KUNGL
TEKNISKA
HÖGSKOLAN

CID-178 ● ISSN 1403-0721 ● Department of Numerical Analysis and Computer Science ● KTH

Production and Management of Events in Electronic Arenas

eRENA ESPRIT Project 25379 Workpackage 4 Deliverable D4.5

John Bowers, Kai-Mikael Jää-Aro, Sten-Olof Hellström, Bernd Lintermann, Michael Hoch, Adam Drozd, Ian Taylor, Greg Whitfield

CID, CENTRE FOR USER ORIENTED IT DESIGN

John Bowers, Kai-Mikael Jää-Aro, Sten-Olof Hellström, Bernd Lintermann, Michael Hoch, Adam Drozd, Ian Taylor, Greg Whitfield

Production and Management of Events in Electronic Arenas

Report number: CID-178, eRENA ESPRIT Project 25379 Workpackage 4 Deliverable D4.5
ISSN number: ISSN 1403 - 0721 (print) 1403 - 073 X (Web/PDF)
Publication date: Aug 2000
E-mail of author: yngve@nada.kth.se

eRENA

Deliverable 4.5

Production and Management of Events in Electronic Arenas

ABSTRACT

This deliverable contains: a description of a system for mapping data from participants to enable its flexible interpretation in media-rich environments, thereby supporting the participation of audience members in the interactive shaping of content; demonstrations of a number of approaches for the control of virtual cameras in an electronic environment so that views of activity can be appropriately obtained; applications designed to support the directors/producers of events in electronic arenas in mixing/editing views from virtual cameras; technologies to provide production personnel with overviews of activity in electronic arenas to faciliate the timely deployment of resources for the support of events; demonstrations of novel sound mixing and spatialisation concepts for electronic arenas along with tools to enable the interactive composition of music as content; and a presentation of a novel tangible interface solution to realise a number of these technologies in an accessible way. Throughout we have been concerned to critically reflect and evaluate the production tools we have developed. In all cases this has involved their use by media professionals who are independent of the eRENA project, as well as, when possible and appropriate, members of the public. Each piece of work in this deliverable is accompanied by an evaluative appraisal based on the experiences of the users of the technologies in question. In addition, several of the production tools developed in this workpackage have been fed back to Workpackage 7 and employed in public demonstrators there..

Document ID	D4.5
Type	Report
Status	Final
Version	1.3
Date	30 Aug 2000
Author(s)	John Bowers, Kai-Mikael Jää-Aro, Sten-Olof Hellström (KTH), Bernd Lintermann, Michael Hoch (ZKM), Adam Drozd, Ian Taylor, Greg Whitfield (NOTT)
Task	4.5

CONTENT LIST

Deliverable D4.5
Production and Management of Events in Electronic Arenas

Preface

John Bowers
Royal Institute of Technology (KTH), Stockholm, Sweden

0.1. Document Overview

This document is the final deliverable arising from Workpackage 4 of the eRENA project of the i3 schema of the ESPRIT-IV research action of the European Communities. eRENA is concerned with the development of electronic arenas for culture, art, performance and entertainment in which the general citizen of the European Community might actively participate supported by advanced information technology. Within this general context, Workpackage 4 has been concerned over the last two years of research with how the production of such events might be supported. The concern in this workpackage is with the 'behind-the-scenes' activity which is necessary in staging an event and how, when appropriate, those activities might be supported technically. Founded on the practical experience in eRENA with staging events gained in Workpackage 7, we present here a number of technologies for the support of the production process.

Specifically, this deliverable contains: a description of a system for mapping data from participants to enable its flexible interpretation in media-rich environments, thereby supporting the participation of audience members in the interactive shaping of content; demonstrations of a number of approaches for the control of virtual cameras in an electronic environment so that views of activity can be appropriately obtained; applications designed to support the directors/producers of events in electronic arenas in mixing/editing views from virtual cameras; technologies to provide production personnel with overviews of activity in electronic arenas to faciliate the timely deployment of resources for the support of events; demonstrations of novel sound mixing and spatialisation concepts for electronic arenas along with tools to enable the interactive composition of music as content; and a presentation of a novel tangible interface solution to realise a number of these technologies in an accessible way. Throughout we have been concerned to critically reflect and evaluate the production tools we have developed. In all cases this has involved their use by media professionals who are independent of the eRENA project, as well as, when possible and appropriate, members of the public. Each piece of work in this deliverable is accompanied by an evaluative appraisal based on the experiences of the users of the technologies in question. In addition, several of the production tools developed in this workpackage have been fed back to Workpackage 7 and employed in public demonstrators there.

0.2. The Production Requirements of Electronic Arenas

Since Year 2 of the eRENA project we have been guided by an image of an electronic arena as *deploying mixed reality technologies to create environments for potentially large-scale real-time participation in media-rich cultural events*. The terms of this image of an electronic arena have given our research agenda in Workpackage 4 a considerable degree of specificity.

For example, we are not merely concerned with virtual reality (VR) technologies but with mixes of the physical and virtual, and of different media. The demand to enable a variety of different media technologies to interwork has motivated the construction of a general mapping toolkit to translate data from different devices and control its mapping and remapping. Consistent with the emphasis on mixed realities, we have introduced work from Workpackage 6 on physical, tangible interfaces to give us ways of presenting many of our production tools to their users.

The emphasis on large-scale participation has led several researchers in this workpackage to devote considerable effort to the design of technologies for visualising the activities of the mass of participants to an event, so that such information can support real-time production decisions.

Our concern for real-time events, and indeed live events with a large component of audience participation and improvisation, has shaped our work in novel ways. We find, for example, many of the high-profile computer animation techniques to be not workable when live events are to be captured in real-time. All of the technologies reported and evaluated in this deliverable are workable in real-time and shaped for following live, improvised action.

0.3. Evaluation and Reflection

Our work has been marked by a focused concern to reflect upon its nature and evaluate its advantages and problems throughout design. Where possible we have put our technologies before both members of the public and media professionals. Also where possible we have deployed our tools in the varied demonstrators in Workpackage 7 or conducted smaller scale demonstrations within this workpackage. In addition, some of the systems we describe have been maturing over the course of two or more years of development in eRENA. Together, these experiences give us a strong basis for critical appraisal of our efforts. While all of our technologies have worked well in the contexts in which they have been used, we are not short of ideas for their improvement. Indeed, many of these ideas come from independent users or trialists and are not just derived from our own experiences.

On a number of occasions we have been able to work on complementary solutions to problems. For example, some of our production tools have been implemented with contrasting interface techniques, e.g. conventional desktop versus tangible interfaces. This has enabled us to have a comparative appreciation of which solution works best and under what circumstances. We have also been able to work on production tools which are consistent with current media practices alongside ones which involve more radical solutions which might provoke a reshaping of production work techniques. Again, this has enabled us to have a broad understanding of when different solutions work, e.g. when modest and when radical solutions might be called for and found acceptable by users.

At the end of three years of work in eRENA, we find ourselves – we believe – able to offer a 'suite' of production tools, all of which have been trialed by independent users, and which together comprise a set of approaches usefully varying the degrees of risk and ambition they embody.

0.4. Structure of this Document

This document is structured as follows. After this Preface, there follow three chapters.

Chapter 1 describes the work conducted at the ZKM on a general mapping toolkit for handling and transforming control data in real-time performances. The initial motivation for this work can be found in the analysis of one of the Year 2 workshops presented in Deliverable D4.3/4.4 last year. It was observed at this workshop that a performance composed of multiple phases, involving multiple performers, each with their own interaction devices, each wishing to interact with computer graphical or sonic displays in different ways was very hard to seamlessly support. Rather the procession of events was marked by long pauses as new programs were launched offering specific ways to map performance data onto control data that were valid for just one segment of the event. Additionally, it was problematic for actors to rehearse with interactive systems when a reconfiguration of a computer graphical system may take several hours for a programmer to implement. To address these problems, a general toolkit has been developed which allows performance data to be transformed in a variety of ways so as to comprise appropriate control data to pass to graphical and sound generative systems. In this way, large-scale events can be much more seamlessly supported and rehearsal with and exploration of interactive systems prior to performance can take place on a much shorter 'feedback loop'. The toolkit developed at the ZKM, the MTK (for Mapping Toolkit), enables the integrated interworking of many graphical and sound control systems and provides a performance 'Swiss Army Knife' for addressing the callibration problems which exist whenever various specialised pieces of equipment have to combine in a production. The MTK has had a key role in a number of important public events. It co-ordinates the interactive panoramic graphics in the Skoda Pavilion at Wolfsburg in Germany - a presentation on permanent public exhibition. It has been worked with in a collaboration between the ZKM and the Frankfurt Ballet. In eRENA itself, it has enabled the integration of the MASSIVE VR system, a video analysis system to capture group interaction, and a sound diffusion set-up reported in workshop document Deliverable D7b.4. An innovative 'panoramic camera' that computes a panoramic view when certain key participants are proximal to each other was also developed in this workshop using MTK.

Chapter 2 describes the work conducted at Nottingham on production support for inhabited TV, particularly focussing on technologies developed for *Avatar Farm* (see Deliverable D7a.3). The chapter argues that in events which accent audience participation, and in particular where such participation involves a degree of improvisation or provokes such on the part of performers, it is ill-advised to hard code all object-behaviours in an interactive VR system in advance. The uncertainties involved in audience participation and improvisation militate against such an approach. The chapter prefers a hybrid solution involving human 'stage-hands' enabling and disabling behaviours in objects and capabilities in the avatars which participants use. This enabled a much more flexible approach in the face of unforeseeable contingencies in *Avatar*

Farm. It also enabled a narrative to be sustained when the content of it was at least in part co-developed by participants drawn from the general public. The chapter also discusses other approaches to event management which have been employed in the eRENA project, notably the systems developed for the event *Out Of This World* (Deliverable D7a.1). That earlier work taken with the current chapter demonstrate a variety of proven approaches for different inhabited TV formats (e.g. a gameshow and a narrative drama). Chapter 2 also discusses enhancements made to the virtual camera interfaces developed for *Out Of This World* as well as a director's interface. Following on from and extending the approach of KTH in Year 2 (see the description of the *Blink* event in Deliverable D4.3/4.4), *Avatar Farm* employed some software based vision mixing technologies to support real-time camera selection and view editing. Importantly, these technologies were developed with close contact between the development group and a TV director whose evaluations of them are reported here.

Chapter 3 presents work led by KTH but involving contributions from all the other partners with effort in this workpackage this year. This work takes as its starting point the argument that *finding the action* is one of the core problems in producing events in large-scale participatory settings, especially live events. When many participants are involved and the action can become distributed and is not necessarily governed by a strongly constraining script, it is easy for events of interest to be missed. Accordingly, effort has been devoted over two years in eRENA to developing tools which might enable production personnel to gain an overview of the action in an electronic arena and, through this, to support the deployment of resources such as virtual cameras and virtual microphones to pick up happenings of interest. Chapter 3 refers to this approach as *activity oriented resource deployment* for electronic arenas. A number of visualisation techniques are described (an attempt to sonify participant-activity was favourably reported in Deliverable D4.3/4.4) and some innovative ideas for virtual camera deployment are specified. Two different strategies for interfacing to these tools are examined. Alongside standard desktop manifestations, researchers at KTH and ZKM have collaborated on building a tangible interface (the Round Table, outlined in Deliverable D6.4) which enables visualisations to be projected onto a table top display and interacted with by means of manipulating small physical blocks. Year 3 has seen a deepening of this work, which was started in Year 2, to embrace more thorough evaluation and comparisons of desktop and tangible interfacing techniques, as well as strategies for integrating the Round Table as a production tool with the MASSIVE record/reply functionality employed in *Avatar Farm*. Importantly, researchers at both KTH and ZKM have also extended their research to address issues of sound control in electronic arenas. Chapter 3 describes some promising sound mixing and interactive compositional tools, which, again, have also been implemented with the Round Table physical interface. The chapter closes with a comparison between different orientations to production support in eRENA and concludes that the tools described in Chapter 3 valuably complement the work reported in Chapters 1 and 2.

0.5. Relationship of this Document to the eRENA Workplan

This document is the culmination of work in Workpackage 4. As such, it can be understood as the workpackage's 'exit deliverable'. It describes and evaluates the most mature work that has been conducted in eRENA on production support. As emphasised already, it gains much of its motivation from our experience in the demonstrator workpackages in the project. Those demonstrators have also been the destinations of many of the tools developed comprising their most rigorous testing grounds. We have been able to effectively cycle between the demonstrator workpackages (within Workpackage 7) and Workpackage 4 over two years and several of our technologies have been through two design cycles offering multiple opportunities for testing and improvement.

In the eRENA workplan, Workpackage 4 is composed of two work tasks. Task 4.3 concerns event design and management. Task 4.4 concerns audience participation and content production. The detailed description of these tasks in the project programme document involved partners in the following commitments:

1. Prototyping a suite of tools for the support of production and direction in electronic arenas, embodying notions of stage and production management, virtual camera deployment, and the high-level supervision of technical resources during an event.
2. Developing a general framework for transforming, mapping and choreographing user-interface data for performance purposes.
3. Supporting audience members and other participants in dynamically shaping and evolving the content of electronic arenas.
4. Identifying and enabling different forms of audience participation, including large-scale interaction.

