Virtual Spaces

VR Projection System Technologies and Applications

Dr. Ping Dai, Dr. Gerhard Eckel, Dr. Martin Göbel,
Frank Hasenbrink, Dr. Vali Lalioti, Uli Lechner, Johannes
Strassner, Henrik Tramberend, Gerold Wesche

GMD

German National Research Center for Information Technology
Institute for Media Communication
Visualization and Media Systems Design

1997

1 Introduction

The motivation for providing multi-sensorial interfaces for human-machine interaction is rooted in the nature of human perception and cognition, which use several sensory channels at the time to construct what is generally referred to as reality. Naturally, the more sensory channels can be stimulated coherently in a human-machine interface, the richer the interaction models can be. The more of our innate and culturally acquired perceptual and cognitive skills can be exploited in an interface, the more refined and efficient the interaction may be. This is especially valid for interfaces which mimic to a large extent certain aspects of our everyday physical environment to create what we call virtual environments or virtual reality.

One of the underlying assumption of these interfaces is that the more a virtual environment perceptually resembles the environment we are familiar with, the easier it will be for us to orient, navigate, and act in such an environment. But it has to be doubted seriously if technology will ever be able to create a synthetic sensory experience completely indistinguishable from the one we experience in our everyday world. Fortunately this is not a drawback of VR technology but its most interesting aspect, because it forces us to develop efficient interaction metaphors which refer to our cognitive skills but which do not necessarily attempt to mimic interaction as it happens in our everyday world. This is how new interaction techniques evolve and become candidates for developing into a new framework or language of expression. This phenomenon can be observed whenever new media are explored in a culture. As cinema or television had to develop their own expressive means adapted to the idiosyncrasies of their media, VR is currently developing its own language of expression which is still very rudimentary.

Cultural techniques of expression tend to mix and merge, reference each other, and are transformed and rethought in the context of new media. They form the rich tissue of a culture's means of expression most consequently explored and developed in its art. But we don't need to get into contemporary art theory to illustrate what is meant here. As an analogy, think of today's advertisement design which more and more often refers in adds to the desktop metaphor of current computer graphics user interfaces. The concept of a window, a menu-bar or pull-down menus suddenly can be used to present different aspects of a product. Such an add never would have been understood before a significant fraction of members of a society became acquainted with modern computer interfaces.

There are two aspects we wanted to clarify in this little excursion. Firstly, virtual environments cannot only be modeled after our everyday world but they have to develop their own interaction metaphors and means of expression.

Secondly, the kind and quality of integration of different sensory channels into one simulation and display system determines to a large extent the basic vocabulary available for the development of an expressive framework which will eventually become part of a culture's communication skills.

1.1 Display Systems

1.1.1 CyberStage

CyberStage is a CAVE-like 4 side room size stereo display system which creates the illusion of immersion within a computer generated virtual environment. Users see large virtual spaces, hear spatially distributed sound. R&D aims to develop intuitive man machine interfaces and cooperation techniques for humans in virtual environments. Projection systems like CyberStage allow a direct and body centered human interaction within virtual worlds as well as team work. Users immersed in a virtual world are physically standing within the display system. The Stage has a 3x3 meter floor on which the virtual space is generated by high performance graphics workstations. CyberStage is a CAVE, the image quality is much more improved and an 8 channel audio display is integrated. The significant characteristic of the Stage is the acoustic floor which allows to generate the sense of vibrations. AVOCADO, the operating software system for the Stage distributed virtual environments has been developed by GMD. It also operates the Responsive Workbench (RWB).

Three wall size rear projection systems are installed orthogonal to the floor projection, each with a size of 3x3 meters. A dual pipe Onyx IR generates 8 user controlled images. The user position is tracked with Polhemus Fastrak sensors. Crystal Eyes shutter glasses are used for stereo image reception. The display resolution is 1024 x 68 pixels at 120 Hz for each of the 4 displays. While the first CyberStage installation in September 96 was fixed in a building, the new version has been developed for „mobile“ usage. Both installations use a wooden skeleton to minimize noise for the electromagnetic tracking. An 8-channel-surround-sound system is fed by the IRCAM´s room acoustic software Spatilisateur. The AVOCADO software allows to import live video sources as well as prefabricated animations into virtual worlds. Virtual actors can be found on Stage in both ways, either in an on-line performance or in a pre-produced manner. Interaction within virtual environments is based on electromagnetic tracking using devices such as 3D pointers or 3D joysticks.

1.1.2 Responsive Workbench

The Responsive Workbench concept is an alternative to the multimedia and virtual reality models of the past decade. In this new concept, the user no longer experiences simulations of the world on the computer, but the computer is (invisibly) integrated into the users world. Everyday objects and activities become inputs and outputs for this environment. Computers are considered as a part of daily life and are no longer isolated on desks. The computer system can use and adapt to the rich human living environment. The project aims to transform the usual dialog concept for man-machine communication. For example adapting multi-media workstations into a more application-oriented form for use in science, medicine and architecture. The display is designed as part of the human working environment. For instance, objects are displayed on a table in 3D. The user interacts with this virtual scenario, manipulates it as if real, an upon request obtains information from the computer in the background.

Virtual objects and control tolls are located on a real workbench. The objects, displayed as computer-generated stereoscopic images are projected onto the surface of a table. The computer screen is changed to a horizontal, enlarged work top version and replaces the two-dimensional flat screen. This view corresponds to the actual work situation in an architect's office, at surgery environments, on the workbench, for three-dimensional atlases, etc. The work action is virtual. A guide uses the virtual working environment while several observers can watch events through shutter glasses. The guide operates within a non-immersive virtual reality environment. Depending on the application, various input and output modules can be integrated, such as gesture and speech recognition systems which characterize the general trend away from the classical human-machine interface. Several guides can work together locally or use global communication networks such as broad-band ISDN.

Responsive Environments, consisting of tracking systems, cameras, projectors and microphones, replaces the traditional computer and is increasingly adapted to human needs. The control tools implement complex actions that can be easily achieved by intuitive movements of the users hand. Each control instrument is represented as a small virtual object that can be activated by grabbing it with the hand and moving it onto an object, which is to be manipulated. Rotations of objects then can be done just by turning the hand. The zoom operation is accomplished by simple up and down movements of a small virtual magnifier, which has been grabbed by the hand.

1.2 Application Areas

1.2.1 Engineering

Ever since Hamming postulated 'the purpose of computing is insight not numbers' visualization has been attached to computation with the purpose to create a visual representation of mostly non-visual phenomena. Computation complexity is steadily increasing, visualization techniques follow. The overlay of different visualization techniques for the simultaneous exploration of many dimensions within data now challenges again the understanding of data. Additional presentation layers like audio, stereo graphics and animations for transient data have augmented the visualization techniques and turned the discipline of scientific visualization more towards a 'multi media experience' of physical phenomena, simulated by a computer. Consequently, virtual reality technology is applied and data visualization has changed to a highly interactive, individual virtual data exploration.

Several approaches have been published in applying virtual reality techniques for scientific visualization. Haase [Haase 97] has done a classification of those approaches reported so far, either using a virtual reality package and adapting visualization techniques to it, or using a visualization package and exporting geometry to a virtual reality presentation. Most work reported, like Bryson's Virtual Windtunnel [Bryson 92] is using head coupled display technology, but more and more experiments are reported in using projection systems.

The Responsive Workbench [Krüger 94] for example, developed 1993 by Wolfgang Krueger, has also been applied for scientific visualization. After 2 years of development together with the engineering research department of Daimler-Benz, it became obvious that the Responsive Workbench Virtual Environment together with a tailor made interactive visualization package is an ideal workspace for engineering applications, filling the gap between immersive and desktop environments.

1. Visualization of Fluid Dynamics

Three dimensional simulations of fluid dynamics reduce costs and time of development processes, especially in the car industry. Conventional visualization tools merely provide a 2D representation or are not able to deal with complex unsteady simulation data.

In our system, a parallel computer (IBM-SP2) and a graphics computer (SGI Onyx RE2) were linked together via a HIPPI connection. The parallel computer has the task of preprocessing the results of the (already concluded) simulation computation and to create appropriate visualization primitives for the rendering on the graphics system.

As a visualization platform, the Responsive Workbench offers powerful control and manipulation mechanisms to the engineer. He is free to choose from a wide variety of visualization methods and presentation modes. Positions can easily be located directly in the space of the simulation area. This flexibility helps evaluating results quickly and therefore reduces the time for the whole simulation and evaluation phase. The user is enabled to have deeper insight into the structure of flow fields in the simulation space, because he perceives the information in 3D, together with the geometry.

2. Interactive Steering

The coupling of parallel super computers, running simulations of time dependent airflow with projection systems like the Workbench or the CyberStage over broad band networks leads to a powerful tool for interactive steering. Although today the resolution of most simulations has to be reduced to match bandwidth and real-time constraints, reasonable visualization is produced. The incoming data has to be converted into computer graphical primitives like surfaces, lines or points. There is no time for a conversion of the multiblock curvilinear grids into more regular structures, therefore a special set of data structures was developed. In combination with a cascade like, multithreaded data flow, a real-time visualization of color coded surfaces, isolines, vector fields, or particles is possible.

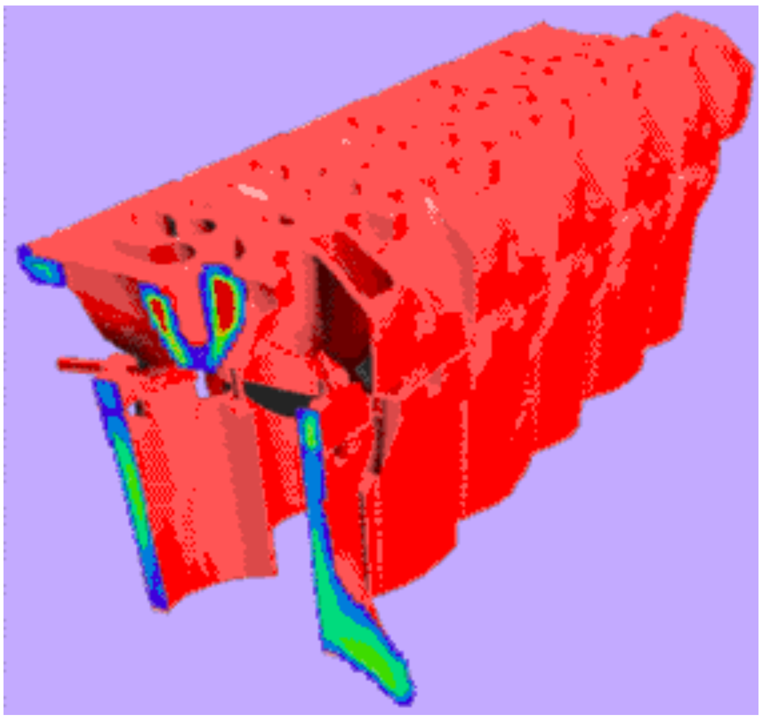


Fig. 1: Fluid dynamics.

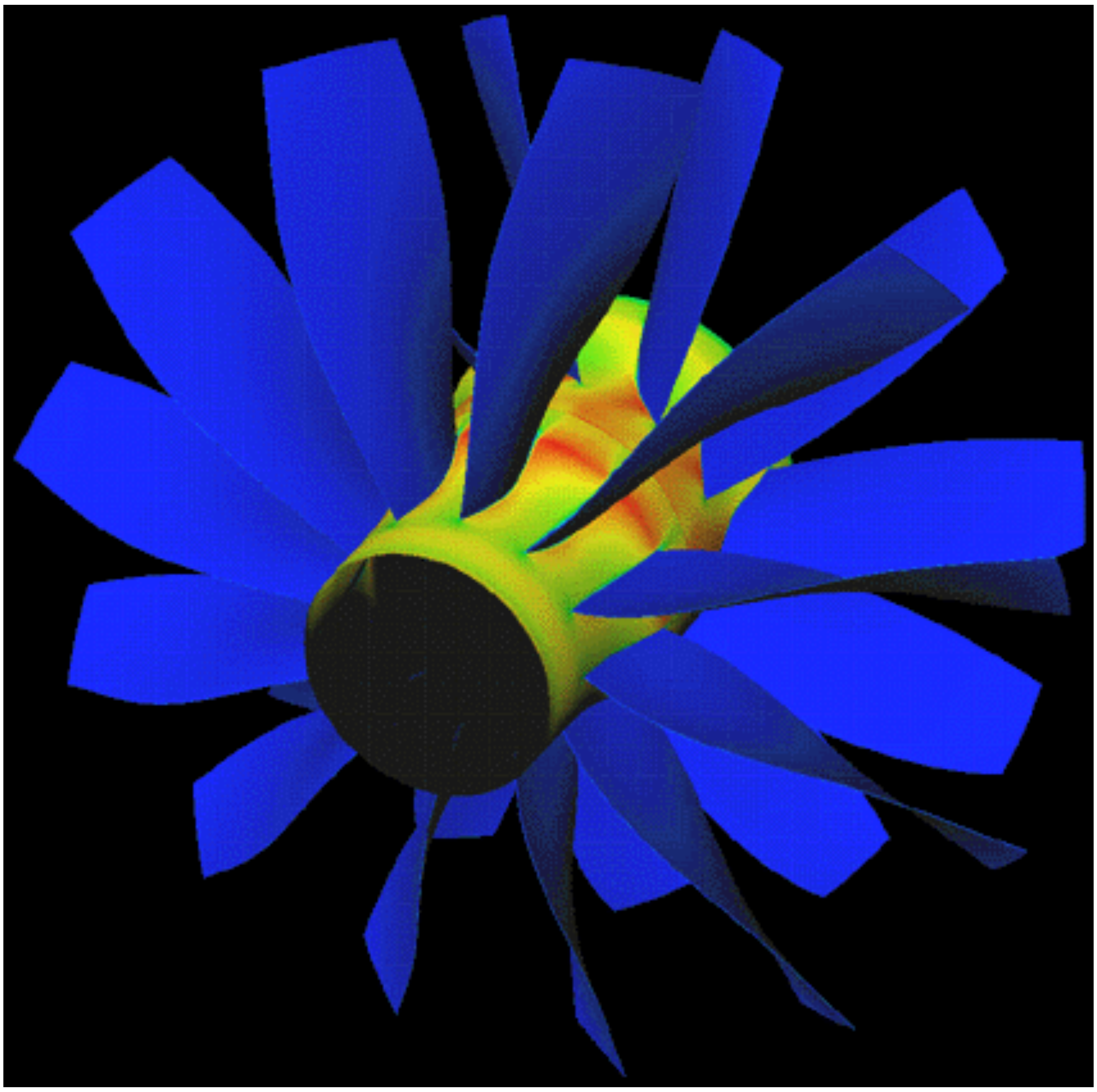


Fig. 2: Visualization of the airflow simulation in an aircraft engine.

3. Surface Modeling

Most applications in virtual environments today are more or less visualization or navigation systems, presenting objects and scenes that can be manipulated mainly through transformations controlled by some interface. Other changes in the shape of objects are mostly results of some background computation. Modeling environments extend the field of applications. On the Responsive Workbench, a free-form surface modeler is being implemented, that allows the user to define an arbitrary network of curves, over which a surface is skinned. Manipulating and changing a curve results in a new triangulation of the adjacent patches. For this purpose, multisided patches are used, because they eliminate some of the problems arising from skinning a B-spline or NURBS surface over sectional curves. The boundary curves of multisided patches need not be compatible in the sense of common knot vectors, degrees and control points. The user is completely free in the shape design of the boundary curves.

As the numerical computation power of the graphics systems increases, surface triangulation can be achieved in real time.

4. Molecular Modeling

Molecular modeling represents another field of application for the Responsive Workbench. Such applications are basically the examination and/or manipulation of complex molecules like protein complexes or other macromolecular systems. Spatial relations of an enzyme/substrate or enzyme/inhibitor complex can be examined stereoscopically. On the Responsive Workbench, molecules can actually be touched, grabbed, and moved manually, which simplifies the detailed examination of distinct molecular regions as well as manual docking. Linked to surface examination algorithms in relation to certain physical and topographical properties, surface segments can be cut out and viewed separately.

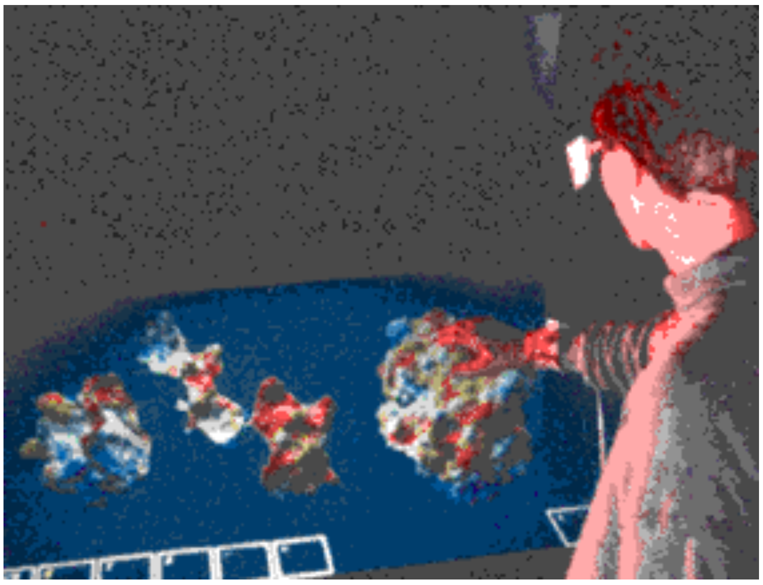


Fig. 3: Molecular modeling.

1.2.2 Teleport

Personal computers equipped with microphone, speakers, camera, and perhaps additional video monitors, are now widely used for desktop video conferencing. Conference participants appear in video windows, or on adjoining monitors, and may access shared applications shown simultaneously on each participant's screen. Several desktop video conferencing systems have been described in the literature [Bly 93, Buxton 92] and commercial products are available. But, while desktop video conferencing has certainly been shown to be useful for a variety of tasks and has many advantages when compared to earlier forms of video conferencing involving special meeting rooms, it is still recognized that there are many situations where desktop video conferencing is not appropriate.

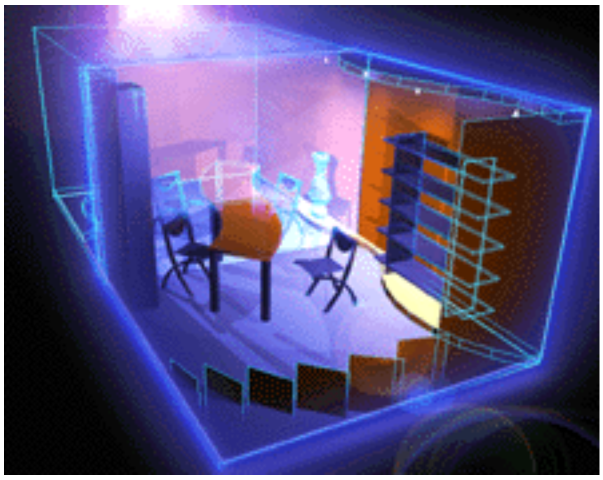


Fig. 4: TELEPORT Display Room

The TELEPORT environment is designed to overcome disadvantages of desktop video-conferencing and to establish life-like conference sessions that bring people together as if face-to-face. TELEPORT has been developed at the Visualization and Media Systems Design Group of GMD, by S. Gibbs, C. Breiteneder, and C. Arapis [Breit 96]. TELEPORT mimics a shared physical context, using 3D modeling and rendering, and provides life-sized display of remote group members placed within a virtual space. The system is based around special rooms, called display rooms, where one wall is a "view port" into a virtual extension as shown in Figure 4. The geometry, surface characteristics, and lighting match the real room to which it is attached. When a teleconferencing connection is established, video imagery of the remote participant (or participants) is composited with the rendered view of the virtual extension (see Figure 5).

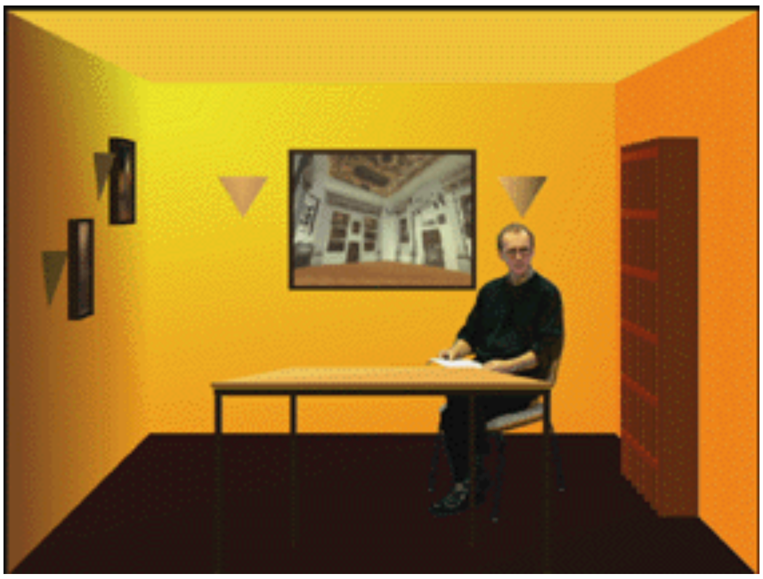


Fig. 5: Remote Participant in Virtual Meeting Area

The viewing position of the local participant is tracked, allowing imagery appearing on the wall display to be rendered from the participant's perspective. The combination of viewer tracking, a wall-sized display, and real-time rendering and compositing, give the illusion of the virtual extension being attached to the real room. The result is a natural and immersive teleconferencing environment where real and virtual environments are merged without the need for head-mounted displays or other encumbering devices. The current system uses a 3m x 2.25m rear-projected video wall attached to a 3m square room. The video-wall is driven by a pair of high luminosity video projectors. Both projectors can display mid-resolution video signals and high-resolution RGB signals. A camera is placed on a stand or a table and set at approximately eye height. The field of view is wide enough to take in a full upper body shot of the local participant. A viewer tracking system determines the position of the local participant within the display room, from which their viewpoint is derived. Two techniques are used for segmentation (for determining the regions of the video signal where a participant appears) chroma-keying and delta-keying. The virtual extension is rendered from the viewpoint of a tracked participant located in the display room. Because this person is free to move within the display room, the virtual extension must be continuously re-rendered. Currently an SGI RealityEngine2 is used to achieve rendering rates, with texturing and full anti-aliasing of up to 25 frames per second. The video imagery of remote participants is combined with the rendered virtual extension (compositing). For audio, each participant wears a small microphone. The audio signals from remote participants are mixed together and sent to speakers mounted on either side of the video wall.

