# Rapid Prototyping 3D Virtual World Interfaces within a Virtual Factory Environment

**Charles Paul Kosta, Dr. Patrick D. Krolak**
Center for Productivity Enhancement
University of Massachusetts Lowell

## ABSTRACT

On going work into user requirements analysis using CLIPS (NASA/JSC) expert systems as an intelligent event simulator has led to research into three-dimensional (3D) interfaces. Previous work involved CLIPS and two-dimensional (2D) models. Integral to this work was the development of the University of Massachusetts Lowell parallel version of CLIPS, called PCLIPS. This allowed us to create both a Software Bus and a group problem-solving environment for expert systems development.

By shifting the PCLIPS paradigm to use the VEOS messaging protocol we have merged VEOS (HITL/Seattle) and CLIPS into a distributed virtual worlds prototyping environment (VCLIPS). VCLIPS uses the VEOS protocol layer to allow multiple experts to cooperate on a single problem.

We have begun to look at the control of a virtual factory. In the virtual factory there are actors and objects as found in our Lincoln Logs Factory of the Future project. In this artificial reality architecture there are three VCLIPS entities in action. One entity is responsible for display and user events in the 3D virtual world. Another is responsible for either simulating the virtual factory or communicating with the real factory. The third is a user interface expert. The interface expert maps user input events, within the current prototype, to control information for the factory. The interface to the virtual factory is based on a camera paradigm. The graphics subsystem generates camera views of the factory on standard X-Window displays. The camera allows for view control and object control. Control or the factory is accomplished by the user reaching into the camera views to perform object interactions. All communication between the separate VCLIPS expert systems is done through VEOS.

## 1. INTRODUCTION

This work is divided into three broad sections covering rapid prototyping, the virtual enterprise, and the VCLIPS architecture. Rapid prototyping is one example domain for VCLIPS development. In this paper we will examine a factory floor enterprise. VCLIPS will be used as both an event-based simulation and a control environment for the factory. The virtual factory section will explore the various aspect of' the physical factory and the identical virtual factory. The final section presents the VCLIPS architecture and related research topics.

## 2. RAPID PROTOTYPING

A prototype is an original type, form, or instance that serves as a model on which later stages are based or judged.

User interface prototyping has evolved into four distinguishable levels. The first level is that of painted or hand drawn diagrams representing static screens which the user views. The second Level is that of prototyping responses, where objects on the screen are no longer simply pictures; they are objects that react to the user. This is usually done though the use of' 'C' code or some script-like language. The next Levels is that of dynamics.

Dynamics includes both responsiveness to user events and simulation of user scenarios. Dynamic systems act as close to the real system as possible. The highest Levels of prototyping contains everything in the previous three levels plus the ability to capture and report usage metrics.

The function of prototyping is to demonstrate that a model serves a useful purpose. At the first Level, one tries to find out if the screens portray the correct meaning. The second level asks whether the prototype can respond in an intuitive manner. The third Level utilizes scenarios which in turn simulate events that the user will encounter on the real system. The highest level prototype would include measures (metrics) of the usefulness of the prototype. It is important to note that the first three levels also have metrics; but they are not integrated into the prototype - they are external: e.g. surveys, video taped sessions, subjective comments of the user community.

### 2.1 The Graphical Display

Today's prototypes not only deal with data models, but also with user models - an example is the use of icons. Icons must somehow depict a similar meaning for all users. Concern for user models inspire the larger role that windowing systems are playing in today's computing environments. Specifically, methods are being created to

specify how information is distributed into windows, such as the *Motif Style Guide.* These methods show how important icons can be for the users' understanding..

New techniques are still being developed [FB89], striving to go beyond the framework of windows of information into what has been termed widgets. A widget is an abstract graphical representation, in the form of an icon or glyph, that provides movement or actuation on some object. An example would be a sliding bar widget. It would look like the sliding bars used on stereo equipment. The user could select the slide bar with the mouse and move it along the axis to set or adjust some scalar value. Widget complexity is only limited by the creator's imagination. They can be as simple as a small radio knob dial or as complicated as the entire front panel of a virtual computer.