Our work in eRENA has addressed all of these but, in the time since the programme for this workpackage was authored at the outset of the project, some matters have emerged as more essential for the support of events in electronic arenas than others. In addition, some matters have been more effectively dealt with or reported in other workpackages or in earlier deliverables. Specifically, this deliverable pays considerable attention to how we have addressed 1 and 2 above. These were also key themes to Deliverable 4.3/4.4, the earlier deliverable from this workpackage. These topics are, we believe, the most important and specific to the production support requirements of electronic arenas. Other aspects of them, which were alluded to in our initial programme of work (e.g. supporting virtual architecture and set design, and story boarding) have been, in our experience, best supported by conventional tools already available in terms of the existing state of the art. As a ‘behind-the-scenes’ pre-production activity, architectuiral and set-design have been most typically accomplished in eRENA using existing 3D modelling packages (though the interactive algorithmic architecture of *Blink* is an exception, see Deliverable D4.3/4.4). Furthermore, pre-production storyboarding has most typically been conducted using manual drawing tools – and these solutions have proven perfectly adequate in most of our work.

There are a number of ways in which audience members can be supported in dynamically shaping and evolving content (3 above). The MTK, especially in its most current form which includes integration with graphical rendering techniques, allows participants to freely experiment with preparing visual content for electronic arenas. The sound manipulation techniques described in Chapter 3 support the real-time composition and mixing of sound and music using readily accessible gestures. No special musical instrumental competence is required to work at the Round Table and yet quite complex algorithmic effects are possible. Chapter 2 presents another one of our preferred strategies for enabling public participants to help shape media content. This is through interaction with a helper or stage-hand who might, more or less, invisibly enable certain object-behaviours or avatar-competencies to come into existence. This hybrid human-automated approach is one we strongly favour as being uniquely suited to participatory environments with a high degree of uncertainty in them.

Finally, we must mention pointers to other deliverables where aspects of 3 and 4 above have been more appropriately dealt with than here. The earlier deliverable to this workpackage, D4.3/4.4, documents a number of techniques for interactively improvising the content of electronic arenas – approaches which we chose not to focus our integrated efforts of Year 3 on but which nevertheless remain available. D7a.1 and D6.3 documented some techniques for using video to support group and large-scale participation in events. These have been perfected in the bFinder and related applications reported and evaluated in D7b.4.

In summary, eRENA has addressed all the relevant commitments made in the specification of Workpackage 4. The current deliverable documents our most mature and focused work in developing production support tools. We hope that, from this preface and what follows, the reader can see the specific research challenges that have been raised by the demand to support electronic arenas and how we have met them.

Chapter One
A General Framework for Transforming, Mapping and Choreographing User Interface Data for Performance Purposes

Bernd Lintermann
ZKM, Karlsruhe, Germany

In the context of performances the generation of adequate control data for driving the graphics as well as the sound plays an important role. Since performances usually have a highly dynamic structure and are based on a certain choreography, the flexibility of data interpretation and mapping onto the graphics and sound parameters is crucial.

Since in a stage situation very different kinds of people are involved at a time, like dancers, actors, choreographer and technical people, it is desirable that the work with the technical equipment fits into the workflow of the participants as far as possible. The requirements on the software used are usability and flexibility to allow quick changes in existing setups and extensibility for the exploration of new ideas.

The Mapping Toolkit, in short MTK, has been designed and developed during the last two years to address these issues exploring a new approach. A first implementation has been described in the eRENA D 4.3/4.4[1].

This document gives first a motivation for the developed software, then summarizes briefly the system architecture, the design strategies and gives a short overview on the state of the first implementation at the time of D 4.3/4.4. It follows a description of the progression since then.

The new developments have been motivated by practical experience using the software in different situations: in an interactive installation created for the Skoda Pavillon in the VW Autostadt in Wolfsburg which has been installed in March 2000, in the development of a dance performance in cooperation with a dancer from the Frankfurt Ballet, and finally in the eRENA workshop “Mixed Reality Group Interaction”[2] held in July 2000 at the ZKM in Karlsruhe.

The document ends with a summary of the contribution to the eRENA project.

1.1. Motivation

The experience in the workshop "Real Gestures, Virtual Environments"[3] held in August 1998 at the ZKM in Karlsruhe has shown, that extending an existing graphics application for the various

[1] D 4.3/4.4 "Production Tools for Electronic Arenas: Event Management and Content Production", ed. Bowers, J et al., 1999

[2] D 7b.4 "Mixed Reality Group Interaction", ed. Lintermann, B et al, 2000

[3] D 4.3/4.4 "Production Tools for Electronic Arenas: Event Management and Content Production", ed. Bowers, J et al., 1999

needs of a performance involving different graphics environments and mappings leads to several difficulties. The application software Xfrog used in the workshop had to be extended at several locations in the code.

The Polhemus™ magnetic tracking device has been integrated as control value generator. Though the new implemented techniques are now part of the Xfrog software and can be useful for other pieces using the Polhemus™ tracker device, it blew up the software in terms of code complexity as well as in the number of offered user interface components. It slowed down the development of the performance pieces itself since the actors had to wait sometimes up to a day until the required techniques were programmed. In this way the software influenced the workflow on stage in a way unfamiliar to actors. It is desirable to compose mappings on the stage interactively and involve the actors in the process of testing.

MTK has been designed and implemented to address the issues of flexibility, usability and extensibility. It is supposed to be used in the context of performances involving computer generated real time graphics and sound. It maps raw hardware device data to high level application control data which controls graphics and sound applications. To account for stage and performance situations it is completely interactive to allow non programmers to create, test and adjust mappings directly on the stage supporting a feedback oriented workflow.

In case a performance requires specialized mappings, which cannot be created with the offered functionality, it is possible to write plug-ins, which can be reused in other performances. The plug-in architecture allows developers to focus on the algorithms rather than on user interface issues.

1.2. Overview on Previous Work

The following paragraph gives a brief overview of the system architecture designed and implemented in the first working period. It is described more extensively in D 4.3/4.4.

1.2.1 System Architecture

The system architecture is designed to support mappings in general where the term mapping is defined as the computation of control values depending on incoming interface data and the actual system state. The system architecture abstracts from concrete applications to allow derivations of the software used for different purposes. The system architecture is designed in layers implementing a basic generic functionality on top of which implementations are built which depend on specific hardware and customize specialized mappings.

The most general scheme of the system architecture distinguishes between the User Interface and Network Layer, which does the actual computations. Via the user interface the underlying application, the network, is manipulated. The User Interface Layer offers dialog elements like buttons, sliders, curves etc.. The User Interface Layer changes the application indirectly, using services of the Manipulation Layer. Since the Manipulation Layer keeps track of all the user's changes, it implements an infinite undo and redo capability. Distinguishing between user interface and application allows the automatic creation of a user interface for a given part of the application, thus freeing the programmer from the need to write code for the user interface for her algorithms. Figure 1.1 shows the diagram of the software architecture layers.

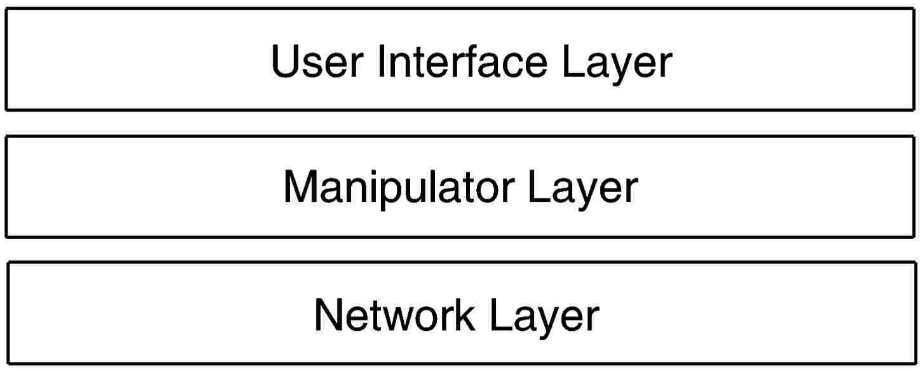


Figure 1.1: Software Architecture Layers.

1.2.2 Network

The application is built out of *Nodes*, basic computation units which communicate data via *Connections*. Nodes implement simple problem solutions such as the interpolation of different values, rescaling and transposing or mapping of data using expressions or splines.

Nodes do their computation involving parameters defined in the context of the application, the so called *Attributes*. These attributes are visible to the user in the graphical user interface. Attributes of different nodes can be linked through connections. A connection invokes the attribute value of the destination attribute to be overwritten by the attribute value of the source attribute, thus making the computation of the destination node dependent on the result of the source node's computation.

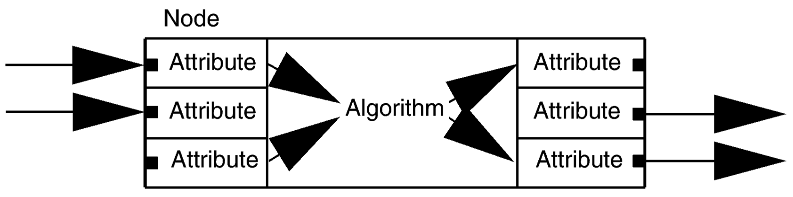


Figure 1.2: Diagram of a Node with Attributes.

The application is built as a *Network* of nodes, each node solving a dedicated part of the overall problem. The connectivity of nodes in a network determines the flow of data between the nodes.

1.2.2.1 Evaluation Strategy

The evaluation strategy chosen for the MTK system architecture is a mixture between a data flow model, such as Maya^{TM} is based on, and a control flow oriented models, such as MAX^{TM} is based on. While the dependency oriented evaluation is more transparent in terms of time behavior, the integrated control flow capabilities allow changing the system evaluation based on the system state and the incoming data.

By default, if a node requires the output of a source node, the system ensures, that the source node is evaluated before the destination node. Additionally, if a node has been evaluated, the system checks all nodes that receive data from this node and evaluates these in case they have not been already evaluated in this frame. In each frame a set of dedicated nodes are evaluated which causes a chain of evaluations in order of node dependencies. These dedicated nodes are so called *Device Nodes*, which are supposed to read data from an input or output device.

The chain of evaluation can be explicitly broken by marking a connection as not to be dependency checked. Additionally a node can explicitly force the evaluation of all nodes connected to an output attribute. Independent of this, if such a node already has been evaluated in the frame it is forced to be reevaluated. This allows on the one hand to selectively evaluate subnetworks dependent on the actual performance state or to evaluate a node with varying input data in the same frame.

In D 4.3/4.4 the integrated strategies were discussed more in detail.

1.2.3 Plug-in Architecture

Registering mechanisms allow programmers to extend the basic functionality of the system. Using dynamically linked code modules, a programmer can register new functionality in the different layers of the system.

Beside new parameter types, the attributes, new nodes and user interface dialogs can be introduced to the system. All registered elements behave like the basic system elements. Registered nodes appear automatically in the graphical user interface. There is a generic user interface builder implemented for editing the parameter of a registered node. Since a programmer can define new attributes she might wish to define also the user interface for editing that attribute. By registering custom code for editing attributes as well she can redefine the system default editors for the basic attribute set like floating point or string attribute editors.

Attributes and nodes are implemented in C++, custom nodes and attributes can be derived from existing ones while inheriting their functionality. By this, programmers can refine the functionality of existing nodes if they encounter a special case that the standard nodes are incapable of handling.

1.3. Recent Work

The first phase of development concentrated on a clean implementation of the system architecture described above. MTK has been used since then in an interactive installation for the Skoda Pavillon in the VW Autostadt in Wolfsburg and in dance performance experiments with a dancer from the Frankfurt Ballet. In the Skoda Pavillon MTK is the main software for controlling imagery and sound as well as for image creation itself. In the eRENA workshop “Mixed Reality Group Interaction” it was used as mapping software allowing groups of people to interact in a dome environment with imagery and sound created by the real time virtual reality software MASSIVE-3[4].

Based on the experience in these different situations MTK has been extended in four directions. Some features have been implemented to increase the general usability independent of the application. The basic available set of attributes and nodes has been extended. In all above mentioned situations special hardware devices were used to control image and sound. It turns out

[4] D 7b.4 "Mixed Reality Group Interaction"

that a generic device and database concept for accessing and storing device data could be developed which is suitable for all situations.

The performance and the workshop situations involving several people working on different topics required quick development of setups. To speed up the development of simple algorithms the script language **Lua** has been integrated. A so called *Lua Node* allows the definition of node attributes while the system is running and changing the algorithm on the fly.

For the interactive installation various 3D capabilities have been integrated.

1.3.1 General Usability

Some features increase the usability of the software for example by making the networks more intuitively readable, by supporting debugging or by speeding up the workflow in standard operations like connecting attributes.

Marking nodes gives a better visual overview of the network during development. Marked nodes are displayed in a highlight color to visually emphasise for example input device nodes or nodes which are often accessed during development of the application.

Moving the mouse over nodes or attributes gives brief node information like node and attribute types. The displayed information can be determined by the programmer, e.g. she can decide to display the values of the most important parameter of a node. Pressing the right mouse button while the cursor is on top of an attribute in the node parameter dialog allows changing some general attribute properties and properties specific for the attribute type, e.g. if an attribute shortcut is visible, if the attribute can be edited in the dialog, accuracy of floating point values etc.

Connections as well as nodes can now be deactivated, thus breaking a connection without manually disconnecting attributes. Connecting attributes by dragging attributes from out the node parameter dialog automatically creates shortcuts for the connected attributes.

