A VRML Integration Methodology for Manufacturing Applications

Sandy Ressler* Afzal Godil
Qiming Wang Gregory Seidman

Information Technology Laboratory
National Institute of Standards and Technology

Abstract

This paper describes several methods for using the Virtual Reality Modeling Language (VRML) as the visualization integration technology for manufacturing simulation systems. One of our goals was to develop an integration methodology based on the use of VRML translators to produce reusable VRML components. The use of readily available off-the-shelf VRML models and systems was a major requirement. In addition to the component libraries we also wanted to add a significant analytic system to demonstrate potential application in a real-world manufacturing simulation system. This resulted in the integration of a near real-time dynamics engine with the VRML world. The production and use of intermediate component worlds demonstrates the potential for component libraries of visual manufacturing elements that can be integrated into larger simulation and visualization environments.

CR Categories and Subject Descriptors: D.3.2 [Programming Languages] Languages/Classifications - Virtual Reality Modeling Language 2.0; H.5.1 [Information Interfaces and Presentation] Hypertext navigation and maps; I.3.2 [Computer Graphics] Graphics Systems - Distributed/network graphics; I.3.6 [Computer Graphics] Methodology and Techniques - Interaction techniques

Additional Keywords and Phrases: virtual environments, user interfaces, manufacturing environment, systems integration.

1 INTRODUCTION

The Virtual Reality Modeling Language (VRML) is a recently ratified ISO standard (ISO 14772) [11] file format for the description of geometry and behavior of 3D computer graphics. We have used VRML as a canonical output format from a variety of systems as well as with off-the-shelf VRML authoring tools, to produce integrated virtual worlds for manufacturing proof-of-concept applications.

*National Institute of Standards and Technology
Bldg. 225, Rm. A216
Gaithersburg MD 20899
email: sressler@nist.gov, agodil@nist.gov,
qwang@nist.gov, gseidman@acm.org

2 BACKGROUND

Our VRML work is part of a larger Systems Integration for Manufacturing Applications (SIMA)[9] program at the National Institute of Standards and Technology (NIST) performing research in advanced manufacturing and the applications of information technology. VRML provides a robust foundation upon which one can build applications that are portable, distributable via the World Wide Web, and integratable with analytic systems. Much of our motivation stems from a desire to provide methods to share engineering information, visually.

VRML, as an international standard, is well suited as a file format for exporting from proprietary systems. When engineers collaborating on a particular project must share information, VRML is an additional tool they can use to facilitate their communication. One major obstacle to increased use of VRML models is the difficulty of integrating models produced by different systems and people. Furthermore, enhanced analytic functionality, often a requirement for the production of a useful system, such as that provided by a dynamics engine, is also difficult to integrate with larger more complex worlds.

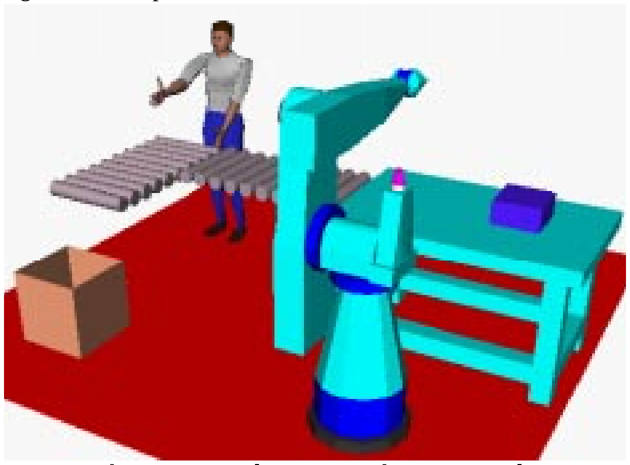


Figure 1. Early Integration Example

Figure 1 illustrates an early proof-of-concept effort at integrating VRML originating from several sources. The world contains models created via three different translators, a robot from Deneb's IGRIP software [12], the conveyor belt from Working Model 3D [13], and a human from Transom's Jack[10]. When the

