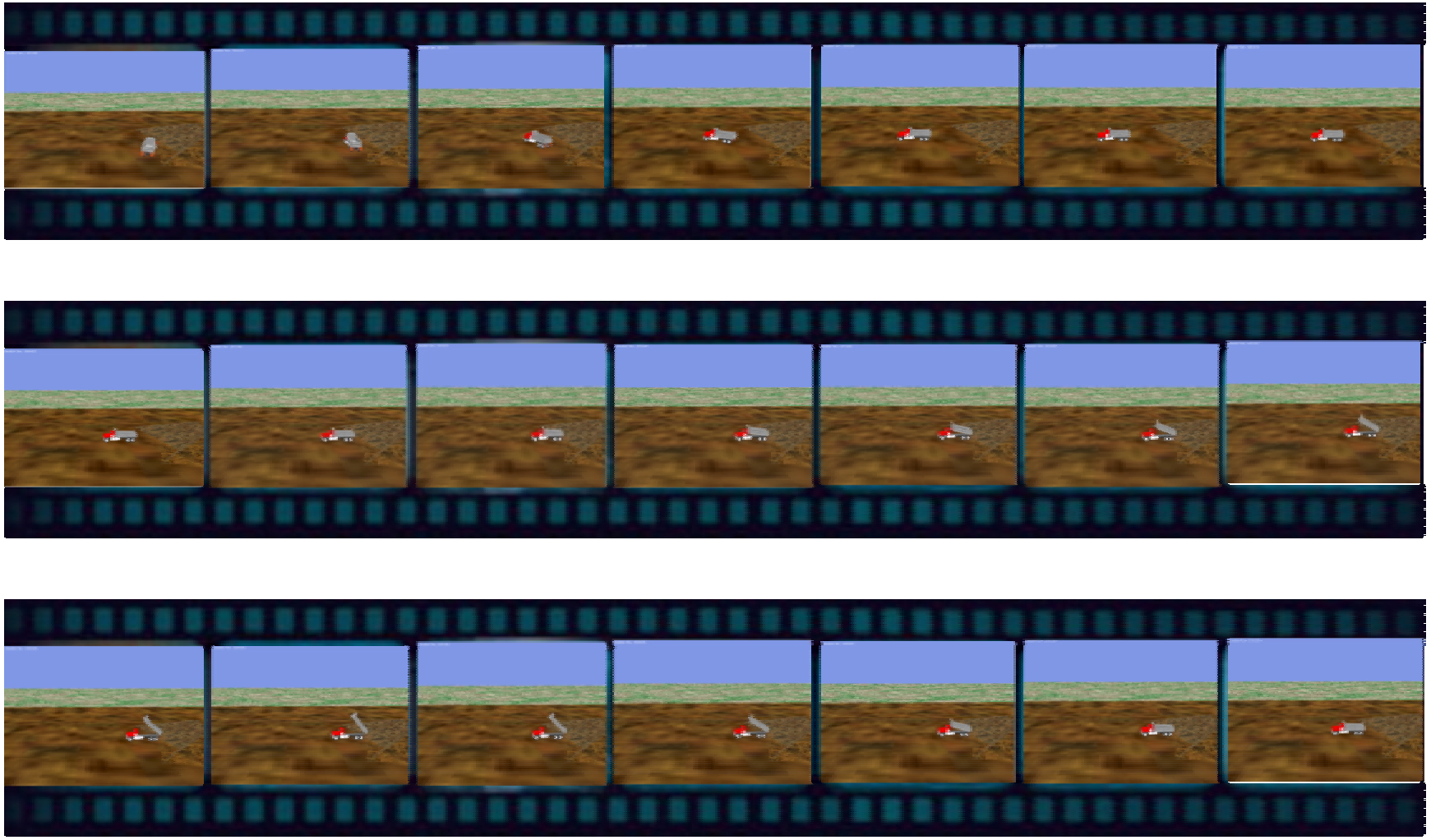


Figure 2-5 Animation Snapshots depicting the User's View from a Vantage Point near the Dumpsite

On a standard laptop computer powered by a 300 MHz Pentium processor, a satisfactory screen refresh rate of 15 frames per second was obtained without the use of any additional specialized graphics hardware. Any increase in the available computing power resulted in a proportional improvement in rendering performance and frame rate. For instance, on a desktop featuring a 600 MHz Pentium III processor, 128 megabytes of RAM, and a Nvidia TNT2 graphics card with 16 megabytes of video memory, a consistent full-screen performance in excess of 60 frames per second was observed. The current version of the DCV visualization engine does not employ any performance improvement techniques such as balancing the graphics pipeline, and optimizing rendering. This makes the performances enumerated above particularly impressive.

2.5 FUTURE RESEARCH

Thus far, the focus of the research has been to develop a methodology to define and depict the realistic motion of construction equipment such as dumptrucks, excavators, cranes, and backhoes, necessary to visualize the performance of construction operations. The capability to depict the constructed facility at various stages of completion as construction progresses is accomplished by “assembling” the facility from discrete components such as CMUs and gypsum boards. This procedure makes it impractical to model all the operations for an entire construction project.

In addition, the current version of the DCV does not support the physical deformation of 3D CAD objects within the visualization. As such, it is not possible to depict certain physical transformations induced by some construction operations. Examples of such transformations include the deformation of the terrain as an excavator digs the earth, the flow of concrete from a concrete pump into forms, and the bending of rebar and cutting of blocks in building construction.

2.6 CONCLUSION

The purpose of using simulation to design construction operations is to obtain insights into alternate designs and thus helping the planner make the most advantageous decisions. Construction process modeling systems provide users with a plethora of

information such as statistical production charts, resource usability, and breakdown times in the system being modeled. There are however, no suitable supporting tools that can graphically illustrate the modeled processes and the resulting products in 3D.

The approach adopted in implementing the DCV language is to allow construction planners to analyze a system at the operation level-of-detail, and plan and design the required construction operations. The designed operations and the resulting product can be subsequently visualized and redesigned as necessary at any desired level of abstraction. The realistic and comprehensible feedback obtained by visualizing modeled operations will greatly improve the accessibility of simulation as an analysis tool in construction.

2.7 REFERENCES

Akinci, B., and Fischer, M. (1998). "Time-Space Conflict Analysis Based on 4D Production Models", *Proceedings of the International Computing Congress on Computing in Civil Engineering,* ASCE, Reston, VA, 342-353.

Aloufa, A.A. (1993). "Modeling and Simulation of Construction Operations", *Automation in Construction,* 1, 351-359.

Colebourne, A. (2000). "AC3D Modeler", *http://www.comp.lancs.ac.uk /computing /users/andy/ac3d.html.*

Griffis, F.H., and Sturts, C.S. (2000). "FIAPP and the Three-dimensional Computer Model", *Proceedings of the 6th Construction Congress,* ASCE, Reston, VA, 996-1000.

Henriksen, J.O. (1999), "General-purpose Concurrent and Post-processed Animation with PROOF™", *Proceedings of the 1999 Winter Simulation Conference*, Society for Computer Simulation, San Diego, CA, 176-181.

Huang R., and Halpin, D.W. (1994). " Visual Construction Operation Simulation: The DISCO Approach", *Journal of Microcomputers in Civil Engineering, 9(6),* 175-184.

Ioannou, P.G., and Martinez, J. (1996). "Animation of Complex Construction Simulation Models", *Proceedings of the 3rd Congress on Computing in Civil Engineering*, ASCE, Reston, VA, 620-626.

Law, A.M., and Kelton, W.D. (1991). *Simulation Modeling and Analysis*, 2nd Edn. McGraw-Hill, New York, NY.

Lipman, R., and Reed, K. (2000). "Using VRML In Construction Industry Applications", *Presented at the Web3D-VRML 2000 Symposium*, Available: http://cic.nist.gov/vrml

Liu, L.Y., and Ioannou, P.G. (1993), "Graphical Resource-based Object-oriented Simulation for Construction Process Planning", *Proceedings Of the 5th International Conference on Computing in Civil and Building Engineering*, ASCE, Reston, VA, 1390-1397.

Martinez, J.C. (1996). "STROBOSCOPE: State and Resource Based Simulation of Construction Processes", Ph.D Dissertation, University of Michigan, Ann Arbor, MI.

Martinez, J. C. (1998). "Earthmover - Simulation Tool for Earthwork Planning", *Proceedings of the 1998 Winter Simulation Conference*, Society for Computer Simulation, San Diego, CA, 1263-1271.

Martinez, J.C., and Ioannou, P.G. (1999). "General Purpose Systems for Effective Construction Simulation", *Journal of Construction Engineering and Management*, 125(4), ASCE, 265-276.

McKinney K., Kim J., Fischer M., and Howard C. (1996). "Interactive 4D-CAD", *Proceedings of the 3rd Congress on Computing in Civil Engineering*, ASCE, Reston, VA, 383-389.

Riley, D.R. (1998). "4D Space Planning Specification Development for Construction Work Spaces", *Proceedings of the International Computing Congress on Computing in Civil Engineering*, ASCE, Reston, VA, 354-363.

Riley, D.R., and Tommelein, I.D. (1996). "Space Planning Tools For Multi-story Construction", *Proceedings of the 3rd Congress on Computing in Civil Engineering*, ASCE, Reston, VA, 718-724.

Shi, J.J., and Zhang, H. (1999). “Iconic Animation of Construction Simulation”, *Proceedings of the 1999 Winter Simulation Conference,* Society for Computer Simulation, San Diego, CA, 992-997.

Slaughter, S. (2000). “The Link Between Design and Process: Simulation Modeling of Construction Activities”, *Proceedings of the 6th Construction Congress,* ASCE, Reston, VA, 1051-1057.

Smith, S.D., Wood, G.S., and Gould, M. (2000). “A New Earthworks Estimating Methodology”, *Construction Management and Economics,* 18(2), 219-228.

Sturgul, J.R., and Seibt, F. (1999). “A Simulation and Animation Model of the Transport of Coal to the Port", *Proceedings of the 8th Mine Planning and Equipment Selection Symposium,* Balkema, Rotterdam, Holland.

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

Wolverine Software Corporation (1995). *Using Proof Animation,* 2nd Edn. Annandale, VA.

Chapter 3
SCENE GRAPH AND FRAME UPDATE ALGORITHMS FOR SMOOTH AND SCALABLE 3D VISUALIZATION OF SIMULATED CONSTRUCTION OPERATIONS

3.1 INTRODUCTION

Construction operations have been modeled for many years using discrete-event simulation (Huang and Halpin 1994, Martinez and Ioannou 1999, Tucker et al. 1998). Discrete-event simulation is a powerful objective function evaluator that is well suited to the design of construction operations. However, decision makers often do not have the training or the time to check the validity of simulation models and thus have little confidence in the results (Ioannou and Martinez 1996). This is largely due to the fact that construction simulation tools provide users with a large amount of numerical and statistical data, but are not designed to illustrate the modeled operations graphically.

The capability to visualize modeled construction operations in 3D can be of substantial help in describing the intricacies of simulation models and in obtaining valuable insight into the subtleties of construction operations that are otherwise non-quantifiable and presentable. 3D visualization can establish the *credibility* of simulation analyses by facilitating the *validation* and *verification* of complex simulation models, thus providing an opportunity to convince all parties involved that models indeed reflect reality.

3.1.1 Background

Numerous researchers have attempted to visualize construction in 3D. The most popular technique focuses on the visualization of the evolving construction product over the period of its construction by animating the transformation of space over time. This technique, referred to as 4D CAD (McKinney et al. 1996), is accomplished by linking a

3D CAD model representing the complete product design and a construction schedule. 4D CAD provides construction planners with significant insights necessary to analyze the project and subsequently plan the required operations at a schedule level-of-detail but not at an operations level-of -detail.

Op den Bosch (1994) capitalized on the prevalent use of CAD during the design phases of projects to design and implement an operations-visualizing application, the Interactive Visualizer Plus Plus (IV++). The IV++ is a virtual environment application for visualizing the process of assembling buildings. Construction procedures, assembly instructions, and other relevant information such as the initial location and orientation of components on the jobsite are associated with each geometric primitive to produce a complete CAD design of the structure requiring assembly. When this augmented CAD file is opened in the IV++ environment, a planning algorithm deciphers the assembly instructions associated with the geometric primitives and transfers them to virtual machines that can be selected and placed in the environment. The “intelligent” machines then assemble the individual geometric primitives to generate the complete structure.

Another 3D operations-visualizing tool, the PlantSpace ® Dynamic Animator ™ from Bentley Systems (1998) allows users to define and move assemblies and individual objects interactively and record the events into scenarios. Recorded scenarios can be manipulated by adding or modifying recorded events, changing their order in a scenario, and editing motion equations to control speed, distance, and acceleration. Combining multiple pre-recorded scenarios then creates animated sequences with several actions occurring simultaneously. Interference analyses can then be performed to detect potential clashes in space and time.

However, none of the available 3D visualization tools are able to visualize construction operations modeled using discrete-event construction simulation systems. 3D visualization of modeled construction operations involves being able to see the graphical depiction of the operations being carried out with the same logical and physical relationships that are embedded in the underlying simulation models.

Model *verification* is the process of determining whether a model accurately reflects the model developer's idea of the existing or proposed system. On the other hand, the aim of *validation* is to determine whether simulation models accurately represent the real-world system under study. Validation is carried out by consulting people who are intimately familiar with the operations of the actual system, but who are not necessarily proficient with modeling tools. Simulation models are termed as credible when the models and their results are accepted as being valid, and are used as an aid in making decisions (Law and Kelton 1991). 3D visualization facilitates both model verification and validation and establishes the credibility of simulation analyses. In addition, it provides subtle visual details about the modeled operations that can be critical in making decisions.

Modeled construction operations have been visualized in several ways in the past. Schematic models capable of dynamically displaying associated model information (Huang and Halpin 1994) and graphical iconic animation (Liu and Ioannou 1993, Shi and Zhang 1999) have been used to illustrate simulated construction operations. Although schematic visualization is somewhat good at model verification it has little applicability in model validation.

2D system visualization tools such as PROOF™ (Henriksen 1999, Wolverine Software Corporation 1995) have been effectively used to visually communicate some modeled construction (Ioannou and Martinez 1996, Martinez 1998) and mining operations (Sturgul and Seibt 1999) as well as operations in other disciplines. PROOF™ graphically illustrates changes in the state of modeled systems in one plane of motion by changing the position, shape, and/or color of icons representing resources. However, although 2D visualization is effective in establishing the credibility of many simulation models, it inherently lacks in the real-world 3D capabilities that are indispensable for the realistic visualization of many complex construction operations.

Some manufacturing simulation systems display real-time 3D animations of modeled operations during simulation runs. Examples of such systems include Deneb Robotics'

Quest™ (Dassault Systemes 2000) and AutoSimulations' AutoMod™ (AutoSimulations Inc. 2000). Some construction operations can be modeled and visualized in 3D using these simulation systems (Tucker et al. 1998). However, in all these systems, the simulation engines are tightly coupled with their built-in visualizers. This compels model developers who desire to visualize their models in 3D to learn and use a different simulation tool than the one they are proficient with. For instance, a GPSS (Schriber 1995) user intending to visualize his/her models in 3D would have to entirely recreate the models in a different 3D-enabled simulation system.

Furthermore, these 3D visualization-enabled systems are tied to their own simulation engines based on *process interaction*. This makes them quite effective for modeling manufacturing systems but not a clear choice for construction. The use of these systems to model and animate construction operations requires a radical change in the frame of thought of construction model developers (Aloufa 1993, Tucker et al. 1998). Simulation strategies (Process Interaction vs. Activity Scanning) and their impact on construction operations modeling are discussed in detail in the literature (Martinez and Ioannou 1999).

Given the current state of affairs, the authors experienced the need for a generic 3D visualization system that would be capable of realistically depicting modeled construction operations as well as the evolving products in 3D virtual space.

3.1.2 The Initiative

New software development technologies emerge at incredible rates, which allow engineers and scientists to create domain-specific applications that far surpass previous attempts. The design of a generic 3D construction operations visualizer presents numerous interesting challenges. Construction sites are highly and at times, unpredictably dynamic. Construction products are built by performing numerous operations that involve complex interactions between multiple pieces of equipment, labor trades, and materials. In selecting suitable high-level computer graphics tools to approach the design of a 3D construction visualization system, the authors saw most promise in a computer graphics technology based on the concept of the Scene Graph. Scene Graph APIs (Application

Programming Interfaces) facilitate and speed up the process of creating complex, domain-specific 3D graphics applications.

The authors capitalized on Scene Graph technology to design and implement a general-purpose 3D visualization/animation system, the Dynamic Construction Visualizer, that enables realistic visualization of modeled construction operations and the resulting products. This paper describes the Scene Graph architecture and the Frame Updating algorithms used in the design of the Dynamic Construction Visualizer.

