An Efficient Object-oriented Authoring and Presentation System for Virtual Environments

Wolfgang Müller

Darmstadt University of Technology
Darmstadt, Germany

Ralf Dörner, Volker Luckas, Arno Schäfer

Fraunhofer Institute for Computer Graphics
Darmstadt, Germany

Abstract

In this paper we describe the design and implementation of a multi-purpose object-oriented authoring system for interactive virtual environments and presentations, semi-interactive 3D visualization, and non-interactive photo-realistic animations. Its main design criteria are reusability and flexibility, especially in the support of a wide range of input and output formats.

Input data may be provided from any type of event-oriented simulation system as well as from an interactive scene modeling tool. The primary output system is a Java-based VRML browser, but a concept of output driver channels allows the use of other rendering systems, such as off-line renderers for the production of photo-realistic video streams.

Objects used in the system consist of both geometry and behavior, where the latter is defined in C++ and/or Java. Objects are provided in different abstraction levels and are stored in an animation element library for reuse in different application scenarios.

Keywords: *Authoring Systems, Virtual Environments, Simulation, Visualization, Animation, WWW, VRML.*

1. INTRODUCTION

Virtual Reality and interactive 3D applications are gaining increasing importance in various application domains. Beyond the large area of computer games and entertainment, these techniques add new dimensions in areas such as presentations [4], visualization of planning and simulation data [3], teaching and training applications, three-dimensional graphical user interfaces, and medical applications [12].

Interactive 3D presentations have some important advantages: for example, they allow the analysis and validation of 3D models, data, and simulation results by interactive visual examination in a natural and intuitive way. Spatial arrangements in three-dimensional scenes in connection with time-related events can be mediated and explored very effectively by the use of interactive navigation and animation in virtual environments. However, not only analytical processes may be supported using these techniques. Sometimes even more important, the ability to create a connection between an abstract 3D representation and real world objects by applying suitable metaphors makes interactive virtual environments a valuable technique for communicating information, data, and ideas to other people, especially to non-experts.

In spite of the advantages of virtual environments for the presentation of data and information, such techniques are rarely applied in multimedia applications until now. Here, the central problem is not the technical feasibility, but the creation of the 3D content and the layout definition of such interactive virtual environments. Using standard tools from the areas of Computer Aided Design (CAD) and 3D Computer Animation, these processes are usually cumbersome, very time consuming, and expensive. This is because tools and techniques in these application areas were primarily designed for high- quality animations, non-interactive media, and well and a-priori defined presentation channels. Moreover, these tools are customized for experienced and well-trained users, which makes their application difficult for non-experts. Flexibility, scalability, reusability of animation components, and ease of use were typically not the primary design criteria. The resulting complexity of content creation explains why 3D virtual environments and animations did not evolve into generally used components in multimedia presentations until now.

In this paper we present the concept of an authoring environment for interactive 3D animations and virtual environments. Our solution is not designed as one large monolithic application but consists of a toolkit with specifically designed subsystems. This approach enables the easy exchange of components such as modeler and viewers. Thus, it provides high flexibility in the world of fast changing interactive media and varying application domains. Moreover, we present a prototype implementation following this concept, based on a newly developed animation package and a specifically designed VRML 2.0[1] browser. Finally we present first results in the application of this prototype in the 3D visualization of simulation data from logistic processes.

[1] Virtual Reality Modeling Language [2]

2. CONCEPT

2.1 System Overview

Instead of having one tool to generate a virtual environment, our concept is based on several independent and coordinated tools at the author's disposal to determine the geometric formation of a 3D scene as well as the behavior of its objects.

This concept is not new for animation and rendering systems as well as visualization systems. Very early animation systems such as Clockworks [7] and state-of-the-art visualization systems such as AVS [15] and Khoros already made use of such a flexible concept. However, this approach has been applied seldom to support the authoring of interactive and behavioral information entities.

The authoring and presentation environment components and their connections are depicted in our architectural concept (see Figure 1). The most important components of the CASUS system are the Animation Element Library, the Animation System, the Scene Editor, and the Presentation Toolkit. Simulators, SPS-Control Units, and additional modelers may be connected to this system as needed. In this chapter, we will describe the CASUS system components in more detail. Moreover, we illustrate their interaction and interfaces.