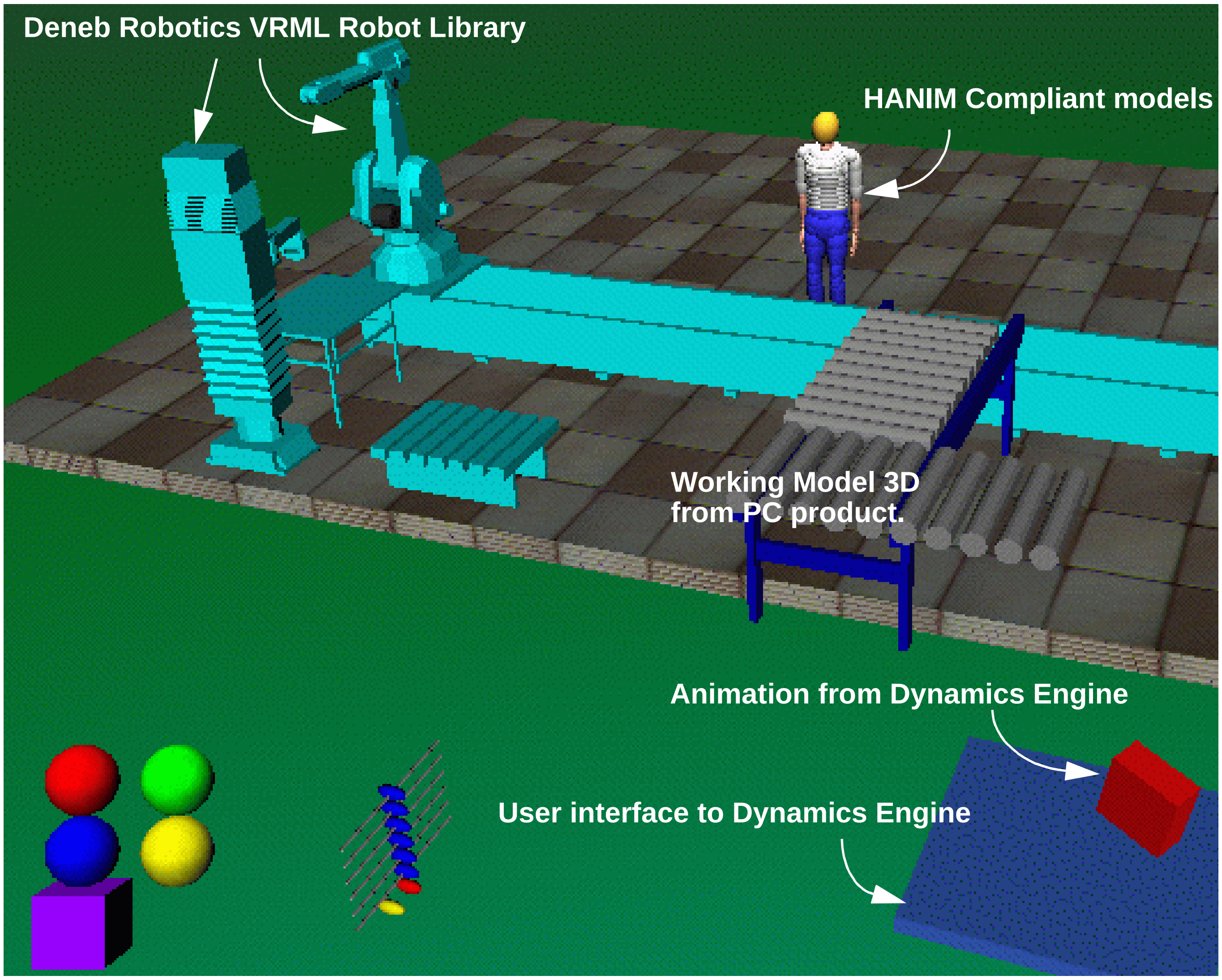

Deneb Robotics VRML Robot Library
HANIM Compliant models
Working Model 3D
from PC product.
Animation from Dynamics Engine
User interface to Dynamics Engine

Figure 2. Overview of entire integrated world

user clicks on the floor the robot picks up the box from the table, places it on the conveyor belt. As the box travels along the conveyor belt, at the appropriate time, the human appears to push the box, which continues along the conveyor belt until it falls off into the box. We wrote the translators for the Deneb and Working Model 3D packages and obtained the Jack translator from the University of Pennsylvania's Center for Human Modeling and Simulation.