1.2.3 Statistics

Dynamic statistical graphics enables data analysts in all fields to carry out visual investigations leading to insights into relationships in complex data that consists of many different variables. The data consists of multiple observations taken on the same object or measured at the same place. Dynamic statistical graphics involves methods for viewing data in the form of point clouds or modeled surfaces. Higher dimensional data can be projected into 1-, 2- or 3-dimensional planes in a set of multiple views or as a continuous sequence of views which constitutes motion through the higher dimensional space containing the data. There is a strong history of statistical graphics research on developing tools for visualizing relationships between many variables.

In the C2 environment [Sym 97] which is similar to a CAVE or Cyberstage familiar tools are being examined in a new technology and the special features of this virtual reality environment are used to develop completely new tools for the visualization of high dimensional data. The applications of the work will extend to almost all areas of science. In particular spatially dependent data, for example, data collected over geographical domains for environmental assessment, and agricultural applications are being examined.

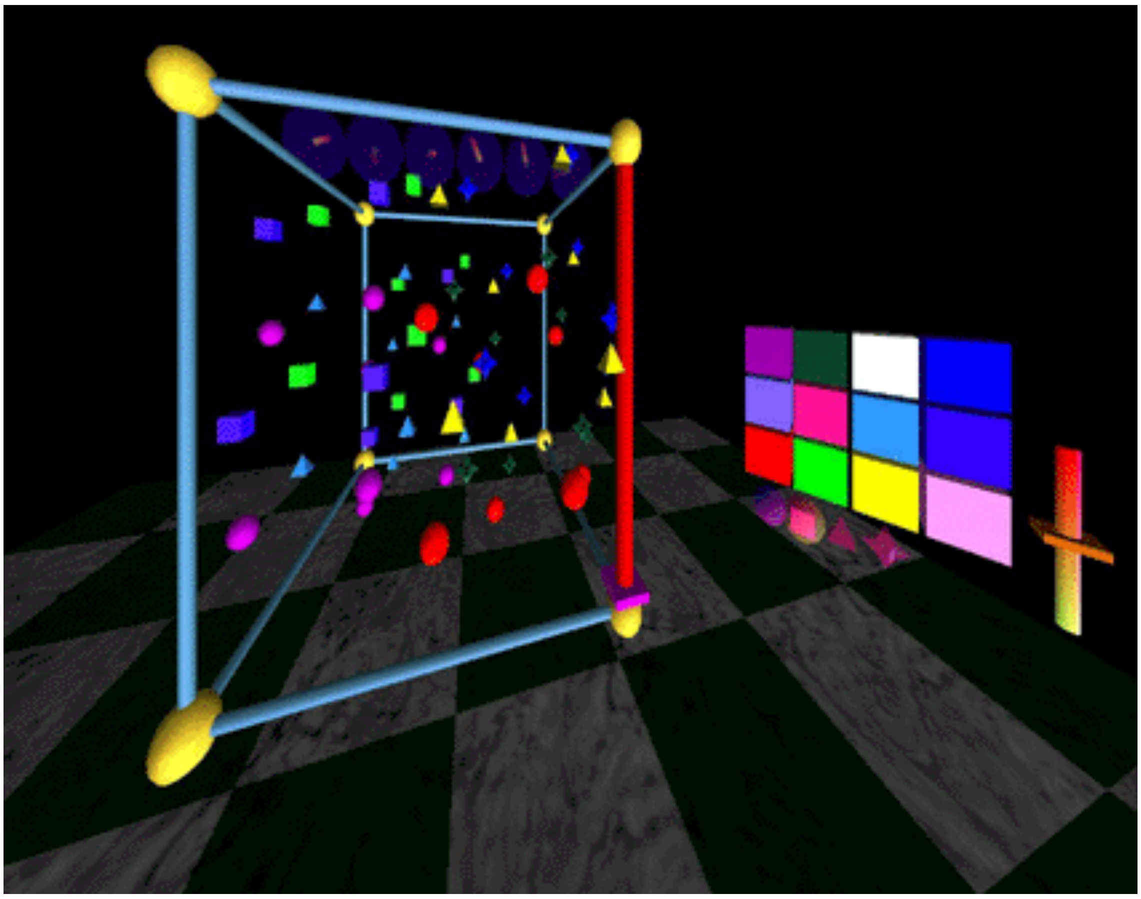


Fig. 6: This image shows the cube containing the data and one of the paint tools.

Several tools allow a user to interact with the environment. The features include a toolbox to select color, glyph type and size. Creating a custom brush is supported by the design box which enables a user to create a paintbrush for marking different data points. The rotation interface allows the user to examine the entire data set from different angles. All interaction employs audio feedback in addition to visual response. People involved at this stage were Uli Lechner, Dianne Cook , Carolina Cruz-Neira , Jürgen Symanzik at Iowa State University.

1.2.4 Artificial Life

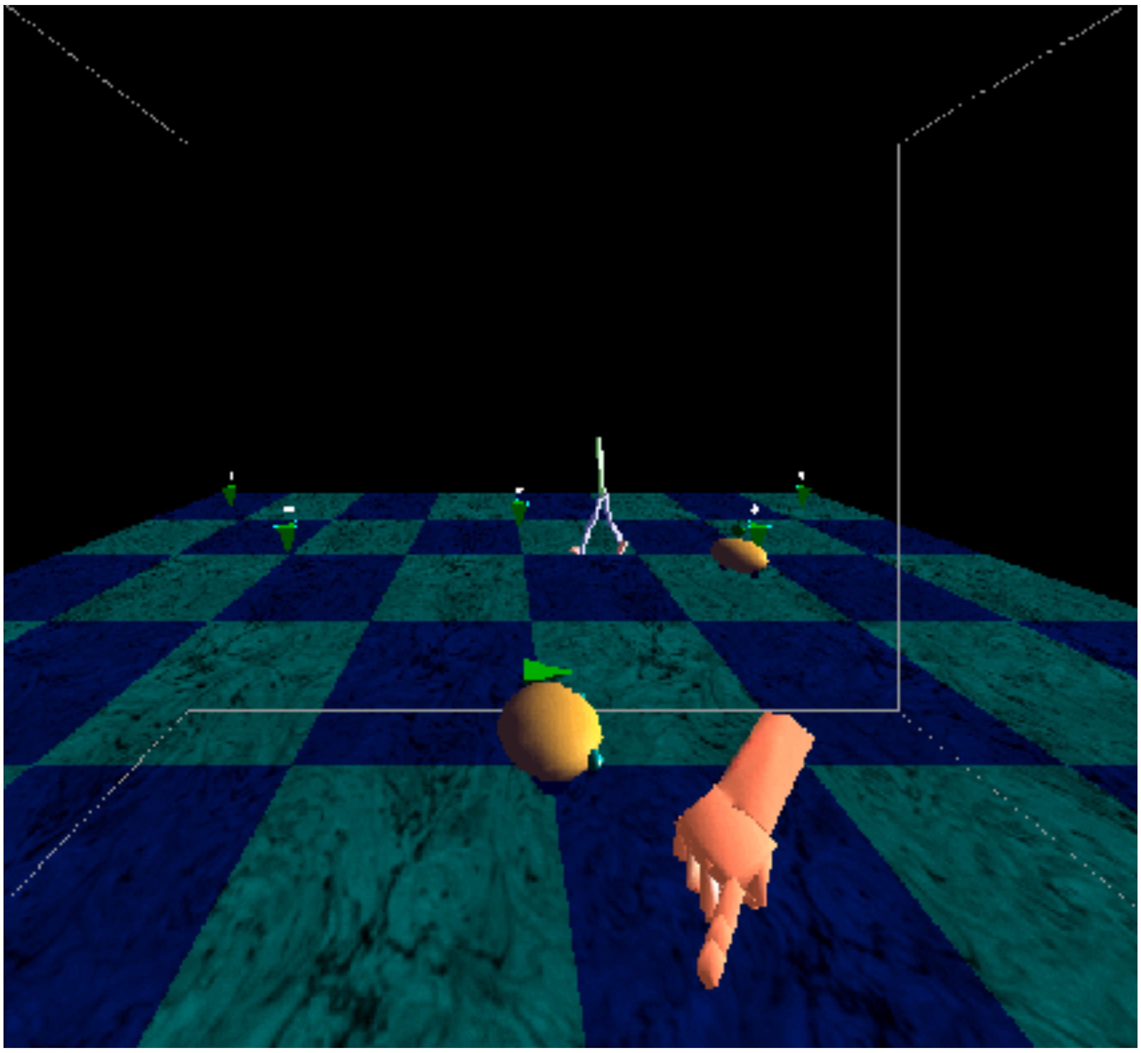


Fig. 7: Evolution of Simple Virtual Robots (Symbots) Using Genetic Algorithms

John Walker with Dan Ashlock, Allen Bierbaum, and James Oliver
Iowa State University, Ames Iowa USA
Contact: volker@icemt.iastate.edu
Funded by the National Science Foundation Young Investigator Award: Grant #DDM-9258114.

This project demonstrates how a genetic algorithm can be used to optimize problems of guidance and control for simple autonomous agents, which we call Symbots. The Symbots are controlled by simple neural networks with input parameters designed by a genetic algorithm. The Symbots learn to find food sources while avoiding collisions with each other.

The evolution is driven by a measure of relative fitness of a group of candidate designs. Fitness is the number of food sources hit without colliding with another Symbot. Once the fittest of the candidate designs are determined, the control system parameters (which can be thought of as a `gene') are cross combined and/or mutated to create new candidate designs.

Users interact in real-time with the resulting evolved Symbots by placing food sources, controlling evolution, and guiding a movable food source. The user may also navigate within the symbot world and capture the preformance of individual symbots for later analysis or rendering in an animation package.

This interactive experimentation facilitates a greater understanding of the Symbot's behaviors and gathering strategies than is possible with a traditional display system. The user gains immediate feedback about the Symbot's performance in the environment, allowing the design and testing of scenarios for training robots to perform tasks in hostile or hard-to-model environments.

2 AVOCADO Framework

The following section shows our approach towards a software system to handle all the devices and types of installations mentioned before. The system is meant as a framework to develop all kinds of applications. We first try to find the way from a more general point of view to the implementation details to use the system afterwards in an example to solve some complex problems.

2.1 VR context

To design a software system in the VR context, we first have to develop an abstract model for the description of virtual worlds. It´s obvious to use an object oriented approach for this description. Everything included in this world is encapsulated in objects and can be manipulated by a variation of the objects attributes. This fits well with the current dominance of object oriented programming languages like C++ and the broad availability of software libraries with C++ Api´s. A directed acyclic graph is used as topological model to describe the relationships between the objects in a so called modeling hierarchy. Objects are nodes of this graph. Together the state of these nodes and their relationship to each other define the state of the virtual world. This state should be complete. This means, all the different display systems aimed at the sensory channels of human perception should be addressed by the nodes in the modeling hierarchy. Every node includes its way of evaluation or better rendering for all available displays. This is very important, since most of todays software approaches are vision oriented, while there are already auditory- and tactile- or even climatic- and olfactory displays in use. Strictly speaking, the objects in the virtual world implement generalized abstractions of the features of the real world, recognized by our sensory channels. The used abstractions normally aim at the possibillities of their target displays and the rendering mechanisms which are used to make them perceptable. For example the visual features of a bird may be described by some geometric primitives and their light interaction, while its auditory features are expressed by a sound sample and its radiation. Every feature domain should have its own set of nodes to describe it. What`s common to all domains, is their localization in space and in time.

Nodes refer to each other as parent and children. A node can have more than one parent, therefore the modeling hierarchy is a graph and not a tree. This is usefull for multiple references to the same object without copying all the data related to this object. To support multi user environments the objects and their state must be shared between the different sites over a network connection. To reduce the amount of data that has to be transferred, only state changes should be transmitted. To handle a virtual world as a state machine, is a big advantage for solving this distribution problem and can also be used to store a copy of the world on disk at any time.

If you think of a virtual world as a 4 dimensional system, the first 3 dimensions describe the euklidian space, which is coded into the modeling hierarchy by nested transformations and some leaf nodes, which define the representation of a part of the virtual world for all available displays. Especially for the visual part a lot of modeling software and description formats are available. Even if this limit seldom is reached, it should be mentioned that the resolution of this euklidian space is limited by the floating point precision of the used hardware. For example it is difficult to represent the relative size of a single atom and the whole galaxy in the same continuous space.

The fourth dimension manipulates the state of the virtual world over time. This leads us to the necessity of some active parts, which control the behavior of each object. For the above mentioned bird, this can be his sole movement or his paticipation in a swarm. To keep things simple, every node in the modeling hierarchy can change his state and the distribution of information about state changes is done through a data flow network, which is orthogonal to the modeling hierarchy. This enables us to encapsulate complex behaviours in single objects or even chains of objects and postulate the results to other objects, defining the visual or auditory features for example. Before addressing the details of the implementation, the limited temporal resolution and absence of temporal continuity of the rendering processes should be mentioned here. A different update rate must be guaranteed for each of the human sensory channels addressed by the already mentioned displays. The visual display for example should have a video rate of more then 50 Hz for each eye to prevent the user of growing weary. The rendering frame rate, which is independent from the video rate, should reach a value of something more then 20 Hz, to give the illusion of continuity or realtime. If there are different realtime systems involved in the same virtual world, which all have their own definition of what the resolution of realtime should be, the software design must support this asynchronous needs.

2.2 AVOCADO System description

To design a system as open as possible, the representation of the virtual world should not be biased by any of the sensory channels addressed by the display system used to experience it. Nevertheless, we had to accecpt such a bias since the system AVOCADO is based on the C++ API of IRIX Performer 2.1 and therefore vision oriented. The development of AVOCADO was started by Henrik Tramberend in April ´96 with the publication of an internal paper named ´Fields for Performer´, which included the following definition of the main goal of such a system:

"This is supposed to be the mission statement for the project. The system has to integrate a variety of different interface devices currently in use at VMSD. Most notably these are the Responsive Workbench, the CAVE, the Communication Wall and the Virtual Studio. It has to be sufficiently general purpose to support application development for all these devices. As new devices are likely to be invented by the more creative minds at VMSD, it has to be easily extensible and adaptable. Most of those interfaces come with a mix of more or less exotic input devices. The system has to be highly interactive and responsive. VMSD is a demo pit. The system must support a rapid prototyping style of application development, without the need to adhere to a tedious write-compile-execute-kill cycle. It has to support the development of truly distributed applications. The System is targeted at high-end SGI workstations not less powerful than the Onyx RealityEngine and has to deliver every jota of performance these machines are capable to deliver."

Since we had been very hungry when looking for a name of the system just described, we descided to call it: AVOCADO. A rationale for this name was quicly found, since the seed, the pulp, and the peel of the avocado fruit perfectly represent the design of our system. There is the core library, which is represented by the seed and which implements the basic functionality of our VR toolkit. The pulp stands for a rapidly developing set of lower and higher level tools derived from the core library, used to configure applications, which build the peel of the system. To achieve the goals defined in the mission statement, we support the following general concepts:

- Browser:

All kinds of configurations of input and output devices can be assembled to so-called browsers. The browser builds up the interface between the user and the virtual world. Typical elements of a browser are the visual, auditory and tactile displays as output devices and spacial trackers, audio or video sources as input devices. In a multiuser environment every user configures his or her own browser.

- Scripting:

All relevant parts of the system's API are mapped to an interpreted scripting language (Scheme). This enables us to specify and change scene content, browser features and object behavior in a running system. This eliminates the rather disagreeable write-compile-execute-kill cycle of the application development process.

- Streaming:

All objects know how to read and write their state from and to a stream.

- Persistence:

Together with streaming support for objects this enables us to write the complete state of the system to a disk file at any time. An initial system state can be read from a disk file as well.

- Distribution:

All objects can be distributed. Their state is shared by any number of participating browsers. Object creation, deletion and all changes at one site are immediately and transparently distributed to every participating browser.

- Extensions:

The System is extendable by subclassing existing C++ system classes. This concerns object classes as well as classes which encapsulate browser features. Compiled extensions can be loaded into the system at runtime via DSOs.

- Interaction:

Browsers provide input/output services which can be mapped to objects in the scene. Objects can respond to events generated from input devices or other objects and can deliver events to output devices.

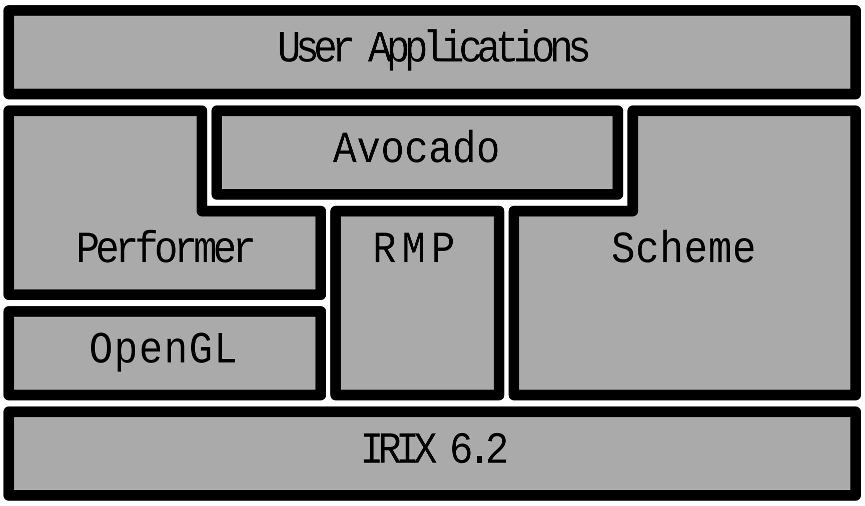


Fig. 8. System Overview

2.3 Implementation details

Because we are focusing on the field of high performance rendering, our system is very hardware-dependent. Performer runs only on SGI machines and even there it is mainly designed to be used on the high end platforms. It uses multiprocessing to work on different tasks in parallel and OpenGL for the visual rendering. OpenGL alone is available on different platforms but there are a lot of extensions only available on SGI machines. It is a very low level library, which does not implement any of the general concepts mentioned earlier and therefore is not used directly. Figure 8 shows an overview of the AVOCADO framework, including Scheme as scripting language and a reliable multicast protocol for the distribution. User applications are mainly build using AVOCADO objects and Scheme. Sometimes it's usefull to have access to Performer directly but most features are allready wrapped by AVOCADO.

Performer can be described from two points of view. First, there is the data processing organized in a pipeline and computed in parallel. This so called rendering pipeline consists of a set of optional units, for:

- a database connection
- a user application
- the visual culling of the scene
- the intersection of objects
- the drawing of the scene

The second view focuses on the data structures used to describe the visual virtual world. There are different types of nodes available which can be connected by parent/child relationships to form a directed acyclic graph. Because Performer only supports visual displays, the nodes contain the information useful to describe the visual portions of the virtual world. If we look at AVOCADO from these points of view, the data processing of Performer is extended by a sound rendering and a tactile feedback which all can be configured through the scripting interface to meet the specific needs of different hardware installations. The data structures available in performer are extended, to meet the stated general concepts. These extensions to the data structures are only available in the application process and have to be Performer compatible, to be used in the processes involved in the pipeline afterwards. This compatibility is achieved by deriving all AVOCADO objects, which have to be rendered visually, from Performer node classes.

2.3.1 Fields

In AVOCADO every object encapsulates its attributes, which describe the state of the object, in fields. All objects are subclasses of a common base class. This base class provides a public interface which allows the retrieval of all field related information. Therefore all objects, even those which are added later as extensions, can be manipulated through a common API. This mechanism allows the effective implementation of the scripting interface, persistence, run-time loading of new object classes and distribution, as the work needs only be done for the common base class. Because there can be more then one field attached to a single object every object is a fieldcontainer. The set of fields available for a specific object is static, which mean that it cannot be changed at runtime. Every field has a name and is typed and therefore can be accessed from the scripting interface easily.

Performer 2.1 has a method based object API with getters and setters for all the different fragments of object state information. This is translated into our field API by subclassing all Performer object classes once using special AdaptorFields to encapsulate method declarations. Further extensions are subclasses only from these adapted Performer classes. This approach ensures full Performer object functionality as a basis for extension development.