3.2 THE SCENE GRAPH

3.2.1 Overview

Conceptually, a 3D computer graphics scene can be created in one of two ways. A scene can either be modeled as a single unit or can be "assembled" from discrete components. Modeling the scene as a single unit in a modeling package involves creating and placing geometrical objects at appropriate positions and with appropriate orientations in the scene using the package's built-in tools. The scene entities so created and placed are fixed in position and orientation and can be manipulated within the design environment of the modeling package. Such static scene models are primarily used for CAD type applications where it is usually necessary to render, visualize, and examine large data sets ranging from individual components, to subassemblies, to entire complex layouts.

On the other hand, scenes "assembled" from discrete components within domain specific applications allow the components to be dynamically manipulated (positioned, oriented, and scaled) within the scene. Such control over individual scene components is essential for animation. The individual scene components can be modeled separately and independently using various modeling packages or can be obtained from disparate sources such as CAD model vendors. Scene Graph technology speeds up and facilitates the creation of domain-specific scene assembling and manipulating applications that can use components modeled with various modeling packages and CAD file formats. The afforded flexibility and usability of the "assembling" utilities (i.e. Scene Graph APIs) and of the applications created therewith increase with their ability to use components created

within different modeling packages such as 3D Studio™ (.3ds), AutoCAD™ (.dxf), MicroStation™ (.dwg), and VRML (.wrl). This capability motivated the authors to explore the possibility of using Scene Graph technology in designing the Dynamic Construction Visualizer.

3.2.2 Scene Graph Characteristics

Scene Graph APIs allow applications to assemble and manipulate scenes in a hierarchical data structure of objects called *nodes* that can be arranged in a directed acyclic graph (DAG) structure. This hierarchical structure is called a Scene Graph. 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. Each node encapsulates the semantics of what is to be drawn but not how it is to be drawn. The user creates one or more scene subgraphs and attaches them to a virtual universe, often referred to as the *root node*. The individual connections between scene graph nodes always represent a directed relationship (i.e. parent to child).

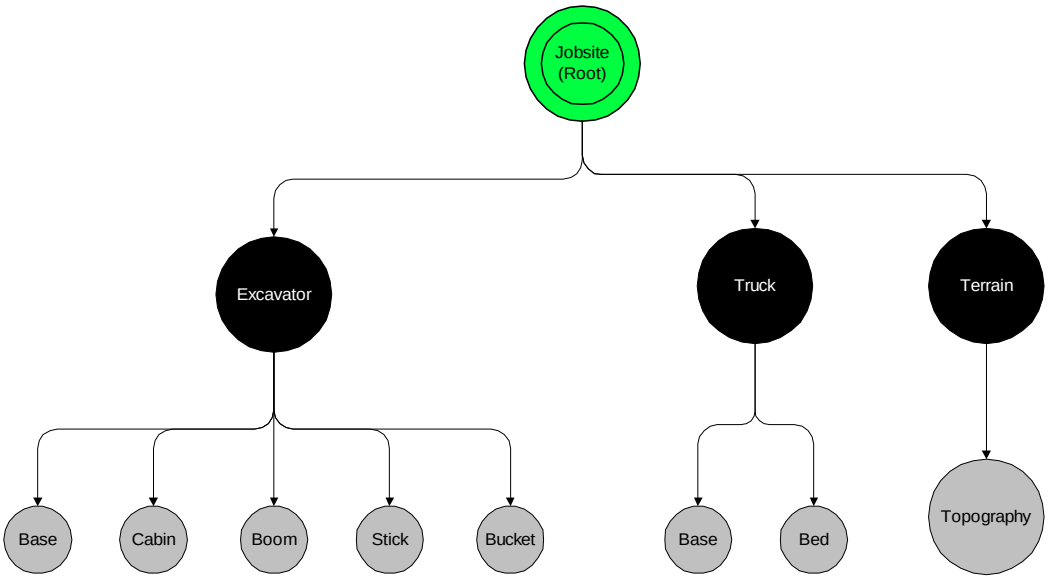


Figure 3-1 Group and Leaf Nodes in a Scene Graph Hierarchy

Figure 3-1 presents the structure of a simple scene graph. Scene Graph nodes can be divided into two subclasses: Group and Leaf nodes. Group nodes assemble together one or more child nodes. A group node can point to one or more children but can have only

one parent. Leaf nodes contain the actual definitions of shapes (geometry), lights, fog, sounds, and so forth. A leaf node has no children.

In figure 3-1, the node “Jobsite” is the root node of the scene graph and is a group type node. Scene subgraphs are created and attached to the root node to completely encapsulate the entire jobsite in the scene graph. For instance, in figure 3-1, subgraphs rooted at nodes “Excavator”, “Truck”, and “Terrain” are created and attached to the root node of the scene graph. The nodes “Excavator”, “Truck”, and “Terrain” are group type nodes. The node “Excavator” groups together all the nodes that completely describe an excavator (i.e. the base, the cabin, the boom, the stick, and the bucket). The nodes “Base”, “Cabin”, “Boom”, “Stick”, and “Bucket” contain the geometrical description of the individual excavator components and are leaf type nodes. These individual excavator component models can be imported from any CAD modeling package whose file format is supported by the Scene Graph API’s geometry loader. In fact, different components may be modeled in different CAD modeling packages. For instance, in the scene graph in figure 3-1, the base and the cabin of the excavator may have been modeled in AutoCAD ™ (in .dxf file format) and the other digging components in 3D Studio™ (in .3ds file format). Similarly, the group node “Truck” groups together the leaf nodes “Base” and “Bed” to completely encapsulate the geometrical description of a truck and add it to the scene. The group node “Terrain” consists of only one child node “Topography” that is a leaf node and contains the geometrical description of the jobsite terrain.

Scene graph nodes hold only data. They encapsulate the semantics of what is to be drawn but not how it is to be drawn. An *action* is an operation that can be carried out on one or more nodes in a scene graph. Actions trigger the events that are prescribed in the scene graph nodes. An action visits each node in the scene graph (i.e. traverses the scene graph) and uses the data contained in each node to display, modify or augment the state of the scene graph. Examples of actions include rendering the scene (drawing), and intersection testing (collision detection). Although actions are a part of the Scene Graph API’s functionality, they do not have a nodal representation. The behavior of an action on a node is a function of both the action and the node. For instance, if a drawing action were

applied to the root node “Jobsite” of the scene graph in figure 3-1, the action would operate on all of the nodes in the scene graph and render all the visible objects (i.e. the terrain, an excavator, and a truck) contained therein. The hierarchy of the scene graph determines the order in which the nodes are acted upon when an action is applied to the scene graph. This hierarchy is established by the order in which the group nodes are added to the sub-branches in a scene graph branch.

3.2.3 Creating Scene Graphs

The coordinate system of the root node in a scene graph is known as the *world space* and is the principal frame of reference. This three-dimensional system is the basis for defining and locating in space all objects in a scene, including the observer’s position and line of sight. On the other hand, a *local space* is used to define the geometry of an object (i.e. size, location, and orientation) independently of the world space. This is done to define geometrical objects independently without giving them fixed specific sizes, locations, and orientations in the world space. Each scene component imported from a CAD modeling package is defined in its own local space.

A scene is created by appropriately sizing and placing geometrical objects (each created and defined in their own local space) at appropriate positions and orientations in the world space. This is accomplished using transformation nodes. Transformation nodes allow scene graph builders to set and manipulate the location (translation), rotation, and scale of their child nodes. Transformation nodes are group-type nodes that translate the local coordinates of their child nodes into the coordinates of their parent nodes. If there is more than one transformation node in a hierarchy of nodes, each transformation node translates the coordinates of its children into the coordinates of its parents all the way up the hierarchy until a final transformation node translates the coordinates of geometrical objects into those of the root node (i.e. world space). Typically, transformation nodes are placed between geometrical object nodes or group-type nodes and the rest of the scene graph. For instance, in figure 3-1, the group type nodes “Excavator”, “Truck”, and “Terrain” need to be transformation nodes in order to be able to place and orient the scene components at desired positions in the scene “Jobsite”.

3.2.4 Visualizing Scene Graphs

A Scene Graph stores the description of a scene by encapsulating relevant data in a hierarchy of suitably arranged nodes. Visualizing a scene encapsulated in a scene graph involves obtaining a suitable view of the hierarchy from any desired viewpoint. Scene Graph APIs provide several utilities to position and orient a viewpoint within the world space. The viewpoint in the world space is analogous to a video camera whose image is continuously transmitted to the viewport (rectangular window on the computer screen) through which a viewer views the scene. The position and orientation of a viewpoint (camera) can be dynamically manipulated within scene graph applications to obtain different views of the same scene graph. Conversely, more than one viewpoint (camera) can be positioned and oriented in the world space to simultaneously transmit different views of the same scene graph to different viewports (windows).

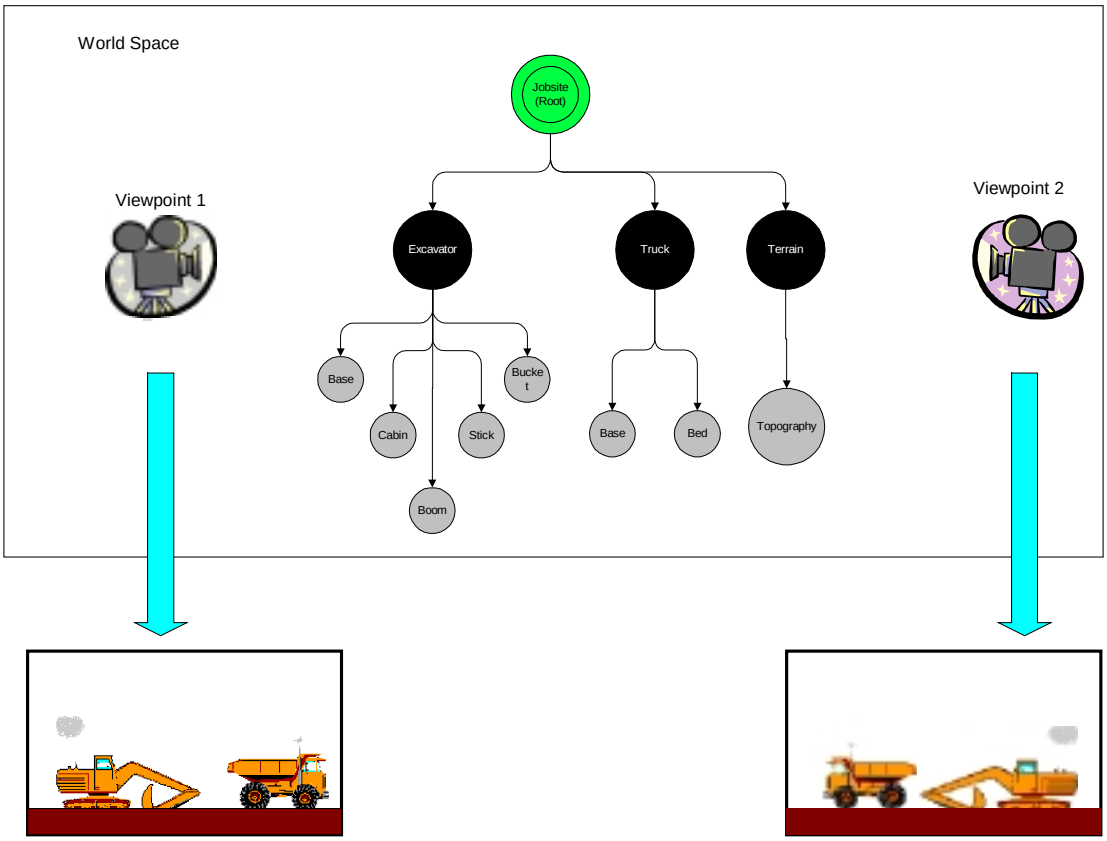


Figure 3-2 Role of the Camera

Although cameras are positioned and manipulated in the world space, they are not a part of the scene graph and hence do not have a nodal representation. They are external mechanisms for visualizing the data encapsulated in the scene graph. Figure 3-2 graphically summarizes the relationship between scene graphs and cameras. Viewpoints placed at different positions in the world space provide different views of the same scene graph.

3.2.5 Animating Scene Graphs

Scene graphs are constructed by developing an appropriate hierarchical node structure of individual scene components that are scaled, placed, and oriented at desired positions in the world space by suitably setting transformation node fields. Depicting the motion of scene entities (i.e. animation) involves a dynamic relationship between scene graph components. Such a relationship is achieved by dynamically manipulating the *fields* (values) of the transformation nodes in the scene graph.

For instance, assume that the truck in the scene graph in figure 3-1 is placed at point (0,0,0) in the world space by setting the translation field of its transformation node ("Truck") to (0,0,0). Imagine that it is required to move this truck along the positive X-axis (in world space) in discrete steps at a rate of 1 unit per second. In order to achieve this motion, the X-axis component of the translation field in the transformation node must be incremented by 1 unit every second. For example, the value of the translation field would be (1,0,0), (2,0,0), and (3,0,0) at the end of one, two, and three seconds. The motion (i.e. discrete jumping action) of the truck described above is schematically represented in figure 3-3. The figure shows the changing position of the truck along the X-axis with the passage of time. At all times, camera(s) transmit view(s) of the current state of the scene graph to the appropriate viewport(s).

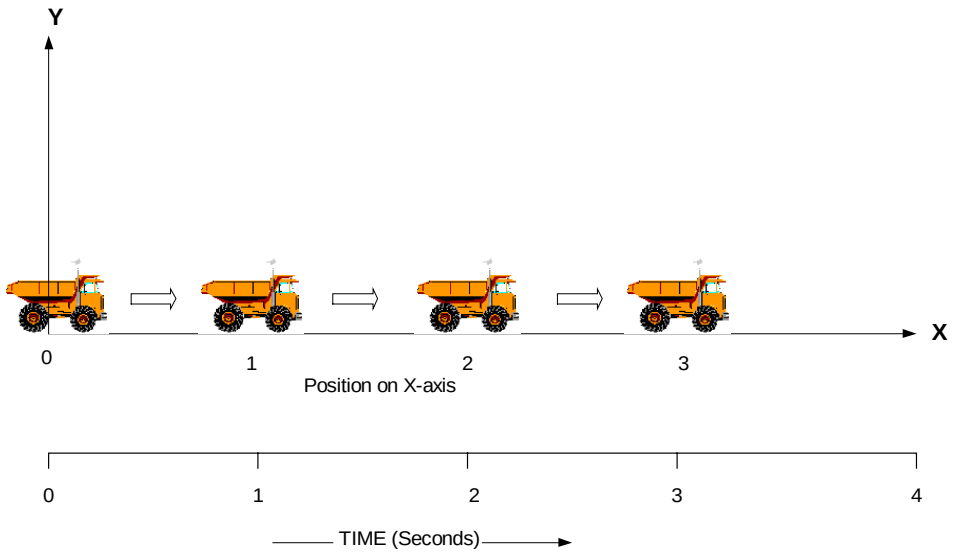


Figure 3-3 Discrete-Step Motion of an Object along the Positive X-Axis

In this example, the truck would appear at point (0,0,0) for 1 second, instantaneously jump to point (1,0,0) and stay there for another second and so on. However, realistic animation requires that objects move smoothly between discrete points with the passage of animation time. This requires constant monitoring and updating of all moving objects in a scene graph. The Frame Updating algorithms that are used to accomplish this are explained in the following section.

3.3 FRAME UPDATING MECHANISMS

3.3.1 Fixed Frame Rate

In movie theaters, motion is achieved by projecting a sequence of pictures at a rate of 24 per second on the screen. Although viewers watch 24 different frames each second, the human brain blends them into a smooth animation. In fact, most modern projectors display each picture twice at a rate of 48 per second to reduce flickering. Typical computer graphics screens redraw the picture (refresh) approximately 60 to 76 times per second (Woo et al. 1997). High-end graphics workstations can refresh up to about 120 times per second. Refresh rates higher than 120 per second are beyond the point of diminishing returns, since the human eye is only so good.

Computer animation generally involves the computing and updating of the positions and orientations of all the dynamic scene objects before the frame can actually be drawn. Imagine that a constant frame rate of 60 per second is desired during computer animation. To obtain a constant frame rate, a routine similar to the one displayed in figure 3-4 will need to be implemented.