3 RELATED WORK

The use of physically based modeling for simulation systems is of course, not a new field. We present work here which combines both the use of physically based modeling integrated with VRML and with off-the-shelf components for robots, humanoids, and motion sequences. The idea is to have the dynamics simulation produce, on the fly, animated sequences representative of the dynamics calculations rather than precomputed keyframes for an animation. S. Chenny et al. [6] have demonstrated the use of a dynamics system rather then keyframes as well. We, specifically set out to use an off-the-shelf dynamics system and not one of our own creation. Also of interest is the work of D. Brutzman [5] and the DIS-Java-VRML Working Group of the VRML Consortium, in the establishment of an infrastructure for linking various simulations. Our work does not rely on a specific protocol; it is a proof-of-concept methodology for demonstrating how distinctly different types of components can be integrated.

4 SYSTEM OVERVIEW

The overall system consists of four pieces. Robots from the Deneb VRML Robot Library, two HANIM[7] compliant humanoids, a Working Model 3D animation produced on a PC, and finally an integrated Working Model Dynamic Modeling Engine (DME) system[8]. Figure 2 illustrates the overall configuration of the final VRML world.

The user clicks on a pallet in the center of the world and one robot picks up a widget and places it into a red box on the table. The other robot transports the red box to the conveyor belt where it is moved to the end of the conveyor belt and falls onto the blue plate. The blue plate may be adjusted by the user which causes the dynamics engine to recalculate the dynamics of the box hitting the plate. The humanoid can walk to a position along the working area when the user clicks on the appropriate command interface (the four spheres in the left side of the scene). A different humanoid may be selected, by clicking on the box on the left side of the scene as well.

5 INTEGRATION METHODOLOGY

Each individual component in our world was created from a translator or from an off-the-shelf source. We wanted to demonstrate the feasibility of using translators and off-the-shelf VRML components to create a coherent integrated VRML world.

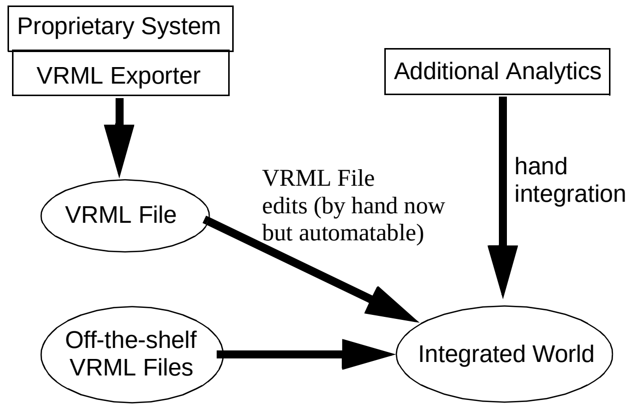


Figure 3. Integration Methodology

In addition to the relatively straightforward geometric integration of different VRML files, we enhanced the overall functionality of the world by adding an analytic component in the form of a dynamics engine.

Two different types of translators were used for the creation of two portions of the environment. First a translator was created on the PC platform and embedded with the Working Model 3D product. As an aside, our translator was eventually incorporated into the Working Model 3D product and is available with the current version of the product. The other translator was the Deneb translator which has existed for approximately two years [12]. Rather than simply use a particular robot model however we first created a library of robots from which we can choose an appropriate robot.