In general, prototyping systems are increasingly becoming object oriented [ZCW+91][FB89] Widget items acquire object properties. These properties can then be linked to widget functionality on the display. When an object value changes the corresponding widget can be updated. The objects can be displayed in two or three dimension.

### 2.2 Object Rendering

In general, a display can be viewed as having multiple objects expressed on the screen using some known output technique. This could be a simple table of integer values or a screen full of text. Each displayed objects rendered onto the display screen. For example consider rendering a *real* number. Doing so might produce the numerals on the screen that represent the real value, or, one might till in the picture of a thermometer with pixels that let the user read the real value from the thermometer [figure 1 ]. In particular, every object is rendered in some manner. Object rendering is usually based on an objects function - is it a number - is it a structure representing a workstation - is it a file - is it a disk drive'? one application development systems that allow for interactive creation of widgets is NASA's TAE+. Another example would be VAPS, a complete environment for designing real-time interfaces.

It is important to consider the user model as a guide to object rendering. Current windowing systems allow the designer to choose different techniques in the management of both windows and objects. The three main types of windows are *tiled, overlapping,* and *pop-up* windows. Tiled windows are those that split up the screen into smaller tiles such that no window ever covers up another. The tile model is based upon the user's ability to deal strictly with spatial relationships. overlapping windows allow for the possibility of data being covered up; however, it usually comes with the ability to resize, move and place the windows over one another. In user model terms, overlapping windows represent the "desktop" paradigm.

Pop-up windows are interesting in that they can represent a user model that goes beyond the "desktop" into models that are based on a virtual technical assistant. The assistant works with the user's "desktop." In particular, current pop up windows are used for: [1] displaying a message about the system that must be dealt with immediately (e.g. someone putting a high priority memo on your desktop), 121 presenting a menu that represents

either local or global choices about the window below it, [3] creating computerized *post up notes* that are attached to objects until you peel them off and throw them away. These pop up examples represent items that an assistant might manage for you, the user.

### 2.3 The Camera Model

Individual windows within an artificial reality interfaces can be represented as a virtual camera. In a three-dimensional (3D) artificial world the one needs to be able to control the location or aspect in the world one is looking at. To perform this using today's two-dimensional (2D) screens, this work centers around the features of camera based controls. Most 3D graphics packages support some form of camera controls, such as HOOPS and PHIGS PLUS. The major advantage being that the user can also manipulate the objects **inside the** views of the virtual cameras.

In a virtual factory world where multiple objects will be changing and moving without regard for the user's view or attention, the interface paradigm becomes that of a camera-person or director. Monitoring of factory processes is achieved using these virtual cameras. The user chooses where the cameras will be pointed and how many will be present.

At the top Levels, a single window on the screen is a window into an artificial three dimensional world. Inside this world you can create objects that represent information and data flow. Further, you can create additional camera views which let you watch what is happening in different places within the artificial world.

## 3 THE VIRTUAL FACTORY ENTERPRISE

### 3.1 Historical perspective on Factory Controls

In the early days of the industrial revolution it was possible, if not required, for machinists to work in close proximity of the machines they were to control. Machines produced their own information in the form of sounds, odors, and "the feel" of the working unit.

Later, the central control room came into being. Here one could find whole rooms full of read-outs, charts, dials, and warning bells. For some people, it was difficult to be away from their machines, even these few yards. No longer were there noises and odors to be had. Often it took a new generation of employees to learn to use the control room gadgets in a productive manner.

As the number of automated machines increased, fewer controls could be kept in a single control room. In today's factories controls are distributed in a clustering manner. These machine clusters then report in a "control room" like manner to centralized monitors and strip charts which allow for recording and monitoring. Programmable controllers handle most of this reporting function. IBM PCs and clones are being used as front ends to these distributed control sites. Graphs and visual programming languages (ladder logic and flow diagrams) are being used to control these machines. Control information can be downloaded from the remote control rooms as well as at the local machine.
In this work we propose that 3D graphics can be used to recreate gadgets such as toggle buttons, numeric readouts, slider controls, and other controls. one can now take the control room to the person, instead of the person going to the control room. In fact, many people can manipulate and view the same control panel at the same time.