For debugging purposes a *step frame* mode allows evaluation of just a single frame and analysis of the result. An *evaluation tracking* displays the names of the evaluated nodes in order of evaluation. A *performance tracking* gives statistics on the performance of the network, e.g. the overall computation time, the system computation overhead, e.g. the time spent for dependency checking, and the computation time spent for each node that has been evaluated.

1.3.1.1 Functions

Often values received by a node have to be mapped by splines, expressions or other mappings dependent of that parameter. Since it often is not desirable to display these simple operations as part of the network, this functionality can be hidden using a function concept. The idea of a function is that wherever a node's algorithm reads an attribute value, a function is called on the received value before it is passed to the algorithm.

Functions are implemented in MTK as simple nodes. In fact each node realizes a function which maps parameters from one domain to another. A programmer can make each node available as a function by declaring two floating point attributes of the node as function input and output parameter. A special connection type indicates that the node is used as function.

In terms of the user interface these functions can be directly applied to attributes within the node parameter dialog. Pressing the right mouse button on the attribute editor opens a menu with the available *Built In Functions*. By default the applied built in function node is not displayed, it can be accessed and edited through the dialog of the related attribute.

1.3.2 Basic Attributes and Nodes

Before summer 1999 a set of attributes and nodes has been implemented which is listed in Table 1.1 and Table 1.3. This set was supposed to be the base set of attributes and nodes useful for mapping purposes. Most of the attributes and nodes chosen as the base set turned out to be useful in the three applications mentioned above. For each of the different applications special plug-ins have been developed. Some have been used in more then one application and became part of the basic set.

Table 1.2 shows the set of the attributes that have been added to the base set. The *Trigger* attribute is an **On** or **Off** value which is On only after setting and valid only for one frame. It is used for passing events indications between nodes or triggering events via user interface. In the user interface the trigger attribute appears as press button.

A *StringEditor* attribute has been derived from the String attribute to allow the input of multiple lines of text. Font and number of lines displayed are adjustable. Additionally it allows displaying an arbitrary number of buttons and applying programmable actions, e.g. for loading and saving of the text.

The *Range* attribute consists out of a begin and an end value. The range value is read by the algorithm with an argument in [0..1] resulting in a value which interpolates the begin and end values. If a function is connected, the interpolated value is passed to the function first. This allows the interactive definition of mappings for iterated values. For example a node might generate 10 values during computation which by default are just linear interpolations between two given limits, e.g. to define an equal step shifting of a geometry. By connecting a spline as a function to the range, the shifting can follow an arbitrary curve, or by connecting an expression follow a mathematical function.

The *Spline3D* attribute realizes spline functionality in 3D space. The spline is edited in the user interface as a 3D perspective projection of an open or closed spline curve allowing to shift, add and remove points, selectively lock axis or constrain points to a movable grid. A *Spline3D* node computes a 3D coordinate dependent on an argument in [0..1].

Table 1.3 shows the nodes implemented until D 4.3/4.4, Table 1.4 lists the nodes which have become part of the base system.

An *Arith* node allows mathematical operations of an arbitrary number of input parameters. Beside the basic +,-,*,/ mathematical operations, it determines minimum and maximum input value, averages the input values or applies the logical operations *and* or *or*.

A *Filter* node implements filtering operations in the time domain by storing a number of received values on which a weighted sum is computed. The weight of the single values is defined by a spline. It can be used to smooth noisy input device values or to average or delay an arbitrary number of input values over time. In the workshop “Mixed Reality Group Interaction” it has been used for smoothing the volume change when setting a speaker's volume to a certain value.

Attribute Type	Description	User Interface	Inherits
Float	Floating point value	Slider	Atomic
Int	Integer value	Slider	Atomic
Boolean	Boolean value	Check Box	Int
Option	Choice between offers	Option Box	Int
String	String of letters	Text Edit	Atomic
Compound	Container Attribute	Browsable	Atomic
Array	Array of arbitrary attributes	Browsable	Compound
FloatArray	Array of floating point values	Browsable, Scale sliders	Array
Vector2D/3D/4D	Array of 2/3/4 floating point values	Browsable, Scale sliders	FloatArray
VectorArray	Array of Array of floating point values	Browsable, Scale sliders	Array
Spline	VectorArray with Spline functionality	Spline Editor, Preview	VectorArray
Transform	3D Translation/Rotation/Scale	Browsable, Scale sliders	Compound
Range	Defines a Range by two float values	Two Scale slider	Vector2D
Image	Pointer to an Image	File Name Edit, browsable	Pointer
Dynamic	Dynamic	-	Compound

Table 1.1: Base Attribute set of the first implementation

Attribute Type	Description	User Interface	Inherits
Trigger	Boolean value	Button	Int
StringEditor	Allows to type in multi line texts	Multi line text editor	String
Spline3D	VectorArray defining a spline in 3D space	Perspective Spline Editor	VectorArray

Table 1.2: Attributes added to the base set

Node	Description
Blend	Interpolates between two values
Spline	Spline function
Expression	User definable expression
Time	Time or Frame based timer
Mutation	Smooth random value generator
Select	Chooses output attribute dependent on the input value
Print	Print floating point value, vector or transformation
Inverse	Invert floating point value, vector or transformation

Table 1.3: Base Node set of the first implementation

Node	Description
Arith	arithmetic operations on an array of values, +, -, *, /, min, max, average, logical and/or
Spline3D	3D spline evaluation
Filter	filters input values over time, computes a weighted sum, weight defined by a spline

Table 1.4: Nodes added to the base set

1.3.3 3D Capabilities

MTK has been used as the main mapping and image creation software in an interactive installation created for the Skoda Pavillon in the VW Autostadt in Wolfsburg. The display environment is a 360 degree dome projection, four projectors fill the dome with seamless graphics surround imagery. The imagery is generated by an SGI Onyx with four video channels output. The user interface is the Panoramic Navigator, a rotatable touch screen with a camera mounted on top of it. The camera image views a rectangle piece of the projection on the touch screen. On top of the camera image icons are displayed which can be touched by the user and trigger events in the projected imagery.

One technical difficulty in a spherical environment is to generate seamless images. One problem is the blending of the projection edges. Next, projecting a plane image onto a spherical surface distorts the image. The graphics projection model implemented in the commonly available graphics hardware is based on the camera obscura, which is a perspective projection model based on linear calculations. A panoramic projection requires a distortion of the 3D geometry to compensate for the distortion caused by projection on a sphere and the implemented hardware camera model.

Because the commercial general purpose graphics software are not capable of doing this kind of distortion in real time, it was decided to integrate a 3D graphics rendering engine into MTK. In fact it was possible to build this 3D capability on top of the existing functionality. A set of classes has been implemented which realizes the usual scene graph concept common to most 3D graphics engines. Objects in the 3D scenery are linked in a scene graph which contains information on 3D geometry, object transformations, lights and materials. The basic scene graph nodes have been designed as C++ classes in an extensible way, using virtual methods for drawing, defining materials, binding textures etc, thus allowing programmers to extend the graphics capabilities via plug-ins.

A set of attributes and nodes have been created which e.g. allow nodes to pass connection points to the scene graph to other nodes. A node can add scene graph elements like materials, textures and geometry which itself may be created by the node itself algorithmically. In each frame the scene graph is created from scratch and deleted after displaying. To make the creation efficient it is possible to cache scene graph elements or even complete subgraphs.

The complete cycle of a scene graph creation and display is built on top of the MTK concept. A 3D camera node, which in fact is an output device, is derived from a device node. The scenery is displayed in the *postCompute* method of the device node (see D 4.3/4.4). The first connection point to the scene graph is offered by a SceneGraph node which again is derived from a device node, here using the *preCompute* method to delete the current scene graph before evaluating the network and passing a connection point to the root of the scene graph.

In Table 1.5 the set of attributes is listed supporting the exchange of 3D data between nodes. The SceneGraph attribute is used to pass scene graphs, a Texture attribute gives basic texture functionality, like scaling, mapping etc. Out of that an image texture attribute is derived, which allows loading image files and using them as textures. Color and SGMaterial attributes let the user define material properties, the SGGeom attribute is a container attribute for materials, textures and geometric transformations. The SGPrimitiv attribute allows setting a geometric primitive, like a sphere, box, cone etc. The Resolution attribute is a floating point value which is scaled by a global variable, thus providing a resolution hint which can be changed globally.

The SGParticle attribute allows passing particle systems and letting a node make computations on a particle system created by another node.

Attribute Type	Description	User Interface	Inherits
SGSceneGraph	Passes connection points in the scene graph	-	Pointer
SGTexture	Offers basic texture definition functionality	Scale sliders, option boxes	Compound
SGImgTexture	Definition of an image texture from file	String dialog, file browser	SGTexture
Color	Definition of a color	Color hexagon, RGB values	Vector4D
SGMaterial	Definition of an OpenGL style material	dialogs for color components	Compound
SGGeom	properties of a geometry , material, texture...	Inherited dialogs	Compound
Resolution	Globally adjustable geometry resolution hint	Scale slider	Float
SGPrimitiv	Basic geometric primitives (box, sphere, ...)	Option box	Option
SGParticle	Allows to pass particle systems	-	Pointer

Table 1.5: Attribute set supporting 3D Capabilities

Table 1.6 shows the additional base nodes supporting 3D graphics creation. A SGCamera node opens a window and displays the 3D scenery using OpenGL. The SGScene node creates the connection point to the root of the scene graph. The SGTransformation node inserts an computationally efficient transformation concatenating two linear transformations. The SGGeom node allows the insertion of materials, image textures and geometric transformations. The SGMaterial redefines the actual material displayed, the SGBoundingVolume node inserts an adaptive bounding volume for culling purposes. The SGImport reads geometry from a file in Wavefront OBJ file format, a commonly used format for exchanging 3D objects. The SGSpline allows moving a geometry or a scene graph along a 3D spline.

Node	Description
SGCamera	Opens a 3D perspective window and renders the created scene graph
SGScene	creates the connection point to the root of the scene graph
SGSubScene	creates the connection point to the actual entry point of the scene graph (convenience node)
SGTransformation	Inserts two linear transformations in the scene graph
SGGeom	Inserts optionally a material, a texture and a linear transformation in the scene graph
SGPrimitive	Inserts geometric primitive in the scene graph, applies a material, a texture and a transformation
SGMaterial	Inserts a material in the scene graph
SGBoundingVolume	Inserts a static or adaptive bounding volume in the scene graph
SGImport	Reads geometry from a file in Wavefront OBJ file format
SGSpline	Moves geometry or a scene graph along a 3D spline

Table 1.6: Attribute set supporting 3D Capabilities

For the interactive installation for the Skoda Pavillon and for the dance performance a number of plug-ins have been created which allow sophisticated graphics creation, but since they don't belong to the basic graphics operations they will remain as dynamical loadable plug-ins. At this point the capabilities are only listed without going into detail.

There are nodes for defining color splines, for painting images and procedurally creating images which can be used as hardware textures, for animating textures as a single frame sequence, for playing back movie files on textures, for using interactively rendered scenes as dynamically changing textures, for blending and overlaying an arbitrary number of textures on the same geometry, for projecting textures onto geometry, for mapping fisheye images onto geometry, for changing OpenGL blend functions, for clipping parts of the scene, for ensuring a constant frame rate, for computing the global transformation of scene graph connection points, for light definition, for switching sub scene graphs with smooth transitions like fading to black, transparent and scaling, for selectively displaying sub scenes in shaded, wireframe or vertex mode, for creating 3D type geometry out of typed in text and more.

A special camera node is derived from the SGC camera that is capable of rendering 4 different viewpoints into 4 windows in parallel. Additionally before rendering the 3D geometry, it distorts the geometry appropriately for a spherical projection.

1.3.4 Script Language

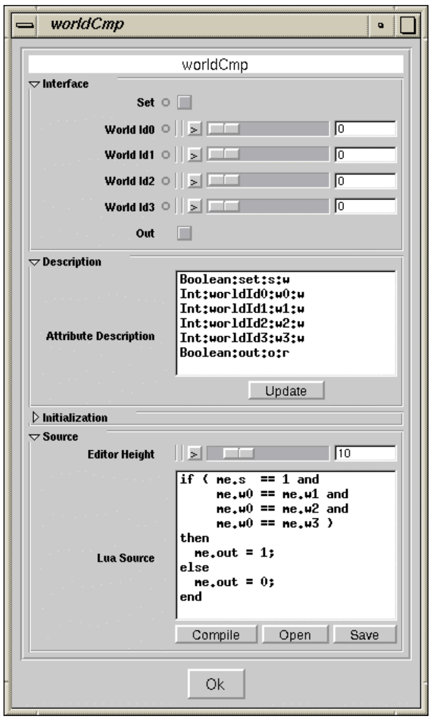
During the development of the dance performance it turned out, that even though MTK offers a lot of possibilities there often appear special cases where the required functionality can be realized only by creating a network which is more complex that the problem justifies. In these cases plug-ins had to be programmed but a more convenient solution allowing the implementation of the required nodes more quickly would be desirable. To face this problem, the script language Lua has been integrated into MTK.

Lua was designed and implemented at TeCGraf, the Computer Graphics Technology Group of PUC-Rio by Roberto Ierusalimschy Waldemar Celes and Luiz Henrique de Figueiredo and is freely available[5]. Lua is a programming language designed to support general procedural programming and has been designed especially to be used as embedded language in a host program.

Since the requirement was to speed up the development of node functionality on the fly, a special node has been created which executes a user definable script instead of a hard coded algorithm. An interface layer has been implemented which generically converts MTK data types into Lua data types and vice versa. By that all attribute types are visible in a lua script, the script can access global variables and all nodes defined in the network plus their attributes. Additionally vector and matrix operations and basic scene graph operations have been implemented in Lua. A script can manipulate the scene graph, insert transformations and geometry. A script node can store local variables which are persistent during successive frames.