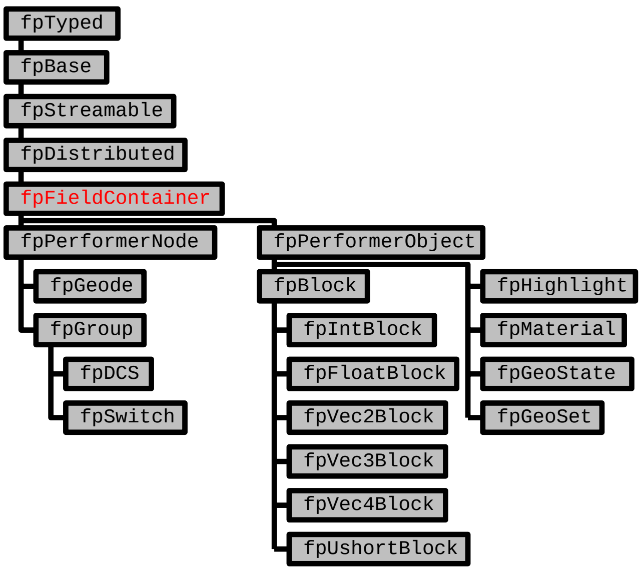


Fig. 9: List of Performer nodes available in AVOCADO and their C++ inheritance structure:

Fields come in four different flavors. A SingleField holds a single, arbitrary typed value. A MultiField holds any number of values of the same type. To adapt the Performer method based object API to our field based API a SingleAdoptorField and a MultiAdaptorField are used. All fields are derived from a single base class, and have methods to set and get field values.

Table of available field types:

fpSFInt	fpSFBox
fpSFUInt	fpSFNode
fpSFLong	fpSFGroup
fpSFULong	fpSFGeoSet
fpSFFloat	fpSFGeoState
fpSFDouble	fpSFHighlight
fpSFBool	fpSFMaterial
fpSFString	fpSFScreen
fpSFVec3	fpSFWindow
fpSFVec2	fpSFBlock
fpSFVec4	fpSFIntBlock
fpSFMatrix	fpSFUshortBlock
fpSFQuat	fpSFVec2Block
fpSFSeg	fpSFVec3Block
fpSFPlane	fpSFVec4Block
fpSFSphere	

Fields can be connected to each other, i.e. field A that is connected from another field B will receive B's value whenever field B is changed. This allows for a dataflow network to be constructed orthogonally to the object hierarchy. This dataflow network is evaluated for each simulation frame. Loops are detected and handled properly.

2.3.2 Object representations

While fields are the basic model to represent data and the dataflow in the AVOCADO system, at a higher level of abstraction (and inheritance) there are 3 types of object representations:

1. Nodes are the already mentioned classes adapted to the performer node classes for the description of the modeling hierarchy. They are field containers and their state is plainly described by their individual set of fields. In a distributed multi user environment only nodes must be shared with other users. While Performer only knows about static visual focused objects, in AVOCADO also audible and tactile properties can be defined.

Table of basic node types:

Table of basic sensor types:

fpSensor	fpViewActuator
fpTimeSensor	fpScreen
fp6dofSensor	fpWindow
fpXvsSensor	fpStereoScreen
fpDeviceSensor	fpStereoWindow
fpADSensor	

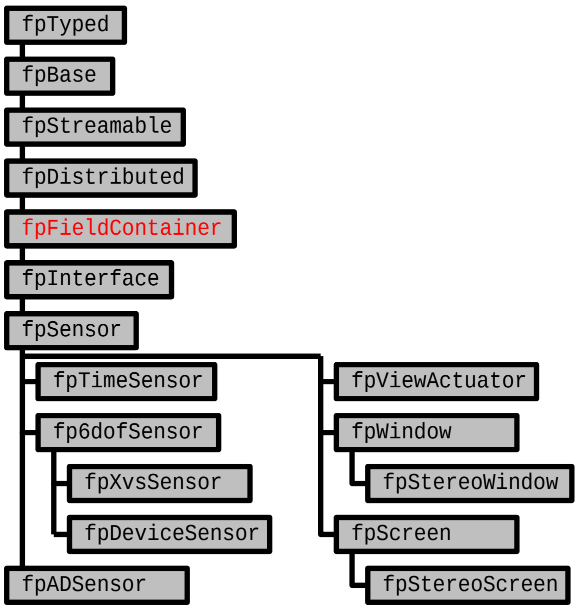


Fig. 10: Inheritance graph for sensors.

fpGroup	fpIntBlock	fpMaterial
fpDCS	fpUshortBlock	fpHighlight
fpGeoSet	fpFloatBlock	fpGeoState
fpGeode	fpVec2Block	
fpSwitch	fpVec3Block	
	fpVec4Block	

2. Sensors are field containers, but not inherited from Performer objects. They are used for data import and export from the AVOCADO system to the outer world. Sensors are not "visible", "audible" or "tangible" on the different displays and therefore don´t have to be part of the modeling hierarchy. They implement the local features of an AVOCADO application and therefore must not be distributed. An example for a typical sensor object are the windows on a workstation screen used as visual display or a graphical user interface used to control some global parameters.

3. Services are neither field containers nor inherited from Performer classes, but they provide a functional Api to unique system features. They are even more local then sensors and therfore must not be distributed either. A service would implement the access to an external device like 6 dof-trackers for example, which exists only a limmited time and therfore should be used from only one location inside the application. A sensor may use this service to access a tracker and to maintain the data related to it to the modeling hierarchy via its fields.

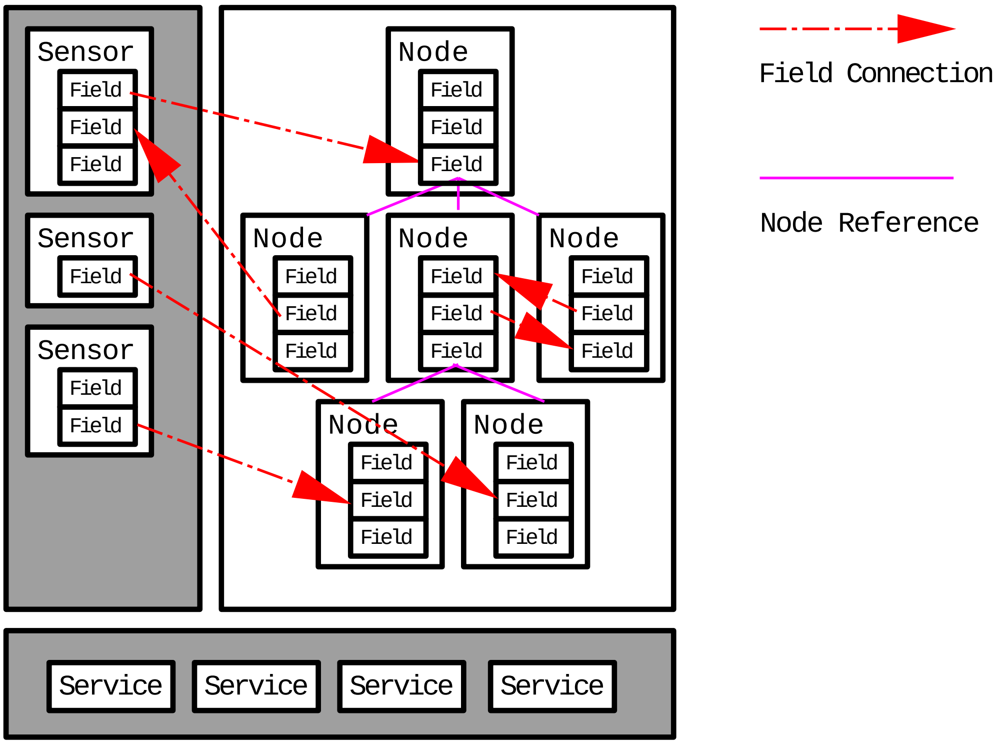


Fig. 11: AVOCADO Application layout. Field connections are orthogonal to the modeling hierarchy and distribute data generated in nodes or imported by sensors. The browser is configured by services and sensors, only the modeling hierarchy is distributed.

2.3.3 Rendering

The different displays all have their own rendering mechanism applied to the modeling hierarchy. Only the visual rendering has a direct access through the performer pipeline involved. The auditory and the tactile rendering are calculated on a second computer, connected to the "master" by a fast network. For the visual rendering the different tasks involved are divided into the already described processes of the pipeline. After the modeling hierarchy was updated to its actual state in the application process it is passed on to the culling process to strip it from all invisible objects. It's important to support this technique by dividing large geometries into smaller, cullable objects. The part of the scene left over after the culling is passed on to the drawing process where it is rendered to the screen with OpenGL. For configurations with more then one visual display system involved, the appropriate number of pipelines is used.

- Auditory Rendering

Rendering the auditory scene has to take into account the position of the observer's head in the virtual world and in the auditory display as well as the characteristics of the auditory display (i.e. the loudspeaker configuration). The auditory rendering process is a two stage process. In the first stage a source signal is synthesized which is then spatialized in the second stage. In the first stage, only the sound model parameters are needed by the rendering process. In the second stage, the signals driving the auditory display are computed as a function of the distance between observer and sound source, the radiation characteristics of the source and the signature of the acoustic environment. With these signals the auditory display produces a sound field creating the illusion of a sound source emitted from a certain position in a certain acoustic environment shared by the observer and the source. The sound rendering process has to be a dynamic process, i.e. movements of the observer's position in the display or in the virtual world or movements of the sound source have to be taken into account. If these movements are faster than about 30 km/h, the pitch changes due to Doppler shift have to be simulated as well.

- Tactile Rendering

The CyberStage display system includes a set of low-frequency emitters built into its floor. This allows to generate vibrations which can be felt by users through feet and legs. There are two main areas of application of this display component. First, low frequency sound (which cannot be localized) can be emitted that way and thus complement the loudspeaker projection. Second, specially synthesized low-frequency signals can be used to convey attributes of objects displayed such as roughness or surface texture. From the point of view of rendering, the vibration display is handled like sound. Sound models are used to generate the low-frequency signals. Sound synthesis techniques generally referred to as granular synthesis are very well suited to produce series of band-limited impulses which may represent surface features. Such features can be displayed through user interaction. For instance, a virtual pointing device can be used to slide or glide over an object's surface which, depending on the gliding speed, provokes the corresponding impulses are produced. Additionally, higher-frequency sound can be produced if necessary. Some of what can be felt usually through the skin of our fingers when sliding over object is presented to our feet. Although the quality of touch cannot be reached with this approach, it can complement sound and vision dramatically.

2.4 Example

To get a better idea of how the described object representations can be used to solve some complex problem, the next section will show an example in the area of interactive steering. It is based on the results of a shared project between GMD and the German Airospace Research Center in Cologne, DLR, finished in November 96, which was funded by the German Research Network, DFN. The simulation is done on a IBM SP2 at the DLR and connected via ATM to a Onyx RealityEngine2 at GMD, where the visualization takes place. The choosen test case simulates the air flow between the turbin blades of an aircraft engine and was already mentioned in the introduction chapter. Three modules had to be developed:

- a distributed, parallel, numerical solver for the SP2,
- a visualization module running on the Onyx,
- and a coupling module, using IP over ATM.

We will focus on the visualization module where the data reduction of multiblock curvilinear grids to some visualisation primitives like points lines or surfaces for AVOCADO is done and skip the simulation and connection details.

2.4.1 Data Structures

We deal with an online problem, therefore we have to handle the incoming data as fast as possible. This leads to a specialisation of the used algorithms to fit the requirements of the incoming data very closly. For example the geometry of the test case include some rotation symmetrie which leads to a subdivision of the whole engine into 12 equal segments, of which only one has to be simulated. Several data structures were developed to represent and analyse the curvilinear grids:

- A cell, which represents the volume covered by 8 corner values and the connections to the 6 neighboured cells. Each corner value consists of a location in space and some scalar values, which represent a physical dimension such as pressure or velocity.

- A block, to represent the single elements of the multiblock structure. This blocks are the smallest logical units in the dataflow managment and contain a 3 dimensional grid of cells.
- A block container, which represents the total of one simulation step.
- A block connection, which represents the neighbourhood of blocks on cell level.
- A particel, which traces the appropriate cell for a given position in space with respect to some time and space dependent heuristics. A new position is always searched for by navigating through the grid of cells relative to the current position. To jump between neighbouring blocks, the just mentioned block connection is used.

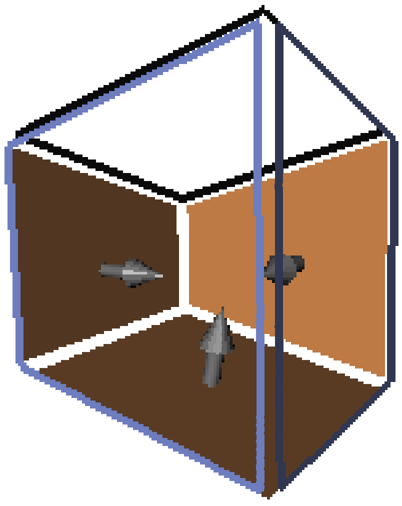


Fig. 12: Cell

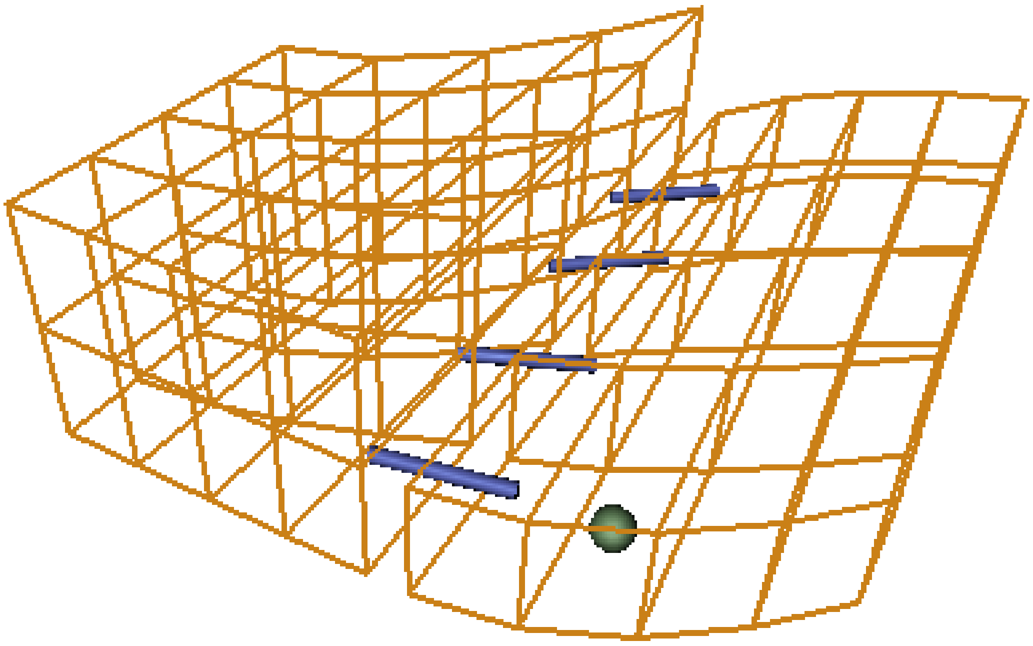


Fig 13: Two blocks consisting of cells, connected by block connections and traced by a particle.

2.4.2 Visualization

The original implementation of this project was done with Open Inventor and the porting to AVOCADO is not finished yet. The object representations are very similar in both systems, therefore the overall structure remains the same. As we are in a realtime environment, which means: the rendering pipeline must not be stoped by any time consuming calculation. Every object has to use a further asynchonous process, to achieve its desired task. Reading and writing of global data, like the simulation grid must be protected by semaphores and kept to a minimum. For the connection with the broadband network (or for the import of some simulation steps from disk for testing or demo purposes) we use a service - sensor combination. The service manages an IP socket connection via ATM and the asychronous data receival. All neccessary data structures are created or updated at this point. The sensor uses this service to check for incomming data and distributes this data to all connected post processing objects by using a special field, which holds a pointer to the current simulation step. After importing the data, the next step is the conversion to a specific visual feature. This is done in a separate sensor for each feature. All this sensors do the feature extraction in an asynchronous process. They are activated by new arriving simulation steps or run permanently, if the feature shows some dynamic properties. Until now, 4 different kinds of visualisations are implemented.

1. A polygonal surface, with a optional color coding of any scalar value available in the grid. The surface assembles a slice of the grid and every gridpoint, lying on the surface corresponds to a vertx in an indexed triangle strip set (fig 14) defining the surface. An average of 50.000 triangles per second computing time was benchmarked on a R10000 Mips processor.

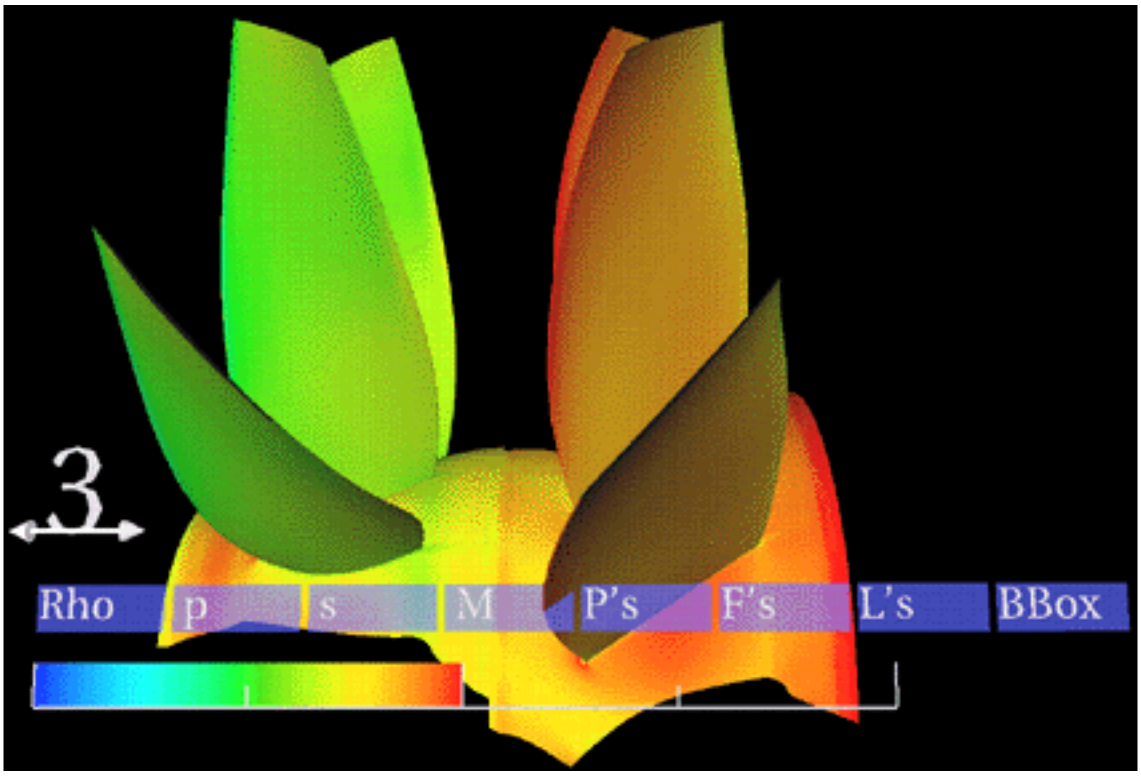


Fig. 14: Visualisation of the surface of 3 segments of the simulated engine with a color coding of the density Rho. The surface consists of about 30000 triangles per segment.

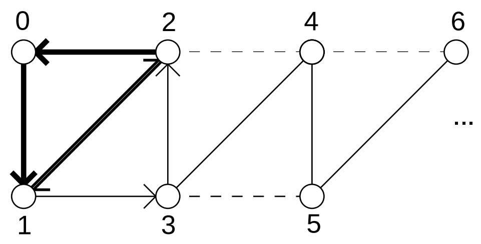


Fig. 15: Structure of an indexed triangle strip set. Every vertex is defined and indexed only once. 6 vertices define 5 triangles with correct ordering (1. triangle [0 1 2], 2. triangle [1 3 2]).

2. Isolines of any of the available scalar values for a given threshhold. The lines are generated as nurbs curves and can be switched on and off for each block individualy. The calculation is done by a 2d adaption of the marching cubes algorithm with a benchmark of 400.000 cells per second.