In addition to creating the control panels for the factory, an artificial reality environment can also reproduce the physical machines and objects. one such example is the ARKola Simulated Bottling Plant developed by Gaver, et. al [GS091]. In this artificial world multiple people manage different parts of the factory and interact with one another.

### 3.2 The Physical Factory

The original concept of the Lincoln Log Factory of the Future was that it be highly autonomous, ideally, needing minimal help from the user. The goal was to use multiple expert systems in a cooperative communication environment to develop an intelligent manufacturing environment. The system controls multiple robots, parts feeders, vision requirements, and a material handling subsystem [figure 2]. In figure 2, each of the rounded boxes represents an actor within the factory floor. The robot expert systems each control a single robot and the cell experts control a single workcell. The dashed lines, with arrows, are meant to show the one to one relationship between the VCLIPS experts and the physical objects.

The current model of the factory of the future utilizes an integrated Computer Aided Design (CAD) package which has knowledge of structural requirements and part constraints. It requires the user to select parts which can actually be placed. The intelligent CAD system creates a work order, represented by structured English sentences. It sends these instructions to the factory scheduling software. The scheduler is then responsible for creating and monitoring workcell and robot processes which actually built the product.

Past years of research has suggested that future factories will not be completely autonomous but will require man and machines to work together in a cooperative manner to produce quality goods.

3.3 The Virtual Factory

The artificial reality version of the factory consists of artificial entities which share a portion of their knowledge base. The knowledge base is used to render a virtual world. Rendering is done by one or more of the entities using a 3D object oriented graphics system called HOOPS. HOOPS is both a rendering and input system developed by Ithaca Software, Inc.. HOOPS allows for both presentation and mouse based input. HOOPS is a trademark of Ithaca Software, Inc.. We use the mouse mainly for picking and menu options. However, you could create any imaginable widget under mouse control.

The artificial world will contain 3D objects .These objects will be placed in the artificial world in a similar arrangement for each person in the environment. This allows the spatial relationships to be shared with others. However, the views of the world are controlled by the individual tailoring the camera objects.

## 4 EVENT BASED SIMULATION AND CONTROL ENVIRONMENT

The CLIPS expert system shell provides for production system based inferencing rules. The CLIPS rules react to each of the users requests as they are presented through the artificial reality graphics engine. These messages are sent via the VEOS message layer to the Interface expert.
The rules are designed with both preconditions that must be true and postconditions that will be propagated within the knowledge base. The work is similar to that reported by Daniel Gieskens and James Foley in their SIGCHI 92 paper titled "Controlling User Interface objects through pre- and postconditions.

The University of Massachusetts Lowell has developed a coarse-grain parallelism version of CLIPS, called Parallel CLIPS (PCLIPS). We used portions of the existing PCLIPS architecture and the VEOS messaging layer to create a new test bed which we call Virtual CLIPS (VCLIPS). To accomplish this merger of both PCLIPS and VEOS, we removed the XLisp layer from the distributed VEOS code. Simply using the "talk" layer of the VEOS environment we have been able to send VCLIPS have similar features: being entity based and having multi-platform capabilities. We chose VEOS because it has the potential to become a de facto standard with the research community

4.1 Rapid Prototyping Control Panels

Artificial reality based control objects such as toggles, sliders, and read-outs can be placed on a virtual control panel. These panels can then be combined in a module-like manner to create larger control stations [figure 3]. To

create new iterations of the control panels the developer can arrange the objects differently on the control station. Figure 3 shows three different widgets are placed on the viewstation interface. Each of the three smaller monitors can be patched into the larger monitor. The *arrow* widget provides coarse camera control for the selected monitor.. The graphical objects (widgets) use a message passing scheme to inform the interface expert which event took place [figure 41-

To get around in the factory (A.R.) we are exploring different models. At present we are using the monitoring camera paradigm. The viewstation that we generate on the screen contains one main simulated monitor and three smaller monitors. The camera that is "patched-in" to the main monitor location can be manipulated by the controls on the viewstation including: Pan, Orbit and Dolly camera options. The three cameras can be looking anywhere in the artificial world, they do not need to be different views of the same work area.