The attributes of a Lua node are declared in a string editor by the attribute type, the name and by setting flags for if they are input or output attributes, if they have a shortcut, are visible in the user interface etc. The attribute declaration and the script code can be changed on the fly while the application is running. The script can be saved and loaded from a file. Figure 1.3 shows the Lua node dialog. The upper string editor contains the attribute description, the lower string editor shows the script code. This script was used in the workshop “Mixed Reality Group Interaction” for detecting if four values were equal and a Boolean value was set. Another script was averaging four positions in space, adding an offset and determining the distance to a certain position is space.

[5] http://www.tecgraf.puc-rio.br/lua



worldCmp

Set
World Id0: 0
World Id1: 0
World Id2: 0
World Id3: 0
Out

Attribute Description

```
Boolean:set:s:w
Int:worldId0:w0:w
Int:worldId1:w1:w
Int:worldId2:w2:w
Int:worldId3:w3:w
Boolean:out:o:r
```

Editor Height: 10

Lua Source

```
if ( me.s  == 1 and
     me.w0 == me.w1 and
     me.w0 == me.w2 and
     me.w0 == me.w3 )
then
  me.out = 1;
else
  me.out = 0;
end
```

Figure 1.3: Example Script Node.

Lua scripts can be registered as presets to the system. These node presets behave in all aspects like any other node, e.g. they are accessible in the node toolbar with their preset name. The script and attribute declaration editors are hidden automatically, such that there is no visual difference between a hard coded and a script node. By that users can build libraries of useful script nodes.

1.3.5 Device and Database Concept

Each of the applications mentioned above, the installation, the performance and the workshop required to read and write data from and to hardware devices. The installation for the Skoda Pavillon reads data from the touch screen and communicates with the sound engine running on an Apple Macintosh via serial ports. The main application communicates with the user interface application, which also is implemented in MTK and running on an SGI O2, via TCP/IP. For the dance performance the Polhemus^{TM} tracker delivered data via UDP.

In both applications the device nodes for reading and writing data have been implemented reusing code fragments of the others implementation. It turned out that the techniques to make the data visible in MTK could be generalized.

Before the workshop “Mixed Reality Group Interaction” a generic concept has been developed and implemented for reading and writing data from and to arbitrary hardware devices. This concept abstracts from a concrete hardware device and offers a generic interface to MTK, that means a set of operations where just the low level operations have to be implemented by a programmer for a specific device.

It offers a generic database concept for storing data received or to be sent, supports automatic parallel processing for slow hardware devices like the serial port and, in order to not slow down the main application, it implements an abstraction layer for ASCII and binary data and introduces a protocol layer for higher level devices.

On top of this implementation for the workshop a MIDI device node for sound control and a TCP/UDP device node have been implemented. The MIDI device node reads and writes the full set of MIDI events and makes them available as MTK parameters. The TCP/UDP device node was used to implement the nodes which communicate with the BFinder application as well as MASSIVE-3 (see D 7b.4).

1.4. Summary

MTK has been designed to support various kinds of mappings for interactive installations and performance purposes. In a first step the basic system architecture has been designed and implemented.

MTK is a network based system with a dependency oriented evaluation strategy, which additionally introduces control flow capabilities. A small number of nodes and attributes supporting standard mappings are part of the base system.

The implementation offers a completely interactive user interface, a visual representation and manipulation of the network and the node attributes for quick creation and adjustment of mappings on the stage.

The basic functionality of the system can be extended by programmers via a plug-in mechanism. Specific mappings and data types can be programmed in the C++ programming language and loaded to the system on demand. This can be utilized for building libraries of mappings. Additionally less complex nodes can be programmed on the fly while the application is running using an integrated script language.

MTK has been used as the main software to implement an interactive real time installation, which is exhibited permanently in the Skoda Pavillon in the VW Autostadt in Wolfsburg. It has proven to be robust and capable of dealing with complex data. The development of the installation as well as the experiments with a dancer from the Frankfurt Ballet led to the development of a useful set of specialized nodes. The practical experience clarified some concepts which could be generalized, implemented and applied in the workshop “Mixed Reality Group Interaction” described in D 7b.4.

Chapter Two
Production Support Technology: Event Management and Software Vision Mixing Tools

Adam Drozd and Ian Taylor
University of Nottingham, Nottingham, UK

2.1 Introduction

This chapter presents details of the production support technology developed for the Avatar Farm demonstrator. This comprised of the Event Management tools, Camera Controls and Software Mixing Environment. These are all enhancements to Nottingham's MASSIVE 3 CVE platform.

2.2 Management System for *Avatar Farm*

2.2.1 Event Management in *Out Of This World*

In *Out Of This World* (*OOTW*, see Deliverable D7a.1) the Event Management was provided using phases. The flow of the game show was split up into what was known as phases. Each phase described the location or the trajectory for objects in the world. The phases could also contain user constraint positions, these constraint positions were used to allow the limitation of navigation for the participants.

The phases consisted of two parts. Firstly there was additional information placed in the world description file. This information was used to define regions for the user constraints and to link objects in the world with a particular phase or phases. The second part was the Event Manager script that contained an entry for each phase of the show. Each phase contained a unique phase identifier, the name of the phase and a user list. This user list gave names of individual users or groups of users and any user constraints that were to be applied to them, information about audio levels, viewpoints for users and any highlighting of users. The Event Manager software then read this script in.

The phases were controlled by the Event Manager software. A phase could either be triggered automatically after a period of time or could be manually triggered by a member of the production crew as the show progressed. The phases were shown to the crewmember as a list, showing the phase names.

The main advantages with this system was that it produced a highly structured narrative for the show, but on the other hand still allowed the participants to explore their environment but the show could be brought back to a known state at any time by the Event Manager selecting a phase.

The Event Manager was also used to provide a means of selecting different paths through the narrative. This was used in cases when the outcome of a game was unknown.

The criticism with this system was that although it produced a coherent show, it sometimes was too severely controlled and this restricted the amount of improvisation by the participants. This was due to the phase structure being fixed and there was no way to override the phases other than to select the next one.

2.2.2 Event Management in *Avatar Farm*

Originally during the development of *Avatar Farm* the approach taken was the same that was used for *OOTW* in that the narrative would be used to produce a set of phases. The phases would be used to apply user constraints to the participants and to position objects as in *OOTW*. However it became apparent early on that this approach would not be adequate for the interaction that was required for the complex narrative that was being developed. The problem occurred that as the narrative progressed through the chapters there were many branches that could be taken. Although the phase system in theory could accommodate this kind of narrative it would become impossible for the crew member that was controlling the phases to know which phase was to be selected next due to the large number of phases that would have been required to accommodate the narrative.

Another problem also emerged during development relating to object interaction. The narrative required very rich interaction with objects. In principle, the interactions required would have been possible in the MASSIVE-3 system. This could have been achieved though the hard coding of interactions for objects in the worlds, but the results of the interactions varied depending on many conditions at the moment of interaction. Coding all of these conditions, even if possible, for the objects would have taken an unacceptable amount of time.

2.2.3 Solution to Management Problems

Due to the two problems given above it was apparent that a different approach was required. The solution that was chosen was to have invisible ‘helpers’ in the virtual world with the actors and participants. This approach is analogous to that of having stagehands in a theatrical production.

The participants (the inhabitants and the actors) were able to undertake only a few very basic direct object interactions themselves, such as picking up, carrying and putting down objects, as well as being able to travel between the worlds through ‘portals’ that were placed around the four worlds. However, the narrative required more complex object interactions. These were created theatrically by various helpers (several stagehands and a world-manager) working behind the scenes to give objects the appearance that they have complex behaviors and functions. The (invisible) world-manager and stagehands were able to carry out a few additional actions, such as making objects visible and invisible or applying movement constraints to participants and objects. Participants had to make a request for an object interaction that the helpers could recognize and react to. This is akin to having invisible stage-crew on a theatre stage who can pick up and move objects or hide and reveal them in response to actors’ dialogue and movements.

This process was not instantaneous, as the helpers had to be able spot that an interaction is being requested in advance and prepare for it. However, often the helpers themselves did not actually have to spot the requests for interactions themselves, instead they were prompted by Artistic Director. Once the 'helpers' realised that an interaction was required, they had to move to the point in the world where the interaction was to take place. This was achieved though the interface the ’helpers’ were given which is discussed later. Once at the location they then may need to undertake several actions to make the interaction happen. For example they may need to move one object and then make a second object invisible. To allow the ’helpers’ to produce this type of interaction, it must be predictable a long time in advance and sufficiently slowly paced so that helpers could improvise it. These interactions were designed so that they took tens of seconds or minutes rather than just a few seconds.

The helpers also need to know clearly what the participants are trying to achieve. This meant that the participants (through help from the actor characters) needed to make their requested interactions very explicit. This was achieved in various ways; such as speaking instructions to objects that are apparently intelligent, reciting magic spells, or participants had to gather and arrange objects at particular locations and then have to publicly speak key phrases so as to achieve a magic effect.

Another important motivation for this approach is portraying interaction to the viewers. Viewers must have been able to follow what was happening when objects were being used. Given that we could not support many of the finer nuances of how people normally interact with objects (e.g., gaze, posture, effort, fine finger manipulations, extension of limbs towards objects), we compensated by making other aspects of the interaction more pronounced (e.g., accompanying speech) and by slowing down the pace of interaction.

As a result of this approach, objects could behave quirkily. They need not be deterministic and can do odd things or change the way they work from moment to moment. This was useful in terms of the narrative and from a technical point of view, given that helpers may have missed some requests for objects interactions (e.g., if many requests were being made in parallel in different worlds) or may not be able to satisfy them properly. From the narrative point of view it was useful, as everything was unpredictable and likely not to work or at least to work in strange ways. This was entirely consistent with metaphors such as living objects, magic spells and incantations.

In addition to the ‘helpers’ there was a ‘world-manager’ who was responsible for enabling access controls on portals between worlds and on objects in the worlds. The ‘world-manager’ received their cues in the same way the helpers did, either from in world events or from the Artistic Director.

In summary, participants needed to explicitly request most complex interactions in the form of spells, prayers, incantations, passwords and so forth. The interactions were then improvised by actors and world 'helpers' working invisibly behind the scenes.

2.2.4 The atoms of interaction

The atoms of interaction are the basic actions that the various kinds of participants could directly carry out for themselves in the *Avatar Farm* worlds. It is from these alone that the more complex interactions must be improvised. We have six kinds of participant:

- Inhabitants – the players
- Desktop-roles – the actors who use the desktop interfaces
- Immersed-roles – the actors who use the immersive interfaces

- Stage-hands (Helpers) – behind the scenes crew who can manipulate individual actors, players and objects within the world
- World-manager – behind the scenes crewmember who could manipulate access control for portals between worlds and objects in the worlds.
- Temporal-links operator (performed by one of the stage-hands) – could select and replay pre-recorded 3D sequences within one of the four worlds.

Inhabitants

The inhabitants were normally able to:

- Move around the ground-plane at will (subject to movement constraints applied by a stage-hand). They cannot fly vertically into the air by default, but may be granted the ability to do so to a fixed height by a stage-hand.
- Talk to others within their immediate vicinity (via a pre-defined 'nimbus', see Chapter 3, that governs the direction and extent of the projection of their speech).
- Pick up and wave around only one object at a time while standing still (subject to proximity to the object and to access controls).
- Pick up only one object at a time and carry it with them while moving through the world (subject to proximity to the object and to access controls).
- Put an object down.

It was also possible to dynamically grant them the following abilities:

- Go between two worlds via a particular portal (as granted by the world-manager).
- Fly vertically at will up-to a preset height.
- Switch their appearance between a number of pre-defined avatars.
- Switch between being visible and invisible.

(Note that these last two abilities were not actually used during the event).

Minor-roles

These were able to do everything that the inhabitants can do. However, they were granted the extra capabilities (switching appearance, flying, and invisibility) by default.

Major-roles

The immersed actors had the same abilities as the Minor-roles but in addition could:

- Move their heads and hands (as they were tracked).

Stage-hands

These were responsible for manipulating individual actors, inhabitants and objects in the following ways:

- Move them around on the ground plane, move them vertically, and turn them to face left and right (as required).
- Make them visible or invisible.

- Switch their appearances between a number of pre-defined 3D models.
- Make them larger or smaller.
- Freeze them to the spot so that they can still gesture, but cannot move around within the world.
- Grant and revoke the ability for inhabitants to make themselves visible or invisible.
- Grant and revoke the ability for inhabitants to change their own appearance (switching between predefined avatars).
- Grant and revoke the ability for avatars to fly.

World-manager

They were responsible for dynamically granting and revoking access controls. They will be able to:

- Dynamically grant and revoke access controls to individual inhabitants for moving through portals.
- Dynamically grant and revoke access controls to individual inhabitants and picking up (access controlled) objects.