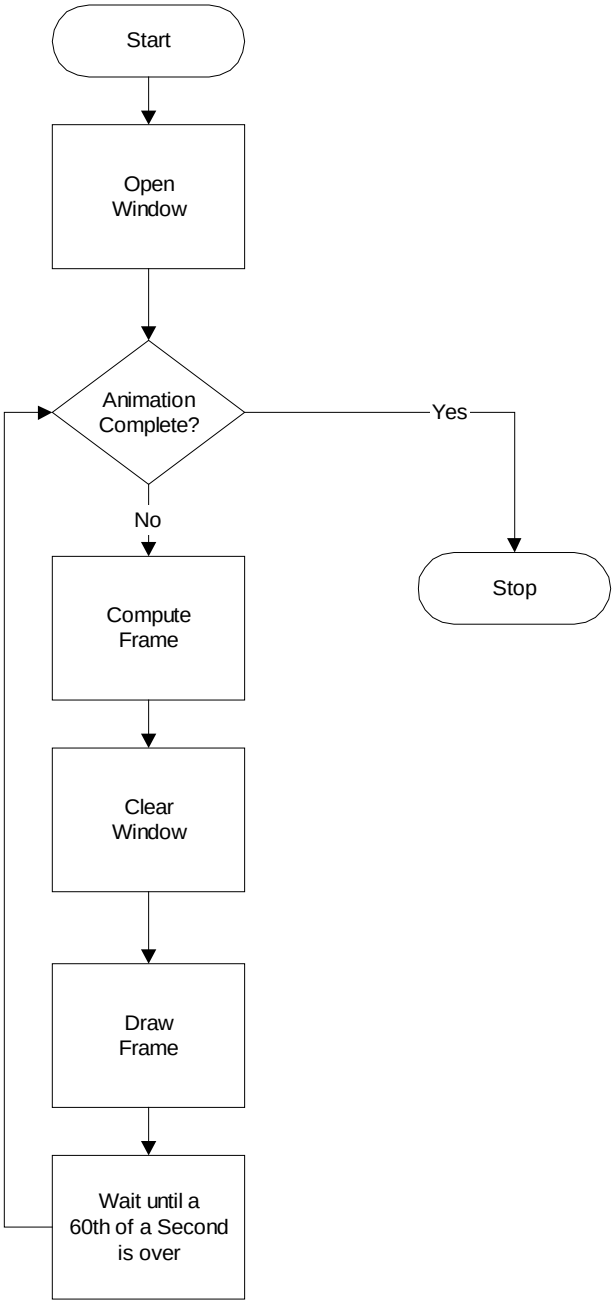


Figure 3-4 Fixed Frame-Rate Algorithm

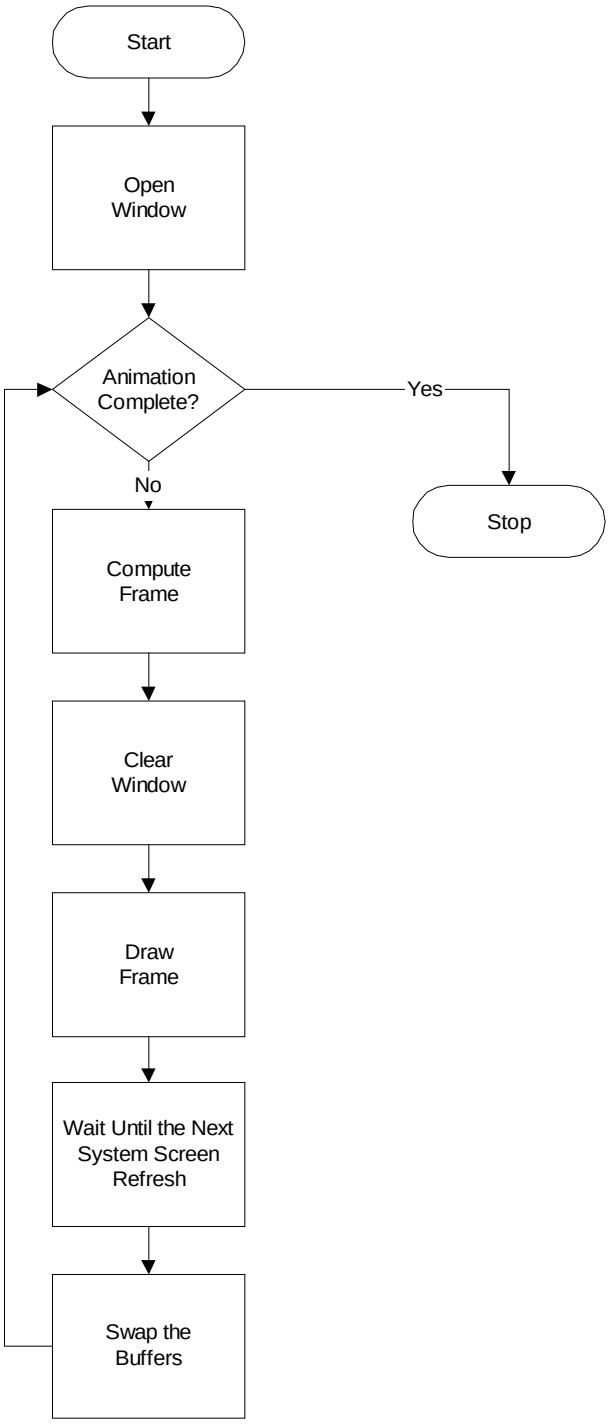


Figure 3-5 – Double-Buffering Algorithm

The critical component in this arrangement is the sum of the times it takes for the system to compute a typical frame and draw it after clearing the screen. The animation results

will degrade progressively depending on how close to 1/60 second it takes to compute, clear and draw. In the limiting case, when the drawing takes nearly a full 1/60 second, scene objects that are drawn first will be visible for the full 1/60 second and present a solid image on the screen. However, scene objects drawn toward the end will be instantly cleared as the system starts on the next frame. This problem is compounded when the time required to draw an entire frame exceeds 1/60 second. In this case, the system will clear the screen for the next frame even before all scene objects in the current frame are drawn, causing unpredictable and distorted images. In this arrangement, the system does not display completely drawn frames. Instead, the viewer watches the drawing as it happens.

3.3.2 Double Buffering

To alleviate the problem of displaying partially drawn frames, most graphics library implementations provide an arrangement known as double buffering. Double buffering allows the system to maintain two screen images at all times. One of the screen images is displayed while the other is being drawn. When the drawing of a frame is complete, the two images are swapped, so the one that was being displayed is now used for drawing, and vice versa. With double buffering, every frame is displayed only when the drawing is complete and the viewer never sees a partially drawn image. This improved arrangement that displays smoothly animated graphics is reflected in figure 3-5.

3.3.3 Variable Frame Rate

In most double buffering implementations, the swapping of the buffers is generally synchronized with the systems' screen refresh rate. This procedure allows the previous buffer as well as the new buffer to be displayed completely, starting from the beginning (Woo et al. 1997). The implication of implementing such a procedure is that the fastest achievable frame rate in an animation is equal to the systems' screen refresh rate. For instance, on a system that refreshes the display 60 times per second, the fastest achievable frame rate is 60 frames per second (fps). If all the frames can be computed, cleared and drawn in under 1/60 second, an animation will run smoothly at that rate.

However, since 3D computer graphics are typically computationally expensive, more often than not typical frames are too complex to be computed and drawn between the screen refreshes of a standard machine. This delay in displaying the new frame results in the previous frame being displayed more than once. For instance, on the same 60-refreshes-per-second system, if it takes 1/40 second to draw a frame, the animation runs at 30 fps, and the graphics are idle for 1/30-1/40 = 1/120 second per frame. Since a system's screen refresh rate is constant, the obtained frame rate in animations is a multiple of the screen refresh rate. For instance, on the 1/60 second per refresh monitor, the obtainable frame rates are 60 fps, 30 fps, 20 fps, 15 fps, and so on (i.e. 60/1, 60/2, 60/3, 60/4, and so on). The obtainable frame rate is inversely proportional to the scene complexity in the animation (i.e. the frame rate degrades proportionately as new scene objects and features are added to the scene).

3.4 THE DYNAMIC CONSTRUCTION VISUALIZER

The Dynamic Construction Visualizer (DCV) is general-purpose 3D visualization/animation system that combines scene graph technology and an effective double-buffering frame-updating algorithm. The DCV allows simulation model developers to visualize modeled operations with chronological and spatial accuracy in 3D virtual space. The system is independent of any particular simulation-modeling program or CAD modeling software.

3.4.1 System Description

DCV language files unambiguously describe the visual configuration of modeled systems with the passage of time. The DCV is as a "post-simulation" visualization engine that possesses the following characteristics:

- Uses Scene Graphs to organize and depict construction jobsite scenarios.
- Allows the user to navigate easily in the 3D virtual space and place himself/herself at any desired vantage point by controlling the camera using the keyboard or the mouse.
- Maintains an independent simulation clock, the speed of which can be controlled by the viewer depending upon the animation speed desired.

- Allows the user to jump ahead or back to any desired location in the simulation by specifying a future or past time value.
- Permits the viewer to start and pause the animation at any time to make static observations in the modeled system.

The DCV language allows the construction and manipulation of complex scene graphs. Modeled operations are visualized in 3D by processing sequential, time-ordered animation commands written in the DCV language. The animation commands are contained in an ASCII text file hereinafter referred to as the trace file. DCV trace files are meant to be generated by simulation software. Any simulation software capable of writing custom text output during a simulation run can generate the trace files automatically. These include most of the programmable generic and special-purpose simulation languages as well as high-level programming languages such as BASIC, FORTRAN, C and C++. Non-language based simulation software may also be adapted to generate trace files during a simulation run (Wolverine Software Corporation 1995).

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

3.4.2 Scene Graphs and the DCV

The DCV has been designed using the Cosmo3D Scene Graph API. Cosmo3D (Silicon Graphics 1998) is a C++ toolkit that brings 3D graphics programming to desktop applications. Cosmo3D facilitates the development of complex graphic applications by allowing application developers to use a higher-level interface than the lower-level OpenGL language that it is based on. The scene graph architecture in Cosmo3D allows developers to arrange and manipulate visible objects in a hierarchical structure that

facilitates the depiction of the realistic motion of complex construction equipment such as dumptrucks and forklifts, as well as that of complex hierarchical assemblies such as cranes and backhoes.

The DCV language is a high-level language that allows users to use simple text statements (commands) to construct and manipulate complex scene graphs. Table 3-1 lists a few of the DCV commands and provides a concise explanation of their functionality. Figure 3-6 presents a sample DCV trace file. Figure 3-7 shows how a scene graph evolves as the statements in the trace file in figure 3-6 are processed.

Table 3-1 Selected DCV Animation Language Commands and their Functionality

STATEMENT	FUCTIONALITY
TIME	Indicates the simulation time at which all subsequent commands take place.
CLASS	Associates a class of simulation entities with their geometric description contained in a CAD file.
CREATE	Creates specific simulation objects by instantiating predefined classes.
ATTACH	Attaches specific simulation objects to one another.
PLACE	Places simulation objects at particular locations or at the beginning of resource movement paths.
MOVE	Moves simulation objects on resource movement paths.
ROTATE	Appropriately manipulates the orientations of specific simulation objects.

The root node of the scene graph is created immediately when the trace file is opened in the DCV application. Two paths, “LoadToDump” and “DumpToLoad”, are defined by specifying the beginning, ending, and all intermediate 3D coordinates of the points constituting the paths. The example defines paths consisting of two segments each. Paths do not have a nodal representation and hence are not part of the scene graph. They are

used to compute fields for transforms that place objects along the path. Defined paths are stored into memory by the DCV application until an animation is complete.

```
                PATH LoadToDump (3,2,1)(0,1,5)(-5,0,3);
                PATH DumpToLoad (-4,0,3)(1,1,5)(2,2,1);

                CLASS Terrain Terrain.wrl;
(A)             CREATE ExTerrain Terrain;
                PLACE ExTerrain AT (0,0,0);

                CLASS Excavator EX1100.wrl;
(B)             CREATE Excvtr1 Excavator;
                PLACE Excvtr1 AT (5,2,1);

                CLASS Truck A30C.wrl;
(C)             CREATE Truck1 Truck;
                PLACE Truck1 ON LoadToDump;

                CREATE Truck2 Truck;
(D)             PLACE Truck2 ON DumpToLoad;
```

Figure 3-6 Sample DCV Trace File to Illustrate the Construction of a Scene Graph

The next 3 statements in the trace file grouped as (A) augment the scene graph as represented in figure 3-7 (A). The “CLASS” statement defines a leaf node, Terrain, which obtains its geometry from the CAD file Terrain.wrl. Terrain.wrl contains the 3D model of the terrain in VRML format. The “CREATE” statement instantiates a transformation node “ExTerrain” and adds a child object conforming to the geometry defined by class “Terrain” to it. The “PLACE” statement places the created object in the scene by adding the “ExTerrain” transformation node to the root of the scene graph. The object is placed at point (0,0,0) in the world space by setting the translation field of the “ExTerrain” transformation node to (0,0,0).

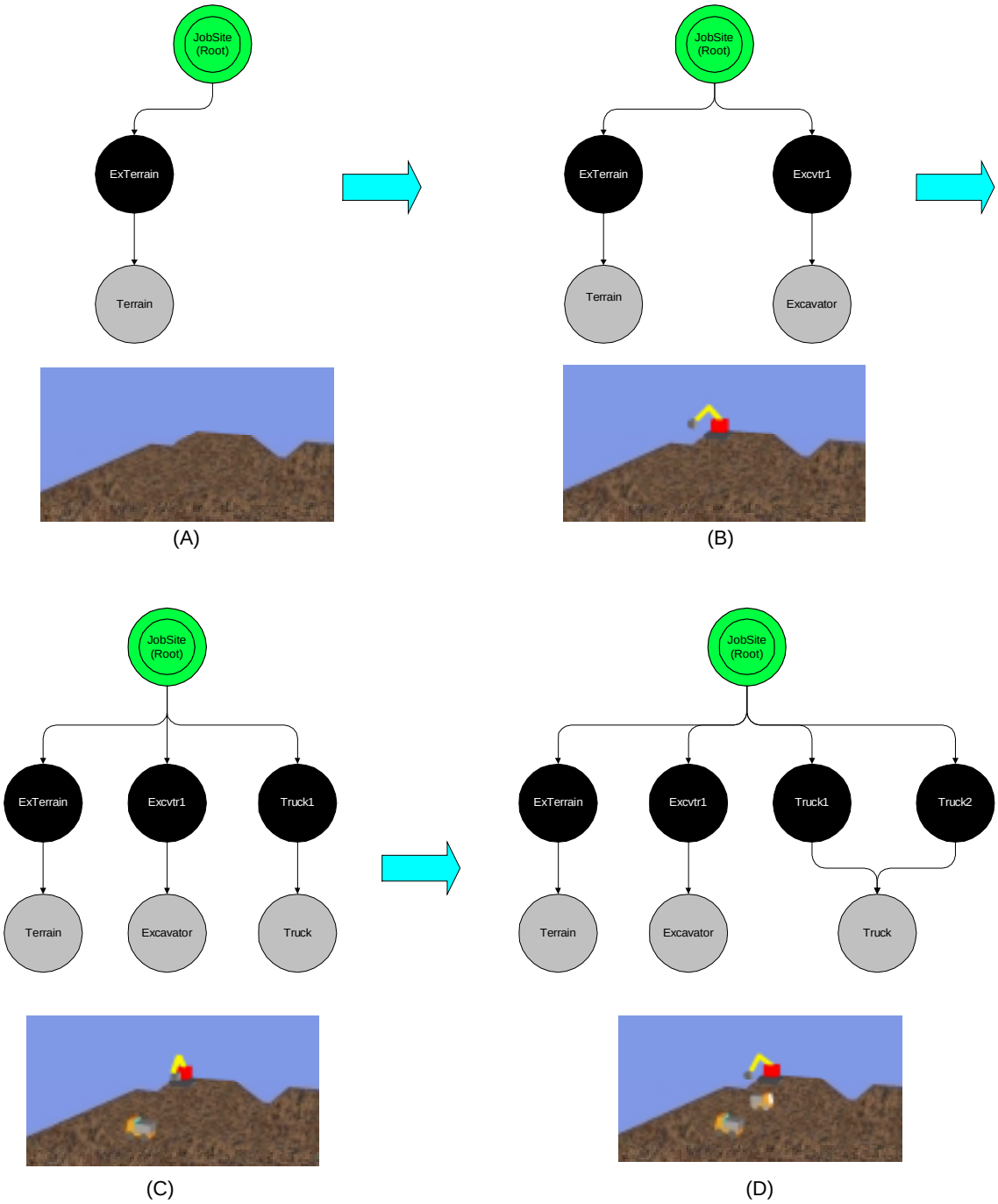


Figure 3-7 Evolution of the Scene Graph corresponding to the Trace File in Figure 3-6

The subsequent 3 statements grouped as (B) similarly define the class “Excavator”, create a transformation node “Excvtr1”, and place an excavator object at point (5,2,1) in the world space. The corresponding augmented scene graph is shown in figure 3-7 (B). The first 2 trace statements in group (C) define class “Truck” and instantiate the object “Truck1” conforming to the geometry defined in class “Truck”. The final statement in group (C) places “Truck1” at the beginning of Path “LoadToDump” (i.e. at point (3,2,1)) with the appropriate orientation by suitably setting the translation and the rotation fields of the transformation node. The modified scene graph is represented by figure 3-7 (C). The trace statements in group (D) similarly define another truck, “Truck2”, and place it at the beginning of the path “DumpToLoad”.