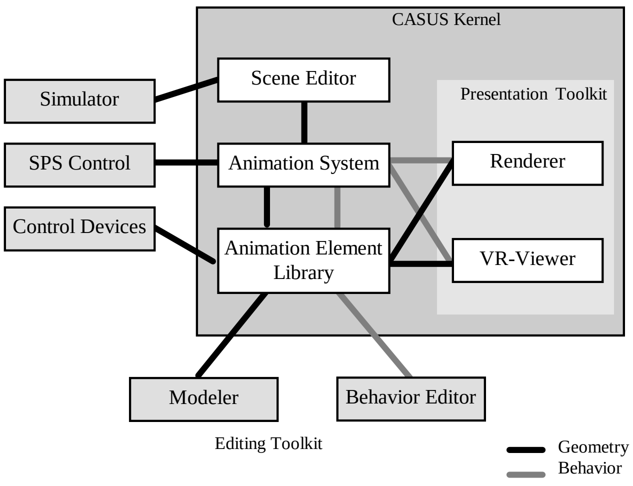


Figure 1: Architecture of the CASUS Authoring and Presentation System

2.2 Animation Element Library

The fundamental idea of our concept is to provide a comprehensive animation element library. Animation elements are the basic objects that can be used to build an interactive 3D scene [10]. An animation element consists not only of a description of its visual representation, but also of a specific behavior provided by this element (see **Fig.2**).

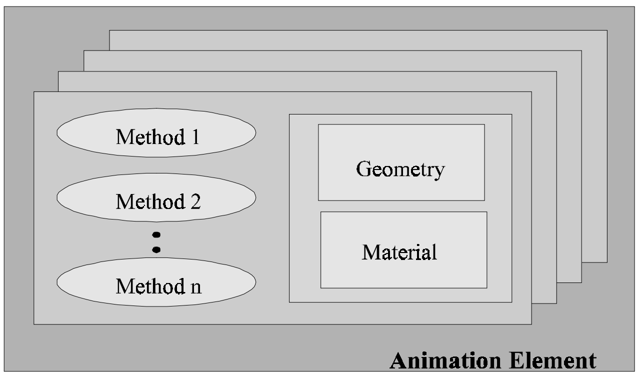


Figure 2: Structure of an Animation Element

While the visual representation is determined by the element's geometry and material characteristics, the behavior is represented by object-specific methods. Each method is linked to a previously described animation. For instance, the element "Person" may provide abstract behavior in terms of methods such as "walk", "sit", "stand up" or "take". Figure 3 shows an example for the use of these methods to create a complex animation.

Figure 3: Example of an animation script

```
peter = new Person( )
peter.scaleLength(1.82)
parcel = new Box
position1 = (3,4,4)
position2 = (3, 4.5, 4)
position3 = ( 7, 7, 7)
time = 0.0
peter.place(time, position1)
parcel.place(time, position2)
peter.take(time, parcel)
peter.walk(time, time+5,
           position2, position3)
```

Two animation elements, a person and a box, are instantiated. In this example the element "Person" takes the box and walks to a certain point. Because of the predefined visual representation and the provided high-level behavior, the author neither needs to model a new visual form, nor to animate the realistic walking of the person - both aspects are provided automatically by the animation element. Additionally, intelligent behavior such as scaling of geometry without distortion is integrated in the animation elements. This leads to a remarkable speedup of the animation, authoring and scene generation process.

Another important concept of the animation elements is the support of a variety of visual representations in animation elements, which correspond to different levels of detail and abstraction. This concept is able to support various media and presentation channels and to switch between such abstraction levels on-the-fly. For example, very detailed representations can be used for high-bandwidth channels such as off-line rendering, while symbolic and more abstract representations may be used for interactive real-time presentations. The same concept holds for the element-specific functionality and behavior, thereby providing animations of varying complexity and levels of abstraction for different client systems.