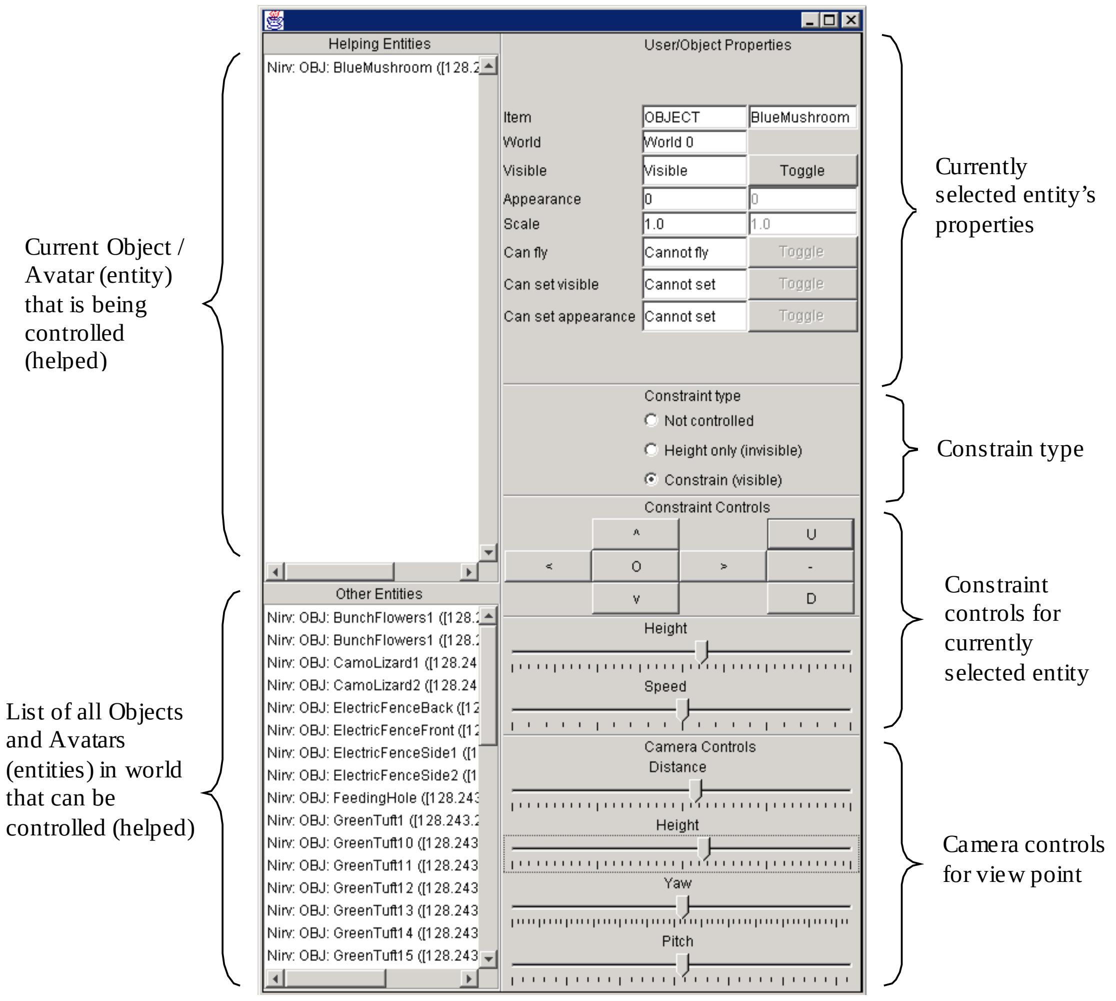
Table summarising the different actions that are available to the different kinds of participant:

Action	Inhabitant	Minor-role	Major-role	Stage-hand	World-manager
Movement on ground plane	subject to constraints	subject to constraints	subject to constraints		
Fly vertically to pre-set height	may be granted	yes	yes		
Move through portals	subject to access controls	subject to access controls	subject to access controls		
Pick up-object – wave, carry.	subject to access controls	subject to access controls	subject to access controls		
Talk (local vicinity)	yes	yes	yes		
Gesture	yes	yes	yes		
Turn visible/invisible	may be granted	yes	yes		
Change appearance of self	may be granted	yes	yes		
Move object or other's avatar				yes	
Make other (avatar or object) visible/invisible				yes	
Change appearance of other (avatar or object)				yes	
Change size of other (avatar or object)				yes	
Freeze other to the spot				yes	
Grant other ability to fly				yes	
Grant other ability to change visibility				yes	
Grant other ability to change appearance				yes	
Grant and revoke other permission to move through portals					yes
Grant/revoke other permission to pick-up object					yes

2.2.5 The Management Tool Interfaces

The Stage-Hand Interface

The stage-hand interface consisted of two windows the first contains the controls (Figure 2.1) and the second contains a view on the world in relation to the current object or Avatar (entity) that is currently being controlled (helped) (Figure 2.2).


Current Object / Avatar (entity) that is being controlled (helped)
List of all Objects and Avatars (entities) in world that can be controlled (helped)
Currently selected entity's properties
Constrain type
Constraint controls for currently selected entity
Camera controls for view point
Helping Entities
Other Entities
User/Object Properties
Constraint Controls
^
U
<
O
>
-
v
D
Height
Speed
Camera Controls
Distance
Height
Yaw
Pitch

Helping Entities

Nirv: OBJ: BlueMushroom ([128.2

Other Entities

Nirv: OBJ: BunchFlowers1 ([128.2
Nirv: OBJ: BunchFlowers1 ([128.2
Nirv: OBJ: CamoLizard1 ([128.24
Nirv: OBJ: CamoLizard2 ([128.24
Nirv: OBJ: ElectricFenceBack ([12
Nirv: OBJ: ElectricFenceFront ([12
Nirv: OBJ: ElectricFenceSide1 ([1
Nirv: OBJ: ElectricFenceSide2 ([1
Nirv: OBJ: FeedingHole ([128.243
Nirv: OBJ: GreenTuft1 ([128.243.2
Nirv: OBJ: GreenTuft10 ([128.243
Nirv: OBJ: GreenTuft11 ([128.243
Nirv: OBJ: GreenTuft12 ([128.243
Nirv: OBJ: GreenTuft13 ([128.243
Nirv: OBJ: GreenTuft14 ([128.243
Nirv: OBJ: GreenTuft15 ([128.243

User/Object Properties

Item	OBJECT	BlueMushroom
World	World 0	
Visible	Visible	Toggle
Appearance	0	0
Scale	1.0	1.0
Can fly	Cannot fly	Toggle
Can set visible	Cannot set	Toggle
Can set appearance	Cannot set	Toggle

Constraint type

Not controlled [✓ Constrain (visible)]
Height only (invisible)

Figure 2.1 : The stage-hand interface

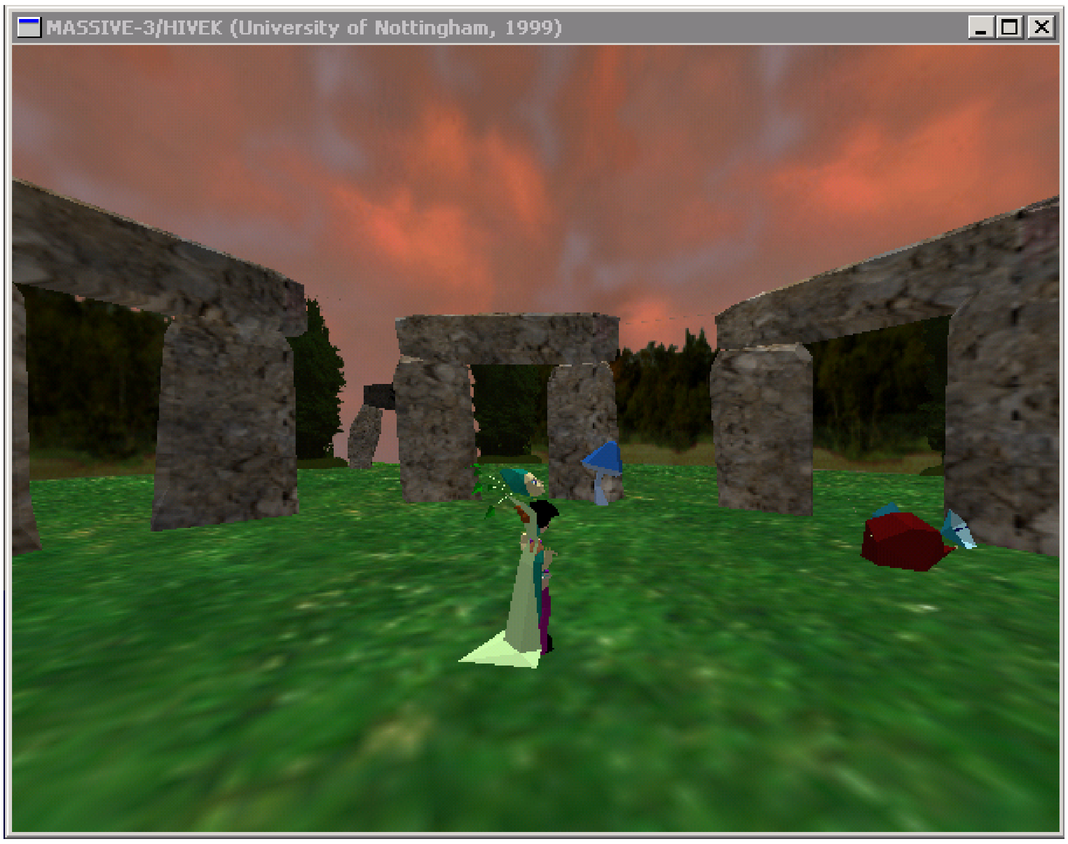


Figure 2.2 : View point on the world.

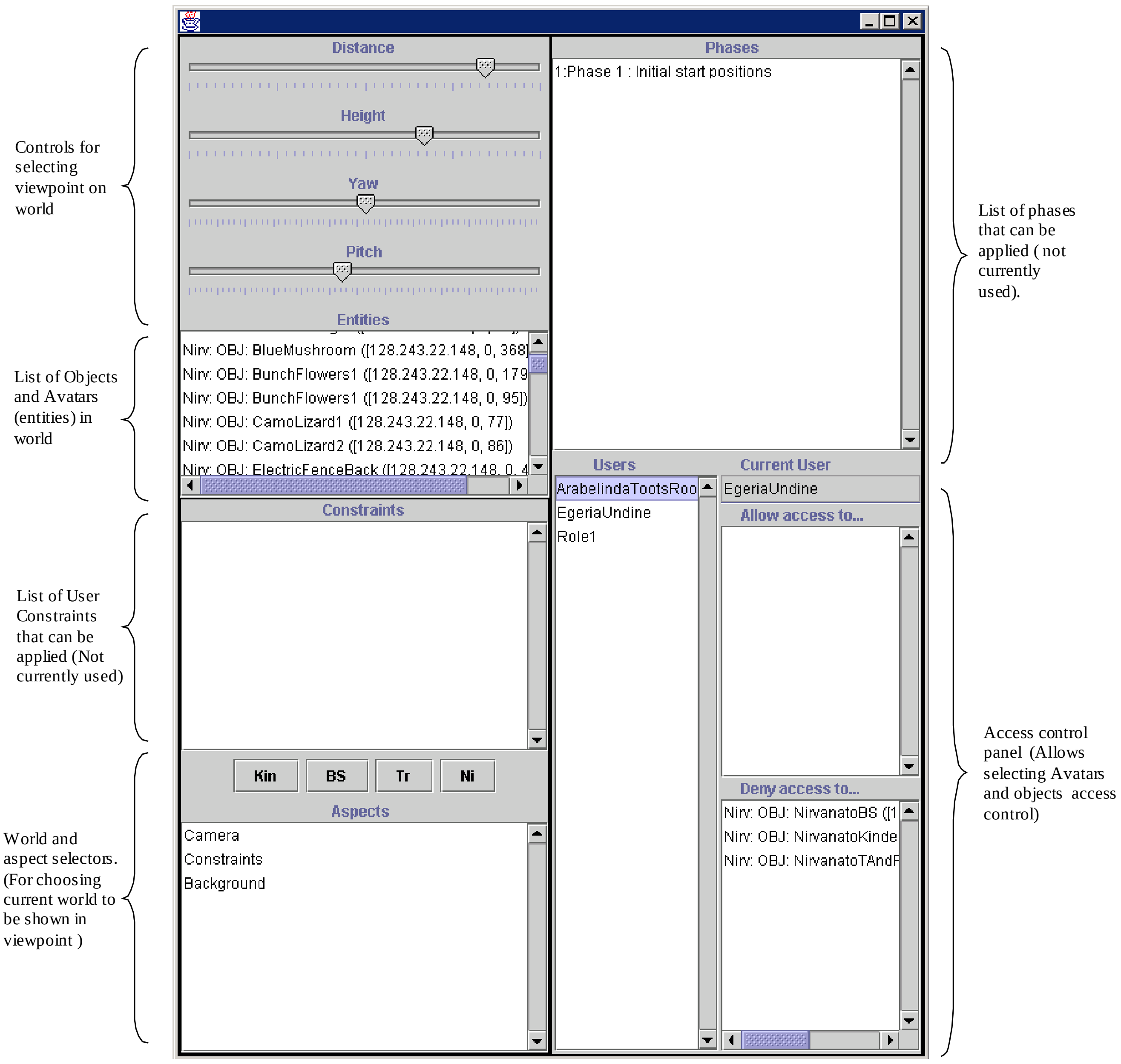
The stage-hand is able to select an entity from the list in the lower part of the interface. Upon selection the stage-hand is taken to the selected entity in the respective world. The stage-hand is then able to adjust their position relative to the selected object using the camera controls. When an entity is selected, it can then be controlled (helped). Using the constraint controls the entity can be manoeuvred around the world, the speed at which they move is determined by the Speed slider on the Constraint control panel. Depending on the type of entity that is selected different properties can be altered. If the entity is an object then the stage-hand can select if the object is visible or not, however if the entity is an Avatar then the stage-hand can perform the following:

- Selection of visibility.
- Altering the appearance of the Avatar from a pre-defined selection of geometries.
- Altering the scale factor of the Avatars geometry.
- Allowing or revoking the Avatars ability to fly.
- Allowing or revoking the Avatars ability to become invisible.
- Allowing or revoking the Avatars ability to change there own appearance (by changing geometry).