To integrate another robot from the robot library the following steps are followed:

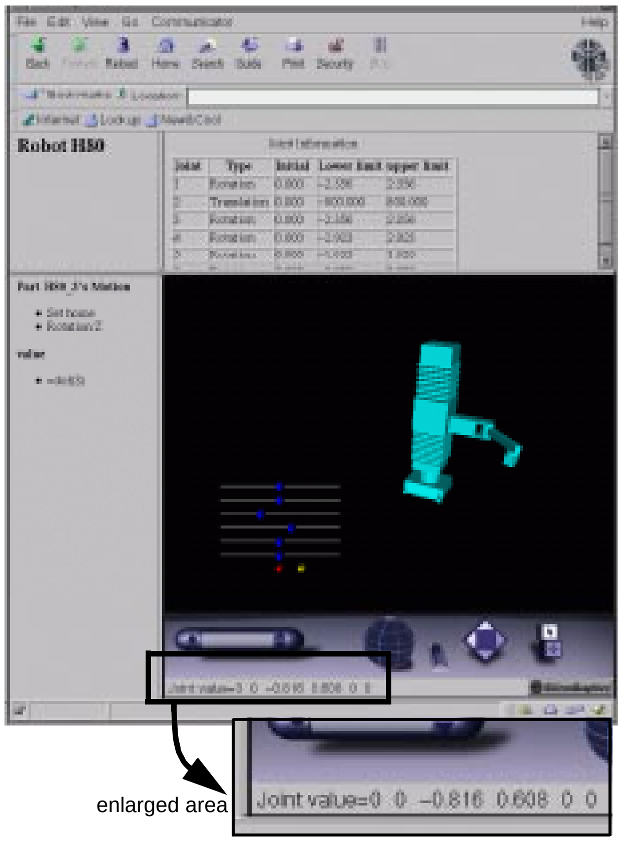
(1) Place the robot VRML from the library into the world. Currently this means simply to copy the main VRML robot file to your world, and all the files in its sub directory to an identical directory structure in the working directory. A utility to automate this is planned.

(2) Comment out the ROUTE WHERE.orientation_changed TO and the ROUTE WHERE.position_changed TO in the PROTO `Dash` (Dash is the dashboard control panel).

(3) Use sliders to identify which joints to move and their values. When the sliders are moved (one for each degree of freedom) values of the joints are displayed on the Browser window pane as illustrated in Figure 4.

(4) Create `ScalarInterpolators` in the `PROTO Dash`, each `ScalarInterpolator` refers to one joint, the keyValue of the `ScalarInterpolator` is the joint value found using the sliders.

(5) Finally, add ROUTEs in the `PROTO Dash` to route the `TimeSensor` in Dash to the `ScalarInterpolator` and route ScalarInterpolator's `value_changed` to the robot joint.


File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location
Internet Lookup New&Cool
enlarged area

Robot H80

Joint Information

Joint	Type	Initial	Lower limit	upper limit
1	Rotation	0.000	-2.356	2.356
2	Translation	0.000	-800.000	800.000
3	Rotation	0.000	-2.356	2.356
4	Rotation	0.000	-2.903	2.903
5	Rotation	0.000	-1.000	1.000

Part H80_2's Motion

- Set home
- Rotation2

value

- -0.0833

Joint value=0 0 -0.816 0.608 0 0

Figure 4. Robot Library Interface

It is our intent to automate this process where possible. Decisions about the animation and how they should trigger other animations and actions are a subject of further research.

In the case of the Working Model 3D (WM3D) VRML files, we did not progress quite so far in terms of the reusability. Each VRML file produced from the WM3D translator is actually a collection of coordinated animations. Each sub-object within the animation has its own timing. The translator simply produces a file. Given the requirements of reuse however we can see that it would be useful to modify the translator to produce VRML code with "hooks" for reuse. These hooks are such capabilities as:

(1) A single point of entry to start the entire animation.

(2) The ability to specify certain sub-animations as having a start and stop time that can be externally controlled.

(3) The ability to specify certain sub-animations as continuing forever.