Referring to the completed scene graph in figure 3-7 (D), it is interesting to note that both “Truck1” and “Truck2” refer to the same geometry (i.e. leaf node). Therefore, the simultaneous depiction of both trucks involves presenting two different views of the same geometrical object via two different transformation nodes (i.e. “Truck1” and “Truck2”) as shown in figure 3-7 (D). This approach is infinitely scalable and is one of the major advantages afforded by scene graph technology. For instance, an entire steel frame structure consisting of hundreds 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. Furthermore, due to the ability of transformation nodes to scale and rotate objects in addition to positioning them, it is theoretically possible to depict the same steel structure using just one CAD model each of a beam and column. Of course, all the beams and columns must use the same type of steel section if this were to be done.

3.4.3 Scene Graph Complexity

The scene graph hierarchy depicted in figure 3-7 (D) is a simplified version of the scene graph created by the DCV. This is done to facilitate this discussion without involving implementation details. The hierarchy depicted in figure 3-7 (D) would only allow the manipulation of the excavator and the trucks as whole units via the transformation nodes “Excvtr1”, “Truck1”, and “Truck2”. In order to depict the articulated motion of the

excavator and the trucks (i.e. realistically display the digging action of the excavator and the dumping action of the trucks), it is necessary to control the motion of individual machine components such as the boom and the stick of the excavator, and the bed of the dumptrucks. Such hierarchical control over individual machine components can be easily achieved by suitably constructing the scene graph.

Figure 3-8 presents an animation snapshot of a modeled earthmoving operation that was used in the initial testing of the DCV. In this animation, the viewer is able to observe the accumulating trucks waiting to be loaded, the trucks maneuvering to get into position under the excavator, the excavator digging the earth and loading the trucks until they are full, the trucks traveling to the dumpsite, accumulating occasionally to enter the dump area, backing up and tipping their load, and then returning to the loading site to begin another cycle.

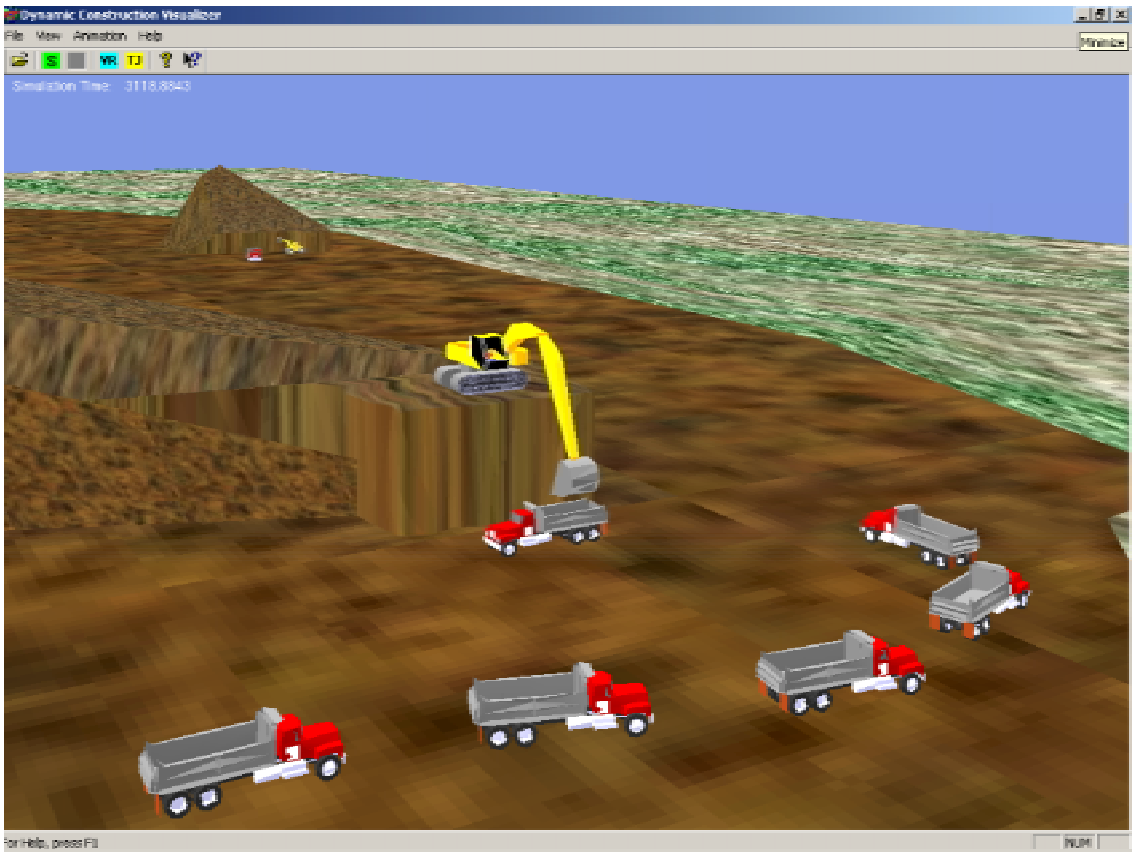


Figure 3-8 Animation Snapshot of the Loading Area in an Earthmoving Operation

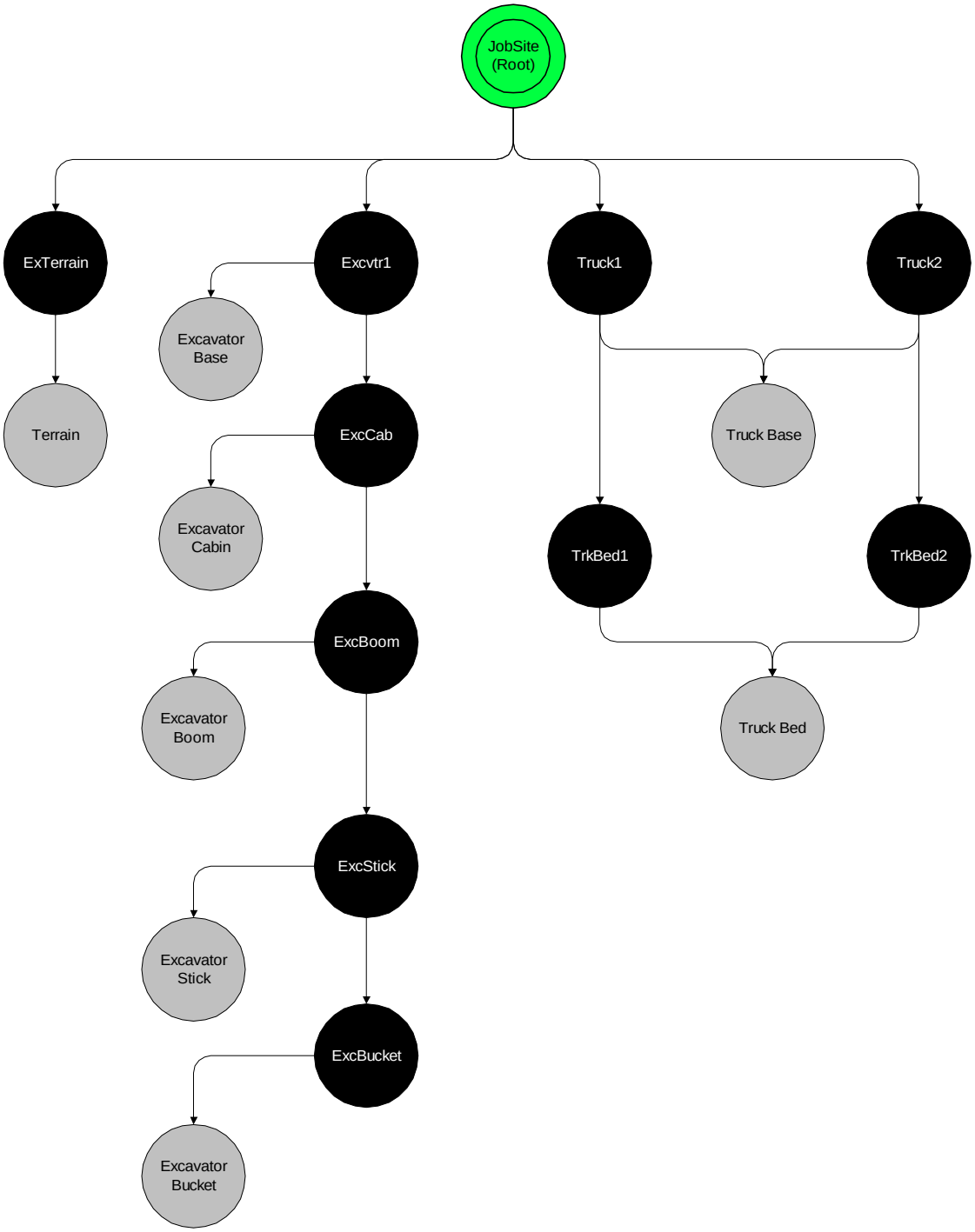


Figure 3-9 Depth of a Scene Graph

Figure 3-9 augments and presents the scene graph hierarchy presented in figure 3-7 (D). This scene graph has the same scene components (i.e. a terrain, an excavator, and two trucks) as the one in figure 3-7 (D). However, scene components are now arranged in a

logical, multi-level hierarchy. The DCV implementation permits the construction and manipulation of similar complex scene graph hierarchies via commands such as "ATTACH" and "DETACH" in the trace file. Such hierarchies permit the realistic depiction of complex machine movements. An attached object can move relative to its parent. For instance, with a scene graph hierarchy similar to the one presented in figure 3-9, it is easy to depict the boom of the swinging excavator being lifted. The hierarchy that develops during the earthmoving visualization depicted in figure 3-8 is similar (except for the number of trucks) to the one depicted in figure 3-9. The hierarchy permits the depiction of realistic digging action of the excavator and the dumping action of the trucks.

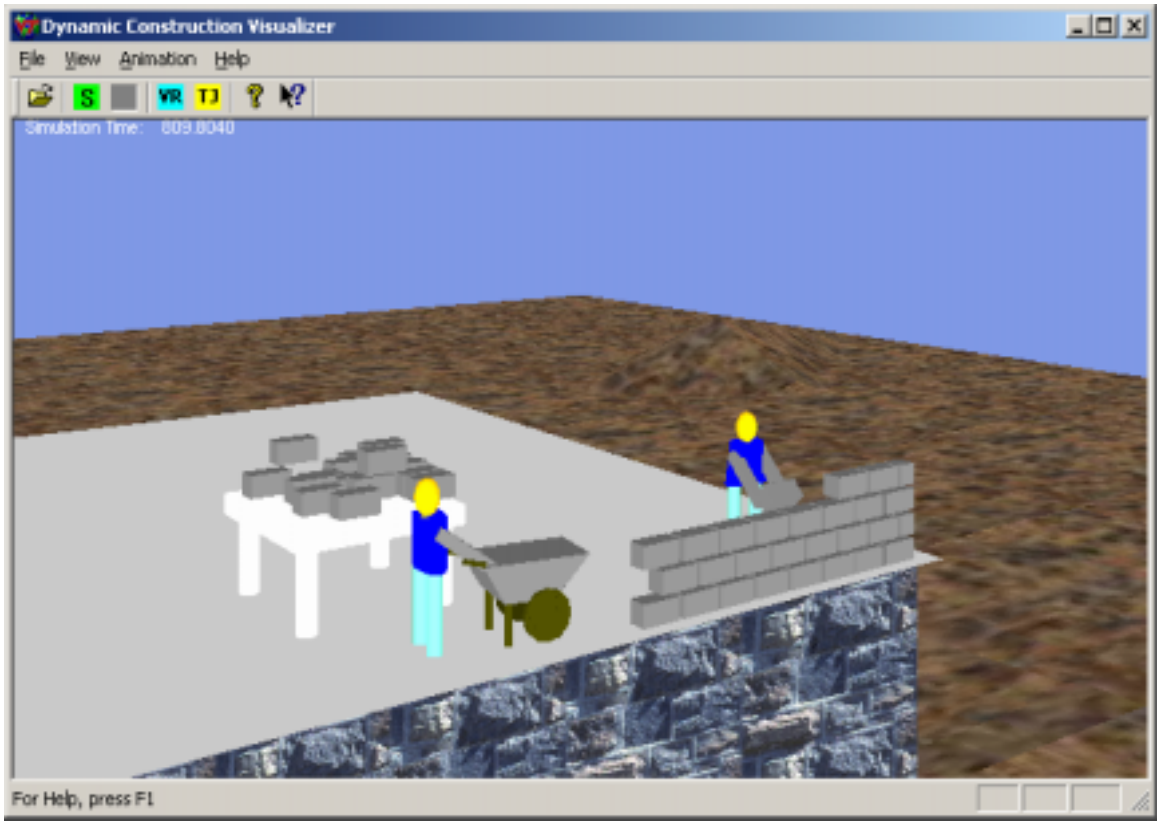


Figure 3-10 Animation Snapshot of a Block-laying Operation

Resources often need to be combined and act as a group. For example, a flatbed and steel shapes often need to travel together as a loaded flatbed in an animation. Besides

facilitating the depiction of complex hierarchical motion, the ‘ATTACH” and “DETACH” commands allow resources to be combined into a compound resource and compound resources to be broken up into its constituents. Figure 3-10 presents an animation snapshot of a modeled block-laying operation that shows a mason and his assistant working on a wall section. This operation was modeled and animated at a very low level-of-detail. The viewer is able to observe a mason constructing a wall section by laying successive courses of individual blocks. The viewer is also able to observe the materials (blocks and mortar) being delivered to the working floor by a lift (not visible in the snapshot) and being transported to the workface by the mason’s assistant.

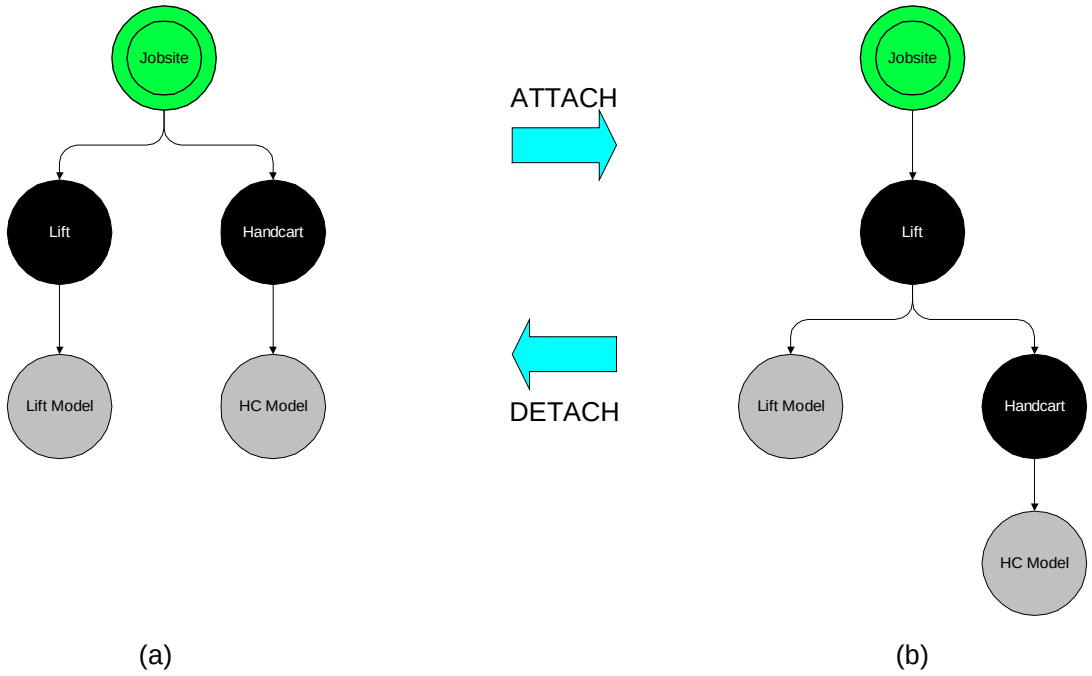


Figure 3-11 Dynamic Scene Graph Modification

In this operation, resources often need to be grouped together at times and then separated again into constituents. For instance, the lift and the material loaded onto it need to move (and hence be grouped) together whenever a loaded lift ascends or descends. When the material is unloaded, each resource (the hauled material and the lift) needs to be independently manipulated and hence needs to be ungrouped. Figure 3-11 presents an example of scene graph modification by the use of the “ATTACH” and “DETACH”

commands. Figure 3-11 represents a portion of the scene graph that develops during the visualization of the block-laying operation in figure 3-10.