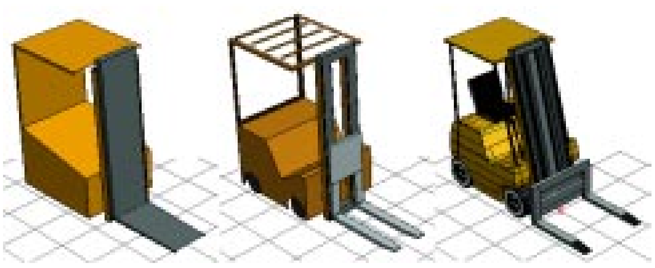


Figure 4: Different Representation forms of a Forklift

In this context, it should be emphasized that animation elements encapsulate geometry and behavior in the sense of the paradigm of object-orientation, making both form and behavior convertible and exportable. In addition, this concept guarantees the reusability of animation elements. It makes it possible to store animation elements in a suitable database offering corresponding retrieval functions. If a required element is not available, the author can create it or modify an existing element accordingly with the help of a modeler. It is also possible for the author to determine the animation element's functionality with a behavior editor.

2.3 Scene Editor

The first thing that has to be done is the creation and initialization of the 3D virtual environment. In order to simplify the creation of the scene's spatial layout a scene editor is offered. With the help of this tool the author is able to interactively place objects for instance symbolic representations of the animation elements and scale them to match the desired layout. After finishing the placement of the objects the initial 3D virtual environment is automatically created making use of the animation elements' encapsulated functionality and considering their initial state.

2.4 Animation System

The purpose of the animation system is to generate time-dependent positions and movements (transformations) of the animation elements from higher level behavior descriptions. Furthermore, the camera can be animated and the rendering parameters (e.g. illumination parameters) can be established. The behavior is specified using the animation elements' specific functionality as shown in the script example in Figure 2. These descriptions must be implemented by the author, or can be acquired through corresponding interfaces to systems such as simulators, SPS[2] programs, or sensors. The animation system transforms this specification into a description suitable for the addressed presentation system.

An intermediate internal representation of the behavioral description is used as a basis for the generation of output to various viewers and media. Temporal conditions such as different frame rates or mappings between simulation and presentation time can thus be handled.

2.5 Presentation Toolkit

In our concept we provide two different kinds of presentation systems, called *renderers* and *VR viewers*. Rendering, which may be performed off-line, produces images or video sequences at a high quality level.

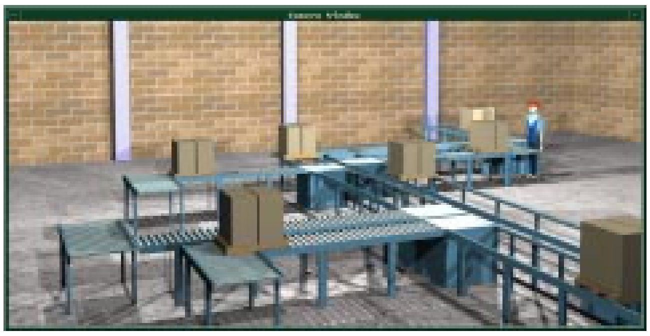


Figure 5: Still image from an example scene produced by an alternative high-quality renderer. In this scene the animation elements were evaluated at a higher level of detail.

These are by nature non-interactive, but can be integrated in interactive virtual environments, for instance, as textures. In contrast to the renderer, a VR viewer offers interaction to the user with appropriate input devices. Interactive camera control and user-controlled movement in virtual space are feasible. The user can also interact with the objects, whose object-specific functionality is still available in the viewer, due to the animation elements' object-oriented approach.

2.6 Discussion

The modular approach allows the exchange of components as well as the use of existing systems, such as animation systems or renderers, after a few adaptations have been made. Furthermore, it is possible to separate animation and visualization into individual subsystems. This is beneficial when distributing animated VR scenes over networks. The scene description can be adapted to various network bandwidths and viewers, due to the flexibility of the animation system. Computation requirements on the

[2] Storage Programmable Steering

machine used for visualization may be lower than those for the animation system. Time and cost efficiency is increased through the scalability and flexibility of the concept presented. The same scene can be easily visualized on various visualization platforms. Behavioral descriptions can be created by external systems and do not have to be defined by the author. However, probably the biggest increase in efficiency is due to reusability. For one, existing systems can be reused for data input, tapping large supplies of existing geometry data. Secondly, the object-oriented animation element concept enables the reuse of modular units.