The multiple cameras create a feeling of presence for the user. Similar work has been reported by [TYT+92] in their work with object oriented video where they have real cameras at real sites with graphic overlays. In our work, we have simulated cameras looking at virtual worlds where the objects seen through the cameras are controllable.

4.2 Shared Virtual Spaces

In the virtual factory of the future there will be teams of professionals. Each participant will share, from separate locations, the controls of the factory floor. Factories in one part of the world can be monitored and controlled from another part of the world. It will even be possible to meet at the same (synchronous) time, and jointly solve an engineering or manufacturing problem. The virtual factory of the future will still contain workers. There will be local technical repair teams who will be coordinating with others via Artificial Reality, Video Conference, or some other high bandwidth communication link.

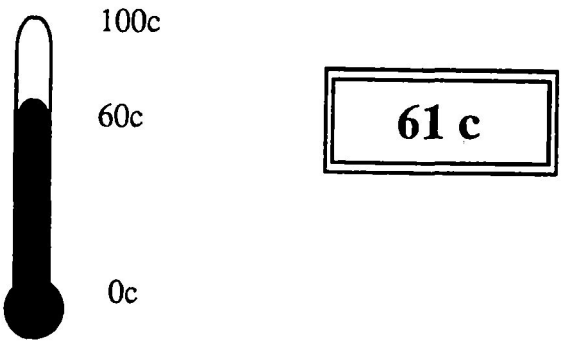


**Figure 1.** Digital vs. pictorial temperature widgets

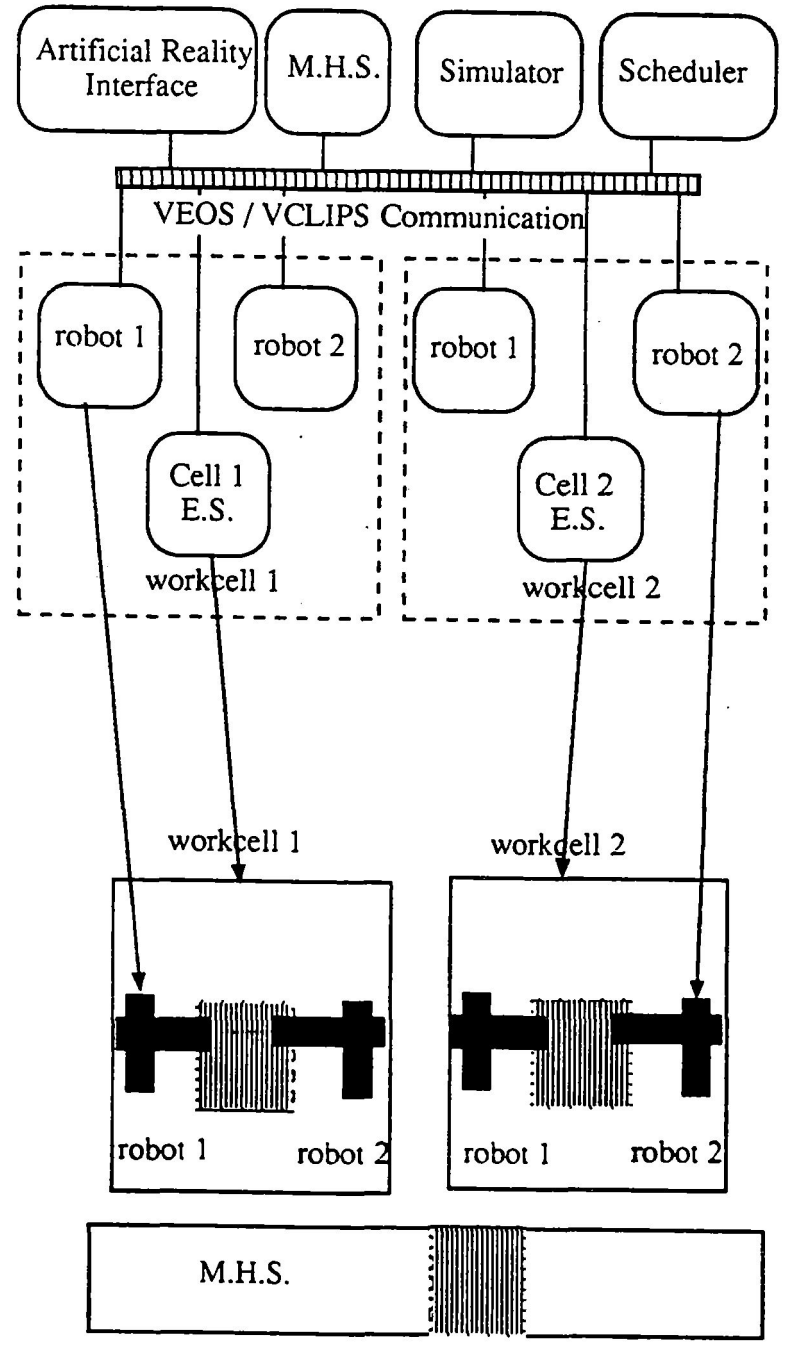