Given a clearly defined set of VRML requirements we can go back and fairly quickly modify the translator to produce VRML that meets the new requirements.

6 DYNAMICS ENGINE INTEGRATION

The most significant component integrated into this world was the dynamics engine. We deliberately chose to use an existing off-the-shelf dynamics engine to get a realistic view of integration issues. We used the Dynamic Motion Engine (DME) [8] from Knowledge Revolution for this task. Knowledge Revolution's primary product is a complete dynamics simulation and animation package that runs on PCs under the Windows OS. The DME system we used however runs on a Silicon Graphics workstation under the IRIX OS.

Physical based modeling has emerged as an important method to simulate animations for virtual environments. Until recently the majority of animations in VRML world has been created with key-framing, resulting in fixed animation sequences. An alternative is to couple physical based modeling to the VRML world which allows for constant changes in parameters and properties, creating new animations. We have implemented a web based virtual environment based on VRML and Java where the animations are based on physical dynamics. Physical based modeling is inherently a CPU intensive task and we did not investigate those issues however others have [6]. The dynamics engine we used was Dynamics Motion Engine (DME) from Working Model Inc. Baraff and Witkin provide a particularly useful reference, available on-line [3].

Our VRML/Java system is based on a two-tier client/server model. Since our client program runs on any web browser that has support for VRML and Java. The server is based on the C++ DME API, hence only runs on a particular server and provides access to real world Newtonian mechanics. Figure 5 illustrate the overall system architecture. The DME calculates the motion of interacting bodies using advanced numerical analysis techniques. The engine can simulate complex multibody systems, compute their motion under a variety of constraints and forces. In addition to user-imposed constraints, such as springs, dampers, or joints, the engine has the capability to simulate collisions, gravity, and external loads conditions. Each body has a set of physical properties, including mass, inertia, position, velocity, and coefficient of restitution (elasticity) and friction.

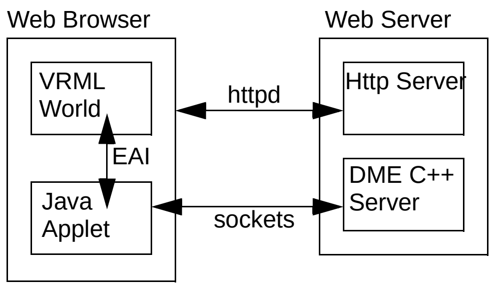


Figure 5. Dynamics Engine Interface Architecture

The DME server must run on the same physical machine as the Web server because of the security restrictions imposed by the use of a Java applet.

Figure 6 illustrates a test world we developed where the user can adjust the elasticity and angle of the target plate. A box travels along a conveyor belt and falls off of the end onto a target plate. The red box (A) is from an animation that is precomputed, created from the WorkingModel 3D PC application and the green box (B) is the animation calculated by the DME server. The user can either type in numbers for the plate angle and elasticity or can adjust the angle of the plate directly with the cursor.

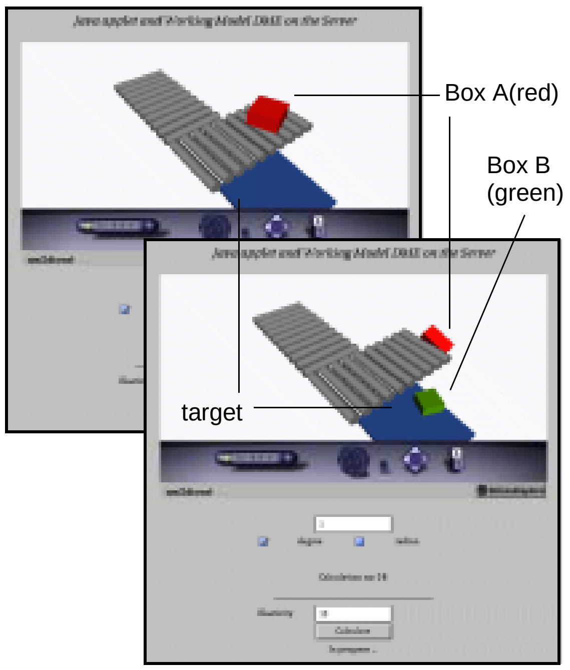


Figure 6. Dynamics Engine Test World

The user adjusts the target plate causing the following sequence of events:

(1) The user directly moves the plate angle, creating an event_out and a new plate angle.