Imagine that the empty lift is in lowered position and a handcart full of mortar is placed onto it. The structure of the scene graph at this point would resemble figure 3-11 (a). The lift now needs to haul the loaded handcart to the work floor. Attaching the handcart to the lift at this point would modify the scene graph as in figure 3-11 (b). Any positional change applied to the lift (via transformation node “Lift”) would also apply to the handcart since the node “Handcart” is a child node of “Lift”. Conversely, positional changes applied to the attached handcart would be relative to the parent lift. When the loaded lift reaches the work floor, the handcart is detached from the lift to revert the scene graph structure to figure 3-11 (a). The lift and the handcart can now be independently manipulated via transformation nodes “Lift” and ‘Handcart” respectively.

3.5 ANIMATING CONSTRUCTION SITE ACTIVITIES

The Cosmo3D API provides several utilities that facilitate the depiction of simple animations of scene graph entities such as the rotation of wheels in a moving car, the constant motion of a swinging pendulum, and other simple translational object motions. However, the dynamic characteristics of typical construction sites demanded the design and implementation of specialized position and orientation-updating algorithms in addition to the scene graph API’s built-in utilities. Understanding the working of the DCV animation clock (explained in the following section) is fundamental to understanding the animation capabilities of the DCV.

3.5.1 Measuring Time

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

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

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

```
TIME timevalue;
```

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

Figure 3-12 augments and presents the sample DCV trace file previously presented in figure 3-6. The implications of visualizing this trace file in the DCV are easily interpretable. Immediately before the onset of the animation, two paths, “LoadToDump” and “DumpToLoad”, and three classes, “Terrain”, “Excavator”, and “Truck” are defined and their representations are stored into memory. The Construction of the scene graph

begins at the onset of the animation (i.e. at time zero) when the terrain and the excavator objects are created and placed in the scene.

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

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

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

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

TIME 18;
MOVE Truck2 DumpToLoad 90;
```

Figure 3-12 Sample Trace File to demonstrate the Time Advance Mechanism

Further reading of statements from the trace file is suspended until the animation time equals 6. Thus, a person viewing the animation sees a motionless excavator in the terrain for six animation time units. At this point, a truck, “Truck1”, is created and placed in the scene at the beginning of path “LoadToDump”. Six animation time units later (at time 12), “Truck1” starts moving along the path “LoadToDump” at a speed that will require 120 animation time units to reach the end of the path. At the same time (12), another truck, “Truck2”, is created and placed in the scene at the beginning of path “DumpToLoad”. At animation time 18, “Truck2” starts moving along the “DumpToLoad” and will require 90 time units to reach its destination. At the moment “Truck2” starts moving, the already in motion “Truck1” will have completed about $1/20^{th}$ of its journey.

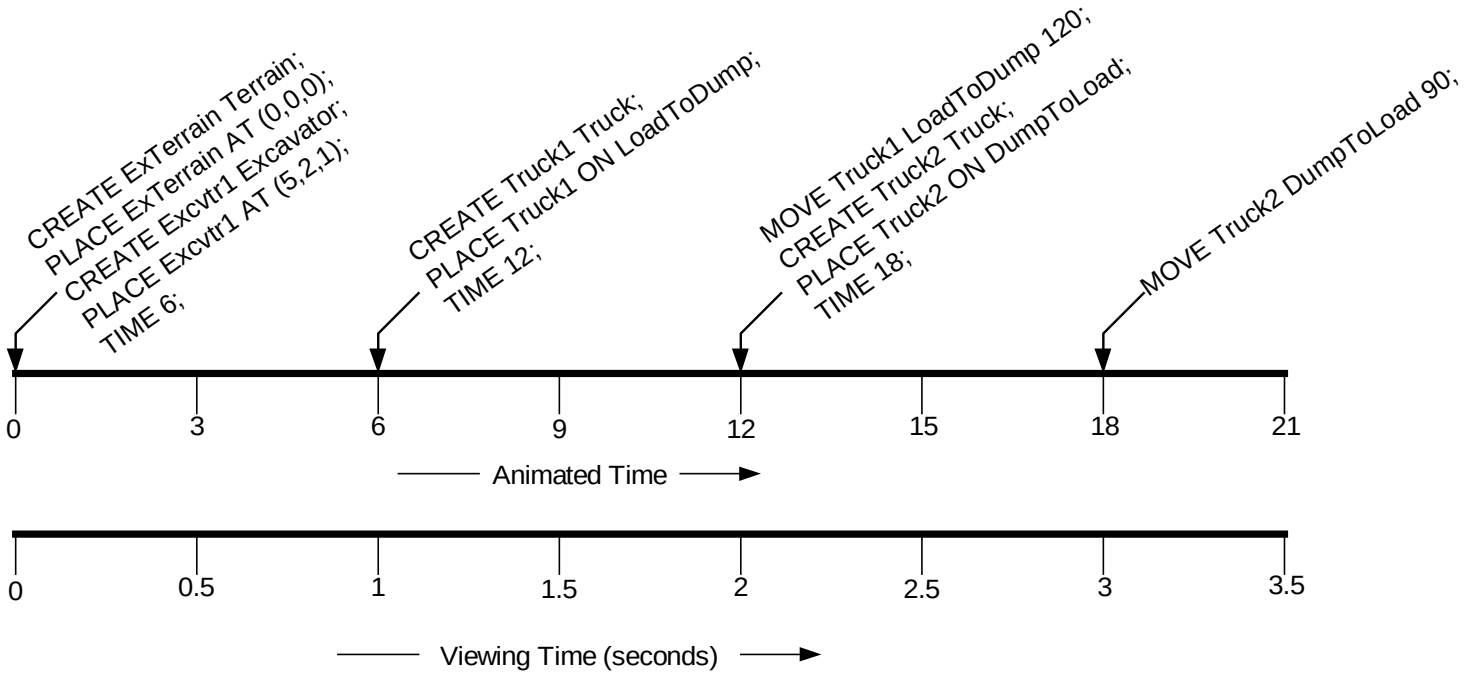


Figure 3-13 Processing of Commands in a Trace File

Figure 3-13 graphically displays the processing of commands in the trace file described above. The real-timeline displayed below the animated-timeline assumes a viewing ratio of 6. Immediately after the trace file commands at time 12 are processed, the scene graph for this animation would exactly resemble the one in figure 3-7 (D). The motion of the scene objects is achieved by manipulating the values of the transformation nodes in the scene graph. As such, the "structure" of the scene graph remains unaltered during the animation of scene objects unless it is explicitly modified using the "ATTACH" and "DETACH" commands. These commands do change the structure of the scene graph by modifying the parent-child relationships between scene graph nodes. In addition, the scene graph is obviously altered (augmented or diminished) if new scene objects are created or existing scene objects are destroyed dynamically during the animation.

3.5.2 The DCV Frame-updating Mechanism

The DCV is intended for the smooth and scalable 3D visualization of modeled construction operations over a wide range of systems ranging from typical laptops and desktops to high-end graphics workstations. As such, the DCV implementation needed to achieve "maximum performance" rendering on all supported systems. Maximum performance means that when there is no contention for rendering resources, and once utilized resources are made available, graphics rendering performance is limited only by the system's raw graphics performance and the graphics software efficiency (Kilgard et al. 1995). The DCV application needed to obtain the *maximum* performance potential of the system on which it was run.

In addition, construction operations range from the relatively simple to the most complex. The complexity of DCV visualizations would depend on the type of operations being visualized as well as on the level-of-detail incorporated therein. For instance, simultaneously visualizing all modeled operations in the construction of a building would be computationally much more expensive than visualizing a single modeled operation such as the construction of a wall section by a crew of masons. The cause of the increased computational load is that the number of scene objects that would need to be monitored

for position and/or orientation changes, updated, and drawn in each frame would be significantly higher in the former visualization.

In addition to the number of scene objects, the desired amount of realism would also influence the complexity of DCV visualizations. For instance, the use of accurate, detailed 3D models and texture-mapped geometrical objects would significantly increase the load on the graphics subsystem and consequently degrade the obtainable frame rate. Texture-mapped geometry is discussed in detail in the literature (Silicon Graphics 1998, Woo et al. 1997). The DCV needed to allow the smooth visualization of modeled operations at a user-specified, but constant animation speed (viewing ratio).

The double buffering, variable frame rate paradigm was found to be most suitable in achieving the design requirements of the DCV and was therefore employed. Figure 3-14 presents the DCV time advancing and frame updating mechanism, which yields satisfactory results on standard machines without the need for any special hardware, and which takes advantage of extra computing power and special graphics accelerators for increased performance. For instance, in visualizing the earthmoving operation depicted in figure 3-8 on a standard laptop computer powered by a 300 MHz Pentium processor, a satisfactory screen refresh rate of 15 frames per second was obtained without the use of any additional specialized graphics hardware. Any increase in the available computing power resulted in a proportional improvement in rendering performance and frame rate. On a desktop featuring a 600 MHz Pentium III processor, 128 megabytes of RAM, and an Nvidia TNT2 graphics card with 16 megabytes of video memory, a consistent full-screen performance in excess of 60 frames per second was observed. This frame-update mechanism is explained in the following sections.

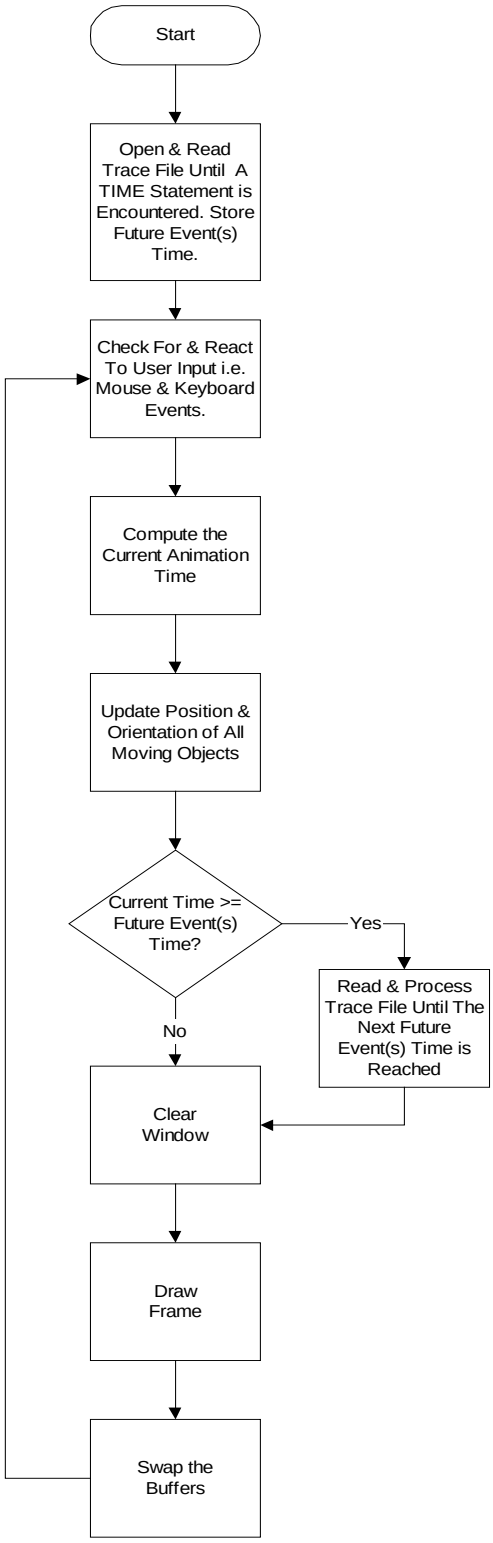


Figure 3-14 The DCV Event Loop

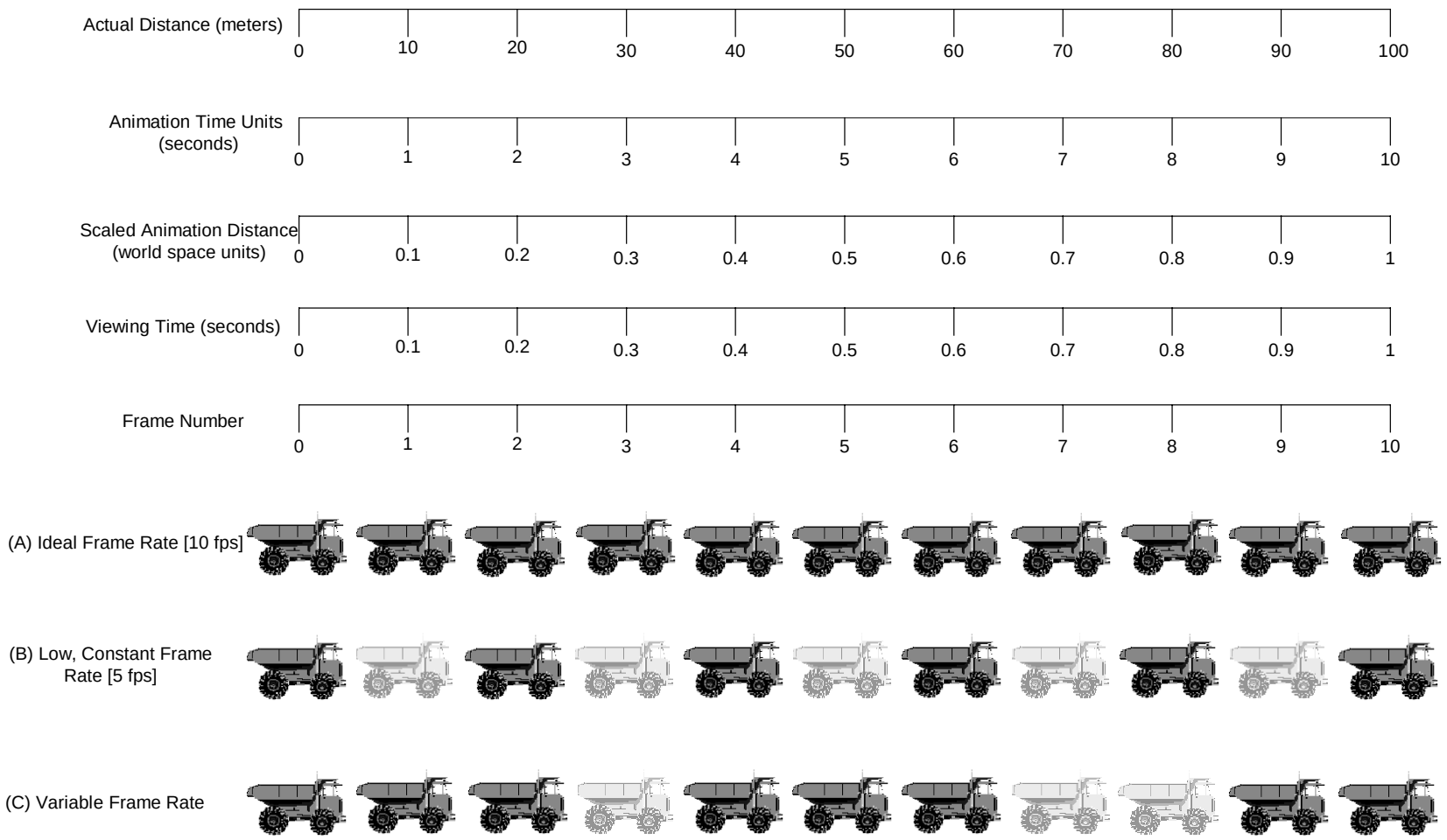


Figure 3-15 Relationship between the Viewing Ratio and the Obtained Frame Rate

3.5.3 Relationship between Frame Updates and the Viewing Ratio

The viewing ratio should always be maintained at a constant value irrespective of the frame rate being attained. For instance, if the animation speed were maintained at 6, a truck needing 24 animation time units to move along a path should complete its journey in 4 viewing seconds on any system, irrespective of the frame rate. Of course, the number of frames displayed in 4 viewing seconds (animation smoothness) would vary depending on the host systems' capabilities. In addition, the frame rate can also vary on the same system depending on the frame complexity at various times during animations.

Figure 3-15 graphically explains the relationship between the constant maintained viewing ratio and the obtained variable frame rate in DCV animations. Consider an example similar to the one depicted in figure 3-3. Imagine that a truck requires 10 seconds in real life to cover a distance of 100 meters and that 1unit of distance in world space (point (0,0,0) to point (1,0,0)) represents 100 meters. If a viewing ratio of 10 were adopted during the animation, the truck would require 1 second to cover the distance between point (0,0,0) and point (1,0,0). Assume, for illustrative purposes, that this animation is run on various systems that refresh their display 10 times per second. In such a scenario, ideally, the journey of the truck should comprise of 10 different frames as depicted in figure 3-15 (a).

On systems powerful enough to compute, clear, and draw each frame within 1/10 second, the ideal frame rate of 10 fps illustrated in figure 3-15 (a) would be attained and the 1 second journey of the truck would be depicted as 10 different and equally spaced consecutive frames. If the same animation was run on systems that require a little more than 1/10 second but less than 1/5 (i.e. 2 times 1/10) second to produce each frame, a constant frame rate of 5 fps would be attained. As demonstrated in figure 3-15 (b), frames 1, 3, 5, 7, and 9 would be skipped and consequently, frames 2, 4, 6, 8, and 10 would each be displayed twice during consecutive screen refreshes.

In the concluding case, imagine that the same animation is run on systems that can produce typical frames within 1/10 second but require more time to generate and draw

certain frames. The increased time to produce certain frames could be due to increased frame complexity or due to a temporary increase in the computational load such as calculating and updating the positions of a large number of objects in the animation. Such a scenario is depicted in figure 3-15 (c).

Frames 1 and 2 are computed and displayed within 1/10 second. However, the subsequent frame (frame 3) is not produced by the system within the next 1/10 second and is therefore skipped (i.e. frame 2 is displayed again during the next screen refresh). The system uses the idle time (accrued by skipping frame 3) and the time allotted for the next frame to compute and draw frame 4. A steady state is temporarily attained and frames 5 and 6 are each produced within 1/10 second. At this point, the system is again unable to produce the next frame (frame 7) within the allotted time and hence skips it displaying frame 6 for the second time. The system tries to use the accrued idle time and the allotted frame time to produce frame 8. However, it fails to do so and skips frame 8 as well, displaying frame 6 for the third time. Steady state is attained subsequently and frames 9 and 10 are produced within their allotted time.

The vital point to be noted in this discussion is that on all systems on which the animation is run (figures 15 (a), (b), and (c)), the journey of the truck requires 1 viewing second irrespective of the frame rate obtained. Frame 10 is drawn at the end of 1 second regardless of the intermediate frames. On all systems, the DCV frame-update algorithm maximizes the number of intermediate frames, and hence the smoothness of the animation, by fully utilizing the host system's frame-generating capacity at all times. The implementation of the DCV thus smoothly animates construction site activities at a fixed user-defined and controllable viewing ratio by achieving maximum possible performance from host systems.

3.6 CONCLUSION

The world of computer graphics is advancing at an astonishing rate. The evolution of standards such as VRML and the availability of high-level toolkits such as scene graph libraries facilitate the development of complex domain-specific graphics applications as

never before. Researchers in various fields such as the physical sciences, engineering, and medicine can capitalize on these advances to develop very advanced and efficient visualization tools that require only the development of certain domain-specific algorithms in addition to the utilities provided by the available high-level graphics tools. Researchers should appreciate their role in developing domain-specific graphics applications and not consider it as a task for computer scientists, as is often done and not achieved due to the lack of understanding of the domain by computer scientists.

The authors seized this initiative to design and implement the DCV for visualizing simulated construction operations and the resulting products in 3D. The DCV is independent of any particular simulation language or software. DCV animations are driven by text trace files that are created by appropriately instrumented simulation models while they run. Since DCV trace files can be generated by a wide variety of simulation software, the system is capable of effectively visualizing modeled operations in other fields such as the manufacturing and service industries. However, it has been designed specifically for visualizing modeled construction operations. This paper explained the underlying scene graph technology and the animation mechanism used to design the Dynamic Construction Visualizer system.

3.7 REFERENCES

Aloufa, A.A. (1993). "Modeling and Simulation of Construction Operations", *Automation in Construction*, 1, 351-359.

AutoSimulations, Inc. (2000). "AutoMod Simulation Software", URL: http://www.automod.com/simulation/simsoftware.html

Bentley Systems, Inc. (1998). *PlantSpace® Dynamic Animator™ Version 1.0 User Guide*, Gaithersburg, MD.

Dassault Systemes (2000). "Factory Simulation Solutions", URL: http://www.delmia.com

Henriksen, J.O. (1999), "General-purpose Concurrent and Post-processed Animation with PROOF™", *Proceedings of the 1999 Winter Simulation Conference*, Society for Computer Simulation, San Diego, CA, 176-181.

Huang R., and Halpin, D.W. (1994). " Visual Construction Operation Simulation: The DISCO Approach", *Journal of Microcomputers in Civil Engineering, 9(6),* 175-184.

Ioannou, P.G., and Martinez, J. (1996). "Animation of Complex Construction Simulation Models", *Proceedings of the 3rd Congress on Computing in Civil Engineering*, ASCE, Reston, VA, 620-626.

Kilgard, M.J., Blythe, D., and Hohn, D. (1995). "System Support for OpenGL Direct Rendering", *Proceedings of Graphics Interface '95,* Quebec City, Quebec. Available: http://trant.sgi.com/opengl/docs/Direct/direct.html

Law, A.M., and Kelton, W.D. (1991). *Simulation Modeling and Analysis*, 2nd Edn. McGraw-Hill, New York, NY.

Liu, L.Y., and Ioannou, P.G. (1993), "Graphical Resource-based Object-oriented Simulation for Construction Process Planning", *Proceedings Of the 5th International Conference on Computing in Civil and Building Engineering*, ASCE, Reston, VA, 1390-1397.

Martinez, J. C. (1998). "Earthmover - Simulation Tool for Earthwork Planning", *Proceedings of the 1998 Winter Simulation Conference,* Society for Computer Simulation, San Diego, CA, 1263-1271.

Martinez, J.C., and Ioannou, P.G. (1999). "General Purpose Systems for Effective Construction Simulation", *Journal of Construction Engineering and Management*, 125(4), ASCE, 265-276.

McKinney K., Kim J., Fischer M., and Howard C. (1996). "Interactive 4D-CAD", *Proceedings of the 3rd Congress on Computing in Civil Engineering*, ASCE, Reston, VA, 383-389.

Op den Bosch, A. (1994). "Design/Construction Processes Simulation in Real-time Object-Oriented Environments", PhD Dissertation, Georgia Institute of Technology, Atlanta, GA.

Schriber, T.J. (1995). "Perspectives on Simulation using GPSS", *Proceedings of the 1995 Winter Simulation Conference*, Society for Computer Simulation, San Diego, CA, 451-456.

Shi, J.J., and Zhang, H. (1999). “Iconic Animation of Construction Simulation”, *Proceedings of the 1999 Winter Simulation Conference,* Society for Computer Simulation, San Diego, CA, 992-997.

Silicon Graphics, Inc. (1998) “*Cosmo 3D Programmer’s Guide*”, Mountain View, CA.

Sturgul, J.R., and Seibt, F. (1999). “A Simulation and Animation Model of the Transport of Coal to the Port", *Proceedings of the 8th Mine Planning and Equipment Selection Symposium,* Balkema, Rotterdam, Holland.

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

Wolverine Software Corporation (1995). *Using Proof Animation*, 2nd Edn. Annandale, VA.

Woo, M., Neider, J., and Davis, T. (1997). *OpenGL Programming Guide*, 2nd Edn. Addison Wesley, Reading, MA.

APPENDICES

A Dynamic Construction Visualizer Language Reference

B Instrumented Earthmoving Simulation Model

C Section of a Trace File for Visualizing Earthmoving Operations

Appendix A
Dynamic Construction Visualizer Language Reference

This appendix lists the statements available in the Dynamic Construction Visualizer (DCV) language. DCV trace file statements can span multiple lines with arguments separated by white space. Arguments that include white space must be enclosed in single quotations. A statement ends with a semicolon.

Comments can be placed in trace files by making the first non white space character after a statement a "/". The comment continues until the end of the line. The statements defined by the DCV language are listed in the table below.

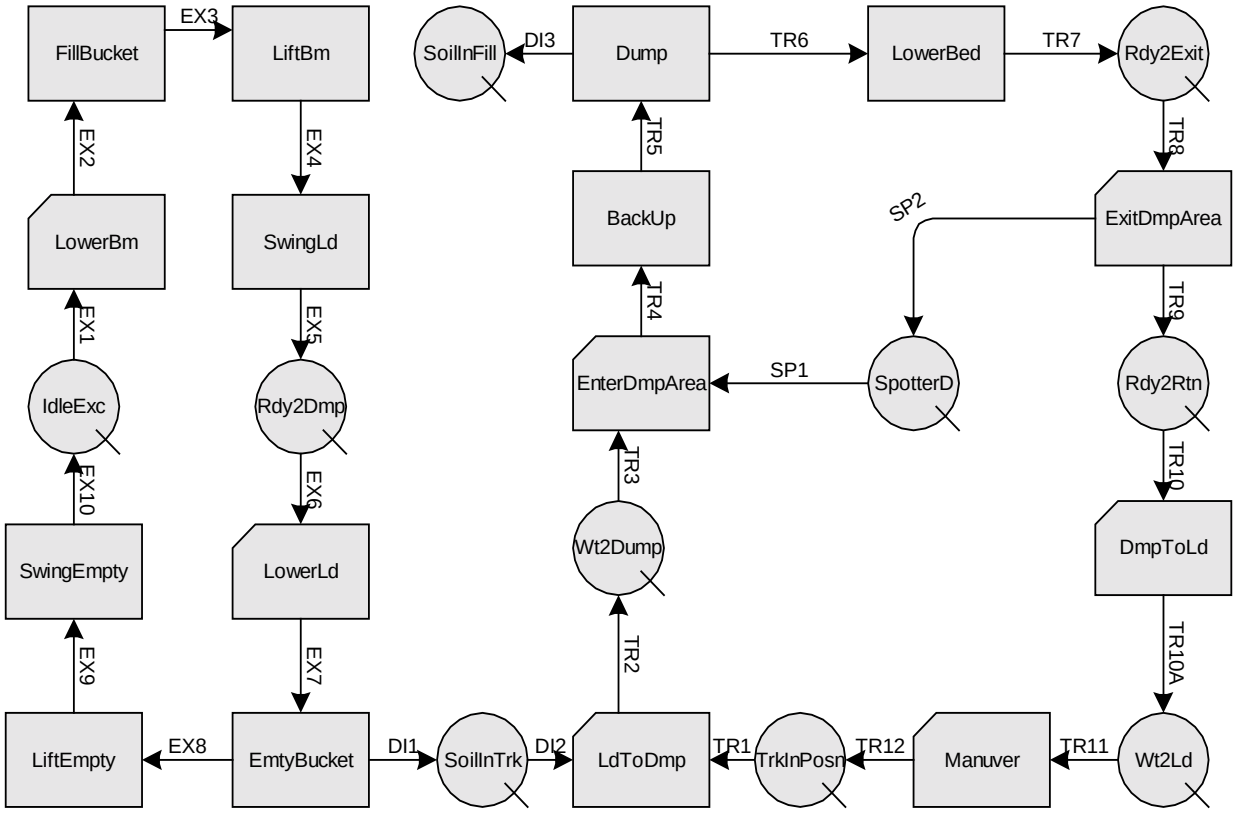
Statement	Arguments
PATH	*PathName Points;*
NONDIRECPATH	*PathName Points;*
TIME	*TimeValue;*
CLASS	*ClassName DiskFileName;*
ORIENTCLASS	*ClassName AboutAxis RotationAmount;*
SET CLASS	*ClassName* RGP *Value;*
SET CLASS	*ClassName* FORECLEARANCE *Value;*
SET CLASS	*ClassName* AFTCLEARANCE *Value;*
CREATE	*ObjectName ClassName;*
SET OBJECT	*ObjectName* RGP *Value;*
SET OBJECT	*ObjectName* FORECLEARANCE *Value;*

SET OBJECT	*ObjectName* AFTCLEARANCE *Value;*
ATTACH	*ChildObjectName ParentObjectName AttachPoint;*
PLACE	*ObjectName* AT *PlacePoint;*
PLACE	*ObjectName* ON *PathName;*
MOVE	*ObjectName PathName Duration;*
SLIDE	*ObjectName StartPoint EndPoint Duration;*
ROTATE	*ObjectName* HOR *RotationAmount Duration;*
ROTATE	*ObjectName* VERT *RotationAmount Duration;*
DESTROY	*ObjectName;*
CAMERA POSITION	*PlacePoint;*
CAMERA HORIZORIENT	*Value;*
CAMERA VERTORIENT	*Value;*
END;	

Appendix B
Instrumented Earthmoving Simulation Model

This appendix presents the STROBOSCOPE network model and the source code for a simulated earthmoving operation. This model has been instrumented to generate a DCV animation trace file during a simulation run.

STROBOSCOPE Network Model for a Simulated Earthmoving Operation