World-Manager Interface

The world-manager interface was as with the stage-hand interface, split into two parts, one window with the controls in it (Figure 2.3) and a second with the view on the world. (Figure 2.2) The world-manger can select the world they wish to view along with the aspects they wish to be shown. They then position themselves either relative to the origin of the world or to an entity (by selecting one from the given list).

To change the access control on either a portal or an object (that has access control enabled), the world-manager selects user from the list in the lower-left hand side of the window. When a user is selected the two panels to the right of the user panel show all object and portals in all four worlds, and whether the user is allowed access to them or if they are denied access.


Controls for selecting viewpoint on world
List of Objects and Avatars (entities) in world
List of User Constraints that can be applied (Not currently used)
World and aspect selectors. (For choosing current world to be shown in viewpoint)
List of phases that can be applied (not currently used).
Access control panel (Allows selecting Avatars and objects access control)
Distance
Height
Yaw
Pitch
Entities
Constraints
Kin
BS
Tr
Ni
Aspects
Phases
Users
Current User
Allow access to...
Deny access to...

1:Phase 1 : Initial start positions

Nirv: OBJ: BlueMushroom ([128.243.22.148, 0, 368]
Nirv: OBJ: BunchFlowers1 ([128.243.22.148, 0, 179
Nirv: OBJ: BunchFlowers1 ([128.243.22.148, 0, 95])
Nirv: OBJ: CamoLizard1 ([128.243.22.148, 0, 77])
Nirv: OBJ: CamoLizard2 ([128.243.22.148, 0, 86])
Nirv: OBJ: ElectricFenceBack ([128.243.22.148, 0, 4

ArabelindaTootsRoo
EgeriaUndine
Role1

EgeriaUndine

Nirv: OBJ: NirvanatoBS ([1
Nirv: OBJ: NirvanatoKinde
Nirv: OBJ: NirvanatoTAndF

Camera
Constraints
Background

Figure 2.3 : Interface for World-Manager.

Temporal Links Interface

The temporal links interface (Figure 2.4) was used by one of the stage-hands to allow the insertion, playback and deletion of the pre-recorded flashbacks. The stage-hand is able to select the replay (flashback) to be loaded using the create/file button. Once the replay has bee loaded the replay can be started using the playback controls. When loading a replay the stage-hand has the option of ‘ghosting’ the replay, this makes all objects and avatars contained in the replay to appear semi-transparent. Once the replay has been finished with it can then be deleted. During the playback of a replay, points can be saved and then at a point later in time the saved points can be jumped to.

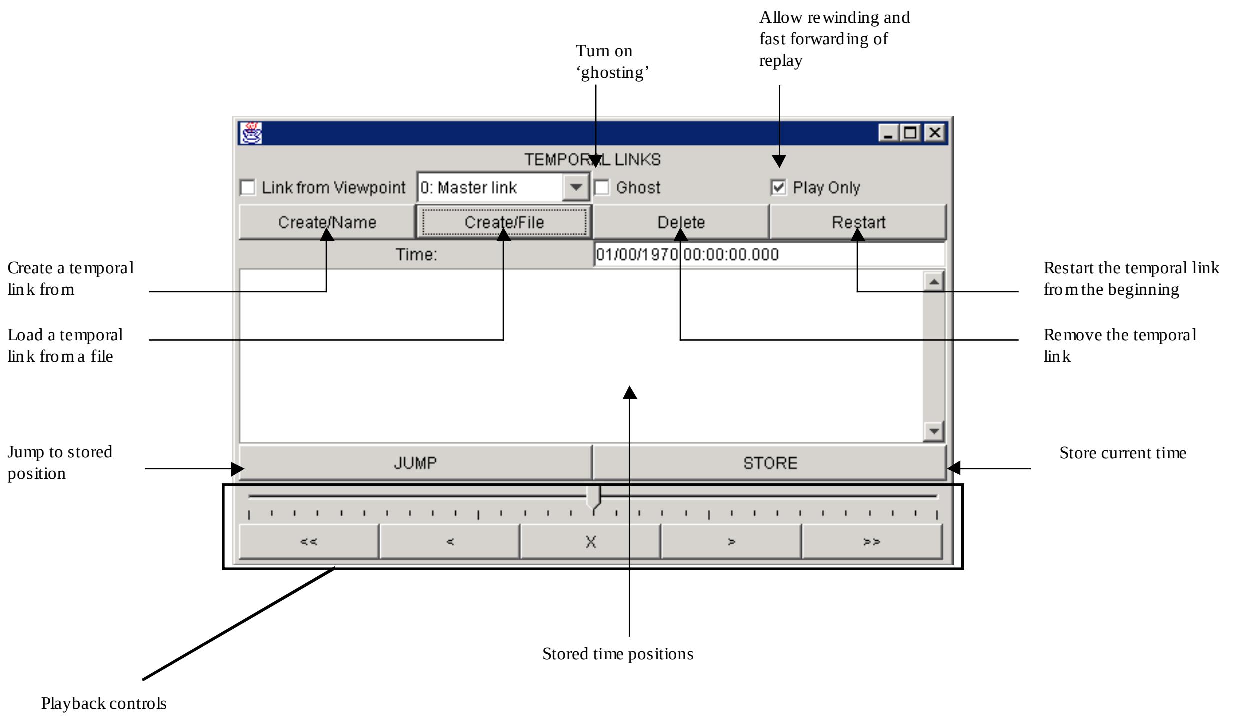


Figure 2.4 : Temporal link interface

2.2.6 Initial Evaluation and Observations

Stage-Hand and World-Manager Interfaces

The stage-hand interface performed well during the event in allowing the stage-hands along with help from the actors in the virtual world to produce interesting and entertaining interactions and also enabling the artistic director to produce narrative that was close to the initial narrative structure that was originally written. The interface allowed objects to be hidden from the view of the participants and to be revealed at the correct moment when required.

Problems that occurred at the beginning were that there was a lack of communication between the actors, artistic director and the stage-hands. This resulted in the stage-hands not being aware of changes to narrative and therefore being slow to react to situations that they were not aware were going to take place. The actors soon became aware of the problems the stage-hands were having (and that occasionally there were technical problems) and often came up with narrative to cover up the fact there was a problem.

The stage-hands also allowed improvising of the narrative. This was useful when there were technical difficulties occurred. One occasion of this was that only one temporal link could be replayed at once, but at certain points the narrative, due to actions of the participants it was necessary to replay more than one temporal link at once. This was overcome by improvisation by the actors with help from the stage-hands. Had all object interactions been pre-programmed then this would have led to many problem (as the replays would have been triggered by one object being moved on top of another).

The improvising was also useful on the occasions when the participants acted in ways that were not predicted and would have been difficult to have dealt with if objects had pre-programmed behaviours.

The following problems were encountered by the stage-hands during the shows:

- When selecting objects from the list quickly, it was difficult with so many objects to work out of which object was the one being referred to by the participants, especially when there were many similarly named objects.
- There were problems with multiple stage-hands trying to select and move the same object or avatar. This situation often ended up with objects either not being moved or confusion as to who was actually moving the entity.

These problems could be overcome with better communication systems between the stage-hands, The actors and the artistic director, as well as some kind of visual feedback in the graphic window as to what had been selected to be controlled by other stage-hands.

2.3 Software Mixing and Camera Control

2.3.1 Introduction

The *Out of This World* event demonstrated the use of a specialised real-time camera control interface to provide broadcast footage for a live inhabited TV event. In *Avatar Farm* this work was extended, after the manner of initial work conducted in Year 2 at KTH (see Deliverable D4.3/4.4), by the addition of software tools for live mixing and camera selection. These tools were developed in collaboration with the director of *OOTW* who provided valuable input at the design phase. A key development was in the use of multiple independent cameras assigned to a single camera operator. Each camera operator continued to use a custom camera interface of the type used successfully in *OOTW*. The director however relied on a new software environment to select a broadcast view from the available camera views, which were presented using a combination of manual and automated selection mechanisms. This section outlines the design process and illustrates the results of the implementation.

2.3.2 Requirements

When applying software substitution to video mixing, a rudimentary solution is to provide a software environment that enables video selection from multiple signals. This selection may be made from viewports distributed across one or more monitors, depending on the preferred configuration. By developing this system, we have essentially duplicated a typical set-up. To go from this to more sophisticated techniques, it was important to consider the following issues:

- Director’s requirements.
- Possible extensions to video presentation and selection.
- Additional visual accessories using data obtained from existing software modules.

Each of these are discussed below.

2.3.2.1 Director’s requirements

Following initial discussions with the director, the following guidelines were emphasised:

- Maintain familiar concepts; ensure the interface does not vary wildly from the director's usual working environment.
- Combining viewports on a single monitor is acceptable (realistically 4 maximum)
- In total no more than 15-20 different viewports can be viewed at once.

The main challenge as perceived by the director of *Avatar Farm* was to select interesting viewports from a number of potentially simultaneous events that may take place in a number of distinct worlds. This contrasts with *OOTW* in which the action took place in a relatively small area and was generated by a small number of participants/participant groups. The analogy given by the director was that of *The Truman Show*, a film in which surveillance cameras are hidden throughout an entire town to broadcast a 'soap opera' style depiction of a man's life. A real world comparison was also made with the use of multiple surveillance cameras in public places such as shopping centres. In these systems, viewpoints can be made to switch periodically, allowing the viewer to maintain awareness of a large number of physical spaces using relatively few video views.

It was agreed that the core strategy was to support more viewpoints than would be visible at any instant.

2.3.3 Camera and Director Suite

The camera and direction tools used in *Avatar Farm* extended the role of software over that in *OOTW* (see Deliverable D7a.1) providing a software-mixing environment for the camera director. In addition to this, the camera interface provided two independent viewpoints, thereby enabling a camera operator to cue a shot whilst providing another pre-cued view. The director was able to preview both shots from each camera operator and select any of the four for broadcast. Figure 2.5 outlines the configuration of camera, director and broadcast during the event.

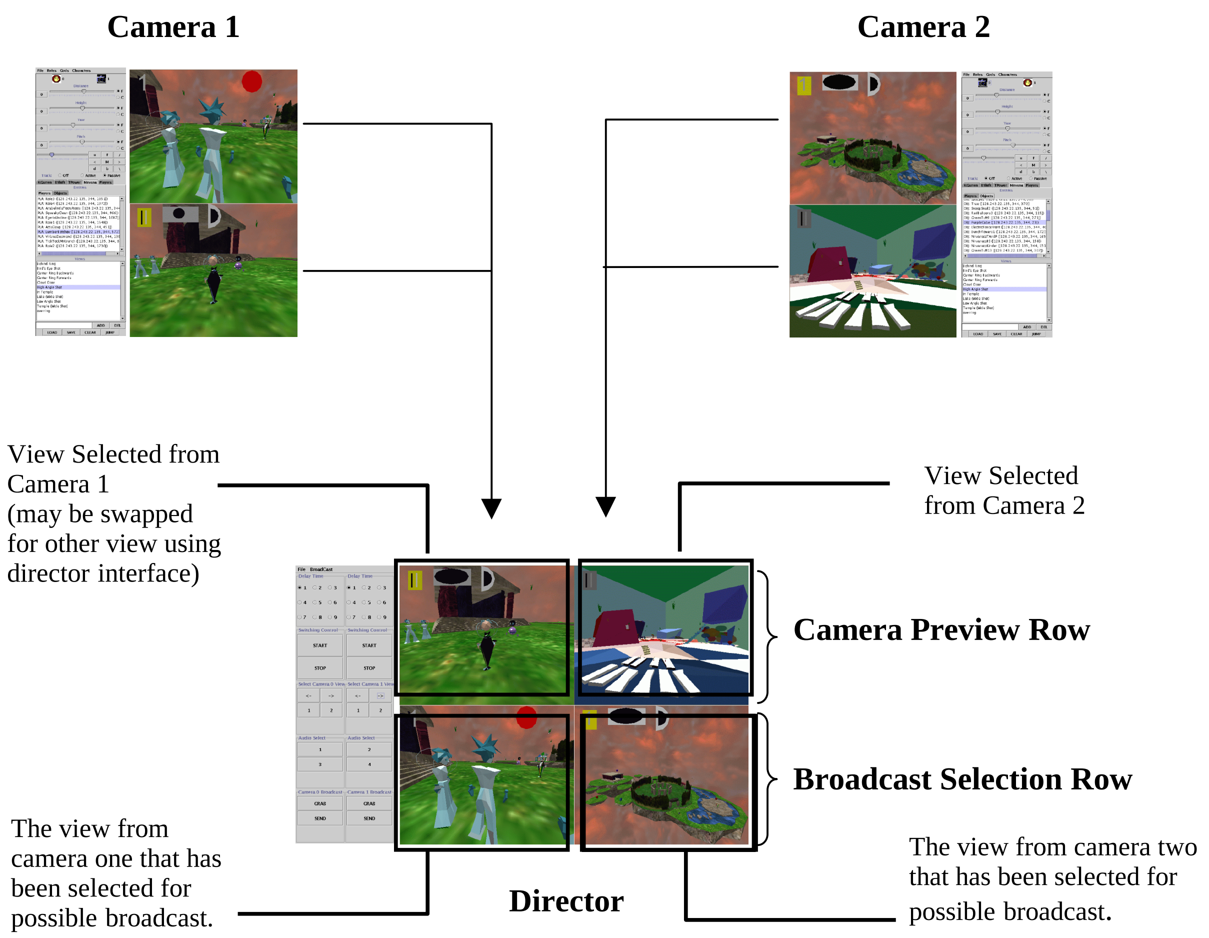

Camera 1
Camera 2
View Selected from Camera 1 (may be swapped for other view using director interface)
View Selected from Camera 2
Camera Preview Row
Broadcast Selection Row
The view from camera one that has been selected for possible broadcast.
The view from camera two that has been selected for possible broadcast.
Director

Figure 2.5. The camera and director interfaces. The upper row of the director's views contains one of the viewports from each camera. Alternate camera views may be switched using the director's interface. The lower row of the director's viewports contains potential broadcast views. Each one is a selection from the camera views that appear in the viewport above. One of the views in the lower row can be broadcast selectively.