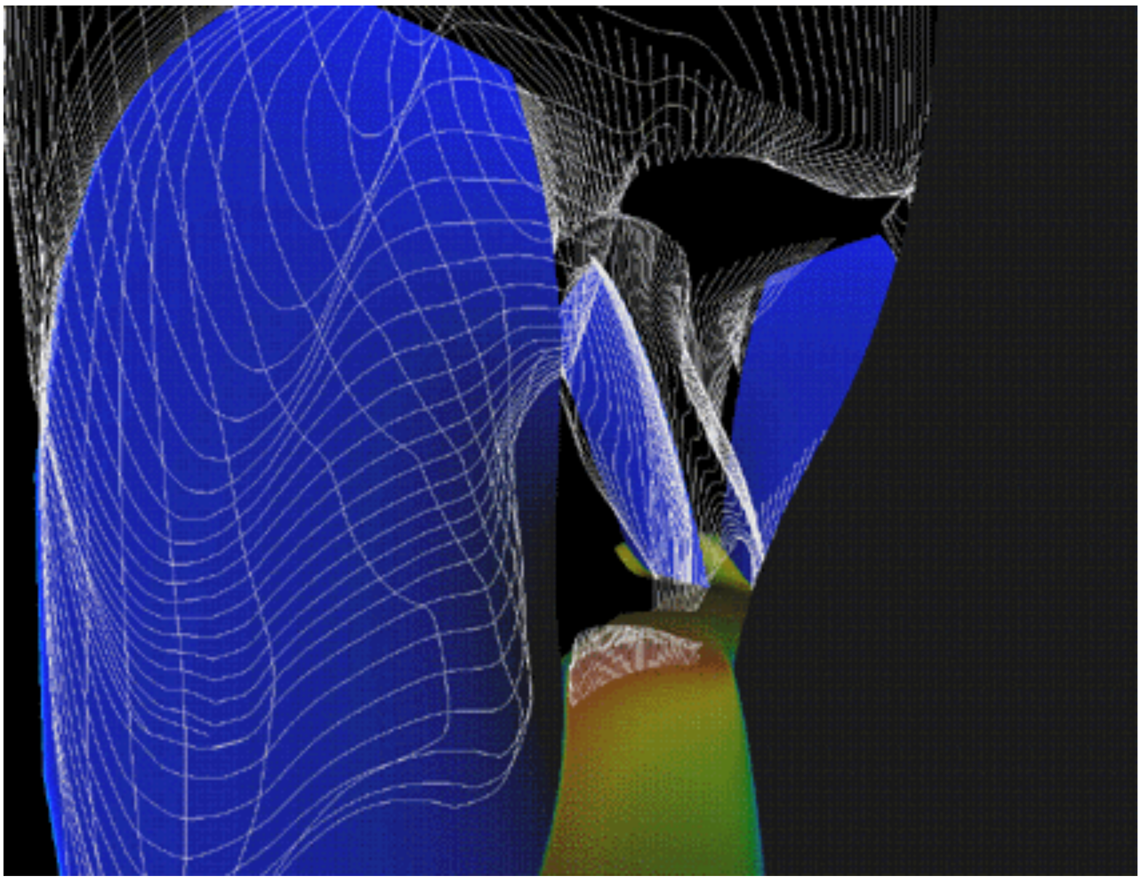


Fig. 16: Visualization of the isolines for the velocity values by a given threshhold of mach 1 for all blocks. This would be to complex in a realtime environment where only single blocks can be viewed.

3. A swarm of particles to visualize velocity values in a local area. The particle sources can be manipulated by changing position and frequency directly in the virtual scene and are shown as points. The calculation of their movement is complicated by the distortions of the simulation grid and the transitions between the different blocks. The calculation is done permanently and about 2000 particles per second can be traced.

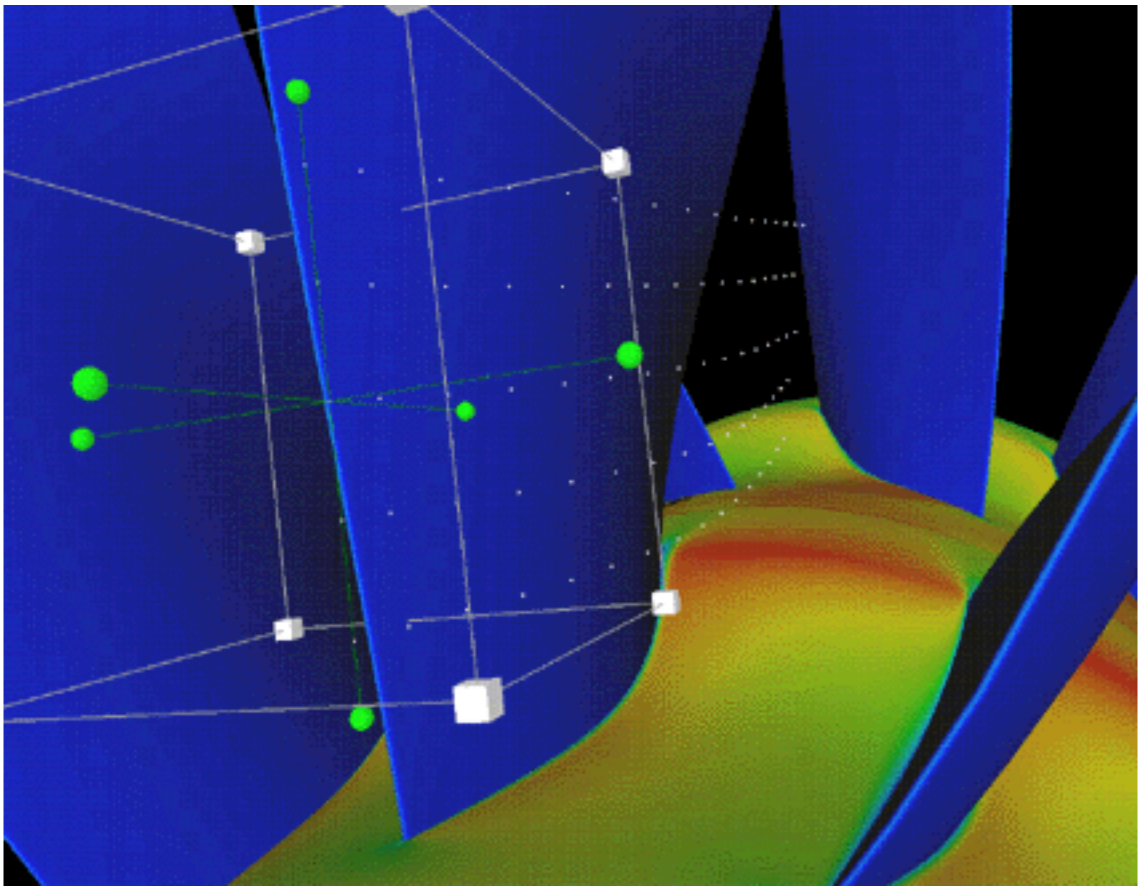


Fig. 17: Visualization of a swarm of particles with its interactive manipulator. The particle sources are bound to this manipulator and can be positioned freely in the entire simulation area.

4. A matrix of velocity vectors, whose position, size and density can be manipulated freely. Every vector in the matrix corresponds to a static particle and only a user related variation of the matrix leads to the tracing also used for the swarm.

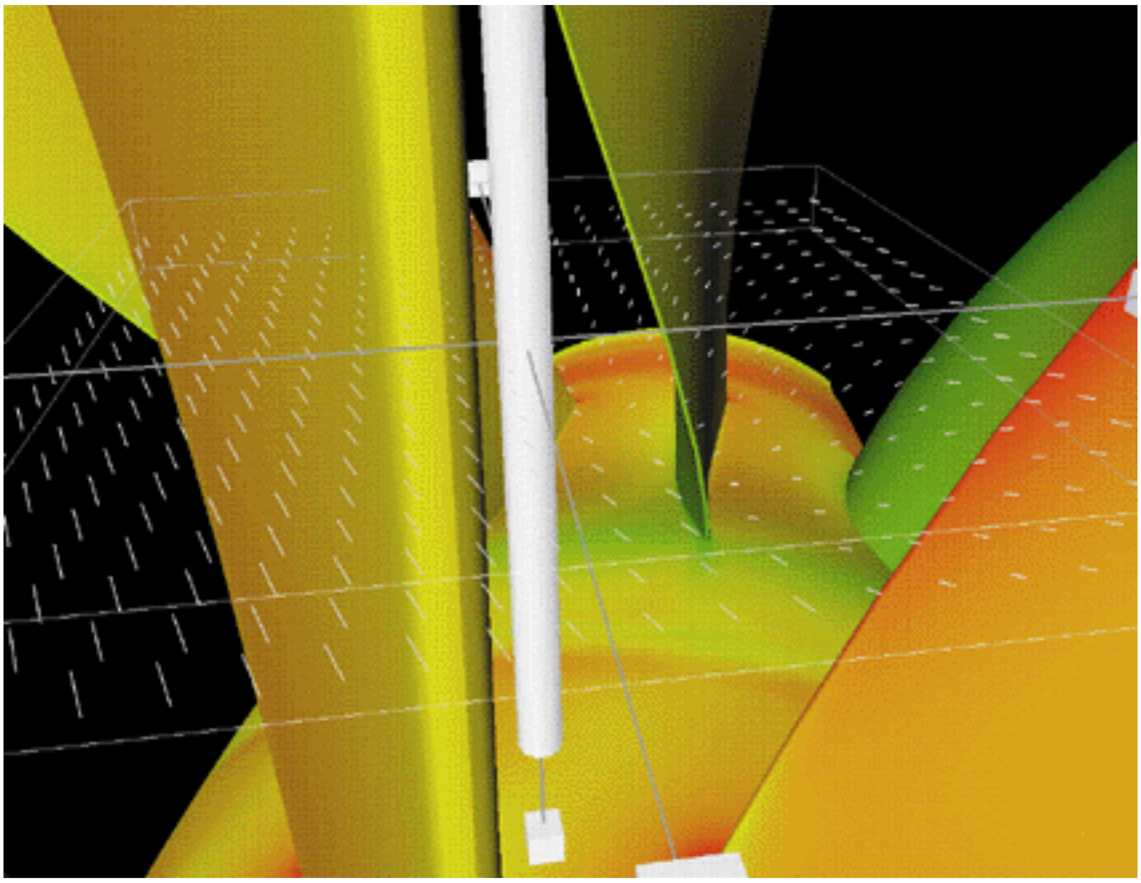


Fig. 18: Visualization of a matrix of velocity vectors whose length and direction correspond to the velocity values at their location. They can be translated, rotated and scaled by the surrounding manipulator.

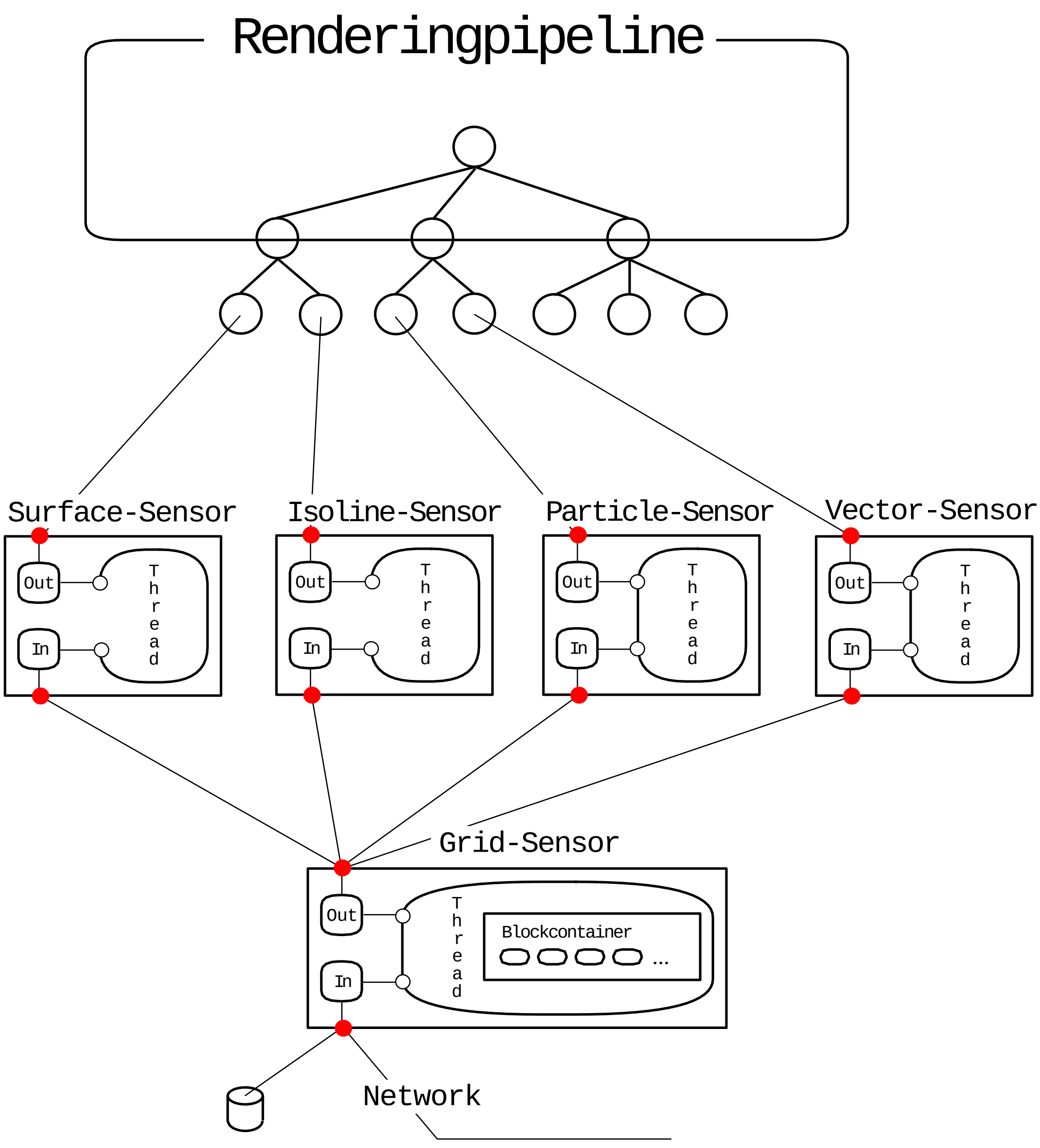


Fig.19: Overall structure of the modularisation

Figure 19 shows the overall structure of the described sensors and their connection to the modeling hierarchy. The calculated graphical data is propagated to specific nodes in the modeling hierarchy via fields. Because the normal copy mechanism of field connections would be to slow, only pointers are used. The modularisation allows an easy scalability for better calculation results by a simple duplication of one of the sensors. For example the amount of particles can be doubled by a second engine, if there are enough CPUs available. It´s very importand to keep track of all the different processes involved. Every sensor and the import service needs its own CPU, if a smooth transition of the simulation data into the virtual world should be achieved and there are already some other processes running which build the performer rendering pipeline.

3 Applications Toolbox

Applications can be seen as special arrangements of several selected tools to solve a given problem. The application areas were already discussed in the introduction chapter and we now will focus on the different parts, neccessary to construct the virtual environments. Several different tools, which means objects with some special behaviour or effect, were already build on top of AVOCADO. Some of them implement only a very simple idea and can be seen as the principal building blocks of the desired virtual worlds, while others represent higher level abstractions and metaphers.

3.1 Lower Level Tools

It is very easy to extend AVOCADO by a simple object, which implements a time or event related behaviour. The following list is a subset of this rapidly growing toolbox.

- Swarm node
- Video texture node
- Motif gui sensor
- Wave node
- Button node
- Tighten node
- Guard node
- Texture stack node
- Metronom node
- Explosion node
- Interpolator nodes
- Pendulum node
- Rotation node
- Rotation motor node
- Linear motor node
- 6 degree of freedom sensor
- Mouse screen
- Intersection service
- Pick node
- Dragger nodes

3.2 Controlling

This section describes general concepts and ideas what types of mechanisms are necessary or desirable for controlling a virtual world. Not all of the concepts have been fully implemented yet. There are two main things that need to be controlled while developing and running a virtual environment:

Hardware Resources

User Interaction and Navigation

3.2.1 Hardware Resources

By resources we think of

CPU time

Texture memory and main memory

Number of triangles that can be rendered in real-time

Number of sound sources that can be rendered in real-time

External Hardware

Virtual worlds tend to be very complex. The geometrical representation requires lots of the hardware resources. While developing an application you have to keep in mind that you need to keep a very high frame rate in order to guarantee a satisfying experience. Therefore the number of triangles that can be rendered at 20 frames per second or higher is strictly limited since you need to render eight images at a time if you run a CyberStage. In addition you cannot allow your application to start swapping main memory or texture memory which causes noticeable delays and disturbs your virtual experience. The same is true for the number of sound sources. The experience gets lost if you cannot render the sound fast enough to display the sound at the same time when it's geometrical representation is visually displayed. While developing custom and special effects you have to make sure that there is enough CPU time left for the main processes like culling, rendering and application management.

External hardware resources like the Sirius video board, which allows the import and export of video signals to the graphics subsystem of high end SGI machines, are also limited. To plug in a camera for some conferencing application or a performance means that there is no possibility for another video source to be connected. Because this limits are underestimated often, they have to be present while any application is designed.

How do we manage our limited resources now? We can separate the mechanisms into three groups:

1. time control

Switching scenes allows you to enable and disable CPU and rendering intensive processes like vertex animations and explosions. It gives you the opportunity to use a storyboard for effects and complexity of the geometrical representation to manage the CPU load and the utilization of your graphics engine. The sound resources can be handled accordingly. Time controlling also enables you to define animations paths a user may follow while exploring the virtual world passively according to a storyboard.

2. spatial control

Spatial control means to manage the resources according to the user's position in the virtual world. There is no need to animate geometry if a user is in a different room or miles away. There are a couple of ways you can accomplish spatial scene management. One possibility is to turn on effects according to a level of detail switch within your geometrical scene graph. Another way is to compute the distance of a user from a effect and turn it on only if a user is getting closer. Computing the distance requires only a minimum of CPU time compared to vertex animations used in an effect. In addition these so called "guards" may be enabled only if a specific scene is switched on via a level of detail.

3. general requirements

There are always a couple of general issues you have to keep in mind while designing your virtual world. Give low priority and asynchronous behaviour to non crucial processes like animation effects. While modeling use level of details as many as possible. Minimize the number of triangles while maximize the texture usage (considering the hardware limits) if texture handling is hardware supported.

3.2.2 User Interaction and Navigation

User related control techniques have to be developed with respect to our trained background in terms of interaction and navigation. Although we are in the virtual world not limited by the physical constraints of the real world, the perception and the acceptance of VR applications strongly depend on the users familiarity with the implemented interaction techniques. This means not, that real worlds interaction and navigation metaphers, like driving or flying for example, are the appropriate choice for controling every virtual environment. On the other hand getting lost in the virtual world because of a more or less exotic response to user interaction is a common problem in many VR applications. Until now for virtual worlds nothing like desktop, windows or menues as for monitor related man machine interaction is available. Even if not every problem can be solved perfectly by these abstractions, they are at least known to a broad community of users, while any VR installation and application comes with its own, unique set of controling devices and metaphers. There is no common input device like the mouse, but a broad range of spacial sensors, with or without buttons, and gloves, bodysuites, eyetrackers, ... Additionally a lot of experiments with even more complex interfaces like speach and gesture recognition are reported. The reaction of the virtual world upon some input from this devices happens always according to an abstraction, like a pointing gesture for instance, which is assigned to the specific device. In a second step this pointing may lead to an reaction of sensitive objects in the virtual world, which interpret this pointing in their own way. A button may generate a single event or a draggable object may move with the pointer, until it´s released.

A very important field of user interaction is the navigation in the virtual world. This means not only to control the movement in euklidean-, but also in information space. Although this information is always arranged in space and time, the navigation metapher should be completely different with respect to the specific topic of information. In the area of scientific visualization for example, the navigation possibilities have to meet the coding principals of the scientific data, to open any included secret spot, whereas applications in the area of training simulators do not explore the virtual world itself, but focus at the navigation or interaction process and its realness directly. If not for training purposes, the neccessity for navigation possibilities increases with the complexity of the virtual world. As in the real world, a guidance of the user through the information space is the basis for a successful navigation. This guidance can passively show the parallel layers of information available, or activaly push and pull the user, who´s in this case more and more changing to a passive consumer, through a sequential story.

3.3 Actors

We all behave with respect to things we can look at and feel, or smell and taste and events we can listen to [Gibs66]. The environment depends on what you can perceive with your sense organs. Gibson named this kind of perception : "direct perception". The sizes and masses of the things in our environment are comparable with that of the humans. You can see a moving car, but not the movement of electrons in an atom. So we can perceive only a special part of the physical processes. Processes in our environment are passing relative to the motions of a mechanical clock.

processes at the level of	time measured in
atom	millionths of a second
environment of the human	seconds, minutes, years
universe	millions of years

Table: Relationship between time and environment

Virtual worlds may also help to understand what happens at a different level. For example we can walk through a virtual molecule and "feel" the distances between the atoms.

3.3.1 Data Representation of Objects

What is a model?

Johnson [John95] defines a model as an encapsulation of the shape, shading, and state information of a character. Shape refers to the geometry of objects. This includes what geometric primitives they're composed of (spheres, polygons, patch meshes, etc.), as well as what geometric transformations are acting on them (scale, skew, translate, rotate, etc.). Shading refers to the subtle surface qualities of an object due to the material properties of the objects in the scene, local illumination effects, and global illumination effects. State refers to other more ineffable, but still perceivable qualities, like elements of personality. The properties of a model that can change over time are called articulated variables.

1. Physically-based Modelling

Physically-based modelling is used to describe dynamic objects by giving physical constraints for the motion of these objects. Objects move according to physical laws, e.g., Newton's Laws of motion. The task is to describe the force or force field which causes the movement of an object. The motions are then calculated using the object's mass properties. Objects within a scene influence each other's movements, e.g. if they are attached to each other or if they collide with each other. All this information must be considered to simulate realistic movement of objects. If objects posess a complex mechanical structure, this structure must be taken into account since all the parts may interact with each other.