3. REALIZATION

The CASUS system implements the concepts presented in this paper. In this chapter we will describe the realization of the three main tools, called CASUS Base, CASUS Anim and CASUS Presenter.

3.1 CASUS Base

In co-operation with professional designers we have modeled over 100 animation elements in three representation forms. This library of animation elements is the key issue of the CASUS Base system [5]. Until now most of the elements belong to an industrial context and, as a result of a survey, they provide 80% of the objects needed for building virtual environments in this area.

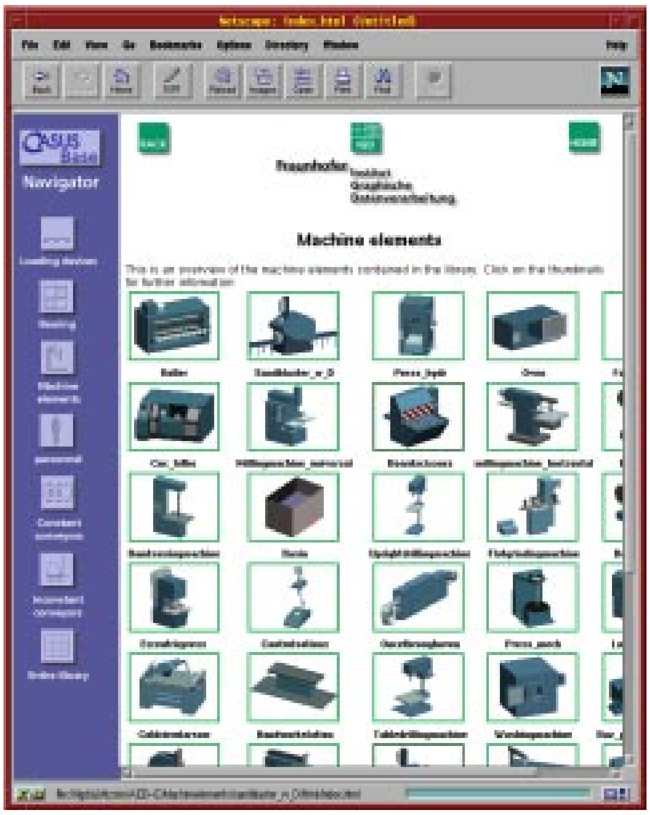
The realization of the elements follows strictly object-oriented design patterns. In addition, we implemented a WWW-based browsing system for the animation elements that shows the object specific functions in a short animation. An extension of the animation element library to other application domains is on the way.

3.2 CASUS Anim

In order to realize the animation system component of our concept we implemented CASUS Anim, an object-oriented three-dimensional animation system, described in detail in [11]. CASUS Anim complies the requirements specified in our conception like supporting the concept of animation elements.

The concept of CASUS Anim does not fit in the classical concepts of animation systems which are known as keyframe animation systems, parametric systems, programmable systems, simulation or model based systems. Based on the desired flexibility and concerning the elements to be animated, a combined animation system was designed, which is parametric as well as programmable. Furthermore, the system must support loading different objects which were saved for specific animation patterns. Therefore, an object-oriented system design seems to be the right choice. This conclusion was already drawn partially in related work [6], however, our strategy follows this concept in more consequence.

There exists the possibility of combining various implementations of animation elements and their behavior patterns. This way, the animation functionality of the entire system can be easily adapted to various conditions. Furthermore, the implementation language for the animation plays a distinct role. It is used to define the desired animation sequences, including the behavior macros as well as the entire animation sequences. It is designed to supply complex structures and be easy to handle, too.



Netscape: index.html (Untitled)
File Edit View Go Bookmarks Options Directory Window Help
CASUS Base
Navigator
Fraunhofer Institut Graphische Datenverarbeitung
Machine elements
This is an overview of the machine elements contained in the library. Click on the thumbnails for further information.

Figure 6: WWW-based Catalogue of CASUS Base

One of the outstanding features of CASUS Anim is that it allows the integration of data from arbitrary event-oriented simulators. For this purpose CASUS Anim provides additional tools including a simulator trace normalizer and a trace translator ([9], [11]). In general a simulator supplies a complete listing of all simulation events and their corresponding objects or components, the so called trace. The trace is converted to a unique format, using the normalizer. Using the normalized trace, the translator generates an animation script, referring the objects or components in the simulation to the animation elements. The animation system creates an executable animation by processing the animation script, which contains a unified description of all simulation events.