```
/*****************************************************************
/* Stroboscope source file generated from Visio drawing c:\Animator\Earth\earthmoving.vsd
/*****************************************************************

/*****************************************************************
/* General section for problem parameters

VARIABLE Animate 1; /Use 0 to not animate.
                    /Use 1 to generate a DCV animation trace file.

/*****************************************************************
/* Definition of resource types

GENTYPE           Dirt; /DI
COMPTYPE          Excavator; /EX
GENTYPE           Spotter; /SP
COMPTYPE          Truck; /TR

SAVEPROPS   Truck AmtOfDirt;

/*****************************************************************
/* Definition of network nodes

QUEUE             IdleExc Excavator;
COMBI             LowerBm;
NORMAL            FillBckt;
NORMAL            LiftBm;
NORMAL            SwingLd;
QUEUE             RdyToDmp Excavator;
COMBI             LowerLd;
NORMAL            EmtyBckt;
NORMAL            LiftEmpty;
NORMAL            SwingEmpty;
QUEUE             Wt2Ld Truck;
QUEUE             SoilInTrk Dirt;
COMBI             LdToDmp;
QUEUE             Wt2Dump Truck;
COMBI             EnterDmpArea;
NORMAL            ShiftGear;
NORMAL            BackUp;
NORMAL            Dump;
NORMAL            LwrEty;
QUEUE             Rdy2Exit Truck;
COMBI             ExitDmpArea;
QUEUE             Soil Dirt;
QUEUE             SpotterD Spotter;
COMBI             Manuver;
QUEUE             TrkInPosn Truck;
COMBI             DmpToLd;
QUEUE             Rdy2Rtn Truck;
QUEUE             TrksInYard Truck;
COMBI             TrkArrToSite;
```

```
/*********************************************************************
/* Definition of network Links

LINK              EX1 IdleExc LowerBm;
LINK              EX2 LowerBm FillBckt Excavator;
LINK              EX3 FillBckt LiftBm Excavator;
LINK              EX4 LiftBm SwingLd Excavator;
LINK              EX5 SwingLd RdyToDmp;
LINK              EX6 RdyToDmp LowerLd;
LINK              EX7 LowerLd EmtyBckt Excavator;
LINK              EX8 EmtyBckt LiftEmpty Excavator;
LINK              EX9 LiftEmpty SwingEmpty Excavator;
LINK              EX10 SwingEmpty IdleExc;
LINK              DI1 EmtyBckt SoilInTrk;
LINK              TR2 LdToDmp Wt2Dump;
LINK              TR3 Wt2Dump EnterDmpArea;
LINK              TR4 EnterDmpArea ShiftGear Truck;
LINK              TR4A ShiftGear BackUp Truck;
LINK              TR5 BackUp Dump Truck;
LINK              TR6 Dump LwrEty Truck;
LINK              TR7 LwrEty Rdy2Exit;
LINK              TR8 Rdy2Exit ExitDmpArea;
LINK              TR9 ExitDmpArea Rdy2Rtn;
LINK              DI3 Dump Soil;
LINK              TR10 Rdy2Rtn DmpToLd;
LINK              SP1 SpotterD EnterDmpArea;
LINK              SP2 ExitDmpArea SpotterD;
LINK              TR1 TrkInPosn LdToDmp;
LINK              DI2 SoilInTrk LdToDmp;
LINK              TR11 Wt2Ld Manuver;
LINK              TR12 Manuver TrkInPosn;
LINK              TR10A DmpToLd Wt2Ld;
LINK              TR11A TrkArrToSite Rdy2Rtn;
LINK              TR12A TrksInYard TrkArrToSite;

/*********************************************************************
/* Entry of resources into IdleExc

/*********************************************************************
/* Startup of LowerBm

ENOUGH            EX1 'IdleExc.CurCount>0';
DURATION          LowerBm 'Uniform[4,5]';

/*********************************************************************
/* Termination of LowerBm

/*********************************************************************
/* Startup of FillBckt

DURATION          FillBckt 'Uniform[5,8]';

/*********************************************************************
/* Termination of FillBckt
```

```
/*********************************************************************
/* Startup of LiftBm

DURATION          LiftBm 'Uniform[4,7]';

/*********************************************************************
/* Termination of LiftBm

/*********************************************************************
/* Startup of SwingLd

DURATION          SwingLd 'Uniform[8,14]';

/*********************************************************************
/* Termination of SwingLd

/*********************************************************************
/* Entry of resources into RdyToDmp

/*********************************************************************
/* Startup of LowerLd