The camera interface contains two independent viewpoints that can also be seen by the director. The upper left viewport in the director's interface displays one of the views from camera one. The upper right viewport performs the same for camera two. The lower viewports in the director's interface contain a choice from one of the camera views. The lower left viewport contains a choice from camera one, the lower right from camera two. One of these viewports can be selected for broadcast.

To understand the operation of this system in more detail, each interface will be described in the following sections.

2.3.3.1 Camera Controls

The camera interface was similar to that used in OOTW but with the following key differences:

- **Multiple Cameras**. The camera operator could control two independent viewpoints. Originally this was to be four, but due to performance problems this was halved.
- **Continuous Movement.** Any of the movement controls could be set in continuous motion, allowing smooth orbit or panning for example.
- **Viewpoint Status Indicators**. Icons appeared in the viewpoint to indicate the following conditions:
    - Viewpoint selected for control.
    - Viewpoint being monitored by director.
    - Viewpoint being broadcast.
- **Separation of Content.** The user interface categorised the entities into worlds and further divided them into players and non-players. A pull-down menu was available to quickly locate any character, irrespective of world location.

Figure 2.6 shows a screenshot from the camera operator's interface.

The functionality offered by the components camera interface can be summarised as follows:

- **Avatar locator menu.**

As with the camera control detailed in Deliverable 7a.1, object-centred control was offered to enable easy tracking of particular avatars. The operator could select Roles, Characters or Gods.

- **Camera selection.**

This determined which of the viewpoints (upper or lower) would be affected by the camera controls.

- **Relative Movement Controls.**

In addition to controls of the type described in Deliverable 7a.1, the operator could select continuous movement of each transformation type. The speed and direction of this movement could be adjusted using the slider. A button was also present to set the slider to its zero position.

- **Free Movement Controls.**

These allowed free navigation at adjustable speed.

- **Tracking type selection.**

Again, similar to those given in Deliverable D7a.1. Any entity could be tracked in both position and orientation (passive) or position alone (active).

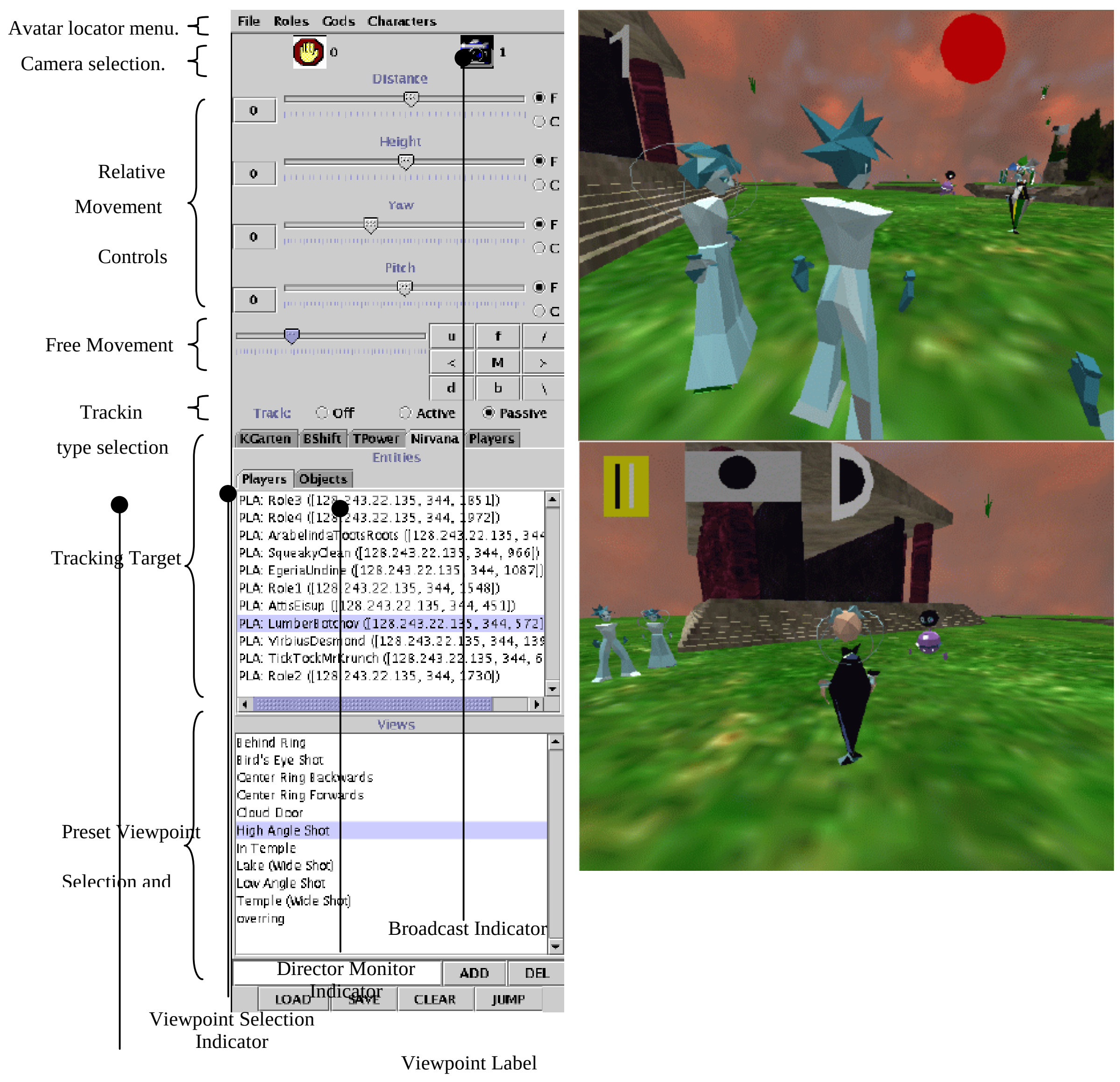
- **Tracking Target Selection Panes.**

This area presents selectable lists of entities categorised by world and sub-categorised by player (role or actor) or non-player. Selecting an item and clicking on a tracking type will instantiate tracking of that item in the appropriate world.

- **Preset Viewpoint Selection and Editing.**

As with *OOTW*, the user was able to define preset viewpoints. In *OOTW* these viewpoints were relative to the object being tracked. In Ages of Avatar this was possible using the preset pane labelled ‘Player’. Also, four lists were available to transport the camera to a pre-defined point in each world.

- **Indicators:**
  - **Viewpoint Label.**
    A numeric label for the viewpoint.
  - **Viewpoint Selection Indicator.**
    An icon to indicate which viewpoint is currently selected for control.
  - **Director Monitor Indicator.**
    This indicator appears when the director is previewing the viewpoint in their camera preview area.
  - **Broadcast Indicator.**
    Appears when the viewpoint has been selected for broadcast by the director.



```
PLA: Role3 ([128.243.22.135, 344, 1851])
PLA: Role4 ([128.243.22.135, 344, 1972])
PLA: ArabelindaTootsRoots ([128.243.22.135, 344
PLA: SqueakyClean ([128.243.22.135, 344, 966])
PLA: EgeriaUndine ([128.243.22.135, 344, 1087])
PLA: Role1 ([128.243.22.135, 344, 1548])
PLA: AttisEisup ([128.243.22.135, 344, 451])
PLA: LumberBotchov ([128.243.22.135, 344, 572]
PLA: VirbiusDesmond ([128.243.22.135, 344, 139
PLA: TickTockMrKrunch ([128.243.22.135, 344, 6
PLA: Role2 ([128.243.22.135, 344, 1730])
```

```
Behind Ring
Bird's Eye Shot
Center Ring Backwards
Center Ring Forwards
Cloud Door
High Angle Shot
In Temple
Lake (Wide Shot)
Low Angle Shot
Temple (Wide Shot)
overring
```

Figure 2.6.Camera interface for Avatar Farm.

2.3.3.2 Director's Interface

For the director, it was important to offer a mechanism by which the viewpoints could be previewed and easily selected for broadcast. The interface was designed to accommodate a number of viewpoints per camera operator. These viewpoints could be cycled automatically using a selectable time delay. Alternatively, the director could stop automatic cycling and manually select available viewpoints. To select a viewpoint for broadcast, the director would first 'grab' a viewpoint. This effectively copied the viewpoint to the lower broadcast row (see Figure 2.5), when satisfied that the viewpoint was ready for broadcast, the director sent the viewpoint to broadcast using the user interface. The concept of 'grab then broadcast' was developed to allow a preparatory step between selecting the viewpoint and ensuring it was at the correct point to cut to.

Figure 2.7 shows the user interface by which the director could preview viewpoints and select broadcast footage.

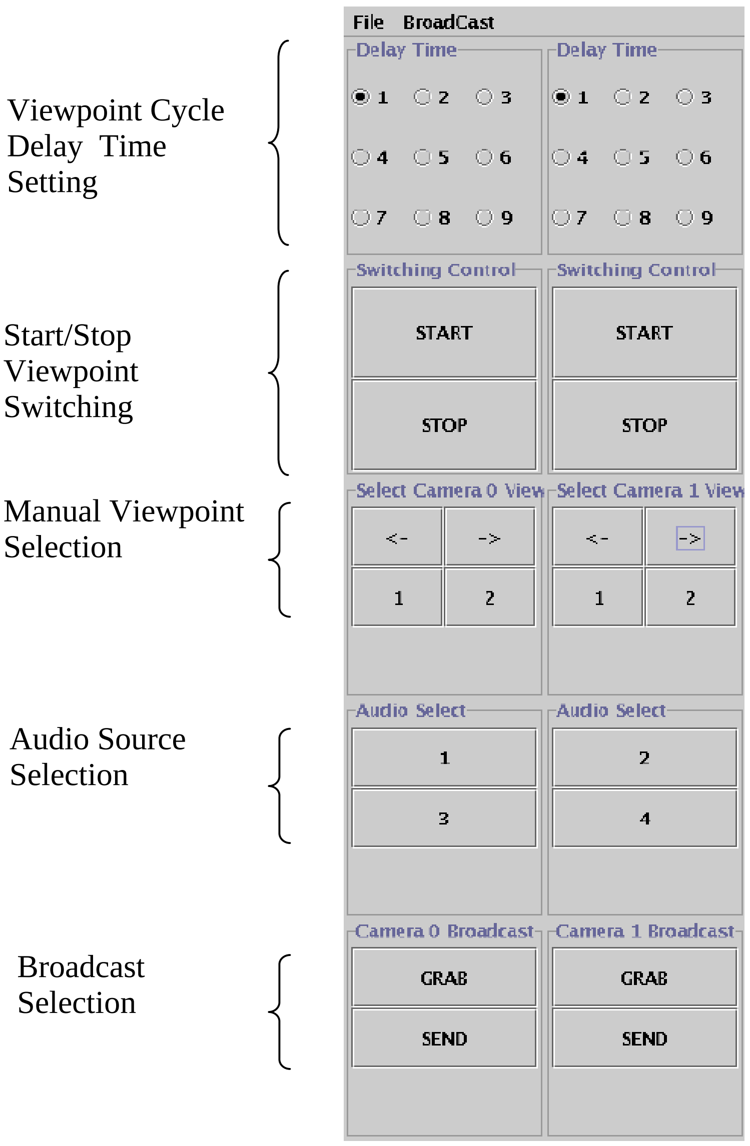


Figure 2.7. The director interface in Avatar Farm.

The director's interface actually consisted of two columns, each containing equivalent controls. The left column was concerned with the views of camera one, the right with camera two. In addition to this interface, the camera view window was always visible. Figure 2.8 presents an example screen image of the director's views during a typical session.