Generally speaking, each object in a dynamic scene interacts with any other object of this scene. According to the type of these relations, different motion rules and constraints are derived: geometric contact directs the motion of objects; kinematic links require calculation of a kinematic solution (forward and inverse kinematic); a magnetic relation leads to a force which depends on the distance between the two objects, etc. Physical and heuristic rules are used to simulate these systems. [Ast93b,Bar91].

The following list contains some categories of events, that change the layout of the environment. Some of them will consider objects, such as collisions or deformations and others describe changes of the layout at the level of the surface, for example deformations and disruptions [Str96,Gib79].

- Rigid translations and rotations of an object These are *displacements*, such as open a drawer or *turns*, for example open door. Furthermore *combinations* consist of both, translations and rotations, for example a rolling ball is such a movement.
- Collisions of an object, we can differ between collisions with or without rebound.
- Nonrigid deformations of an object. Objects can be classified as inanimate or as animate. An example for deformations of inanimate objects are the drops of fluid. An example for animate objects is the change of posture of a human.
- Surface deformations, examples for surface deformations are waves, or the flow of a liquid. Deformations can cause elastic or plastic changes of a surface. Note that a deformation will not cause a disruption of the surface or disturbing the continuity of the surface.
- Surface disruptions, rupture occurs when the continuity of a surface fails. A surface can be disrupted for example by rupturing, cracking, disintegration or an explosion.

2. Objects in Virtual Space

The possibilities to interact in the real world are enormous. It's impossible to find an categorization for all possible objects in the world. There are 20,000 - 30,000 everyday objects [Norm88] and nobody knows all the objects in the world . There are natural objects, such as stones and artificial objects, such as furniture. New objects are also created or available objects are changed by phenomens of nature or by manufacture. J.J.Gibson [Gib 97] is trying to find a description of the different objects. He is talking for example about hollow, elongated and rigid objects. But finally he is saying: "The list of examples could go on without end". So he almost only distinguishes between attached and detached objects or between animate and inanimate objects. For a first attempt we will follow this cathegorization, but further cathegorizations are maybe useful to learn about special domains.

Inanimate Objects are described in terms of the environment, e.g. by their surfaces. They are static, but they have of course a function. E.g. the ground can support walking or a hole affords to fill something in.

Animatable Objects have the Potential for Change. What happens, if you pull on this part, if you push here, or prod there? A model is an immutable thing; static and stiff. An animated model is one which is changing in some particular way over time. Each animated part can be accessed through an articulated variables. Looking at a model from the outside, the only thing that is visible are the variables (or the fields). Some of the variables are read only, some are both readable and writable. Some change over time, some don't. To manipulate the model is to write a value to one of the model's variables. In order to write to a variable, a process must first attach to a variable. Once they've successfully attached to a variable, a process can write new values to it. When they are done writing to the variable, they must detach from it.

3. How to build a model

In animation, the "actor" is a combination of the animator drawing the scene and the voice talent doing the character's voice. Many an animated character's look is directly informed by the voice talent that makes them speak. Johnson suggests:"The animator has to step into the character and look into the mirror."

1. Before you can start to build a character you have to write down it's role. The character is the particular instantiation of that role for a given performance situation. It is constrained by the role, which itself must work with in the bounds of the story. Available choices are always constrained by the role and its place in the story.

2. Now you should describe it's physical appearance, it's typical body postures and the contact points with physical objects. How does the character hold itself? Is it happy or unhappy? Does it walk like a soldier or like a frightened schoolboy? An important part of an actor is the face. This has not to be a human face with eyes, a nose and a mouth. Here a "face" refers more to the meaning of the character What does this character's face look like? Is it warm, denying, uncomfortable, obtrusively, affords it some kind of communication or is it only a blank undefined nothingness.

3. Now it is time to specify the activities that the character will be asked to perform. Again: Does it physically has everything it needs? What are it's everyday objects? Is it able to interact with the objects in it's environment and to perform the required tasks? At this point, we're still building up the character. If we suddenly realize that our character needs some particular property, we might look in the script to see if there's any information. A system has to have features to recognize such missing parts to change interactively parts of a character or to exchange an agent by another.

4. On which level have we to interact with the character? Zeltzer discusses a three part taxonomy of animation systems: guiding, animator level, and task level. Guiding includes motion recording, key-frame interpolation, and shape interpolation systems. Animator level systems allow algorithmic specification of motion. Task level animation systems must contain knowledge about the objects and environment being animated; the execution of the motor skills is organized by the animation system.

4. The Illusion of a lifelike and autonomous Character

The creature must be able to adapt and to learn from past experience and its behavior should be lifelike. Maes [Maes95] defined this term as non-mechanic, non-predictable and spontaneous. That means:

- the character must react
- the character must be seen as having an independent existence
- the character must have choices to make
- the character must adapt (e.g. to the situation, to the experience level of the user)
- the character must display variability in movement and response

With a few exeptions, the behavioral complexity of the creatures created to date has been limited. Typically the creatures pursue a single goal or display a single competence. For example work has been done in locomotion [Zel90b,Bad93], flocking [Rey87], grasping [Kog94], and lifting [Bad93] Tu and Terzopoolos' autonomous animated fish [Terz94] which incorporate a physically based motor-level, synthetic vision for sensing, and a behavior system which generates realistic fish behavior. But learning is not integrated in the actrion-selection mechanism , the fish address only one goal at a time, and the action-selection architecture is hard-wired, reflecting the specifics of the underlying motor system and the repertoire of behaviors they wished to demonstrate. Tosa [Tos93] used neural networks to model an artificial baby that reacts to the sounds made by an user. McKenna and Zeltzer [Zel90a, Zel90b]. demonstrated an articulated figure with 38 degrees of freedom, that uses the gait mechanism of a cockroach to drive a forward dynamic simulation of the creature moving over even and uneven terrain. It is an example of how successfully biologically-based control schemes can be adapted for computer animation. Sims designed a system for making creatures that, using inverse kinematics and simple dynamics, could navigate over uneven terrain [Sims87]. This system was notable in that the notion of "walking" was generalized enough that he could generate many different kinds of creatures that all exhibited different behavior very quickly. More recently, Sims has developed a system for quickly prototyping creatures embodying a set of physically-based behaviors by breeding them [Sims94]. He presents a genetic language that he uses to describe both the shape and the neural circuitry of the creatures. His work is most interesting in the context of building systems in which creatures are bred by using aesthetic decisions as fitness functions. This work, more than any other, shows the power of genetic techniques when applied to complex computer graphic character construction problems. Even more recently, Blumberg and Galyean [Blum95a] demonstrated a "directable" dog character that can respond autonomously, in real-time to user input in the context of a larger, scripted narrative activity. He is using etological theories for his work, but he is also looking at classical animation.

5. Designing Reactive Behavior

It has been learned that Artificial Intelligence is not sufficient to build reactive behavior as in the real world. Here is the process of knowledge or reasoning less important than adaption and learning. Biology and especially ethology helps to understand the mechanism which animals use to demonstrate adaptive and successful behavior. So a number of action-selection algorithms have been proposed by etholists, for example Lorenz in [Lor73], Tinbergen [Tin50] and Toates [Toat91]. But more important is very often the observed behavior over time in a special domain and not to study is biological theories. Blumberg paraphrases Papert "biology should be on tap, not on top" [Blum96b]. It is important to describe behavior on the right level, to find a good level of abstraction in a model. Important features should be described, but the description and the implementation should still be understandable. It should be possible to decompose a complex system in many independent parts and also to exchange parts, to debug parts and to change the configuration of the system. Very useful in this context may be Marvin Minsky's definition, how our mind might be organized (in his book "The Society of Mind"):

"Any part or process of the mind that by itself is simple enough to understand - even though the interactions among groups of such agents may produce phenomena that are much harder to understand."

Think about a world which consists of thousands of autonomous sources. Each source is able to produce a short colored flash followed by a beeping sound. If a source in it's neighbourhood will hear or see this event, it will also beep and flash, but in a little bit different color. A source will sample it's environment time by time. To make these autonomous sources to individuals we give each of them a different sample rate, a different beep and a different color. An observer who is walking around in this scenario will find sources which are building sequences. He will walk through centers of confusion, see harmony and disharmony in very close neighborhood, chaos and regularity and then there is only dark silence. It is hard to describe your impression, to figure out how it works, to see the structure behind these irregular structures.

3.3.2 Autonomous Agents

Autonomous agents are computational systems that inhabit some complex dynamic environments, sense and act autonomously in the environment, goals or tasks. They can "live" in 2D or 3D physical worlds. Maes describes in [Maes94] "knowbots", software or interface agents, which inhabit the digital world of computers and computer networks. A real world agent [Nag96] can support its user in performing tasks in real world environments, such as place-to-place location guidance, instruction in physical tasks, and augmentation of human knowledge related to the physical environment. Animated autonomous objects are endowed with the possibility to decide what to do and to change their skills or to modify their geometry accordingly or the structure of the system itself. The animation arises out of the actions of these autonomous objects rather than from the keyframes of an animator. Johnson defines as a software module; an autonomous black box that has a well defined purpose, a separate namespace and its own thread of control it explicitly refers to such a black box [John95]. It explicitly refers to a black box. If you open this black box you will find a process inside which should not be too complex and easily understandable by a programmer.

1. Implementing Reactive Behavior

- Perceiving the Environment

Following Rasmussen [Ras83] we characterize knowledge about the figure and the world as follows:

"Signals represent sensor data-e.g., heat, pressure, light-that can be processed as continuous variables. Signs are facts and features of the environment or the organism."

In a virtual world signals are directly messurable in the environment. This can be an open door, the elbow position of a character or the curvature of a surface. Signals are discretely sampled using a mechanism called a receptor. Receptors are used by agents to discretely sample signals variable at a given frequency. Each receptor has a sampling frequency associated with it that can be modified by the agent. If their value changes by some epsilon from one sample to the next, they transmit a message containing the new value to the appropriate agents, which prompts the agent to recalculate itself. All receptors can be embedded into a single system process, but it's also possible to distribute them in several processes accessable over a network.

A sensor agent computes a boolean value using the information gathered by its receptors. It produces an assessment of the item it perceives (i.e. LightIsOn, ItIsToLoudHere, FriendIsNearby, iAmSitting, aCupIsNearby, etc.). Sensor agents corresponded directly to signs. They enable the character to perceive itself and its environment at a level above the articulated variables of a model. Sensor agents does not depend on available hardware or on the current environment. They have not to differ between real and virtual actors, between the real and the virtual world. So different characters and installations can use the same sensor agents, but the result may look completely different, because the receptors are handled in a different way. A receptor will be evaluated many times relative to the sensor agent. They represent an assumption that communication is expensive compared to computation; it is cheaper for a sensor agent to embed a computation somewhere else than to keep asking for the value and determining itself if the value has changed enough for it to recalculate itself. The sample frequency corresponds to the minimum reaction time of a character. It can also be used to model the attention of virtual actors.

E.g. it can be changed, if the character is awake or falling in sleep or if it should pay less attention to events happening in greater distance.

- Skill Agents

The skill agent has some activity that it's trying to perform over some period of time, and it does this by measuring and manipulating the variables that it has access to. A skill can be executed, if it's preconditions are true. These preconditions are calculated by sensor agents. As the skill agent begins, it attaches to the variables in the character and the environment that it will be manipulating. When the skill agent finishes, it detaches from the variables in the character and the environment that it was manipulating. Many times the best way is to begin by constructing the skill agent is simply as a process which invokes single reflex, where the skill agent merely sets the articulated variables of the parts of the model it is concerned with. When building a skill agent, ideally, we want to always be thinking about how to write it for more general use.

- Goal Agents

Goal agents embody an explicit goal, and are described in terms of sensor agents. Different kinds of goals, characterized by their lifespan are:

- persistent, constant goal: e.g. stay alive
- transitory: e.g. hold the book, be hungry, get tired

- Motor Goal Parser

A motor goal parser will take care of translating a given task from a natural language interface to a set of task primitives which can then be passed to the skill network. The skill network will be responsible for finding out which skills need to be executed in what order and for invoking the appropriate skill agents corresponding to those skills. Complex activities like "making breakfast" are composed of component acts such as "make coffee" which are in turn composed of yet other behaviors - get the coffeepot, boil the water and so on. That means also a task has to be devided in a number of *sub-tasks*. For example "grasp a cup" is a sub-task in the example above. Such a sub-task can furthermore be devided in subtasks until you get task-primitives. Task primitives limit the decomposition of a task. Schanks Conceptual Depency (CD) theory [Scha75] introduce a set of primitive "ACTs", some of which describe physical actions and some of which describe abstract "mental actions". Zeltzer and Johnson develop in [Zel94] a set of task primitives, which are based on Schanks theory.

2. Virtual Humans

The model of the virtual actor has to be realistic enough to be believable, but it has also to match realtime requirements. To perceive the posture of a human body it is not required that cloth of the suit of the virtual actor will be perfect deformed during moving. The posture can be perceived, if the limbs are in a natural configuration. We try to make the face and the hands of the actor as realistic and movable as possible. The rest of the body will be builded by more rigid elements. However added textures and video textures will equip these parts with the required realism. The kinematic structure of the model is based on a human skeleton. Forward kinematic skills can be realized by finite state machines with usually more than one state. Inverse kinematic skills usually have just one state and contain an inverse kinematics calculation procedure. Realistic walking is also a hard problem, because humans are skilled to see even small mistakes in the way other people walk. An interaction environment can be inhabited by different types of virtual actors, e.g. personal agents or agents offering a special service to the user.

- Virtual Representants

A virtual actor can act instead of a real person in some domains, e.g. to arrange a meeting or to introduce somebody in a domain or to present a product. The face of the human could be captured and mapped onto the face of the virtual actor. This representant could be dressed like it's owner. Some gestures and postures of the human can be recorded in the cave, by the parrot or by cameras which are installed at the terminals. This data can be used to improve the movements of the representant. The virtual actor has then some characteristics in common with it's owner and all actors move in a different way.

- Social Interaction

Interactions between humans are multimodal. A composition of verbal and nonverbal communication improves the believability of a virtual actor. Such a virtual actor can be part of a "social interface". It can replace parts of common interfaces. Instead of pressing buttons, filling in forms it could be easier to interact instead with a virtual actor and tell him the task of the system. Since nonverbal communication is an important part of such an interface we will give a short introduction in this field.

- Nonverbal communication

Nonverbal communication is used e.g. to express interest, friendliness, anger, disgust, fear, happiness and sadness. This form of communication can serve as a fast feedback and reduces the complexity of verbal communication. It tells about the context of a conversation. Furthermore it shows how interested the communication partner is in a communication and if he is experienced in this domain or not. The communicating partner has to observe and to react to special postures or facial expressions.

- face-to-face communication

One of the major features of face-to-face communication is its multiplicity of communication channels that act on multiple modalities . 65 percent of a face to face communication is conveyed by nonverbal elements [Bech96].To realize a natural multimodal dialogue, it is necessary to study how humans perceive information and determine the information to which humans are sensitive. A face is an independent communication channel that conveys emotional and conversational signals, encoded as facial expressions integrates speech dialogue and facial animation. A virtual actor can be equipped with a number of facial patterns, belonging to conversational situations like "yes", "no", "I donot understand","thanks" . While the actor is speaking they will be displayed on it's face belonging on the actual context. Nagao and Takeuchi display a artificial face on a monitor and use it for a human computer dialog [Nag94]. Their experiments have shown that facial expressions are helpful, especially upon first contact with the system. They mention also that featuring facial expressions at an early stage improves subsequent interaction.

- Body Postures

Changes of the body modulate ongoing activities. Movements of the arms will support the way of talking to another person. The emotinal state of a person will affect the way he walk or move.

3.4 Conferencing

Today's technology and advances in telecommunication rapidly change the way business is carried out, making it a globally distributed and thus a more electronically-based process. Support for interaction, coordination and group work in the emerging distributed businesses, should be adapted accordingly, both in terms of the way it is carried out and in terms of the tools used. Social interaction, between members of widely dispersed working groups, becomes more difficult and must be supported by appropriate collaboration facilities. CSCW systems, video-conferencing and electronic meeting systems, facilitate the process of preparing and carrying out meetings, and at producing material via a group effort in a distributed business. Although desktop video-conferencing or electronic meeting systems address such situations, it is still recognized that there are many situations, such as distributed negotiation, conflict resolution meetings or remote seminars and teaching, where body-language and eye contact become important. Ideally, in such situations, we would like to provide geographically separated group members with a sensation of being in the same room at the same time and meeting face-to-face. This is what is called "co-presence".

The degree of co-presence can be enhanced by providing the group with a sensation of sharing the same physical space, by maintaining body size, eye contact and gaze awareness. However, meetings of local or geographically dispersed groups require additional support for preparing documents, use of visual aids and generation of various artifacts during and after a meeting. A variety of tasks are performed by different members of the group during this process. Members might need to schedule co-presence sessions, prepare slides or electronic documents, generate reports, review diagrams or charts depending of the nature of the enterprise. Therefore, before, during and after a co-presence session support is needed also for co-working.

3.4.1 Requirements

A system that provides high degree of co-presence and was described in a previous section, is TELEPORT. It is an experimental teleconferencing system aimed at enabling small groups of people, which are geographically separated, to meet as if face-to-face. The prototype has been used in various applications areas such as tele-conferencing, telelearning for the academic or business communities, and distributed musical rehearsal. From our experience, while using the TELEPORT system for co-presence, became evident that additional support is needed for co-working. For example, remote meetings would normally include the use of documents, slides or other visual aids; tele-learning requires teaching material to be prepared in advance and included in the tele-conferencing session; collaborative architecture design requires 2D or 3D models to be shared. In addition, various CSCW tools are commonly used by members to facilitate different aspects of group work. The tools used vary from shared calendars, editors and shared workspaces, to collaborative modeling and design tools. Advanced tools such as visualization, animation and simulation tools, already in common use for architecture design, engineering and manufacturing, are also becoming a major support for teaching in the academic as well as in the business community.

Therefore, tele-conferencing sessions in order to be successful and cost effective, should not only provide a high degree of co-presence but also support the different tasks that are performed by geographically dispersed groups. The following sections present ways of achieving co-presence and support co-working by the use of VR systems, such as TELEPORT and CyberStage. To exemplify the approach, different applications areas are considered, ranging from business seminars and teleteaching to collaborative modeling and manufacturing.

3.4.2 Meeting example

A variety of CSCW tools provide facilities for collaborative information sharing and co-working. Such tools could be used before, during and after a TELEPORT session to provide support for preparing and changing any material necessary for the session and most importantly for supporting the brainstorming, structuring and evaluation of ideas during the session. Real-time composition of live-video, synthetic backgrounds and electronic documents is then projected on the wall-sized display of TELEPORT. However, artifacts needed for the co-presence session such as documents, spreadsheets, tables etc., should be prepared prior to the meeting. The preparation process might involve participants from different organizations in different locations. For the preparation phase of the co-presence meeting a system such as the BSCW Shared Workspace system [Bent 97] could be used. BSCW (Basic Support for Cooperative Work) is a system developed by GMD.FIT, which provides support for cross-platform information sharing for groups of users over the World-Wide Web.

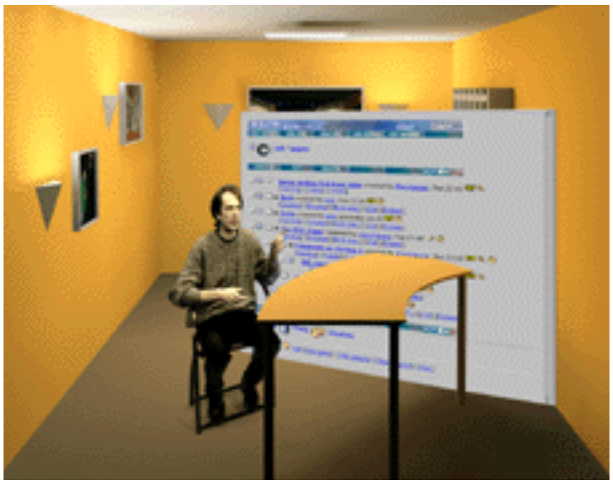


Fig. 20: Remote participant with virtual projection wall in TELEPORT's virtual extension

Once the preparation phase is completed, group members from different sites are ready to meet in the TELEPORT display rooms in each site. During the co-presence session information in the form of electronic documents, presentation material and shared workspaces is displayed on a virtual projection wall which is blended together with video imagery from remote participants into the virtual extensions of the display rooms, as shown in Figures 20 and 21.