(2) The Java applet, via sockets, sends the DME server the new plate angle.

(3) The DME server calculates a set of new position and orientation values.

(4) The DME server sends back to the Java applet the new positions and orientations.

(5) The Java applet parses these new values and modifies the Position and Orientation Interpolators of the animation via the External Authoring Interface (EAI).

The entire sequence typically takes from 3-5 seconds, and the display is static during that time.

7 INTEGRATION OF ANIMATION

The result of a simulation created on a PC with Working Model 3D is a VRML animation. This animation was placed into the larger environment and had to be "wired up" by hand. The logical wiring was simply a matter of creating ROUTES from the appropriate interpolators to start and stop animations at the correct time. In our particular example the conveyor belt animation was started by the completion of the robot animation. The end of the conveyor belt animation started the dynamics engine animation. The visual alignment of these elements however, was tedious and time consuming. The animations were lined up by hand to flow in a visually seamless manner. While it is clear that the logical elements could be automatically wired by identifying start and stop interpolators, the visual aspect is much more problematic.

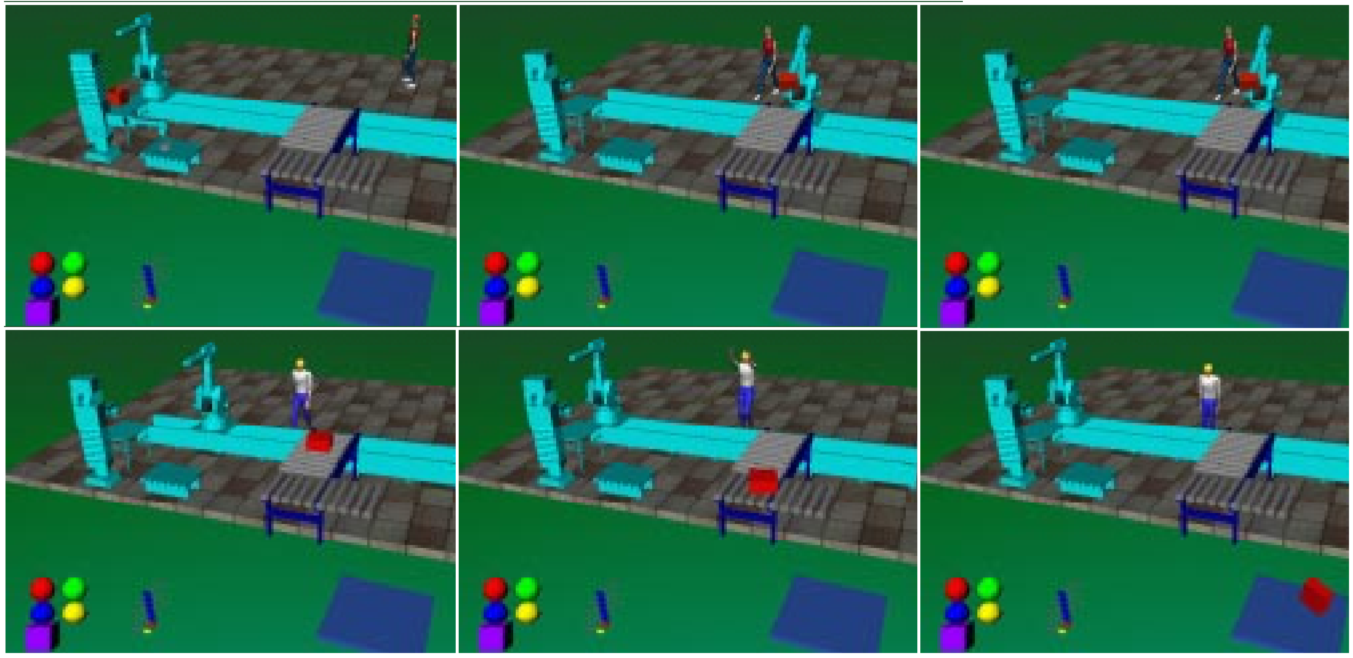


Figure 7. Animation sequence in fully integrated world.