SEMAPHORE         LowerLd 'TrkInPosn.CurCount';
ENOUGH            EX6 'RdyToDmp.CurCount>0';
DURATION          LowerLd 'Uniform[4,6]';

/*********************************************************************
/* Termination of LowerLd

/*********************************************************************
/* Startup of EmtyBckt

DURATION          EmtyBckt 'Uniform[4,6]';

/*********************************************************************
/* Termination of EmtyBckt

RELEASEAMT        DI1 '5.0';

/*********************************************************************
/* Startup of LiftEmpty

DURATION          LiftEmpty 'Uniform[4,6]';

/*********************************************************************
/* Termination of LiftEmpty

/*********************************************************************
/* Startup of SwingEmpty

DURATION          SwingEmpty 'Uniform[6,10]';

/*********************************************************************
/* Termination of SwingEmpty
```

```
/*********************************************************************
/* Entry of resources into Wt2Ld

/*********************************************************************
/* Entry of resources into SoilInTrk

/*********************************************************************
/* Startup of LdToDmp

ENOUGH                  DI2 'SoilInTrk.CurCount>=10';
DRAWAMT                 DI2 'SoilInTrk.CurCount';
DURATION                LdToDmp 'Uniform[100,140]';

/*********************************************************************
/* Termination of LdToDmp

/*********************************************************************
/* Entry of resources into Wt2Dump

/*********************************************************************
/* Startup of EnterDmpArea

DURATION          EnterDmpArea 'Uniform[40,60]';

/*********************************************************************
/* Termination of EnterDmpArea

/*********************************************************************
/* Startup of ShiftGear

DURATION          ShiftGear 'Uniform[4,6]';

/*********************************************************************
/* Termination of ShiftGear

/*********************************************************************
/* Startup of BackUp

DURATION          BackUp 'Uniform[15,25]';

/*********************************************************************
/* Termination of BackUp

/*********************************************************************
/* Startup of Dump

DURATION          Dump 'Uniform[8,12]';

/*********************************************************************
/* Termination of Dump

RELEASEAMT        DI3 '10';

/*********************************************************************
/* Startup of LwrEty
```

```
DURATION          LwrEty 'Uniform[4,6]';

/*********************************************************************
/* Termination of LwrEty

/*********************************************************************
/* Entry of resources into Rdy2Exit

/*********************************************************************
/* Startup of ExitDmpArea

DURATION          ExitDmpArea 'Uniform[15,25]';

/*********************************************************************
/* Termination of ExitDmpArea

RELEASEAMT        SP2 '1';

/*********************************************************************
/* Entry of resources into Soil

/*********************************************************************
/* Entry of resources into SpotterD

/*********************************************************************
/* Startup of Manuver

SEMAPHORE         Manuver '!Manuver.CurInst & !TrkInPosn.CurCount';
ENOUGH            TR11 'Wt2Ld.CurCount>0';
DURATION          Manuver 'Uniform[20,30]';

/*********************************************************************
/* Termination of Manuver

/*********************************************************************
/* Entry of resources into TrkInPosn

/*********************************************************************
/* Startup of DmpToLd

ENOUGH            TR10 'Rdy2Rtn.CurCount>0';
DURATION          DmpToLd 'Uniform[100,130]';

/*********************************************************************
/* Termination of DmpToLd

/*********************************************************************
/* Entry of resources into Rdy2Rtn

/*********************************************************************
/* Entry of resources into TrksInYard

/*********************************************************************
/* Startup of TrkArrToSite
```

```
SEMAPHORE         TrkArrToSite '!TrkArrToSite.CurInst';
DURATION          TrkArrToSite 'Uniform[30,60]';

/*************************************************************************
/* Termination of TrkArrToSite

/*************************************************************************
/* Initialization of Queues, Running the Simulation, Presenting Results

IF Animate;

OUTFILE DEVIL "Earthmoving.vtf";

PRINT DEVIL

"PATH LdToDmp
      '(3.7, 0.0, -2)'
      '(3.7, 0, -1.7)'
      '(2.75, 0.0, -1.6)'
      '(2, 0, -1.5)'
      '(1.2, 0, -1)'
      '(1.2, 0, 0.5)'
      '(1.0, 0, 2.5)'
      '(0.7, 0, 4.8)'
      '(-1.5, 0, 5.5)'
      '(-2, 0, 5.5)'
      '(-3.9, 0, 5)'
      '(-4.2, 0, 3)'
      '(-4.5, 0, 1)'\059

PATH EnterDmpArea
      '(-4.5, 0, 1)'
      '(-4, 0.0, -1.5)'
      '(-3, 0, -1.5)'\059

NONDIRECPATH BackUp
      '(-3, 0, -1.5)'
      '(-4.7, 0, -1.5)'\059

PATH ExitDmpArea
      '(-4.7, 0, -1.5)'
      '(-3.5, 0.0, -1.5)'
      '(-3.5, 0.0, -0.5)'\059

PATH DmpToLd
      '(-3.5, 0.0, -0.5)'
      '(-3.5, 0.0, 1)'
      '(-3.3, 0.0, 3)'
      '(-3.0, 0, 4)'
      '(-2, 0.0, 4.5)'
      '(-0.5, 0, 4.5)'
      '(0.0, 0, 3.5)'
      '(0, 0.0, 3)'
      '(0, 0.0, 1)'
      '(0, 0.0, -1)'
      '(0, 0.0, -2)'\059
```

```
PATH Manuver
      '(0, 0.0, -2)'
      '(1, 0, -3)'
      '(2, 0.0, -4)'
      '(2.5, 0, -3.5)'
      '(3, 0, -3)'
      '(3.2, 0, -2.9)'
      '(3.4, 0, -2.75)'
      '(3.7, 0, -2.5)'
      '(3.7, 0, -2)'\059

PATH Excavator
      '(5.0,1,-2.1)'
      '(5.0,1,-2.3)'\059\n";

PRINT DEVIL
"TIME 0\059

/Position and Orient Camera
CAMERA POSITION '(-8,5,15)'\059
CAMERA HORIZORIENT -26\059
CAMERA VERTORIENT -10\059

CLASS Truck minitruckbase.csb\059
CLASS TruckBucket minitruckbucket.csb\059
SET CLASS Truck AFTCLEARANCE 1\059

CLASS ExcTerrain excterrain.csb\059
CREATE ExcTerrain ExcTerrain\059
PLACE ExcTerrain AT (0,0.0,0)\059

CLASS ExcBase excbase.csb\059
CLASS ExcCabin exccabin.csb\059
CLASS Boom excboom1.csb\059
CLASS Stick excboom2.csb\059
CLASS ExcBucket excbucket.csb\059
CREATE ExcBase ExcBase\059
CREATE ExcCabin ExcCabin\059
CREATE Boom Boom\059
CREATE Stick Stick\059
CREATE ExcBucket ExcBucket\059
ATTACH ExcCabin ExcBase (0,0.375,0)\059
ATTACH Boom ExcCabin (0,0.48,0)\059
ATTACH Stick Boom (0.6,0.68,-0.20)\059
ATTACH ExcBucket Stick (0.71,-0.68,0)\059
PLACE ExcBase ON Excavator\059\n";

ONSTART LowerBm PRINT DEVIL
"TIME %.2f\059
ROTATE Boom VERT -27 %.2f\059
ROTATE Stick VERT -10 %.2f\059\n"
SimTime LowerBm.Duration LowerBm.Duration;

ONSTART FillBckt PRINT DEVIL
"TIME %.2f\059
ROTATE ExcBucket VERT -90 %.2f\059\n"
SimTime FillBckt.Duration;
```

```
ONSTART LiftBm PRINT DEVIL
"TIME %.2f\059
ROTATE Boom VERT 27 %.2f\059
ROTATE Stick VERT 10 %.2f\059\n"
SimTime LiftBm.Duration LiftBm.Duration;

ONSTART SwingLd PRINT DEVIL
"TIME %.2f\059
ROTATE ExcCabin HOR 90 %.2f\059\n"
SimTime SwingLd.Duration;

ONSTART LowerLd PRINT DEVIL
"TIME %.2f\059
ROTATE Boom VERT -35 %.2f\059
ROTATE Stick VERT 20 %.2f\059\n"
SimTime LowerLd.Duration LowerLd.Duration;

ONSTART EmtyBckt PRINT DEVIL
"TIME %.2f\059
ROTATE ExcBucket VERT 90 %.2f\059\n"
SimTime EmtyBckt.Duration;

ONSTART LiftEmpty PRINT DEVIL
"TIME %.2f\059
ROTATE Stick VERT -20 %.2f\059
ROTATE Boom VERT 35 %.2f\059\n"
SimTime LiftEmpty.Duration LiftEmpty.Duration;

ONSTART SwingEmpty PRINT DEVIL
"TIME %.2f\059
ROTATE ExcCabin HOR -90 %.2f\059\n"
SimTime SwingEmpty.Duration;

ONSTART LdToDmp PRINT DEVIL
"TIME %.2f\059
MOVE Truck%.0f LdToDmp %.2f\059\n"
SimTime LdToDmp.Truck.ResNum LdToDmp.Duration;

ONSTART EnterDmpArea PRINT DEVIL
"TIME %.2f\059
MOVE Truck%.0f EnterDmpArea %.2f\059\n"
SimTime EnterDmpArea.Truck.ResNum EnterDmpArea.Duration;

ONSTART BackUp PRINT DEVIL
"TIME %.2f\059
MOVE Truck%.0f BackUp %.2f\059\n"
SimTime BackUp.Truck.ResNum BackUp.Duration;

ONSTART Dump PRINT DEVIL
"TIME %.2f\059
ROTATE TruckBucket%.0f VERT 55 %.2f\059\n"
SimTime Dump.Truck.ResNum Dump.Duration;

ONSTART LwrEty PRINT DEVIL
"TIME %.2f\059
```

```
ROTATE TruckBucket%.0f VERT -55 %.2f\059\n"
SimTime LwrEty.Truck.ResNum LwrEty.Duration;

ONSTART ExitDmpArea PRINT DEVIL
"TIME %.2f\059
MOVE Truck%.0f ExitDmpArea %.2f\059\n"
SimTime ExitDmpArea.Truck.ResNum ExitDmpArea.Duration;

ONSTART Manuver PRINT DEVIL
"TIME %.2f\059
MOVE Truck%.0f Manuver %.2f\059\n"
SimTime Manuver.Truck.ResNum Manuver.Duration;

ONSTART DmpToLd PRINT DEVIL
"TIME %.2f\059
MOVE Truck%.0f DmpToLd %.2f\059\n"
SimTime DmpToLd.Truck.ResNum DmpToLd.Duration;

BEFOREEND TrkArrToSite PRINT DEVIL
"TIME %.2f\059
CREATE Truck%.0f Truck\059
CREATE TruckBucket%.0f TruckBucket\059
ATTACH TruckBucket%.0f Truck%.0f (-0.6,0.3,0)\059
PLACE Truck%.0f ON DmpToLd\059\n"
SimTime TrkArrToSite.Truck.ResNum TrkArrToSite.Truck.ResNum
TrkArrToSite.Truck.ResNum TrkArrToSite.Truck.ResNum
TrkArrToSite.Truck.ResNum;

ENDIF; /Animate

INIT SpotterD 1;
INIT IdleExc 1;
INIT TrksInYard 5;

SIMULATEUNTIL Soil.CurCount>=5000;
REPORT;
```

Appendix C

Section of a Trace File for Visualizing Earthmoving Operations

This appendix presents a portion of the Dynamic Construction Visualizer Animation Trace File for visualizing an earthmoving operation. This trace file has been generated automatically during a simulation run by the instrumented simulation model presented in Appendix B.

```
PATH LdToDmp
      '(3.7, 0.0, -2)'
      '(3.7, 0, -1.7)'
      '(2.75, 0.0, -1.6)'
      '(2, 0, -1.5)'
      '(1.2, 0, -1)'
      '(1.2, 0, 0.5)'
      '(1.0, 0, 2.5)'
      '(0.7, 0, 4.8)'
      '(-1.5, 0, 5.5)'
      '(-2, 0, 5.5)'
      '(-3.9, 0, 5)'
      '(-4.2, 0, 3)'
      '(-4.5, 0, 1)';

PATH EnterDmpArea
      '(-4.5, 0, 1)'
      '(-4, 0.0, -1.5)'
      '(-3, 0, -1.5)';

NONDIRECPATH BackUp
      '(-3, 0, -1.5)'
      '(-4.7, 0, -1.5)';

PATH ExitDmpArea
      '(-4.7, 0, -1.5)'
      '(-3.5, 0.0, -1.5)'
      '(-3.5, 0.0, -0.5)';

PATH DmpToLd
      '(-3.5, 0.0, -0.5)'
      '(-3.5, 0.0, 1)'
      '(-3.3, 0.0, 3)'
      '(-3.0, 0, 4)'
      '(-2, 0.0, 4.5)'
      '(-0.5, 0, 4.5)'
      '(0.0, 0, 3.5)'
      '(0, 0.0, 3)'
      '(0, 0.0, 1)'
```

```
      '(0, 0.0, -1)'
      '(0, 0.0, -2)';

PATH Manuver
      '(0, 0.0, -2)'
      '(1, 0, -3)'
      '(2, 0.0, -4)'
      '(2.5, 0, -3.5)'
      '(3, 0, -3)'
      '(3.2, 0, -2.9)'
      '(3.4, 0, -2.75)'
      '(3.7, 0, -2.5)'
      '(3.7, 0, -2)';

PATH Excavator
      '(5.0,1,-2.1)'
      '(5.0,1,-2.3)';
TIME 0;

/Position and Orient Camera
CAMERA POSITION '(-8,5,15)';
CAMERA HORIZORIENT -26;
CAMERA VERTORIENT -10;

CLASS Truck minitruckbase.csb;
CLASS TruckBucket minitruckbucket.csb;
SET CLASS Truck AFTCLEARANCE 1;

CLASS ExcTerrain excterrain.csb;
CREATE ExcTerrain ExcTerrain;
PLACE ExcTerrain AT (0,0.0,0);

CLASS ExcBase excbase.csb;
CLASS ExcCabin exccabin.csb;
CLASS Boom excboom1.csb;
CLASS Stick excboom2.csb;
CLASS ExcBucket excbucket.csb;
CREATE ExcBase ExcBase;
CREATE ExcCabin ExcCabin;
CREATE Boom Boom;
CREATE Stick Stick;
CREATE ExcBucket ExcBucket;
ATTACH ExcCabin ExcBase (0,0.375,0);
ATTACH Boom ExcCabin (0,0.48,0);
ATTACH Stick Boom (0.6,0.68,-0.20);
ATTACH ExcBucket Stick (0.71,-0.68,0);
PLACE ExcBase ON Excavator;
TIME 0.00;
ROTATE Boom VERT -27 4.84;
ROTATE Stick VERT -10 4.84;
TIME 4.84;
ROTATE ExcBucket VERT -90 5.50;
TIME 10.34;
ROTATE Boom VERT 27 5.78;
ROTATE Stick VERT 10 5.78;
TIME 16.12;
ROTATE ExcCabin HOR 90 9.92;
```

```
TIME 45.94;
CREATE Truck1 Truck;
CREATE TruckBucket1 TruckBucket;
ATTACH TruckBucket1 Truck1 (-0.6,0.3,0);
PLACE Truck1 ON DmpToLd;
TIME 45.94;
MOVE Truck1 DmpToLd 117.27;
TIME 77.05;
CREATE Truck2 Truck;
CREATE TruckBucket2 TruckBucket;
ATTACH TruckBucket2 Truck2 (-0.6,0.3,0);
PLACE Truck2 ON DmpToLd;
TIME 77.05;
MOVE Truck2 DmpToLd 123.78;
TIME 122.90;
CREATE Truck3 Truck;
CREATE TruckBucket3 TruckBucket;
ATTACH TruckBucket3 Truck3 (-0.6,0.3,0);
PLACE Truck3 ON DmpToLd;
TIME 122.90;
MOVE Truck3 DmpToLd 110.06;
TIME 163.21;
MOVE Truck1 Manuver 23.38;
TIME 169.85;
CREATE Truck4 Truck;
CREATE TruckBucket4 TruckBucket;
ATTACH TruckBucket4 Truck4 (-0.6,0.3,0);
PLACE Truck4 ON DmpToLd;
TIME 169.85;
MOVE Truck4 DmpToLd 109.69;
TIME 186.59;
ROTATE Boom VERT -35 5.08;
ROTATE Stick VERT 20 5.08;
TIME 191.66;
ROTATE ExcBucket VERT 90 4.79;
TIME 196.45;
ROTATE Stick VERT -20 5.03;
ROTATE Boom VERT 35 5.03;
TIME 201.47;
ROTATE ExcCabin HOR -90 7.72;
TIME 209.19;
ROTATE Boom VERT -27 4.05;
ROTATE Stick VERT -10 4.05;
TIME 209.33;
CREATE Truck5 Truck;
CREATE TruckBucket5 TruckBucket;
ATTACH TruckBucket5 Truck5 (-0.6,0.3,0);
PLACE Truck5 ON DmpToLd;
TIME 209.33;
MOVE Truck5 DmpToLd 114.29;
TIME 213.25;
ROTATE ExcBucket VERT -90 6.92;
TIME 220.17;
ROTATE Boom VERT 27 5.03;
ROTATE Stick VERT 10 5.03;
TIME 225.20;
ROTATE ExcCabin HOR 90 11.34;
```