**Figure 2.** Factory controlled by VCLIPS expert systems.

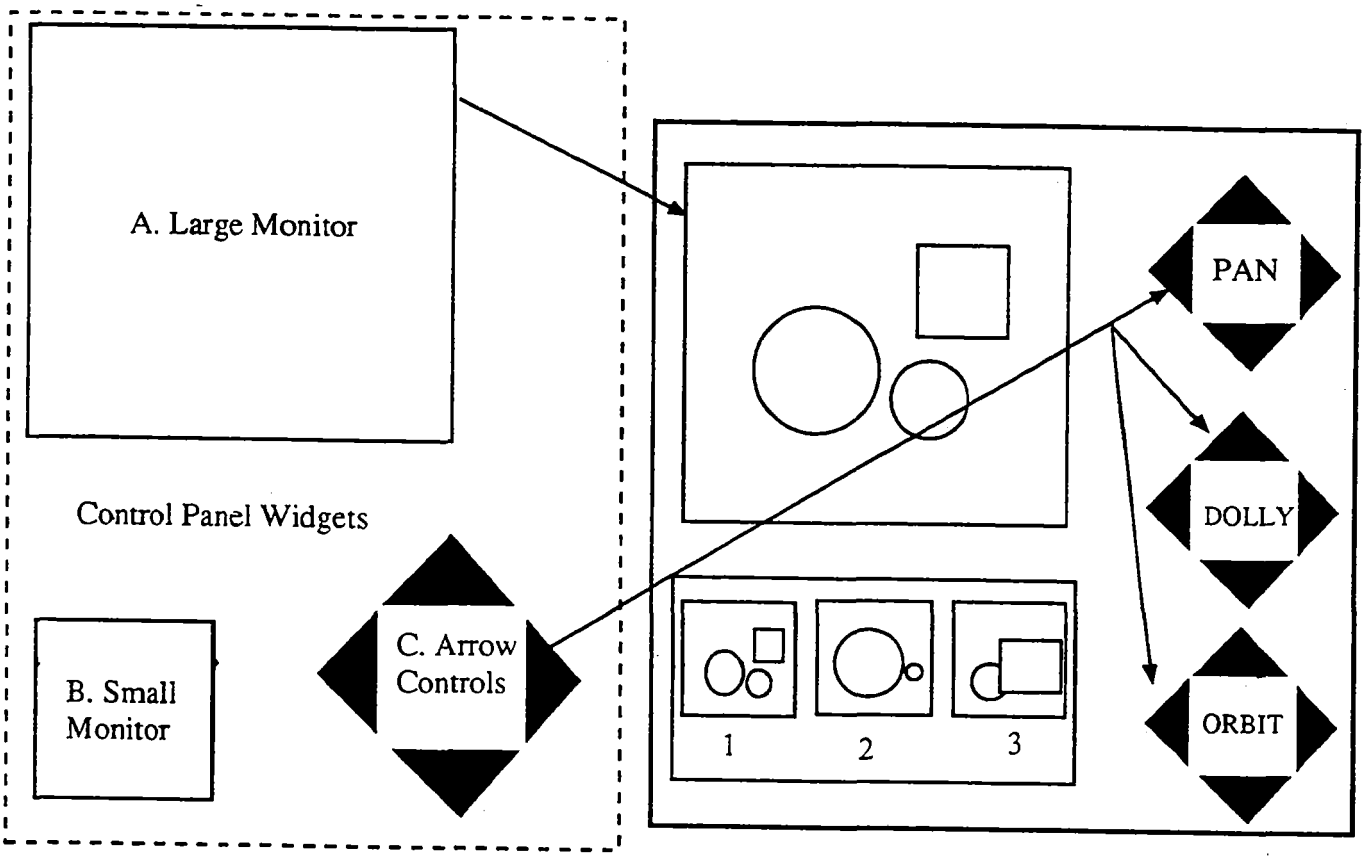


**Figure 3.** Viewstation Example

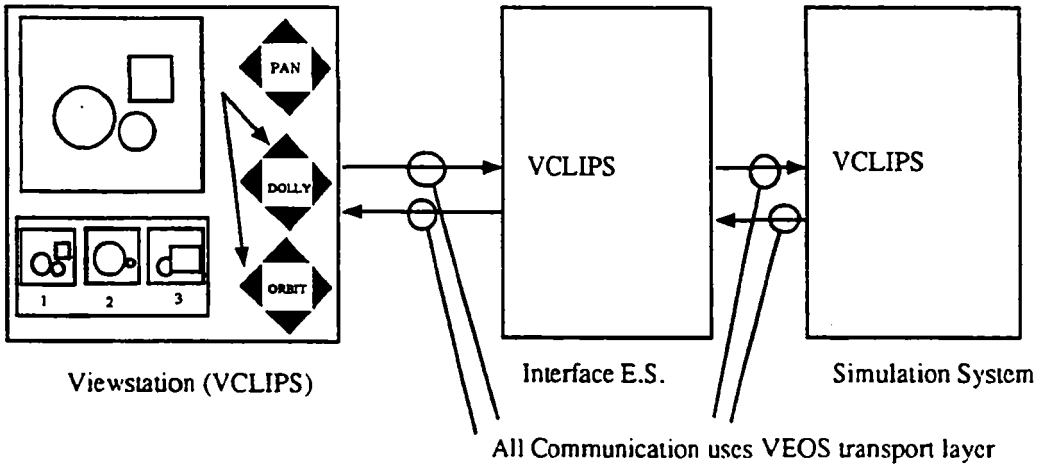


**Figure 4.** Artificial Reality based on VCLIPS model.

## 5 SUMMARY

Further research will be done in the areas of artificial reality-based user interfaces, virtual vehicles which can be used to move around in the artificial worlds and real-time control of physical objects from within the artificial

reality. Additionally, we are seeking industrial and research partners who are interested in experimenting with artificial reality based monitoring of an actual factory floor. The next version of the control panels will provide camera creation and minimal task-based control. By task-based, we mean that commands such as "follow object 123" will keep the camera targeted at object-123 wherever it may move. We will also deal will more long distance instruction and multi-user interaction.