Moreover, CASUS Anim is specifically adapted to CASUS Base, which contains a complete description of all animation elements available. Also, the various events of simulation are translated to a descriptive form which can be used within the animation. The animation system processes this script and creates a rendering script depending on the connected visualization system. For instance VRML can be used as output format to achieve distributability via WWW. The rendering script holds all information necessary for the 3D animation, again referring to the library of animation elements.

The flexibility of the output format raises the problem how an animation system can convert its output to different media types making use of various scene description methods, e.g. frame-based or procedural.

CASUS Anim uses both frame-based and procedural description components. The frame-based concepts are needed to support key-framing while procedural components are used to realize the programmable scripting interface. In order to make clear the advantages of our concept we will exemplary discuss the different ways to generate VRML as an output format. Referring to the internal structure of CASUS Anim there basically exist three possible solutions. First, one can implement a conversion of frame-based output into frame-based VRML. Although this is a working solution, the generation of an online visualization is obviously impossible to establish in this way. The conversion is independent from the animation system but cannot use the animation elements' internal functionality. This is equivalent with a severe information loss during the presentation generation.

An alternative is to implement a functionality to directly generate frame-based VRML referring to the internal data structure of the animation system. The animation system itself is responsible for parsing and processing the necessary information. Hence the information is already translated into the specific media when send to the presentation system using a specific driver. So it is possible to provide online visualization allthough there is still an information loss because of the frame-based concept. The last and most efficient way is the synthesis of frame-based and procedural information following the element-based concept directly. In this case the scripting interface of VRML is extensively used with the help of Java programs. These programs represent the procedural (object-specific) information of the animation elements and fit directly into the object-oriented concept. Allthough this solution is absolutely flexible and extensible there is still a high effort as the basic animation system functionality must be provided in Java.

We currently support the first and the second way to automatically generate VRML output format.

In all cases not only during the generation of VRML the generation of 3D animations is done automatically. This is possible because of the transparency during the whole generation process that allows direct access to the internal data structures of the animation system. Thus, for the author the connection of a system that provides the behavioral description of the objects in the virtual environment is easy to realize with CASUS Anim. The various example figures (Figure 4, 5, 6) in this paper show the result of the automatic scene generation in the context of a logistic simulation that can be used for validation or presentation purposes. It can be seen in the examples that three-dimensional visualization has decisive advantages, for instance the customer will receive a complex display of the entire simulation scenario not only giving an overview but enabling an interactive walkthrough. Events and strategies are visible at once and can be followed better, so a decision can be made easier and faster [9].

Another advantage of CASUS Anim is that the author is able to focus on rendering parameters of the animation such as camera position, view-point, light sources and background that may be defined interactively giving the user direct feedback on his changes (see Figure 7). Positioning of the camera and multiple light sources can be done through exact input or intuitively by dragging different three-dimensional icons into the scene. The resulting camera view with light and specified background as well as the animation is presented in a separate camera view. The author can choose to view the entire animation or can specify a part of the animation by defining special beginning and ending times.