8 HUMANOIDS

One of the key components in a real manufacturing environment are the people. People on assembly lines in semiautomatic plants are often the most difficult element to characterize and simulate. In our integrated VRML world we wanted to demonstrate the ability to introduce humans of different types. Given the existence of the HANIM 1.0 specification for the description of VRML humanoids, we felt it important to at least provide some human models in the environment. We simply “borrowed” some existing well known HANIM models. Nancy created by Cindy Ballreich of 3Name3D[2] and the Bimos model from Matt Beitler[4] of the University of Delaware were used and their owners generously allowed the use of their models for our demonstration.

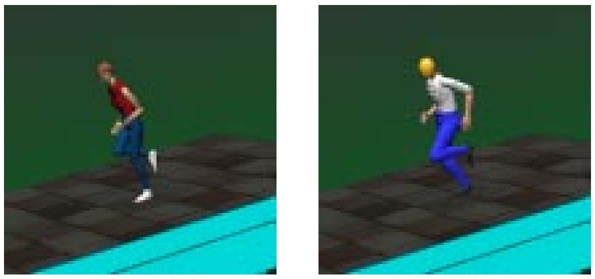


Figure 8. Nancy and Bimos Switchable Humanoids

Our goal was to demonstrate the interoperability of these HANIM models with our environment. The animations in the Nancy model were able to drive both the Nancy and Bimos figures. We simply added a screen element (a cube) to function as a toggle to switch between the Nancy and Bimos figures. In addition we modified the animation to cause the figures to walk along a fixed path so that the figures took their proper place on the assembly line.

The path walking utilized a `TimeChain` PROTO we developed which allows a `TimeSensor` driven animation to, upon completion, trigger another animation. The walking of the humanoids were a series of three animations linked by `TimeChain` Nodes. This PROTO is also useful for coordinating other animations.

We currently have software capable of generating VRML humans of arbitrary sizes based on work dealing with child anthropometry[1]. The VRML figures however are not HANIM compliant and the files are somewhat large so we decided to leave conversion to HANIM for the future.

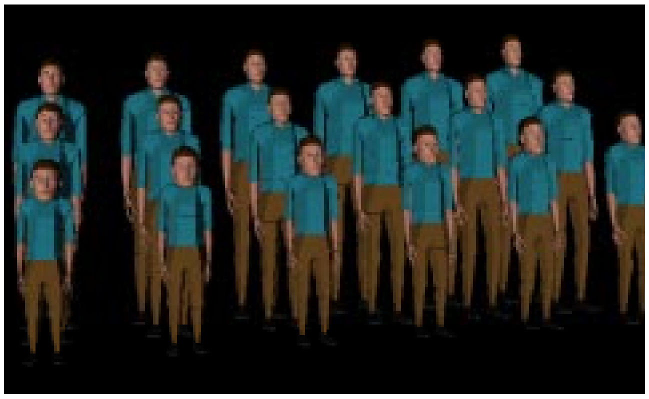


Figure 9. VRML Humans of Different Sizes from Transom Jack

9 INTEGRATION ISSUES

The integration of worlds produced from different sources is clearly a difficult problem. Some of the difficulties can be mitigated by the use of automated translators. As we discover new integration issues we can modify the translator to accommodate our new requirements. The integration problems can be classified in three ways, (1) geometric, (2) behavior, and (3) interaction. Geometric issues are those such as rotated coordinate systems and vastly different scales. Behavioral issues are issues concerning a lack of simple hooks for starting and stopping animations dependent on one another. Interaction issues concerns problems integrating a user interface, often created separately in stand-alone worlds which must be combined in a coherent integrated world. Each of the three integration categories presents its own unique challenges some relatively simple (geometric) and others clearly open to further research.