## References


[Bro91] Rodney A. Brooks. A *robot that walks. emergent behaviors from a carefully evolved network*, pages 99-108. Morgan-Kaufmann, 1991. Editor(s): Badler, Norman I.; Barsky, Brian A.; Zeltzer, David.

[CJK+92] Christopher Codella, Reza Jalili, Lawrence Koved, J. Bryan Lewis, Daniel T. Ling, James S. Lipscomb, David Rabenhorst, Chu P. Wang, and Greg Turk. Interactive simulation in a multi-person virtual world. In *Proceedings ACM SIGCHI '92*, pages 329-334, 1992. (IBM Watson R. C., Veridical User Envs.).

[CRM91] Stuart K. Card, George G. Robertson, and Jock D. Mackinlay. The information visualizer, an information workspace. In *Proceedings ACM SIGCHI '91*, pages 181-188, 1991. (Card.PARC@Xerox.com).

[DB92] Paul Dourish and Sara Bly. Portholes: Supporting awareness in a distributed work group. In *Proceedings ACM SIGCHI '92*, pages 541-548, 1992.

[ES90] Margaret A. Ellis and Bjarne Stroustrup. *The Annotated C++ Reference Manual.* Addison-Wesley, Reading, MA, 1990.

[FB89] Jay Fenton and Kent Beck. "playground": An object-oriented simulation system with agents rules for children of all ages. In *Proceedings OOPSLA '89*, pages 123-136, October 1989. (Apple Vivarium Project ) .