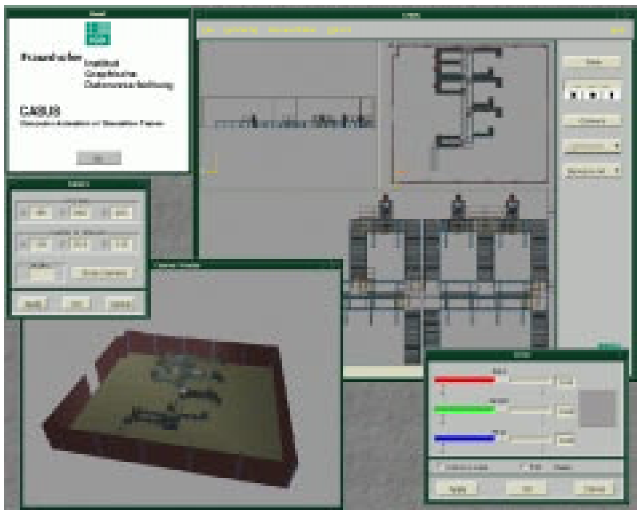


Figure 7: Graphical User Interface of CASUS Anim

3.3 CASUS Presenter

CASUS Presenter [14] is a general-purpose VRML 2.0 Browser which is used as a VR viewer and presentation tool and was developed for interactive visualization in the context of the CASUS system. Since CASUS Presenter is based on VRML 2.0 and implemented in Java, it can be used as a flexible VRML 2.0 browser in various application scenarios, especially in World Wide Web applications. The use of Java combined with the industry standard 3D graphics libraries *Open Inventor* and *OpenGL* in the realization of CASUS Presenter allowed to combine very good performance with high portability. CASUS Presenter currently supports a subset of the VRML 2.0 specification, including prototypes, events, sensors, and interpolators. Additionally, it enables direct reuse of Java behavior descriptions in animation elements from CASUS Base, by supporting the VRML 2.0 Java API. Stereo viewing using shutter glasses is possible. CASUS Presenter takes advantage of existing 3D hardware through OpenGL. It runs on multiple platforms, including SGI IRIX, Sun Solaris, and Windows NT.

CASUS Presenter accesses the Open Inventor library through the Kahlua Interface [16], which is a freely available Java wrapper for the Open Inventor library. Kahlua uses the native code interface of Java to make the Open Inventor classes available to Java applications. By using Kahlua, it is possible to develop 3D graphics applications in Java using Open Inventor, just as one would do in C++. When an Open Inventor object is created or a method is called from within a Java program, the corresponding method in the C++ library is invoked.

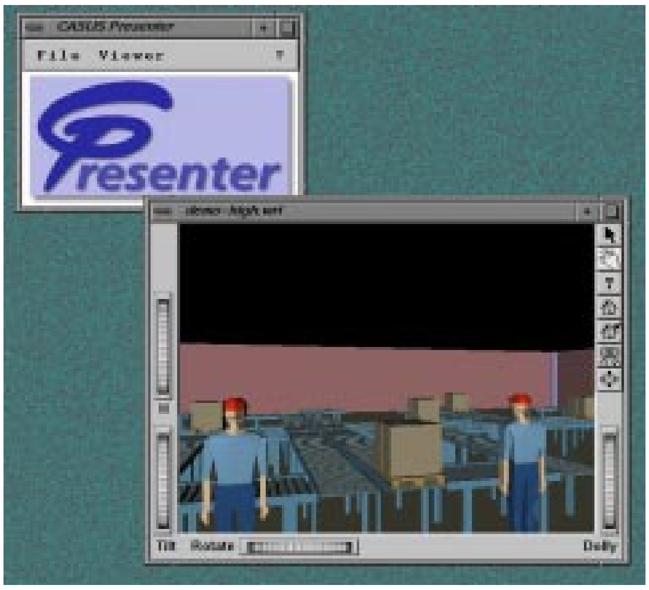


Figure 8: Example Scene in CASUS Presenter

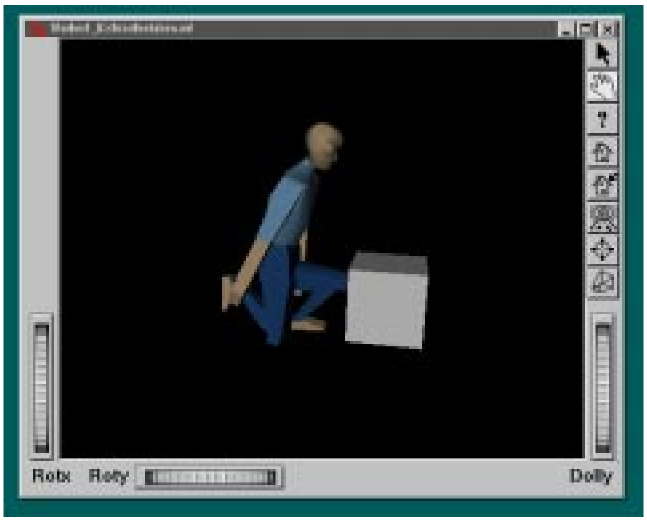


Figure 9: Example scene from the animation sequence described in Figure 3 in CASUS Presenter

Thus, one takes advantage of performing computation intensive 3D operations in a graphics library on the system level, while retaining the robustness and ease of use of Java.