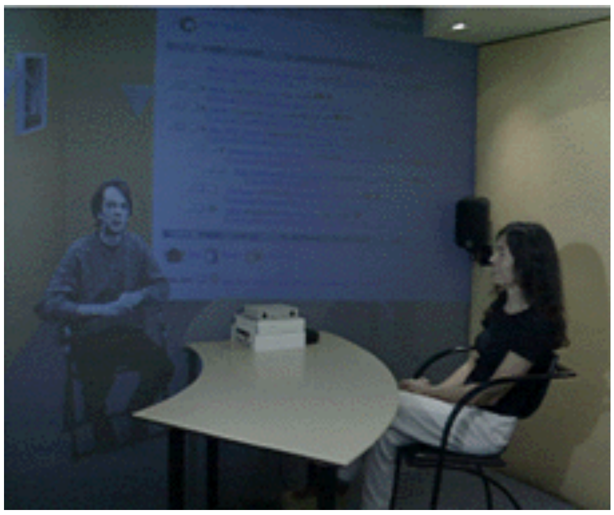
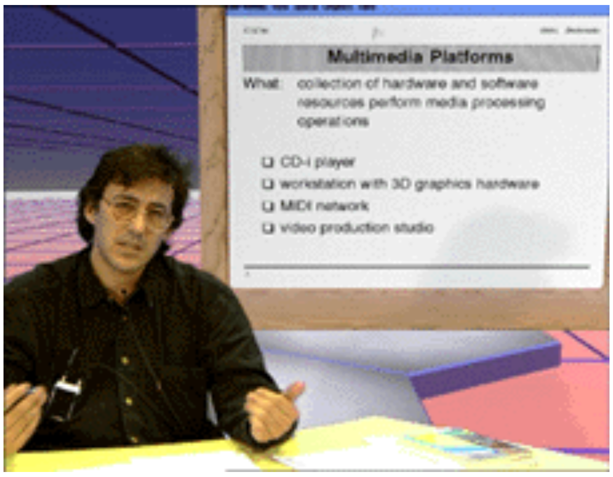


Fig. 21: Session between two remote sites as seen from within a TELEPORT's display room

This is achieved by introducing a virtual projection wall that is mixed in the 3D model of the virtual rooms whenever that is necessary. This virtual projection wall is directly connected, for example, to the participant's notebook. Thus group members can choose what to show to the other participants, i.e. slides, pie charts or the way a software tool it's been used.

3.4.3 Telelearning example

In today's distributed companies seminars and skill improvement courses could be very costly both in terms of time and money, because of the travelling involved, either for the expert giving the seminar or for the people participating. Also seminars very often are about or make use of specific software tools which require the use of a well prepared seminar room with the necessary hardware and software installed for this particular situation. To save time and reduce costs of traveling, in our approach the expert and students meet in a TELEPORT session and are able to view teaching material, ask questions and have a seminar that provides the illusion of been in a seminar room together with other participants and face to face with the expert.


Multimedia Platforms

What: collection of hardware and software resources perform media processing operations

- ❑ CD-i player
- ❑ workstation with 3D graphics hardware
- ❑ MIDI network
- ❑ video production studio

Fig. 22: Integrating teaching material in a co-presence session

Universities so far dispense courses only regionally to their students. However, instruments for global delivery of teaching content exist and are becoming cheaper by the day. In the fall term of 1996, a weekly high-bandwidth ATM connection between the University of Geneva and GMD headquarters in Bonn has been used to conduct regular tele-learning sessions. Our goals were to gain experience with high quality video conferencing technology and to broaden the perspective of course presentations while simplifying the logistics for remote participants. During these sessions, the teacher's image was extracted from a controlled background and blended with teaching material and virtual backgrounds, as shown in figure 22.

Another important aspect of this approach, is the possibility of using animation and visualization tools to enhance the understanding and facilitate the learning process. Animation and visualization tools have been extensively used for teaching within the academic community [Brown 88, Brown 92] and proved to be a successful teaching aid. Different possibilities exist for incorporating animation and scientific visualization in a co-presence session. For example, animations could be shown to students as pre-recorded video displayed on a virtual video wall, in a way similar to that of displaying transparencies. Another possibility, is to integrate the 2D or 3D visualization as part of the 3D model of the synthetic background of a TELEPORT session.

3.4.4 Collaborative VR example

Collaborative Virtual Reality systems are currently used in various application areas, such as landscape architecture or automobile manufacturing. Co-working in such application areas requires a higher level of interaction and model manipulation, making CAVE and VR systems an ideal tool. However, the high communication and hardware costs still limit the extensive use of CAVE systems.

The AVOCADO system presented earlier, will provide the possibility of connecting different VR systems, such as TELEPORT, Responsive Workbench and Cyberstage, thus reducing the cost of hardware and allowing sites with different VR systems to be connected for collaborative work. One or more TELEPORT rooms can be connected to CyberStage. The 3D model used in the CyberStage site could become part of the virtual extension of a TELEPORT room. In addition, the video image of the remote participants can be extracted and blended into the virtual extensions of the TELEPORT rooms. For the Cyberstage site, the participants of the remote TELEPORT rooms could also be included within the virtual 3D space. Interaction and manipulation of the 3D model could be restricted to the CyberStage site, or also allowed within the TELEPORT rooms, depending on the infrastructure of each site.

3.5 Sound Rendering

3.5.1 Integrating Visual and Auditory Display

It is generally agreed on that in our vision-centered culture the second important sensory channel is the auditory channel. Once the visual channel has been sufficiently developed in a new medium, usually the next one to be integrated is the auditory one. This can be seen for instance in the evolution of cinema, which was mute in the beginning or in the development of the computer, which hardware manufacturers like to call multi-media workstation today. Which aspects have to be taken into account when combining a visual with an auditory display can be clarified by a comparison of visual and auditory scene analysis. Both analyses carried out by the human sensory system are concerned with grouping problems. We can think of scene analysis as a process of forming hypotheses about the way the sensory evidence is to be grouped in order to form as unambiguous entities as possible. In vision, it is mainly color, texture, and spatial information combined with a set of rules (e.g. concerning occlusion and extension) which allows us to identify objects in a visual scene and organize them in an inner representation called reality. All information is carried by light, which reaches our eyes as reflections from objects.

Auditory perception works similarly but with a completely different type of sensory evidence. First of all, sound doesn't represent objects but events. Sound is only generated if there is some sort of motion or action (i.e. an energy source) is involved. Information about acoustic events in our environment is transported to our ears by sound waves. The problem to be solved by the auditory scene analysis process is to decide which auditory cues (spectral and temporal aspects of the sound signal) belong to which sound source. Our ears are not primarily concerned with reflections (they may even become a source of confusion) but they are interested in the nature of the energy source. Vision uses the reflections of light and usually cares less about the light sources themselves.

These and many other differences account for the different, often complementary kinds of information about our environment the two sensory channels can supply us with. When looked at separately, the visual and the auditive channel provide us with different perspectives of the environment we analyze. But when experienced together, they form a whole which is more than the combination of its parts. Sensory cues from both channels are used to analyze an audio-visual scene. Evidence from one channel can be used to complement missing information in the other one. Sound informs us about events we cannot see because they are not in our field of view (e.g. happen behind us or behind other objects), or they are too small or move too fast to be seen (e.g. vibrating objects), or can't be seen at all (e.g. vibrating air volumes). Through sound we learn about the material of an object colliding with another one or about the texture of an object sliding over another one. Sound informs us about the nature of interaction between objects and about the forces involved. And, last but not least, sound informs us about the environment in which we perceive it and in which it was produced and emitted. This is how we distinguish inside from outside spaces, small and large, empty and fully furnished rooms.

So there are many good reasons for combining auditory cues with visual cues in virtual environments. We saw that this combination provides virtual environments with redundant information (e.g. spatial or temporal cues), which is important to reduce perceptual ambiguity. We showed how auditory cues complement visual cues in an integrated audio-visual scene analysis process. And we may add two other aspects here. Firstly, the presence of sound in an interface masks environmental sounds and therefore increases the degree of immersion in a virtual environment. And secondly, experience has shown that fine grain auditive feedback to user actions enhances the sense of presence in virtual spaces.

3.5.2 Integrated Simulation of Image and Sound

But integrating visual and auditory displays into one display system is not enough. Such an integration has to be reflected also on the level of the simulation process which generates the visual and auditory stimuli. These have to be presented such that they don't contradict but support each other with respect to a set of constraints characterizing the kind of reality to be simulated. When simulating an environment structured by the laws of causality as we know them from our everyday world (which is the normal case), then the coherence of presentation mainly concerns spatial and temporal cues. In such a reality, a sound produced by a (visible) sound source has to be emitted at exactly the same position at which the source is visible, i.e. visual and auditory space have to be coincide. What has to hold for spatial cues in such a world is also required with respect to time. Visual and acoustic aspects of the same event have to happen, i.e. have to be simulated at the same time. Since sound propagates much slower (~340 m/s) than light (~300000 km/s) it always reaches the observer later than light stemming from the same event. Whilst the propagation delay of light can be ignored in the simulation process, the speed of sound has to be taken into account for distances bigger than about 10 m (i.e. ~30 ms time delay) between the observer position and the event location.

Since other types of realities, as we can find them for instance in artistic applications, may need other sets of constraints to be met, it is important that the simulation system is flexible enough to allow for modeling of such realities as well. As an example we may think of experimental interaction metaphors, which may need special simulation modes (e.g. cut-like discontinuous navigation). Furthermore, a system architecture open at this level would invite for experimentation, as it is required to discover innovative interaction or navigation metaphors. Generally speaking, an ideal audio-visual simulation system should allow any relationship between visual and auditory cues to be expressed explicitly. Naturally, this would include the special relationships necessary to simulate worlds similar to our everyday causal world, which still is the first requirement and most important validation criterion for a virtual reality system today.

3.5.3 Auditory Scene Synthesis

Realizing an integrated image and sound simulation system means integrating sound synthesis and its control into the well understood and validated scene graph based framework of visual simulation. Auditory scene synthesis (the counterpart to auditory scene analysis) serves as the conceptual framework underpinning this integration. By auditory scene synthesis we understand the whole process of defining auditory scene elements, organizing them in an audio-visual scene, and rendering this scene for an auditory display. The result of the auditory scene synthesis process is a sound field which is presented to the user through the auditory display. The user (re)constructs the auditory scene by analyzing the synthetic sound field.

1. Auditory Scene Elements

An auditory scene element represents the sound to be produced upon a certain event. An event is usually generated in the simulation process as a direct or indirect (scheduled, scripted) reaction to a user action. In the simplest case, an event only triggers the playback of a prepared sound sample (e.g. sound of a switch to be played once the switch is operated). But an event may also include event attributes or it may consist of several sub-events, each of them carrying its own attributes. Such compound events control more complicated auditory scene elements. Typically such scene elements represent continuous sounds, which start at some time in some specified way and stop later in another controlled manner (e.g. a motor switched on, running for a while, eventually changing speed, and then switched off again). Event attributes are used to characterize the event and are used to shape the sound accordingly (e.g. intensity of a collision sound depending on the velocities of the objects involved). Thus, event attributes can be used as (or are mapped on) sound model parameters. A sound model [Eckel 93] describes a class (in the perceptual sense) of sounds. Instances of this class are distinguished by their model parameter values, which are needed to synthesize the sound. There are models that simulate the sound production mechanism (e.g. physical modeling [Cadoz 91]) and others which directly model perceptual attributes of the sound signal (e.g. spectral modeling). Examples of physical modeling parameters are mass, stiffness, velocities, spatial position whereas spectral modeling parameters specify amplitudes, frequencies, durations.

Which model to choose depends on the sound quality needed, the control needed over the sound, and the computational resources available. As a rule of thumb it can be said that physical modeling usually needs more CPU cycles than signal oriented synthesis techniques, which do not simulate the total comportment of a vibrating object but only mimic the sound signal resulting from all these vibrations. Physical models are preferable when the sound radiation is an important aspect in an auditory scene, because they often use spatial representations, i.e. they distinguish how an objects vibrates on one or the opposite side. Naturally, physical models are better suited to simulate complex interactions between objects (e.g. brushing or scratching) but they are usually much harder to control than signal models. This is why mostly signal models are used when sampled sounds cannot be used in an auditory scene. The disadvantage of sound samples is that they can only be transformed (i.e. parametrized by event attributes) in a very narrow range. Individual perceptual attributes of sampled sound material are not accessible because of the lack of a real sound model (the sample is class and only instance at a time). Sound samples can only be transformed by being reproduced at different pitches, amplitudes, and durations (by playing only a fragment of a sample). Sound models using a combination of sound samples and signal modeling techniques can combine advantages of both (e.g. modeling sound by filtering and mixing sound samples).

2. Auditory Scene Structure

Auditory scene elements represented by sound models and parametrized by events are assembled in what we refer to as an auditory scene. It is in the auditory scene where the scene elements are provided with a spatial location, a radiation pattern, and their acoustic environment. The auditory scene is naturally represented in the same scene graph than the visual scene - visual and auditory scene elements form together what we call the audio-visual scene. From the point of view of representation in the scene graph the only difference between auditory and visual elements are the node types used. Auditory scenes are composed of Avocado nodes (c.f. [2.3.2]) representing the sound model, the sound radiation, and the sound environment. The sound model node defines the sound synthesis process used to create the source signal. The sound model parameters are accessible as fields (c.f. [2.3.1]) of the sound model node, which will be connected to event generating nodes used to control the synthesis process. The sound radiation node defines the spatial position of the sound source and its radiation pattern, which may be omni-directional or directed in a frequency dependent manner. The acoustic environment in which the source signal will be propagated is defined by the sound environment node which defines the acoustic signature of the space the sound source is located in.

How do these nodes compare to nodes defining visual scene elements? The radiation and environment nodes define how the sound source will be rendered and are thus comparable to visual property nodes (e.g. transformation and appearance node; actually, the radiation node is a transformation node). And the sound model node is comparable to the visual shape nodes.

3.5.4 Example System: CyberStage

So far only the conceptual framework of the auditory scene synthesis process has been discussed. In the next paragraphs we will report on a concrete implementation of the process in the CyberStage II audio-visual display system.

1. Sound Projection

While the first generation CyberStage system (1996) used only 4 loudspeakers providing 2d localization, the second generation (CyberStage II, 1997) uses 8 loudspeakers in the classical cube configuration for full 3d localization. We use Genelec 1030A active near-field studio monitors which we chose for their sound quality, small size, integrated amplifiers and overload protection (an important feature when working and especially when experimenting with direct sound synthesis). The upper 4 loudspeakers are mounted directly above the 4 corners of the cubical projection screen frame (3 x 3 x 3 m). The 2 lower speakers in the back are mounted on the floor at the lower corners of the projection frame. A special arrangement had to be made for the 2 lower speakers in the front as they have to be positioned behind the projection screens in order not to occlude the visual projection. The deformation of the frequency response caused by the screen has been measured and filters are used to compensate this effect. The acoustics of the room behind the front projection screen has been adapted to avoid reflections and prevent sound paths other than through the front projection screen. All 8 loudspeakers are oriented towards a point in the center of the projections system at average ear level.

Additionally to the 8 speakers there are 4 low frequency vibration emitters installed in the floor of the CyberStage. These are used to generate vibrations up to around 100 Hz which can be perceived directly through the body via feet and legs. Unlocalizable high-amplitude low-frequency components are presented that way.

2. Sound Server

The 8 loudspeakers and the 4 vibration emitters are fed by a sound server designed in the context of a GMD research project on the integrated simulation of image and sound (ISIS). In order to meet the requirements described above, the sound server architecture has to be as open as possible. Therefore we decided to build the sound server entirely in software and minimize the amount of external hardware required. The sound server runs on an SGI Octane machine using an Alesis ADAT recorder connected via a fiber optic link as 8-channel D/A converter. The vibration emitters are fed by the analog stereo output of the Octane which is passed though an external low-pass filter and a dynamic limiter before being sent to power amplifiers.

The sound server is based on IRCAM's Max/FTS real-time sound processing system originally built for computer music applications [Lindemann 90, Dechelle 95]. FTS is an extensible signal processing kernel providing all necessary low-level modules to build sophisticated sound synthesis and processing applications. Max is a graphical programming environment [Puckette 95] used to interactively build FTS programs. Max allows to control and monitor the state of a signal processing program running in FTS. The spatialization algorithms used in the sound server are based on IRCAM's Spatialisateur toolkit [Jot 95] developed in Max/FTS. The software built on top of these components consists of parts realized in Max (synthesis control, resource management, message parsing) and FTS extensions written in C (efficient spatialization modules, sound sample manager, custom synthesis algorithms, network communication). The sound server is not a closed application but an open toolkit adapted to a large class of applications. The application designer chooses among many templates provided by the server to solve standard problems.

3. Controlling the Sound Server from Avocado

The information represented in the auditory scene by a graph of Avocado nodes has to be passed to the sound server to invoke the corresponding sound synthesis and rendering processes. The Avocado sound nodes use the Avocado sound service abstraction (c.f. [x]) to communicate with the sound server. The sound service defines a message passing protocol for this communication which is based on the Internet User Datagram Protocol (UDP) to insure low-latency and low-jitter control as it is needed to meet the mentioned synchronization requirements of an integrated simulation of image and sound. Since UDP is an unreliable protocol we use a dedicated Ethernet link and a simple message counting scheme to prevent messages from being lost. Whenever a state change in the scene graph concerns an aspect of the auditory scene (i.e. when some kind of sound-related event occurs), the sound nodes notify the sound server by sending the corresponding messages. We distinguish two kinds of messages: messages which allocate or free synthesis and rendering resources and messages which update active synthesis or rendering processes.

4. Summary: Features of the Sound Server

The sound server defines a set of general purpose sound synthesis and rendering modules which are selected by the application designer through parametrization of the Avocado sound nodes. As sound sources the sound server currently supports real-time sound input (e.g. from a microphone or CD player), sound sample playback from memory, sound file playback from disk, and direct sound synthesis based on sound models as described above. Real-time sound input, sound sample and sound file playback works for 1 to 8 channel input, samples, or files. For sound rendering (i.e. spatialization) a collection of modules are available ranging from very fast and to very sophisticated solutions among which the application designer may choose. For instance, static simple spatialization which ignores the movements of the source and the observer (which is reasonable for short percussive sound) is very fast and thus many sound sources can be rendered at a time. On the contrary, fully fetched spatialization including directivity effects, Doppler shift, propagation delay, fine-grain early reflections simulation, and high-quality diffuse reverberation [Jot 95] allows only a couple of sound sources to be rendered on the same hardware. And in the case of ambient sound sources, (e.g. background music) no spatialization is needed at all. This scalability allows the application designer to optimize and balance the use of computational resources.

Since the base system used to implement the sound server (FTS) doesn't allow dynamic allocation of modules for efficiency reasons, static banks of modules are allocated at startup time and individual modules (therefore also called voices) are switched on and off on demand. This implies that the application designer has to foresee the maximum number of each module type ever needed at the same time in an application. This may be tedious, but the application designers are rewarded by a maximum use of the available processing power.

5. Example Application: Sound Spheres

Sound spheres is a sound installation conceived as a part of the virtual environment caveland (c.f. [x]). The installation explores the basic features of the CyberStage audio-visual display system with the aim of shaping some of the yet unstructured vocabulary of musical expression and experience in virtual space. The concentration on a few fundamental aspects of integrated audio-visual simulation was a conscious decision in the design process and led to the abstract and minimalist character of the installation.

- Description

In essence, sound spheres is about localization of moving sound and light sources. The audience is immersed in a space of weightlessness enclosed by a large sphere. This space is populated by small rotating spheres slowly moving along circular paths. The audience activates these spheres by inflating them with a virtual pump. The more a sphere is inflated, the longer it will keep on emitting percussive sounds and light flashes in simple rhythmical patterns. But pointing the virtual pump at a sphere for too long a time will lead to its explosion accompanied by a violent detonation flash and noise. While freely floating in between the flashing and sounding spheres, the audience experiences an ever changing rhythmical tissue of spatialized sound and light. The perceptual plausibility and coherence of this experience is achieved by a careful adjustment of the dynamic behavior of the light flashes, the light model parameters, the sound material, and the room acoustics attributes. Sound spheres also serves as an example for the high degree of immersion achievable by perfect synchronization of image and sound rendering.

- Sound Model

The sound model used in Sound Spheres is a good example for the sound modeling capabilities of the sound server and shall therefore be described in some detail here. The rhythmical patterns audible in Sound Spheres are formed by streams of sounds of a certain timbre presented in regular repetition. The complexity of the resulting pattern is achieved by slightly different repetition rates. The resulting temporal phasing effect typical for minimal music can only be obtained through a careful choice of the sound material. The sound material has to meet three requirements. First, each sound should be easily localizable in space. Therefore we decided to use percussive sounds. Second, each sound should be easily identifiable by its timbre in order to form a clear perceptual stream when presented in repetition. The identification should work even if several streams were audible at a time. Third, since the rhythmical structure is based on repetition, each time a sound is presented, it should sound slighly different, every time displaying another facet of its class. This can be obtained by micro-variations in the spectrum which are typical for all percussive sounds.

It is clear that these requirements are hard to be met with sound sampling which cannot provide such richness of variation and strength of identification at the same time. Model-based direct sound synthesis was chosen instead. A bank of 10 resonating 2nd-order filter is used in a subtractive synthesis setup. The filter parameters (center frequency, bandwidth, amplitude for each filter describe one partial in the spectrum) are generated by a spectral model describing are large but perceptually clearly characterized class of wood-like sounds. The spectra are described by ranges of possible variation for each of the 30 parameters. Within these ranges random decisions are taken to produce an instance of the sound class. The filter bank is excited by short noise bursts. Due to the random nature of noise, the burst vary slightly from excitation to excitation and thus excite the filters differently every time. The timbre variations achieved this way carry a high degree of perceptual plausibility because the model mimics a typical case of excitation/resonance based sound generation we know very well from our everyday world (all percussive sounds are generated that way).