[GA90] Michael Girard and Susan Amkraut. "eurhythmy": Concept and process. *Journal of Visualization and Computer Animation, Vol. 1*, No. 1, August 1990.

[GF92] Daniel F. Gieskins and James D. Foley. Controlling user interface objects through pre- and postconditions. In *Proceedings* A *EM SIGCHI '92*, pages 189-194, 1992. (foley@cc.gatech.edu).

[Gre91] Mark Green. Using dynamics in computer animation: control and solution issues. In *Making them move. mechanics, control, and animation of articulated figures.*, pages 281-314. Morgan-Kaufmann, 1991. Editor(s): Badler, Norman I.; Barsky, Brian A.; Zeltzer, David.

[GSO91] William W. Gaver, Randall B. Smith, and Tim O'Shea. Effective sounds in complex systems: The arkola simulation. In *Proceedings of HCI '91*, pages 85-90, ,1991.

[Hou92] Stephanie Houde. Iterative design of an interface for easy 3d direct manipulation. In *Proceedings ACM SIGEHI '92*, pages 135-142, 1992.

[KL90] Dennis Kafura and Keung Hae Lee. "act++": Building a concurrent c++ with actors. *Journal of Object Oriented Programming*, pages 25-37, May/June 1990.

[KWN90] Dennis Kafura, Doug Washabaugh, and Jeff Nelson. Garbage collection of actors. In *ECOOP/OOPSLA '90 Proceedings*, pages 126-134. ACM, August 1990.

[LKL91] J. Bryan Lewis, Lawrence Loved, and Daniel T. Ling. Dialogue structures for virtual worlds. In *Proceedings ACM SIGCHI '9l*, pages 131-136, 1991. (IBM Watson R.C., Veridical User Envs.).

[Luc87] S.E. Lucco. Parallel programming in a virtual object space. In *OOPSLA '87: Conference on Object Orientated Programming, Systems, Languages and Applications*, pages 26-34. AEM, Oct 1987.

[McF91] Tim McFadden. *Cyberspace. first steps*, chapter Notes on the structure of cyberspace and the ballistic actors model, pages 334-362. MIT Press, 1991. Editor: Benedikt, Michael.

[Mes90] Jose Meseguer. A logical theory of concurrent objects. In *Proceedings ECOOP/OOPSLA '90*, pages 101-115, October 1990. (SRI International) .

[MF91] Chip Morningstar and Randall Farmer. *Cyberspace. first steps*, chapter Habitat, pages ?-?? MIT Press, 1991. Editor: Benedikt, Michael.

[MR89] Naftaly H. Minsky and David Rozenshtein. "controllable delegation": An exercise in law-governed systems. In *Proceedings OOPSLA '89*, pages 371-380, October 1989. (minsky@aramis.rutgers.edu).

[Rub90] Kenneth S. Rubin. Reuse in software engineering: An object oriented perspective. *IEEE Publication Num. CH2343-l/90*, pages 340-346, 1990.

[SLGS92] Chris Shaw, Jiandong Liang, Mark Green, and Yunqi Sun. The decoupled simulation model for virtual reality systems. In *Proceedings AEM SIGEHI '92*, pages 321-328, 1992. (Univ. of Alberta).

[TYT+92] Masayuki Tani, Kimiya Yamaashi, Koiehiro Tanikoshi, Masayasu Futakawa, and Shinya Tanifuji. Object-oriented video: Interaction with real-world objects through live video. In *Proceedings AEM SIGEHI '92*, pages 593-598, 1992.

[WH91] Jakub Wejchert and David Haumann. Animation aerodynamics. *Computer Graphics, Vol.* 25, No. 4:19-22, July 1991. ().

[ZCW+91] Robert E. Zeleznik, D. Brookshire Conner, Matthias M. Wloka, Daniel G. Aliaga, Nathan T. Huang, Philip M. Hubbard, Brian Knep, Henry Kaufman, John F. Hughes, and Andries van Dam. An object-oriented framework for the integration of interactive animation techniques. *Computer Graphics, Vol.* 25, No. 4:105-111, July 1991 .