Because of Java's nature as an interpreted language, it is slower to execute than native machine code, and special attention must be given to the performance of the browser. Tests with large VRML models confirm that good interactive performance can be achieved using the approach described. No significant difference is noticed when compared to the performance of Open Inventor applications or VRML browsers written entirely in C/C++. Moreover, use of Open Inventor guarantees that the browser automatically takes advantage of existing 3D graphics hardware through the OpenGL library, as soon as an OpenGL binding exists for the respective hardware. This distinguishes CASUS Presenter from VRML browsers using pure software rendering.

4. APPLICATION OF THE SYSTEM

In the application of the system three user types have to be distinguished:

- programmers, i.e. persons who model and program the animation elements
- authors, i.e. persons who create and edit scenes
- users, i.e. persons to whom the scenes are presented or who interact with the scene

In this paragraph we describe how each group makes use of the proposed system.

Programmers can be assisted by 3D modelling tools to create the geometry of animation elements. For the programming of the objects' behaviour a C++ / Java class hierarchy of animation elment classes is provided. To implement an animation element the proper class is inherited and the object-specific methods are written. A reimplementation is necessary if standard methods (like scale, rotate) have to be adapted to a specific behavior of the element.

Authors use a scene editor to specify the static scene description. The editor allows to choose animation elements and place them in the scene. Each element can be picked to scale and orient it properly. In our sytem a simple scene editor is used where the elements are represented in 2D as rectangle with a specific symbol inside. After the scene description is given the author defines the animation where he can use a simulator to complete this task. Then each object in the simulation model has to be mapped to its correspondend animation element by name. If a software adaptor for the chosen simulator exists the rest can be done automatically. If no simulator is applied the author has to program the behavior using the methods provided by the animation elements. Methods are documented in the animation elements library and illustrated with short animations. The last step is to load the scene and behavior description in the animation system. Here the author can edit animation parameters (e.g. concerning illumination) and timing parameters. Then one or more output format are chosen and the virtual environment is saved in the specific formats.

Users can interact with the virtual environment according to the output format the author has chosen and according to the presentation tool they are using. For instance, if VRML is the output format and a VRML browser is chosen as the preferred presentation environment, the user can navigate through the 3D world and may interact with the object specific functions (see Figure 8).

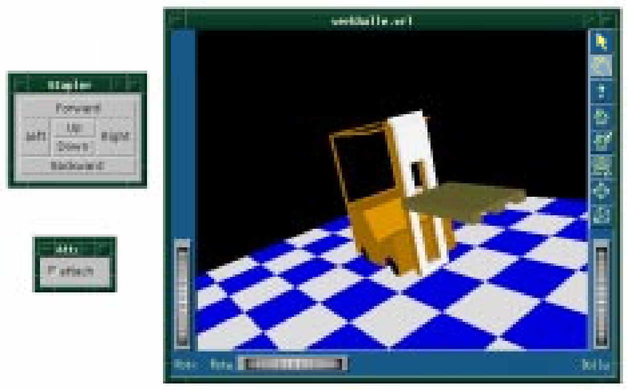


Figure 10: Manipulating the Fork of a Forklift

5. CONCLUSION

In this paper, we have presented a flexible and efficient authoring system for virtual environments. The approach described can significantly reduce cost and development time in a wide range of application scenarios. This is achieved by encapsulating behavior inside the animation elements in an object-oriented manner and by using a flexible driver concept for input and output. The object-oriented approach enables reuse not only of model geometry, but also of programmed behavior, and allows hiding the details of complex actions such as movement and scaling inside the objects. Additionally, both geometry and behavior can be defined at different levels of abstraction, making it possible to generate both high definition VR worlds or photo-realistic renderings and lower definition VRML files from the same input source for distribution over the Internet. The driver concept supports flexible adaptation of the system to changing needs and environments.