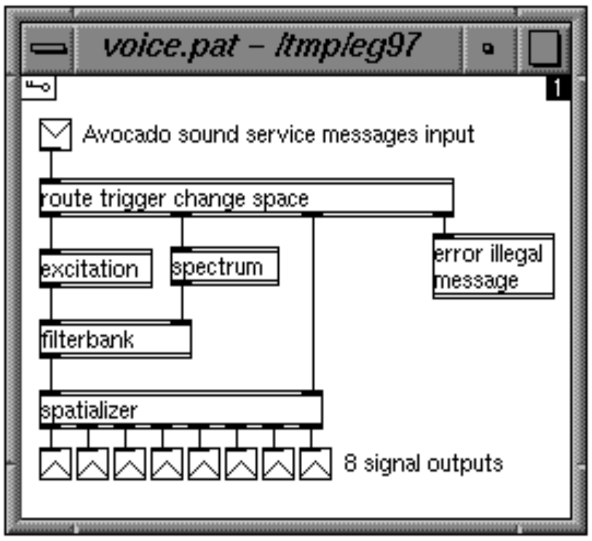


Fig. 23 shows one voice of the described sound model as a visual Max program. The voice receives messages from the Avocado sound service. A parser directs the messages to the corresponding objects. The excitation object can receive a trigger message to generate one noise burst. Whenever the spectrum object receives a change message, it will generate a new spectrum. The new spectrum is passed to the right input of filter-bank object which receives at its left input the signal from the excitation object. The output of the filter-bank is sent to the left input of spatialzer object which can receive space messages from Avocado on its right input. These space messages indicate where the sound source is located with respect to the observer and his or her position with respect to the loudspeakers. The output of the spatializer object is sent to the 8 loudspeakers.

6. Summary: Audio features of the CyberStage

The auditory display of the CyberStage can be characterized by the specific combination of the following key features:

- An open sound field is created by 8 loudspeakers in cube configuration.
- Sound projection is complemented by vibration emitters built into the floor to produce low-frequency vibrations.
- Model-based real-time sound synthesis is used to overcome sample-based approaches (complements sample playback).

- Life-audio input and sound file playback from disk are possible.
- Scalability of sound rendering (from a few high-quality up to 50 low-quality concurrent voices).
- Entirely software-based sound server (no special DSP or "off-the-shelf" MIDI sound processing hardware needed) using 32-bit floating-point digital signal processing for high audio quality.

3.6 Physical Simulation

3.6.1 Problem Statement

Virtual reality can make a quantum progress towards a higher level realism, when the virtual objects are able to react and change in a similar way as their counterparts in the real world. Indeed, adding temporal and reactive behavior to the objects in the virtual world is one of the most important issues of VR [Baldler 91] and the key feature of VRML 2.0. It is hardly possible to rely on the animator's intuitive specification and intervention in achieving this goal, so that computer programmed mathematical models are indispensable. As classical physics, i.e. Newtonian physics, is the well established mathematical theory about behavior of physical objects, it becomes naturally the theoretical basis for this domain, therefore it is commonly referred to as physically based modelling. In fact, robotics represents another important theoretical basis of physical simulation, since their problems have much in common.

In principle, all the attributes of a physical object, e.g. position, orientation, shape, colour, etc. may change. However, at the actual state of the art, physically based modelling deals mainly with motion and deformation of physical objects [Dai 97]. In this section, we will focus on the problem of motion control of solid bodies in VR. In this case, we may view our task as

- building movable or animatable objects representation

An animatable object representation is a visual data base which has an appropriate structure to allow on-line specification of motion variables. A data representation of a human skeleton, for example, needs to be sectioned into parts, and the spatial parameters, i.e. position and orientation of each part are represented by on-line modifiable variables. In walk-through type virtual reality, the world model is most often not animatable.

- adding guidance model - building guided moving objects

A guided moving object is an animatable object, associated with a guidance model. This guidance model is typically a one-to-one mapping from animator input variables to the motion parameters of the animatable data structures. We qualify it guided because the model itself contains no information about how to move, and such information is provided completely by the animator.

- adding autonomy - building semi-autonomous and autonomous moving bodies

Autonomy of a virtual physical body here means the capacity that enables it to behave like the real counterpart without the animator's intervention. Adding autonomy implies the embedment of object dynamics.

It is possible that some objects are only guided and some others are only autonomous, however the actual trend is towards multi-modal motion control, that is an autonomous object should have different control modes, ranging from low level to high level autonomy.

3.6.2 Major Issues and Approaches

It turns out that models offered by classical physics are not sufficient for the task of the domain. Principle problems that remain to be treated include the following.

1. Interference Handling

Newtonian physics helps us much in finding mathematical models of physical behaviour of virtual objects. However it provides us in fact only with piecewise mathematical formulations. If a free physical body is in the air, we can model the external forces including gravity, air drag, and magnus force and use the second Newtonian law to formulate the ball motion. When it collides against other body, the external force becomes infinitely big and the second Newtonian law can no more directly be employed. We need instead the conversation of momentum principles. Therefore, we can only have segmental behavior models for its flight, rebound, and sliding. A higher level behavior model has to implemented to detect its qualitative state and control the above low level behaviors. This is the task of the important problem of collision detection and response or more generally interference handling [Baraff 95].

Indeed, the essential difficulty in interference handling is not the correct formulation of the collision detection models, but the improvement of the run time efficiency. The efforts to speed up the collision detection include the following:

- specialization

The most general formulation of the collision would be based on the free-form object assumption. It is obvious that more specific formulations for collisions between special types of forms such as particles, plan, sphere, box, cylinder, polygons, etc., should be much more efficient.

- spatial approximation

Replacing complex geometry by simpler ones, e.g. sphere, box, etc. is an important method to speed up collision detection, of course at the cost of the precision. On the other hand, we can also represent the virtual space with rough "points", e.g. cubic regions - the Octree-method [DAI 97] falls into this category.

- temporal coherence exploitation

Every physical body has inertia, so that the continuity of motion is one of the most basic assumptions. Therefore the past results of the collision detection may be used to reduce the computation time for that of the present. In addition, a more precise dynamic model of objects can be also used to predict the next collision, which is important to avoid the detection error when objects are moving too fast.

- optimal scheduling

When a large number of physical bodies are involved, the problem of choosing the best order in which pairwise detections are processed becomes very important. This scheduling may be based on the possibility of collision and the computation complexity of each pair.

2. Kinematics and Dynamics of Multibody

The classical physics does not provide us with a solution to the problem of modelling physical bodies with complex structures. This becomes a special problem commonly referred to as multi-body modelling. The most interesting structures are body chain and body tree. Many concepts and methods introduced by robotics, such as those described in the following, provide the basis for physical simulation of multibodies.

- multibody kinematics

Kinematics of a multibody is the relationship between the motion variables, i.e. position, velocities and accelerations of its body parts. Forward kinematics of a body chain is the mapping form the joint motion variables to those of its terminal part. Inverse kinematics is the inverse mapping of the forward kinematics.The major difficulty of inverse kinematics resides in redundancy, real-time computational speed, and joint limits. The solutions to the inverse kinematics fall mainly into two categories: algebraic (i.e. closed form) and iterative.

- multibody dynamics

The dynamics of a body (including multibody) is the relationship between its motion variables and torques or forces applied to it. Forward dynamics of a body chain is a mapping from the applied joint torques or forces to their motion variables. Inverse dynamics refers to the mapping from the motion specifications to the desired joint torques or forces. Implementation of dynamics is essentially the problem of solving a system of differential equations. Methods used here can be also grossly divided as closed form, iterative and recursive.

Although methods in robotics are very helpful, they are not fully suitable for the simulation of bodies of virtual animals and humans, as those have more complex structure than most of the existing robots. Therefore new methods have to be studied. The closed form solution is more efficient than the iterative ones, and has no problem of convergence, while the iterative ones have the advantage to be more general.

Normally, the physical simulation of multibodies needs only the forward kinematic and dynamic models, and the inverse kinematics and dynamics are used for goal oriented object like robots and living beings. However, they are employed for inanimate objects simulation in the approach called teleological modelling [Baldler 91].

3. Physical Activities of Virtual Living Beings

Simulating physical activities of living beings, especially those of humans is a very interesting and hard topic.

- action primitives

Motion of virtual living beings involves coordinated motions of a very big number of degrees of freedoms, so that we need some higher level abstractions, e.g. actions, instead of treating them directly in terms of numeric formulations. Much effort has been made by many researchers on locomotion and manipulation [Zel 90b, Badler 93, Magnenat-Thalmann 96]

- perception

Visual and other exterioceptive perception of humans are essential, because they can hardly rely on their previous knowledge about their environment and apply open loop control to succeed in their physical activities.

- reflexive and intelligent behavioral controls

The control mechanism needed for connecting the perception to actions is one of the most interesting research topics. It has a very strong flavour of AI and automatic controls.

Traditional approaches in AI tends to represent an intelligent entity such as a human as a centralized system consisting of perception, reasoning and action modules, while actually the distributed intelligence approaches dominate either the domain of AI and the simulation of virtual creatures. These are known as emergent behaviors, autonomous agents, sensor actuator network etc. They are successfully applied in the examples such as the artificial fishes [Terz 94], Jack [Badler 93]. A generic programming method is also applied to modelling of structural behaviour, i.e. evolution [Sims 94].

3.6.3 AutoMove - An Autonomous Movement Simulation System

AutoMove is an experimental system which aims at integrating the major aspects of physical simulation into a unified framework based on system theoretical concept.

1. A System Theory based Unified Approach

As the complexity of the virtual world increases, a general framework becomes important. This fact for example was also stressed by David Zelter and Tom Calvert in [Badler 91]. Our approach is based on general system concepts [Yang 89, Mesarovic 89, Lin 93], so that we can represent the physical world and our physical simulation system in more formal and systemic terms.

On the one hand, such general framework may be useful in putting actually rather ad hoc and piecemeal methods into a more coherent theoretical structure. On the other hand, such formulations are especially adequate for the Object Oriented programming paradigm, and may further provide high level intuitive programming interfaces for physical simulation.

Additionally, we also try to combine the results from robotic visual servoing [Dai 90, 92, 93a, 93b] techniques and qualitative modelling and reasoning [Patrick 85] into our general framework.

2. Implementation

AutoMove is first directly implemented on the top of the C++ API of Performer. Its actual integration with the AVOCADO system and the CyberStage display environment is achieved through plug-in technology, and tighter integration is ongoing.

Unlike conventional animation where the data update more frequently than the frame rate, which is a waste of computing power, dynamic behaviour simulation gets more accurate as its updating frequency increases. Therefore, in our system, physical behaviour simulation is executed in a process running concurrently with the visualization process, and its update frequency is adjusted based on a compromise between its precision and time consumption balanced with that of the visualization process.

Up to now, the following specific models and scenarios have been implemented.

- An Autonomous Moving Ball

A physically realistic ball in the virtual world is a simple yet basic example of physical bodies. The ball has the general structure of a general behavioral model, in which the sensor's task is to detect collision and a high level logic controls its lower level behavoirs. i.e. flight, rebound and sliding. In order that the ball behaves physically, its static environmental objects need to be all physical. In the caveland scenario, the visitor can pick-up this ball and throw it. It can then fly, bounce or slide in the environment accompanied by a coordinated sound effect produced by the sound server in AVACADO. It adds some physically realistic interaction between the visitor and the caveland world.

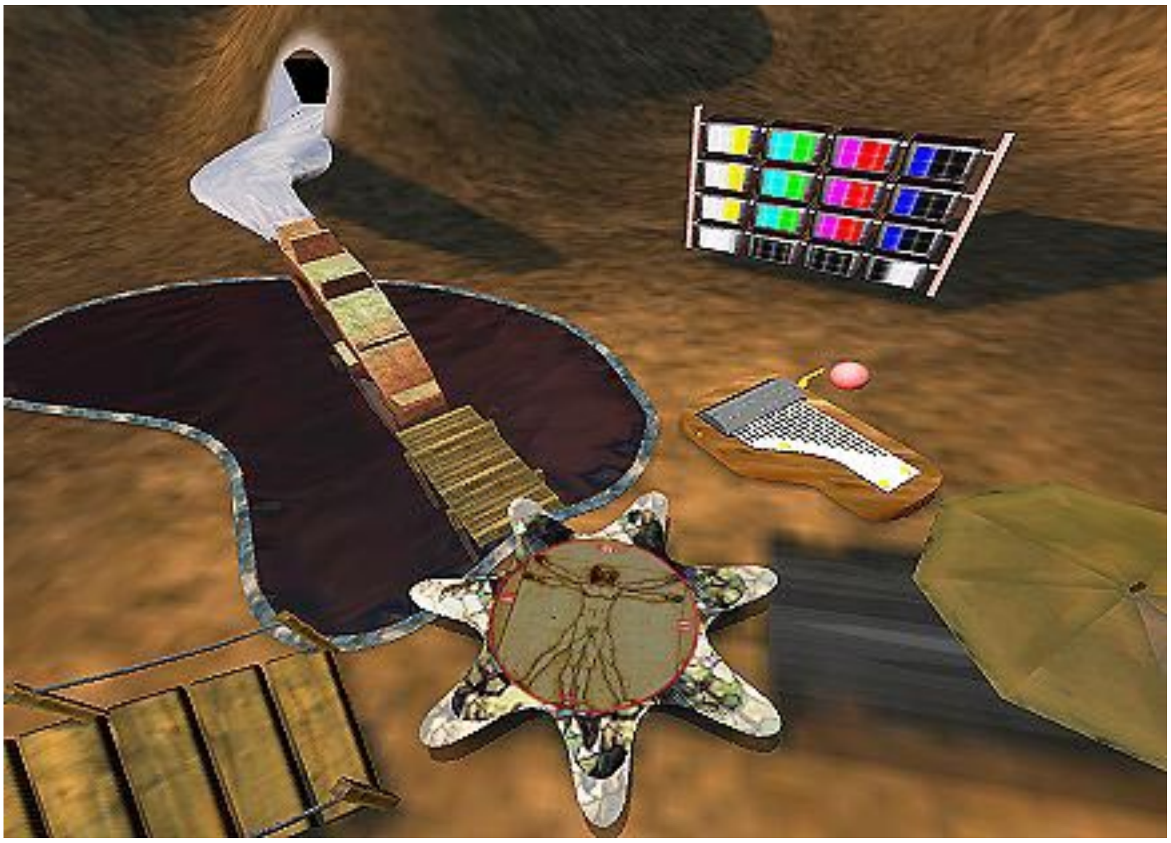


Fig. 24: An interactive autonomous ball in caveland

- A kinematics and dynamics based human body

This human body model can be guided through three modes: forward kinematics, inverse kinematics, and dynamically constrained inverse kinematics. Figure 2 shows a snapshot where the right arm is guided under dynamic constraint to follow the moving target, which is controlled by the user.

- A virtual ping-pong game

The virtual ping-pong game integrates all the important ingredients discussed above, so that it provides a good testbed of our unified approach. Some theoretical study and a simple implementation based on Open Inventor has been done previously [Dai 96]. Here the ping-pong scenario under the system AutoMove, consists of a guided player which is the participant's avatar, an autonomous virtual player, a virtual ping-pong ball, and the game environment. The virtual player perceives the ball motion, predicts its trajectory, then invokes some appropriate playing actions such as push, lob and side stepping etc. The trajectory predictor and decision maker of the player involve some qualitative modelling and reasoning about the ball, the environment and his own body. The playing actions are themselves sensor-controller-actuator loops at lower levels.

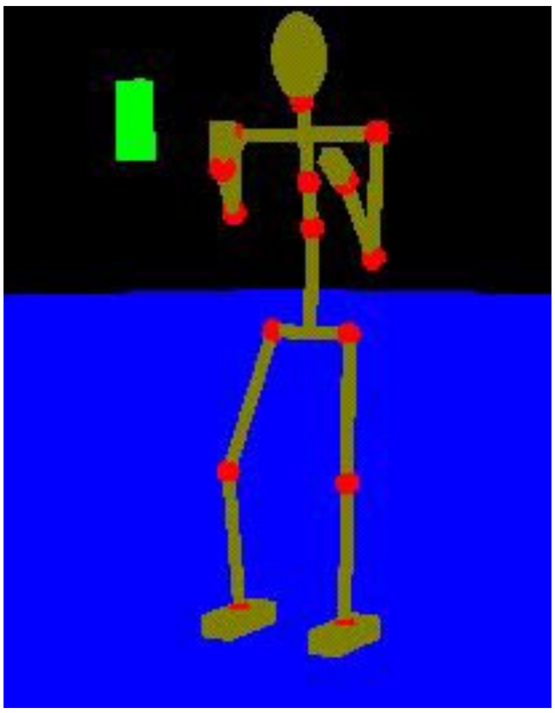


Fig. 25: A kinematics and dynamics based human body

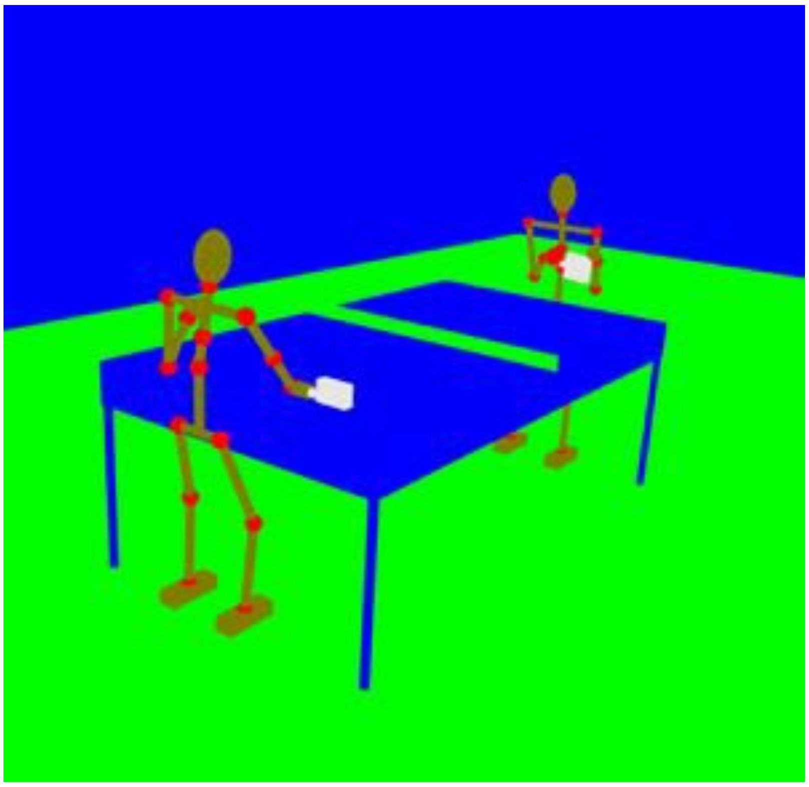


Fig. 26: A virtual ping-pong game

4 Caveland on Cyberstage

Many people wonder about the process of building a complex application for a virtual space like the CyberStage. The following describes the making of caveland that was first shown in a CyberStage at Cebit 1997 in Hannover/Germany.

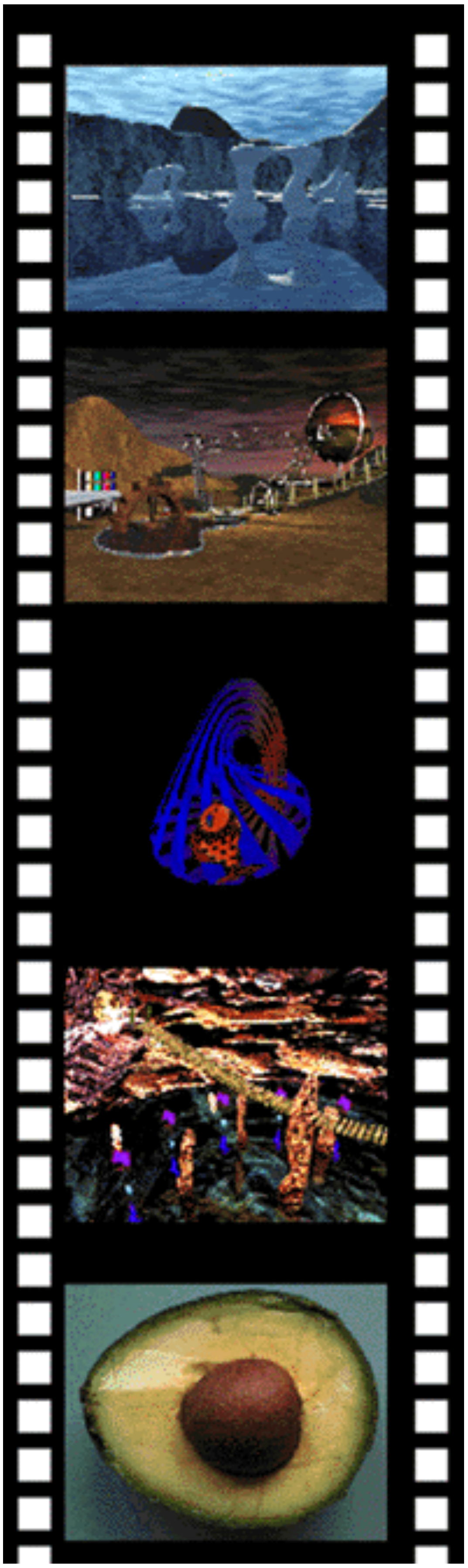
4.1 Team and Tasks

People involved can be ranged in the following areas:

- project coordination (5 people)
- software supervision (1 people)
- art supervision (1 people)
- software development (9 people)
- modeling (7 people)
- custom effects (5 people)
- music and sound rendering (5 people)
- system installation and support (2 people)
- special support (11 people)

4.1.1 project coordination

The most important job is to coordinate the whole project. It starts with the planning and developing of a storyboard while considering and evaluating the existing resources including software, hardware, manpower, time, money. It has to be decided whether additional resources need to be provided. The second step is to divide the task into subtasks and to assign them to the people involved in the project. All not-yet existing resources like software tools or additional specialists have to be ordered or hired. After the project has started the coordinators are responsible for controlling the schedule, the integration process and the managing of alternative solutions if unresolvable problems occur.