```
TIME 236.54;
ROTATE Boom VERT -35 5.11;
ROTATE Stick VERT 20 5.11;
TIME 241.64;
ROTATE ExcBucket VERT 90 5.15;
TIME 246.79;
ROTATE Stick VERT -20 4.78;
ROTATE Boom VERT 35 4.78;
TIME 246.79;
MOVE Truck1 LdToDmp 136.84;
TIME 246.79;
MOVE Truck2 Manuver 23.24;
TIME 251.57;
ROTATE ExcCabin HOR -90 7.52;
TIME 259.09;
ROTATE Boom VERT -27 4.11;
ROTATE Stick VERT -10 4.11;
TIME 263.20;
ROTATE ExcBucket VERT -90 5.65;
TIME 268.85;
ROTATE Boom VERT 27 5.58;
ROTATE Stick VERT 10 5.58;
TIME 274.43;
ROTATE ExcCabin HOR 90 10.62;
TIME 285.05;
ROTATE Boom VERT -35 4.36;
ROTATE Stick VERT 20 4.36;
TIME 289.42;
ROTATE ExcBucket VERT 90 5.86;
TIME 295.27;
ROTATE Stick VERT -20 5.10;
ROTATE Boom VERT 35 5.10;
TIME 300.37;
ROTATE ExcCabin HOR -90 7.97;
TIME 308.34;
ROTATE Boom VERT -27 4.74;
ROTATE Stick VERT -10 4.74;
TIME 313.08;
ROTATE ExcBucket VERT -90 6.11;
TIME 319.19;
ROTATE Boom VERT 27 4.14;
ROTATE Stick VERT 10 4.14;
TIME 323.33;
ROTATE ExcCabin HOR 90 9.54;
TIME 332.87;
ROTATE Boom VERT -35 4.92;
ROTATE Stick VERT 20 4.92;
TIME 337.78;
ROTATE ExcBucket VERT 90 5.55;
TIME 343.33;
ROTATE Stick VERT -20 4.68;
ROTATE Boom VERT 35 4.68;
TIME 343.33;
MOVE Truck2 LdToDmp 137.14;
TIME 343.33;
MOVE Truck3 Manuver 25.67;
TIME 348.02;
```

```
ROTATE ExcCabin HOR -90 9.12;
TIME 357.13;
ROTATE Boom VERT -27 4.29;
ROTATE Stick VERT -10 4.29;
TIME 361.42;
ROTATE ExcBucket VERT -90 5.76;
TIME 367.18;
ROTATE Boom VERT 27 5.26;
ROTATE Stick VERT 10 5.26;
TIME 372.44;
ROTATE ExcCabin HOR 90 8.92;
TIME 381.36;
ROTATE Boom VERT -35 4.56;
ROTATE Stick VERT 20 4.56;
TIME 383.63;
MOVE Truck1 EnterDmpArea 46.22;
TIME 385.92;
ROTATE ExcBucket VERT 90 5.89;
TIME 391.81;
ROTATE Stick VERT -20 5.77;
ROTATE Boom VERT 35 5.77;
TIME 397.57;
ROTATE ExcCabin HOR -90 6.71;
TIME 404.29;
ROTATE Boom VERT -27 4.83;
ROTATE Stick VERT -10 4.83;
TIME 409.12;
ROTATE ExcBucket VERT -90 5.39;
TIME 414.51;
ROTATE Boom VERT 27 5.44;
ROTATE Stick VERT 10 5.44;
TIME 419.94;
ROTATE ExcCabin HOR 90 9.36;
TIME 429.30;
ROTATE Boom VERT -35 5.50;
ROTATE Stick VERT 20 5.50;
TIME 434.08;
MOVE Truck1 BackUp 18.96;
TIME 434.80;
ROTATE ExcBucket VERT 90 5.23;
TIME 440.03;
ROTATE Stick VERT -20 5.65;
ROTATE Boom VERT 35 5.65;
TIME 440.03;
MOVE Truck3 LdToDmp 139.89;
TIME 440.03;
MOVE Truck4 Manuver 22.66;
TIME 445.68;
ROTATE ExcCabin HOR -90 8.62;
TIME 453.04;
ROTATE TruckBucket1 VERT 55 8.96;
TIME 454.30;
ROTATE Boom VERT -27 4.50;
ROTATE Stick VERT -10 4.50;
TIME 458.80;
ROTATE ExcBucket VERT -90 7.70;
TIME 462.00;
```

```
ROTATE TruckBucket1 VERT -55 4.77;
TIME 466.50;
ROTATE Boom VERT 27 5.02;
ROTATE Stick VERT 10 5.02;
TIME 466.77;
MOVE Truck1 ExitDmpArea 24.42;
TIME 471.52;
ROTATE ExcCabin HOR 90 9.19;
TIME 480.71;
ROTATE Boom VERT -35 4.20;
ROTATE Stick VERT 20 4.20;
TIME 484.91;
ROTATE ExcBucket VERT 90 4.44;
TIME 489.35;
ROTATE Stick VERT -20 5.16;
ROTATE Boom VERT 35 5.16;
TIME 491.19;
MOVE Truck2 EnterDmpArea 41.71;
TIME 491.19;
MOVE Truck1 DmpToLd 127.52;
TIME 494.51;
ROTATE ExcCabin HOR -90 9.19;
TIME 503.70;
ROTATE Boom VERT -27 4.01;
ROTATE Stick VERT -10 4.01;
TIME 507.71;
ROTATE ExcBucket VERT -90 5.37;
TIME 513.08;
ROTATE Boom VERT 27 4.40;
ROTATE Stick VERT 10 4.40;
TIME 517.48;
ROTATE ExcCabin HOR 90 8.93;
TIME 526.41;
ROTATE Boom VERT -35 5.97;
ROTATE Stick VERT 20 5.97;
TIME 532.37;
ROTATE ExcBucket VERT 90 5.66;
TIME 538.03;
ROTATE Stick VERT -20 4.72;
ROTATE Boom VERT 35 4.72;
TIME 538.03;
MOVE Truck4 LdToDmp 112.93;
TIME 538.03;
MOVE Truck5 Manuver 28.37;
TIME 538.36;
MOVE Truck2 BackUp 24.68;
TIME 542.75;
ROTATE ExcCabin HOR -90 6.64;
TIME 549.40;
ROTATE Boom VERT -27 4.70;
ROTATE Stick VERT -10 4.70;
TIME 554.10;
ROTATE ExcBucket VERT -90 7.41;
TIME 561.50;
ROTATE Boom VERT 27 6.45;
ROTATE Stick VERT 10 6.45;
TIME 563.03;
```

```
ROTATE TruckBucket2 VERT 55 11.75;
TIME 567.95;
ROTATE ExcCabin HOR 90 11.22;
TIME 574.79;
ROTATE TruckBucket2 VERT -55 5.81;
TIME 579.18;
ROTATE Boom VERT -35 5.66;
ROTATE Stick VERT 20 5.66;
TIME 580.60;
MOVE Truck2 ExitDmpArea 22.05;
TIME 584.83;
ROTATE ExcBucket VERT 90 5.07;
TIME 589.90;
ROTATE Stick VERT -20 5.87;
ROTATE Boom VERT 35 5.87;
TIME 595.78;
ROTATE ExcCabin HOR -90 6.12;
TIME 601.89;
ROTATE Boom VERT -27 4.80;
ROTATE Stick VERT -10 4.80;
TIME 602.65;
MOVE Truck3 EnterDmpArea 56.26;
TIME 602.65;
MOVE Truck2 DmpToLd 111.40;
TIME 606.70;
ROTATE ExcBucket VERT -90 5.12;
TIME 611.81;
ROTATE Boom VERT 27 5.80;
ROTATE Stick VERT 10 5.80;
TIME 617.61;
ROTATE ExcCabin HOR 90 12.99;
TIME 630.60;
ROTATE Boom VERT -35 5.43;
ROTATE Stick VERT 20 5.43;
TIME 636.04;
ROTATE ExcBucket VERT 90 4.69;
TIME 640.73;
ROTATE Stick VERT -20 4.57;
ROTATE Boom VERT 35 4.57;
TIME 640.73;
MOVE Truck5 LdToDmp 129.51;
TIME 640.73;
MOVE Truck1 Manuver 24.76;
TIME 645.29;
ROTATE ExcCabin HOR -90 7.47;
TIME 652.76;
ROTATE Boom VERT -27 5.00;
ROTATE Stick VERT -10 5.00;
TIME 657.76;
ROTATE ExcBucket VERT -90 6.62;
TIME 663.59;
MOVE Truck3 BackUp 23.11;
TIME 664.38;
ROTATE Boom VERT 27 6.67;
ROTATE Stick VERT 10 6.67;
TIME 671.06;
ROTATE ExcCabin HOR 90 9.01;
```

```
TIME 680.07;
ROTATE Boom VERT -35 5.45;
ROTATE Stick VERT 20 5.45;
TIME 685.52;
ROTATE ExcBucket VERT 90 5.63;
TIME 686.70;
ROTATE TruckBucket3 VERT 55 11.16;
TIME 691.15;
ROTATE Stick VERT -20 5.24;
ROTATE Boom VERT 35 5.24;
TIME 696.39;
ROTATE ExcCabin HOR -90 7.98;
TIME 697.86;
ROTATE TruckBucket3 VERT -55 5.59;
TIME 703.45;
MOVE Truck3 ExitDmpArea 24.01;
TIME 704.37;
ROTATE Boom VERT -27 4.59;
ROTATE Stick VERT -10 4.59;
TIME 708.96;
ROTATE ExcBucket VERT -90 6.72;
TIME 715.69;
ROTATE Boom VERT 27 6.93;
ROTATE Stick VERT 10 6.93;
TIME 722.62;
ROTATE ExcCabin HOR 90 9.83;
TIME 727.45;
MOVE Truck4 EnterDmpArea 58.01;
TIME 727.45;
MOVE Truck3 DmpToLd 116.20;
TIME 732.45;
ROTATE Boom VERT -35 4.89;
ROTATE Stick VERT 20 4.89;
TIME 737.34;
ROTATE ExcBucket VERT 90 5.84;
TIME 743.18;
ROTATE Stick VERT -20 5.92;
ROTATE Boom VERT 35 5.92;
TIME 743.18;
MOVE Truck1 LdToDmp 132.03;
TIME 743.18;
MOVE Truck2 Manuver 29.46;
TIME 749.11;
ROTATE ExcCabin HOR -90 6.96;
TIME 756.07;
ROTATE Boom VERT -27 4.90;
ROTATE Stick VERT -10 4.90;
TIME 760.97;
ROTATE ExcBucket VERT -90 5.38;
TIME 766.35;
ROTATE Boom VERT 27 4.85;
ROTATE Stick VERT 10 4.85;
TIME 771.20;
ROTATE ExcCabin HOR 90 13.61;
TIME 784.81;
ROTATE Boom VERT -35 5.12;
ROTATE Stick VERT 20 5.12;
```

```
TIME 789.93;
ROTATE ExcBucket VERT 90 5.44;
TIME 790.04;
MOVE Truck4 BackUp 15.48;
TIME 795.37;
ROTATE Stick VERT -20 5.39;
ROTATE Boom VERT 35 5.39;
TIME 800.77;
ROTATE ExcCabin HOR -90 6.22;
TIME 805.52;
ROTATE TruckBucket4 VERT 55 8.24;
TIME 806.99;
ROTATE Boom VERT -27 4.46;
ROTATE Stick VERT -10 4.46;
TIME 811.45;
ROTATE ExcBucket VERT -90 6.74;
TIME 813.76;
ROTATE TruckBucket4 VERT -55 4.27;
TIME 818.03;
MOVE Truck4 ExitDmpArea 16.21;
TIME 818.19;
ROTATE Boom VERT 27 4.79;
ROTATE Stick VERT 10 4.79;
TIME 822.98;
ROTATE ExcCabin HOR 90 10.63;
TIME 833.61;
ROTATE Boom VERT -35 5.41;
ROTATE Stick VERT 20 5.41;
TIME 834.24;
MOVE Truck5 EnterDmpArea 51.26;
TIME 834.24;
MOVE Truck4 DmpToLd 120.24;
TIME 839.02;
ROTATE ExcBucket VERT 90 5.99;
TIME 845.01;
ROTATE Stick VERT -20 4.93;
ROTATE Boom VERT 35 4.93;
TIME 845.01;
MOVE Truck2 LdToDmp 111.10;
TIME 845.01;
MOVE Truck3 Manuver 20.62;
TIME 849.93;
ROTATE ExcCabin HOR -90 8.55;
TIME 858.48;
ROTATE Boom VERT -27 4.07;
ROTATE Stick VERT -10 4.07;
TIME 862.55;
ROTATE ExcBucket VERT -90 6.53;
TIME 869.08;
ROTATE Boom VERT 27 6.97;
ROTATE Stick VERT 10 6.97;
TIME 876.06;
ROTATE ExcCabin HOR 90 12.44;
TIME 888.50;
ROTATE Boom VERT -35 4.60;
ROTATE Stick VERT 20 4.60;
TIME 891.17;
```

```
MOVE Truck5 BackUp 21.45;
TIME 893.10;
ROTATE ExcBucket VERT 90 5.24;
TIME 898.34;
ROTATE Stick VERT -20 4.94;
ROTATE Boom VERT 35 4.94;
TIME 903.28;
ROTATE ExcCabin HOR -90 7.69;
TIME 910.97;
ROTATE Boom VERT -27 4.86;
ROTATE Stick VERT -10 4.86;
TIME 912.62;
ROTATE TruckBucket5 VERT 55 9.46;
TIME 915.83;
ROTATE ExcBucket VERT -90 6.55;
TIME 922.08;
ROTATE TruckBucket5 VERT -55 5.59;
TIME 922.38;
ROTATE Boom VERT 27 6.60;
ROTATE Stick VERT 10 6.60;
TIME 927.67;
MOVE Truck5 ExitDmpArea 20.16;
TIME 928.98;
ROTATE ExcCabin HOR 90 10.13;
TIME 939.10;
ROTATE Boom VERT -35 4.27;
ROTATE Stick VERT 20 4.27;
TIME 943.37;
ROTATE ExcBucket VERT 90 4.05;
TIME 947.42;
ROTATE Stick VERT -20 4.02;
ROTATE Boom VERT 35 4.02;
TIME 947.42;
MOVE Truck3 LdToDmp 139.85;
TIME 947.83;
MOVE Truck1 EnterDmpArea 47.94;
TIME 947.83;
MOVE Truck5 DmpToLd 116.64;
TIME 951.44;
ROTATE ExcCabin HOR -90 7.74;
TIME 954.48;
MOVE Truck4 Manuver 23.26;
TIME 959.18;
ROTATE Boom VERT -27 4.60;
ROTATE Stick VERT -10 4.60;
TIME 963.78;
ROTATE ExcBucket VERT -90 6.03;
TIME 969.81;
ROTATE Boom VERT 27 5.60;
ROTATE Stick VERT 10 5.60;
TIME 975.41;
ROTATE ExcCabin HOR 90 8.93;
TIME 984.34;
ROTATE Boom VERT -35 4.68;
ROTATE Stick VERT 20 4.68;
TIME 989.02;
ROTATE ExcBucket VERT 90 5.60;
```

```
TIME 994.62;
ROTATE Stick VERT -20 5.09;
ROTATE Boom VERT 35 5.09;
TIME 999.71;
ROTATE ExcCabin HOR -90 9.83;
TIME 1000.90;
MOVE Truck1 BackUp 21.40;
…
```

VITA

Vineet R. Kamat was born on 11th January 1977 in Margao, India. He received his primary and high school education at Deepvihar in Vasco-Da-Gama. After graduating from high school in 1994, he started to pursue his baccalaureate degree in Civil Engineering at the Goa College of Engineering, Farmagudi. During the course of his undergraduate education, he became interested in the engineering and project management aspects of construction. After graduating with honors in 1998, he chose Virginia Tech to continue his academic pursuit by working towards his M.S. degree. At the time of completing this thesis, he hopes to continue his academic endeavor by pursuing his Ph.D. degree next. His parents live in India. He was married in 1999. His wife, Sonia is a dentist who now wishes to pursue her MBA.