6. REFERENCES

[1] Curtis Beeson. *An Object-Oriented Approach To VRML Development*, Proceedings of VRML 97, Monterey, 1997

[2] G. Bell, R. Carey, and C. Marrin: *The Virtual Reality Modeling Language Specification, Version 2.0*, http://vag.vrml.org/VRML2.0/FINAL, August 1996

[3] S. Bryson: *Virtual Environment Techniques in Scientific Visualization, Tutorial*, IEEE Visualization, 1992

[4] C. Cruz-Neira, D.J. Sandin, T.A. DeFanti: *Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE*, Proceedings of SIGGRAPH '93, Anaheim, 1993

[5] R. Dörner, V. Luckas, U. Spierling: *Ubiquitous Animation - An Element-based Concept to Make 3D Animations Commonplace*, Visual Proceedings of SIGGRAPH '97, Los Angeles, August 1997

[6] M. Gervautz, O. Beltcheva: *An Approach for Object-Oriented Animation Design*, Institute for Computer Graphics, Technical University of Vienna, August 1994

[7] P. H. Getto and D. E. Breen: An Object-Oriented Architecture for a Computer Animation System, The Visual Computer, Vol. 6, No. 2, 79-92, March 1990

[8] J. K. Hodgins, N. S. Pollard: Adapting Simulated Behaviors For New Characters, Proceedings of SIGGRAPH '97, Los Angeles, 153-162, August 1997

[9] D. Krömker, F. Loseries, V. Luckas, S. Wenzel, U. Jessen: *Realitätsnah planen – Die 3D-Visualisierung als ideale Ergänzung zur Simulation*, Proceedings zum Workshop Visualisierungsverfahren beim Einsatz der

Simulationstechnik in Produktion und Logistik, 7. ASIM Fachtagung, Dortmund, 1996.

[10] J. C. Lafon, M. Mahieddine: *An object-oriented approach for modelling animated entities,* N. Magnenat-Thalmann, D. Thalmann, *Computer Animation '90,* 177-187, Springer-Verlag, 1990

[11] V. Luckas, T. Broll: *CASUS - An Object - oriented Three - dimensional Animation System for Event-oriented Simulators*, to be published in: Ed. N. Magnenat - Thalmann, D. Thalmann: Proceedings of Computer Animation '97, University of Geneva and Swiss Federal Institute of Technology in Lausanne, Geneva, 1997

[12] W. Müller, R. Ziegler, A. Bauer, H. Edgar*: Virtual Reality in Surgical Arthroscopic Training*, In: Journal of Image Guided Surgery, Wiley-Liss, New York, 1996

[13] S. Nakagawa, H. Ishida: *Visual Behavior Programming with Automatic Script Code Generation* Visual Proceedings of SIGGRAPH '97, Los Angeles, 1997

[14] A. Schäfer, W. Müller, V. Luckas: *A Java-based VRML 2.0 Browser: CASUS Presenter*, Proceedings of 6th International World Wide Web Conference, POS 734, Poster Presentation, Santa Clara, 1997

[15] C. Upson, Th. Faulhaber, D. Kamins, D. Laidlaw, D. Schlegel, J. Vroom, R. Gurwitz, A. van Dam: *The Application Visualization System: A Computational Environment for Scientific Visualization,* IEEE Computer Graphics & Applications, 30-42, July 1990

[16] J White: Kahlua: *A Java Wrapper to the Open Inventor 3D Graphics Library*,
http://www.cs.brown.edu/~jsw/kahlua/home.html,1996

Authors:

Wolfgang Müller, staff scientist at the Department of Computer Science of Darmstadt University of Technology.
Address: Rundeturmstraße 6, D-64283 Darmstadt
E-mail: mueller@gris.informatik.tu-darmstadt.de

Ralf Dörner, staff scientist at the Department Animation and Image Communication of Fraunhofer Institute for Computer Graphics.
Address: Rundeturmstraße 6, D-64283 Darmstadt
E-mail: doerner@igd.fhg.de

Volker Luckas, staff scientist at the Department Animation and Image Communication of Fraunhofer Institute for Computer Graphics.
Address: Rundeturmstraße 6, D-64283 Darmstadt
E-mail: luckas@igd.fhg.de

Arno Schäfer, staff scientist at the Department Animation and Image Communication of Fraunhofer Institute for Computer Graphics.
Address: Rundeturmstraße 6, D-64283 Darmstadt
E-mail: aschaefe@igd.fhg.de