1. software supervision

One person is supervising the whole software development process. His job is to

- decide which extensions need to be programmed
- estimate efforts and time ("nice effect but takes years to develop")
- supervise integration process ("do the pieces work together?")
- estimate the complexity and resulting performance of the whole application ("cool but too slow?")
- give guidance for software developer ("help me guru!")
- evaluate modules and suggest improvements ("still too time consuming?")
- communicate with modelers due to special requirements for certain special effects

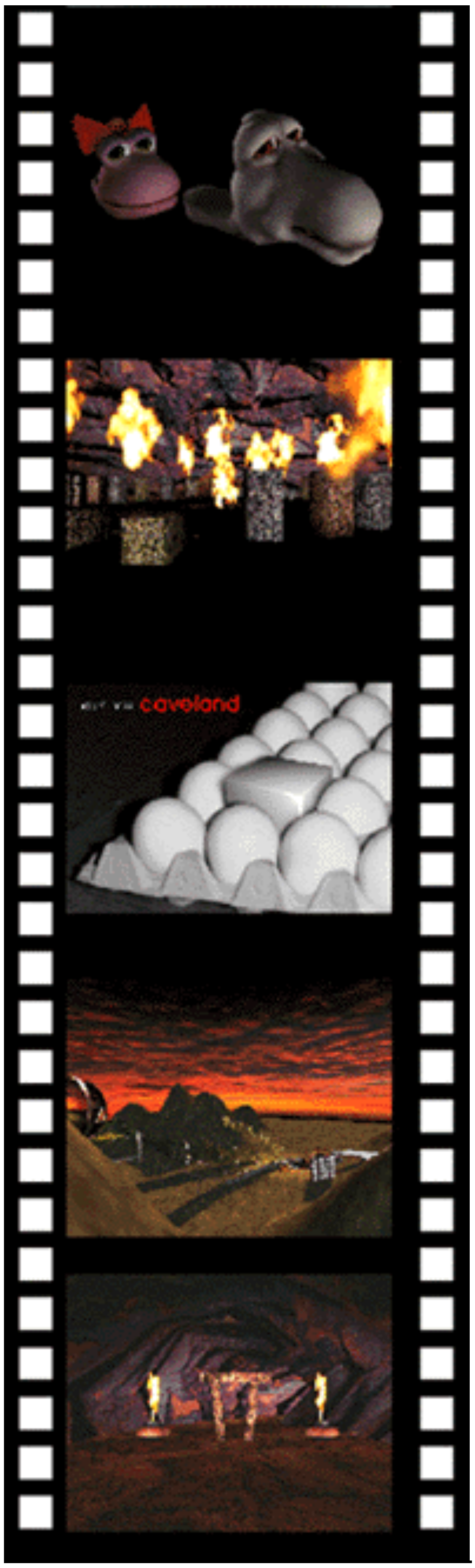
2. art supervision

Another person is supervising the whole modeling process. His job is to

- estimate efforts and time ("nice idea but takes years to model")
- supervise integration process ("do the pieces fit together?")
- give guidance for modelers ("how do I create shadows?")
- evaluate the different models and suggest improvements ("doesn't look good in the CyberStage environment")

3. software development

A couple of people are necessary to extend the current software system. Their job is to



- develop new modules including sound rendering and special effect modules
- integrate the modules in the software system
- test, evaluate and improve the modules especially considering performance issues
- communicate directly with modelers while working on a specific effect

4. modeling

Several modelers need to create the geometric representation of the virtual world. Their job is to

- create models while considering the special requirements (limited number of polygons, textures, ...)
- cooperate with software developers and adjust models due to the needs for special effects
- test and evaluate the models in the virtual environment due to the different light conditions etc.

5. custom effects

Since we wanted to include virtual actors guiding a user through certain parts of the virtual world we started a cooperation with a small company specialized in real-time virtual actors. Their job includes

- modeling of the virtual actors including geometric representation considering behaviour and gestures
- synchronization of sound and model
- cooperation with software developers to create a interface to the Avocado system

6. music and sound rendering

In order to provide music and real-time sound effects a couple of people has to take care of various issues including

- edit, process, and synthesize sound material
- synchronize image and sound
- test and evaluate the spatial sound experience
- compose music and record samples

7. system installation and support

At least two people are necessary to take care of the system. Their job is to

- administer the system while developing the project
- reconfigure the system according to the specific needs (sirius video,...)

- supervise the moving of all the hardware to the demo location
- reinstall and test everything at the event location

8. <u>special support</u>

You always need a couple of people assisting you. These people

- prepare video material used for special effects
- just make things possible by paving the way
- support you with ideas or give software tips
- represent models scanned in for special effects
- provide samples of their speech

9. <u>scheduling</u>

We started planning and developing the caveland project at the beginning of January 1997. Cebit took place in the middle of March. The internal deadline was the 28th of February, although all the modeling had to be finished by February 21th in order to plug in all the effects. The assembling of all parts and final adjustments of the software took one week with three people working 25 hours a day at Cebit location.

4.2 Software Tools

We used the following software packages while creating caveland:

4.2.1 CyberStage rendering

- Avocado 1.5 (in house) based on

    o SGI IRIS Performer 2.1
    o Scheme

1. <u>geometrical modeling</u>

- Alias|Wavefront Advanced Visualizer
- MultiGen II
- SoftImage
- In House Converter
- SGI Converter
- SGI Inventor Tools

2. <u>image pre and post processing</u>

- PhotoShop
- SoftImage Eddie

- SoftImage StudioPaint
- SGI Tools

3. sound synthesis and sound processing

- Digidesign SoundDesigner II
- Digidesign ProTools III
- Emagic LOGIC Audio 2.6
- Emagic SoundDiver
- IRCAM AudioSculpt

4. sound rendering

- IRCAM Max/FTS
- IRCAM Spatialisateur
- CCRMA snd

4.3 Special Effects

A lot of special effects and unique features of the Avocado system contribute to the life and beauty of caveland:

- vertex and norms animations make water swashing and glinting and let suspension bridges swinging
- texture and material animations create blazing flames
- moving animations make a lift working, a pendulum swinging, wheels turning
- interpolated paths a user may follow to explore the world
- live video input (video textures)
- reflection maps and reflection faces makes it look realistic
- quicktime movies imported as texture generate a lava stream
- localized sound sources enhance the visual effects
- level of detail effect switches
- exploding sound objects
- virtual actors accompanying a user

4.4 @ Cebit location

4.4.1 graphics installation

- SGI Onyx 2 Pipes IR-Graphics, 4 RM6 RasterManager, 10 R10000 CPUs, 1 GB
  RAM, 64 MB Texture Memory, 2 Sirius video boards
- CyberStage projection system, wooden projection room 3m^3

- Polhemus tracking system (head, two input devices)
- 16 CrystalEyes shutter glasses
- video cameras used for live video input

1. sound installation

- SGI Indy R5000, 160 MB RAM
- 4 channel surround sound system
- acoustic floor

2. schedule

The installing took approximately one week. Assembling of all parts and final adjustments of the software required three people working 25 hours a day. Almost all modules have been included in the final version of caveland. In parallel the installing of the projection system was completed.

3. presentation

For the presentation we needed to have counter personal taking care of the mass of people and arranging the schedule. Two presenters were showing caveland at a time. One responsible for handing out glasses and giving short instructions another showing the demo. Several people presented the demo alternately. Over 2700 visitors have seen the caveland. Fortunately nobody got injured or sick.

4. comments

A couple of problems arise when we started showing our project: The air condition to cool the projection room didn't work properly which caused high temperature within the CyberStage. In addition the air was not very fresh. Since our projection mirrors were mounted on the same floor as the rest of the CyberStage the mirrors began to waggle a little while the visitors entered the CyberStage. Due to the number of visitors a tight schedule had to be met. Showing the caveland was very exhausting for the presenters. Fortunately the caveland project itself ran very stable during the entire Cebit conference.

5 References

[Ast 93b] Astheimer, P., Dai, F., Dynamic Objects in Virtual Worlds: Integrating Simulations in a Virtual Reality Toolkit. European Simulation Symposium '93, Delft, October 1993.

[Bad90] "Animation from instructions", N. I. Badler, B. Webber, J. Kalita and J. Esakov. In Making Them Move: Mechanics, Control, and Animation of Articulated Figures, Morgan-Kaufmann, 1990, pp. 51-93.

[Badler 91] Baldler,N., Barsky bB. and Zeltzer, D. (eds): "Making Them Move". Morgan Kaufmann Publishers Inc. 1991.

[Bad 93] N. I. Badler, C. B. Phillips, and B. L. Webber. (1993) "`Simulating Humans: Computer Graphics, Animation, and Control." Oxford Univ. Press, 1993.

[Bar 91] Baraff, D. (1991), "Rigid Body Concepts", in Course notes, Siggraph 91

[Baraff 95] David Baraff, "Interactive Simulation of Solid Rigid Bodies", IEEE Computer Graphics and Applications, May 1995.

[Bech 96] P. Becheiraz, D. Thalmann, The Use of Nonverbal Communication Elements and Dynamic Interpersonal Relationship for Virtual Actors, Proc. Computer Animation 96, IEEE Computer Society Press, June 1996, pp.58-67.

[Bent 97] Bentley, R., Horstmann, T. and Trevor, J., The World Wide Web as enabling technology for CSCW: The case of BSCW, Computer Supported Cooperative Work: The Journal of Collaborative Computing. Special issue on CSCW and the Web, Vol. 6, Nos 2-3, Kluwer Academic Publishers, 1997

[Blum 95a] Blumberg, Bruce M.; Galyean, Tinsley A (1995). " Multi-Level of Autonomous Animated Characters for Real-Time Virtual Environments ", Siggraph '95 Proceedings

[Blum 96a] Blumberg, B., P. Todd and P. Maes(1996). No Bad Dogs: Ethological Lessons for Learning. In: From Animals To Animats, Proceedings of the Fourth International Conference on the Simulation of Adaptive Behavior, September 1996, MIT Press. Cambridge Ma.

[Blum 96b] Blumberg, Bruce (1996). Old Tricks, New Dogs: Ethology and Interactive Characters. PhD Dissertation. MIT Media Lab.

[Bly 93] Bly S.A., Harison S.R., Irwin S., MediaSpaces: Bringing People Together in Video, Audio, and Computing Environment, CACM, Vol 36, No. 1, pp. 29-47, January 1993

[Breit 96] Breiteneder C., Gibbs S. , Arapis C., TELEPORT- An Augmented Reality Teleconferencing Environment, Proc. 3rd Eurographics Workshop on Virtual Environments Coexistence & Collaboration, Monte Carlo, Monaco, February 1996

[Brown 88] Brown M.H., Perspectives on algorithm Animation, CHI'88: Human Factors in Computing Systems, Washington, D.C., 33-38, 1988

[Brown 92] Brown M.H., Zeus: A System for Algorithm Animation and Multi-view Editing, DEC Systems Research Center, Research Report, 75, February 28, 1992

[Bryson 92] Steve Bryson and Creon Levit, "The Virtual Windtunnel", IEEE Computer Graphics and Applications, July 1992

[Buxton 92] Buxton W., Telepresence: integrating shared task and person spaces. Proc. Graphics Interface `92, 123-129

[Cadoz 91] Cadoz, C., "Timbre et causalité", in: J.B. Barrière, ed. Le timbre, métaphore pour la composition, Christian Bourgois Éditeur / IRCAM, Paris, 1991.

[Cook 97] Cook, D., Cruz-Neira, C., Kohlmeyer, B., Lechner, U., Lewin, N., Nelson, L., Olsen, A., Pierson, S., Symanzik, J., (1997): "EXPLORING ASSOCIATIONS AMONG MID-ATLANTIC STREAM INDICATORS USING DYNAMIC MULTIVARIATE GRAPHICS AND GEOGRAPHIC MAPPING IN A HIGHLY IMMERSIVE VIRTUAL REALITY ENVIRONMENT;, EMAP meeting, Albany, NY, April 1997

[Dai 90] Ping Dai, "Etude et réalisation d'une commande de robot industriel polyarticulé par vision en mode dynamique", Doctoral thesis, 1990.5, UPL, Strasbourg, France.

[Dai 92] Ping Dai, "On Dynamic Visual Control of Industrial Robots", IEEE International Symposium on Industrial Electronics, 1992.5, Xian, China.

[Dai 93a] Fan Dai and Ping Dai, "Graphics Simulation of Dynamic Vision based Robotic Workcell", IFIP. International workshop on Graphics and Robotics, Dagstuhl, Germany, April 1993.

[Dai 93b] Ping Dai and Fan Dai, "Analysis and Modeling of Dynamic Vision Based Robotic System", in Proceedings IEEE TENCON'93-IEEE regionten Conference on Computer, Communication, Control and Power Engineering, International Academic Publishers, 1993.10, Beijing, China.

[Dai 96] Ping Dai, "Virtual Ping-Pong Game - Autonomous Objects in Virtual Reality", internal report FhG-IGD, July 1996.

[Dai 97] Fan Dai, Lebendige virtualle Welten: "Physikalisch-basierte Modelle in Computeranimation und virtualler Realität", Springer Verlag, Berlin Heidelberg 1997.

[Dechelle 95] Dechelle, F., DeCecco, M., "The IRCAM Real-Time Platform and Applications", Proceedings of the 1995 International Computer Music Conference, International Computer Music Association, San Francisco, 1995.

[Eckel 93] Eckel, G., "La maîtrise de la synthèse sonore", Les Cahiers de l'IRCAM, recherche et musique, No. 2, Paris, January 1993.

[Gaf 94] Gaffron, S. (1994)," SkillBuilder: A Motor Program Design Tool for Virtual Actors", M.S. Thesis, Februray 1994, Massachusetts Institute of Technology, Cambridge,MA.

[Gibs 66] Gibson, J.J. (1966), "The senses considered as perceptual systems, " Boston: Houghton Mifflin

[Gib 79] Gibson, J.J. (1979), "The Ecological Approach To Visual Perception", Houghton Mifflin Company Boston

[Haase 97] H. Haase, and F. Dai and J. Strassner and M. Göbel, "Immersive Investigation of Scientific Data", Scinentific Visualisation, IEEE Press, 1997

[John 95] Johnson, Wavesworld: A testbed for 3d, semi-autonomous animated characters. Mar 1995. MIT, Media Lab Ph.D. dissertation.

[Jot 95] Jot, J.-M., Warusfel, O., "A Real-Time Spatial Sound Processor for Music and Virtual Reality Applications", in: Proceedings of the1995 International Computer Music Conference. San Francisco: International Computer Music Association, 1995.

[Kog 94] Y. Koga, K. Kondo, J. Kuffner, and J.C. Latombe (1994). "Planning Motions with Intentions.", Proceedings of SIGGRAPH'94 (Orlando, Florida, July 24-29, In Computer Graphics Proceedings (July, 1994), pp.395-408.

[Krüger 94] Wolfgang Krüger and Bernd Fröhlich, "The Responsive Workbench", IEEE Computer Graphics and Applications, May 1994

[Lass 87] J. Lasseter. Principles of traditional animation applied to 3d computer animation. Computer Graphics, 21(4), July 1987.

[Lin 93] Y. Lin and Y. Ma, "System - A Unified Concept," Cybernetics and Systems, 24:375-406, 1993.

[Lindemann 90] Lindemann, E., Starkier, F. & Dechelle, F., "The IRCAM Musical Workstation: Hardware Overview and Signal Processing Features." In: S. Arnold and G. Hair, eds. Proceedings of the1990 International Computer Music Conference. San Francisco: International Computer Music Association, 1990.

[Lor 73] Lorenz,K., (1973), "Foundations of Ethology", Springer-Verlag, New York.

[Magnenat-Thalmann 96] N. Magnenat-Thaalmann and D. Thalmann, "The Simulation of Virtual Humans", Proc. GraphiCon'96, Moscow 1996.[Min 87] Minsky, M. The Society of Mind, Simon and Schuster, N.Y. 1987.

[Mesarovic 89] Mesarovic, M.D., and Y. Takahara., "Abstract Systems Theory," Springer Verlag, Berlin 1989.[Min 94] Minsky, Marvin L., Will Robots Inherit the Earth?, Scientific American, Oct, 1994

[Nag 96] Katashi Nagao and Jun Rekimoto, Agent Augmented Reality: A Software Agent Meets the Real World, To appear in Proceedings of the Second International Conference on Multi-Agent Systems, 1996

[Nag 94] Katashi Nagao and Akikazu Takeuchi, Speech Dialogue with Facial Displays: Multimodal Human-Computer Conversation, Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL-94)", 1994

[Norm 88] Norman, D.A. (1988), "The Psychology of Everyday Things, " New York: Basic Books.

[Patrick 85] Patrick, J. Hayes, "The second Naive Physics Manifesto," in: "Formal theories of the common sense world", Ablex: Norwood, 1985.

[Puckette 91] Puckette, M., "Combining event and signal processing in the Max graphical programming environment", Computer Music Journal, vol. 15, no. 3, MIT Press, Cambridge, MA, 1991.

[Ras 83] Rasmussen, J. Skills, Rules and Knowledge; Signals, Signs and Symbols and other Distinctions in Human Performance Models, IEEE Trans. on Systems, Man, and Cybernetics, vol. SMC-13, no. 3 (May/June 1983).

[Reed 96] Reed, Edward S., "Values and knowledge", Lawrence Erlbaum Associates , 1996.

[Rey 87] Reynolds, Craig W. (1987). "Flocks,Herds and Schools: A Distributed Behavioral Model, " Siggraph 87

[Scha 75] Schank, R.C. (1975), "Conceptual Information Processing. 1975, , " New York: American Elsevier Publishing Company

[Seeg 90] Seeger, L. Creating Unforgettable Characters, Henry Holt and Company, New York, 1990.

[Sims 87] Sims, K. (1987), "Locomotion of Jointed Figures over Complex Terrain, SMVS Thesis, , " MIT Media Lab, 1987.

[Sims 94] Sims, K. (1994), "Evolving Virtual Creatures", Computer Graphics (Siggraph '94) Annual Conference Proceedings, July 1994, pp. 15-22.

[Str 96] Strassner, J., (1996), "Autonomous Actors in Virtual Environments - physical object properties to constrain behaviour", Diploma Thesis, Computer Science, Technical University of Darmstadt, Germany.

[Sym 97] Symanzik, J., Cook, D., Kohlmeyer, B. D., Lechner, U., Cruz-Neira, C. (1997): "Dynamic Statistical Graphics in the C2 Virtual Environment", Second World Conference of the International Association for Statistical Computing, Pasadena, California, USA, February 19-22, 1997

[Terz 94] Terzopoulos, D. et al. (1994). "Artificial Fishes with Autonomous Locomotion, Perception, Behavior and Learning, in a Physical World, " Proceedings of the Artificial Life IV Workshop, Pattie Maes and Rod Brooks (ed.), MIT Press 1994.

[Tin 50] Tinbergen, N. (1950), "The Study of Instinct", Clarendon Press, Oxford

[Toat 91] Toates F. & Jensen, P. (1991). "Ethological and Psychological Models of Motivation: Towards a Synthesis", In: From Animals to Animats, Proceedings of the First International Conference on the Simulation of Adaptive Behavior, Edited by Meyer,J. and Wilson, S.W., MIT Press 1991

[Tos 93] Tosa,N. (1993), "Neurobaby, "SIGGRAPH-93 Visual Proceedings, Tomorrow's Realities, ACM SIGGRAPH 1993, pp. 212-213, 1993

[Yang 89] Yang, Z., "New Model of General Systems Theory," Cybernetic. Syst. 20: 67-76, 1989.

[Zel 90a] McKenna, M., S. Pieper and D. Zeltzer (1990). "Control of a Virtual Actor: The Roach", Proc. 1990 Symposium on Interactive 3D Graphics, Snowbird UT, March 25-28, 1990, pp. 165-174.

[Zel 90b] McKenna, M. and D. Zeltzer (1990). "Dynamic Simulation of Autonomous Legged Locomotion", Proc. ACM SIGGRAPH 90, Dallas TX, pp. 29-38.

[Zel 91a] Zeltzer, D. (1991), "Task Level Graphical Simulation: Abstraction, Representation and Control", in Making Them Move: Mechanics, Control and Animation of Articulated Figures, N. Badler, B. Barsky and D. Zeltzer, eds., San Mateo CA, Morgan Kaufmann, pp. 3-33.

[Zel 91b] Zeltzer, D. and M. Johnson (1991). "Motor Planning: Specifying and Controlling the Behavior of Autonomous Animated Agents", Journal of Visualization and Computer Animation, 2(2), April-June 1991, pp. 74-80.

[Zel 94] Zeltzer, D. and M. Johnson (1994). "Virtual Actors and Virtual Environments: Defining, Modeling and Reasoning about Motor Skills", in Interacting with Virtual Environments, L. MacDonald and J. Vince, ed., Chichester, England, John Wiley & Sons, pp. 229-255.

[Zel 96] Zeltzer, D. and S. Gaffron (1995). "Task Level Interaction with Virtual Environments and Virtual Actors", International Journal of Human-Computer Interaction, 8(1), pp. 73-94.