The geometric integration issues are the most straightforward of the three categories. Integration can be made much simply by development of a geometric normalizer which would orient and scale all geometries to their “standard” orientation and size.

For behavioral integration a standard naming convention for interpolators would enable automated ROUTEs a

In the case of the Deneb robot library behavioral integration was the primary issue we encountered. We can substitute (theoretically) any of the robots for each other. If we clearly identify the particular joints of interest we can wire up interpolators to those joints. In the case of the Working Model 3D animations, we can modify the translator to identify the last action of the animation which is used subsequently to start the animation produced by the dynamics engine. A recommended practice and conventions for the production of named interpolators would be of great use in these cases.

The integration issues for interaction are perhaps the most difficult. Small component worlds often have little widgets or other graphical elements for user interaction. As one integrates several component worlds the user interface elements must be either reworked or integrated. The simplest approach to address this area is to create worlds that do not contain embedded controls, and rely on external, in the HTML page control that takes advantage of the EAI. Clearly this is also not a sufficient and further research is necessary.

10 CONCLUSIONS AND FUTURE WORK

We have demonstrated the integration of collections of various types of VRML worlds. In addition we have also demonstrated the integration of an additional analytic component, a dynamics engine, into this complex VRML environment. This type of integrated world allows engineering systems to be visualized and communicated via the Web. Proprietary systems retain their value and their strengths in performing particular tasks, and the addition of a VRML export and integration path allows these proprietary systems to leverage the value of Web publishing.

In the future we will develop automated and/or semi-automated programs to ease the integration process. Lessons learned from the various integration issues we encountered, will lead to improved integration tools that take advantage of built in hooks embedded in the generated VRML.

11 ACKNOWLEDGEMENTS

We would like to thank the continuing sponsorship of the Systems Integration for Manufacturing Applications (SIMA) program for supporting this work. Thanks to Cindy Ballreich and Matt Beitler for allowing us to use their Nancy and Bimos HANIM humanoids. Thanks to Gary Yang, one of our intrepid summer students for his work with Jack. Finally thanks to Sharon Laskowski for her typically ruthless editing of this paper.

References

[1] AnthroKids web site. *http://ovrt.nist.gov/projects/anthrokids/*

[2] Cindy Ballreich, 3Name3D *http://www.ywd.com*

[3] D. Baraff, A. Witkin, Siggraph 97 Course notes from “Physically Based Modeling: Principles and Practice” *http://www.cs.cmu.edu/~baraff/sigcourse*.

[4] Matt Beitler, Bimos, *http://www.asel.udel.edu/~beitler*

[5] D. Brutzman “Distributed Interactive Simulation DIS-Java-VRML” *http://www.stl.nps.navy.mil/dis-java-vrml/#Overview*

[6] Chenney, Ichnowski, Forsyth, Efficient Dynamics Modeling for VRML and Java, in *Proceedings of VRML98* Monterey CA Feb 1998.

[7] Humanoid Animation Working Group (HANIM) *http://ece.uwaterloo.ca:80/~h-anim/*

[8] Knowledge Revolution Inc., San Mateo, CA. *Dynamic Motion Engine API Reference*, Version 1.2, Sept. 1997.

[9] Systems Integration for Manufacturing Applications (SIMA) Program *http://www.nist.gov/sima*

[10] Transom Jack User Manual V 1.1 Ann Arbor MI 1997.

[11] VRML. *VRML 2.0 Specification ISO/IEC CD 14772*, 1996.

[12] Q. Wang, S. Ressler, Translating IGRIP Workcells into VRML2, NISTIR 6076, *http://www.nist.gov/itl/div878/ovrt/projects/vrml/deneb2vrml.html,* Sept. 1997.

[13] Working Model 3D Translator to VRML, *http://ovrt.nist.gov/work3d/*