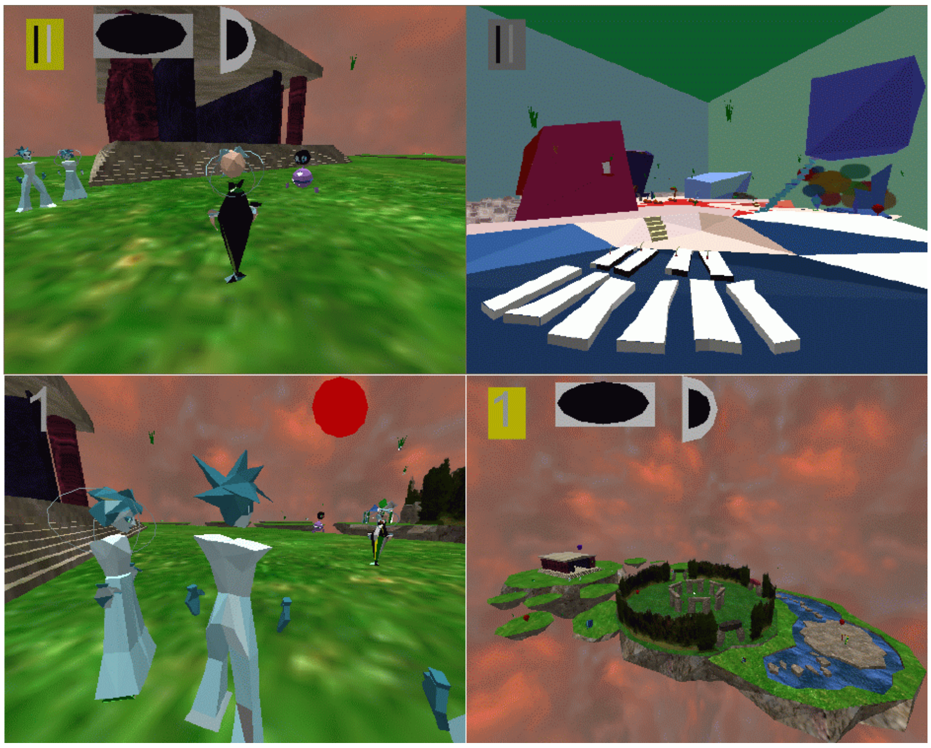


Figure 2.8. The director's views as taken from the two camera operators. The left column is dedicated to views from camera one, the right from camera two. Upper views are affected by the switching controls whilst the lower views are copied from the upper views and are available for broadcast.

The components of the director's interface behaved as follows:

- **Viewpoint Cycle Delay Time Setting**
  The director could select the time (in seconds) taken to switch between the two viewpoints. A range of one to nine seconds was available.
- **Start/Stop Viewpoint Switching.**
  Automatic view switching can be stopped or restarted depending on director preferences.
- **Manual Viewpoint Selection.**
  When automatic switching is disabled, the director can select viewpoints using these buttons. They can be selected explicitly or cycled through using left/right buttons.
- **Audio Source Selection.**
  Audio could be selected from any of the four viewpoints that were visible to the director. The director could then preview audio from any viewpoint as opposed to say, hearing only broadcast audio.
- **Broadcast Selection.**
  Having identified a viewpoint for potential broadcast, the director would 'grab' a viewpoint. This would copy the viewpoint to the lower window. When satisified that the shot was in the correct position, the director would 'send' the viewpoint to the broadcast machine.

2.3.4. Evaluation Issues and Observations

This section details the immediate observations gained during the performance, and from discussions with the director and camera operator.

2.3.4.1 Camera Control

The key difference between the camera interface of *OOTW* and *Avatar Farm* was in the use of multiple viewports. This was originally intended to be tested using four viewpoints, but the slow rendering speed prohibited any more than two (a similar constraint to two was worked with in the work reported by KTH on software vision mixing in Deliverable D4.3/4.4). However, the use of two viewpoints proved effective in enabling tracking of an avatar, whilst cueing a shot from a different perspective. It was especially useful for maintaining static shots of distant views. These could then be selected without the distracting the operator from a close-up of action. Continuous movement also proved useful for automating the process of circling static objects. However, the pace of the event meant that it was difficult to combine continuous movements (such as yaw and distance) using the interface. Although this may have produced interesting motion, the slider based interface made it too cumbersome to perform several movements simultaneously.

The use of a number of tabbed panes for separation of information proved useful due to the sheer number of objects in the four worlds. However, clearer indication of tracking target would have helped. Ideally, more information should have been placed within the viewpoint rather than on the interface. For example, a marker above the target avatar that was only visible to the camera operator. Some confusion was also caused by the subtlety of the camera selection indicator. A much more visible method of indication would have been preferable, such as reduction in size or brightness of the view that was not under control.

In general, the use of multiple viewpoints worked well but the use of alternate interfaces and a greater clarity of status indicators should be explored.

2.3.4.2 Director Interface

The main strength of the director’s interface was in offering multiple viewports without the need for several monitors. This worked to a degree, but the process was mainly frustrated by the high degree of parallel activity that took place during the event. Although the director had a number of views at their disposal, it was difficult to quickly locate areas of interest. Also, the fast pace meant that the interface was often distracting and it would have been useful to separate the controls into a tactile interface or to within the viewpoints (see arguments also to this effect in Deliverable D4.3/4.4 and Chapter 3 of this deliverable). However, the director initially felt comfortable with the interface but it was only during the live performance that it was used in earnest.

The ability to select audio sources was a new concept that was not fully explored. Ultimately, the director concentrated on the broadcast audio whilst being satisfied with visuals from the other viewpoints. Again, some form of dial or more intuitive switching mechanism may have overcome this problem. Chapter 3 of this deliverable also explores some software audio mixing solutions which could offer enhancements over the techniques used here for selecting and pre-listening to sound sources.

Reassuringly the director was enthusiastic about the use of software and accepted new techniques as a means of exploration. In an initial demonstration, the director was interested in having the camera controls in their interface to allow personal control. It was said that the use of software in this way could overcome the physical limitations of both direction and camera work that are inevitable in a physical studio.

In general, the use of multiple viewpoints was received well and future work should concentrate on providing directorial awareness of key activity when such a high degree of parallel activity is taking place. It is precisely this research topic which focuses much of the work reported in the next chapter.

Chapter Three
Production Support Tools for Electronic Arenas: Using Tangible Interfaces for Media Editing

John Bowers, Kai-Mikael Jää-Aro and Sten-Olof Hellström
Royal Institute of Technology (KTH), Stockholm, Sweden

Michael Hoch
ZKM, Karlsruhe, Germany

Greg Whitfield
University of Nottingham, Nottingham, UK

3.1 Introduction

This chapter describes the outcomes of a major strand of work conducted in Workpackage 4 devoted to the support of event management in electronic arenas, culminating in a demonstration of a prototype production support 'suite' – a mixed reality environment intended for use by production personnel concerned with managing events in electronic arenas.

This work has a character that has been strongly influenced by the ethnographic analyses of production work which were extensively carried out in Year 2 of eRENA. Social scientific study of the production of the demonstrators in Year 2's work led to a rich set of suggestions for future requirements of electronic arenas and how to respond to these. The work in Workpackage 4 has been our response to these suggestions. To be more concrete, we have devoted ourselves to developing techniques for enhancing the deployment of virtual cameras in an electronic arena, for managing their relations, and for enabling production staff to search out the action in a potentially mass-participation environment. Early versions of this work were available at the end of Year 2. Here, we report on the refined versions of our applications that we close the project with – together with more detailed accounts of our evaluations of our work in practice. Important and specific to the work in Year 3 has been the extensions of our approach into sound control. In this way, we offer an integrated set of ideas for the interactive control of both visual and sonic media in an electronic arena.

To look ahead, we propose to present production staff with real-time visualisations of participant position, orientation and activity while an event in an electronic arena unfolds. These visualisations can be interacted with so as to deploy virtual cameras or virtual microphones. We propose a number of algorithms for the near-optimal initial deployment of cameras and a visualisation strategy that clearly highlights areas of 'sonic interest'. We propose a number of ideas for algorithmically controlled camera paths and mobile virtual microphones. However, we envisage such algorithmic techniques being used in concert with human manipulation and control. For example, algorithms might determine initial camera deployment but, following this, finer control may be achieved using a camera interface much like that described in Chapter 2 of this deliverable. Indeed, the work reported there is entirely complementary to the ideas here as we envisage complex mass-participation electronic arenas requiring combinations of automatic and manual techniques to facilitate capturing an interesting selection of visual and sonic sources.

Of central importance in all this has been the development of what we call *activity-oriented resource control and deployment*. In many ways, this can be regarded as a particular instantiation of a general concept we would like to offer: *activity-oriented navigation*. Conventional virtual reality systems support *avatar-centred navigation* through the control of the position and orientation of the embodiment of the user. The camera control interface developed for inhabited television applications in eRENA supports *object-centred navigation* so that movements can be made in relationship to entities in the field of view (see Deliverable D7a.1, especially Chapter 3, and this deliverable, Chapter 2). In the current work, we are proposing a further paradigm of activity-oriented navigation whereby deployments in space of production resources can be influenced by activity within it. In the current chapter, we examine specific applications of this navigational paradigm for the deployment and control of virtual cameras and microphones.

To integrate our work with that in Workpackage 6, we have adopted one of interfacing technologies developed there to provide the means by which our production support applications are presented to users. The RoundTable supports the projection of a visual display onto its top surface and the detection and tracking through video analysis of objects placed upon it. All of the applications described in this chapter have been trialed upon the RoundTable as well as in more conventional desktop variants. Using the RoundTable is important to our work as it indicates how mixed reality technologies (here mixing the manipulation of physical objects with projected computer generated displays) can be deployed in support of production activities in an electronic arena. In this way, the mixed reality theme of eRENA is consistently carried through in our technologies for supporting ‘behind the scenes’ personnel.

The structure of this chapter is as follows. In Section 3.2 we discuss our applications for camera deployment and control. The early parts of this section update material first presented in Deliverable 4.3/4.4. The latter parts give more details of our experience implementing our technologies with physical interfaces and evaluating our effort. We also present our work integrating these technologies with the record/replay functionality of the MASSIVE VR system. In Section 3.3 we discuss our extensions of our approach to sound control, giving examples of applications to support sound mixing and also the real-time interactive composition of music. Again implementations of these applications with a tangible interface are highlighted together with their evaluation in the light of user-experience. Section 3.4 concludes this chapter with an overview of what has been achieved, the advantages and limitations of the work we have done, and its general place in the repertoire of technologies developed in Workpackage 4 and in eRENA in general.

3.2. Camera Deployment and Control

There exists a considerable literature in the VR and computer graphics research fields discussing how cameras in a virtual environment can be controlled. In Deliverable 4.3/4.4 we presented a detailed critical review of this research, finding it to be lacking in a number of respects crucial to the support of events in electronic arenas. Let us summarise some of these points as they enable us to lay out requirements for the technologies we have developed to support camera control in a characteristically novel way for electronic arenas.

Real-time interaction. Although real-time interactive VR systems are increasingly commonplace, much existing work has been developed with animation or other non-real-time applications in mind. We regard electronic arenas as typically involving events which essentially accent real-time interaction. This is the case with the most demanding of the inhabited TV and artistic applications we have examined in eRENA.

Mass social participation. The demonstrator work in the eRENA project both on inhabited TV and mixed reality performances critically emphasises the real-time participation by a number (perhaps a very large number) of individuals. In contrast, many existing computer graphical techniques are devoted to computing a single user's view and not all need to make multiple active, mobile participants the subject of shots.

Understanding rules of practice. A number of existing attempts to support the computer graphical rendering of events have implemented various 'rules of cinematography' in composing sequences of shots or juxtaposing multiple visual projections. For example, a number of systems have reified the supposed rules of continuity editing in generating edited sequences of computer graphical material. However, ethnographic work in the project (see Deliverables D4.4/4.5, D6.2, D7a.1, D7b.1) would gainsay the view that TV directors and producers, still less media artists, obey 'rules of cinematography' without deviation. When considering production tools for electronic arenas, we think it is important to avoid building systems around formal reductions of cinematic (or other) practice.

Hybrid interaction methods in a working division of labour. Our ethnographic research has persistently shown how interactive technologies for electronic arenas need to be developed to fit a 'working division of labour' between differently skilled participants to a production, between (say) camera operators and a director, or between a sound technician and a video tracking technician. We believe that technologies that offers varied combinations of manual and automatic (or 'delegated') control are most suitably flexible for complex co-operative work settings. This is not to say that autonomous cameras (and other such automatic processes) have no role. On the contrary, we can see potential for automatically computed sources being available, from time to time, for a director to cut to if this yields material she judges as relevant and interesting. It is hybrids of this sort that we are interested in developing, hybrids where automatic and manual control coexist, and where humans can variably interact with, intervene upon or delegate control to autonomous processes.

Scripted and improvised action. Most of the events we have conducted in electronic arenas have a noticeable element of improvisation in them. In each of the inhabited TV events, participants improvised around an existing framework (a game format or a narrative). In the artistic events involving public participation (e.g. *Desert Rain*), the introduction of the public brings with it a degree of unpredictability in exactly how action will unfold. In many respects, the greatest design challenge for developing camera deployment and control technologies is to design for such improvised situations. If a system can be shown to be feasible in situations with a high degree of real-time unpredictability, then it is reasonable to imagine that they might also be workable when a script exists.

Following and capturing action. A fundamental issue for production and direction personnel in electronic arenas is the depiction *of action.* How best to follow action, capture it, display it to an audience. How to avoid missing it. How to maintain a set of options from camera operators so that a director does not find herself without anything to cut to. In existing work on computational support for camera control, we do not find efforts precisely devoted to this topic. More common concerns are with geometrical problems to do with the composition of shots or the projection of a 3D scene onto a 2D view. Camera animation is commonly conceived of as a virtual physical problem (e.g. computing a path through an environment to avoid bumping into things). These approaches do not directly address the requirement to support finding (social) activity in the environment and depicting it. For these reasons, in what follows in this section, we devote ourselves to exploring techniques that are directly concerned with supporting what we call *activity-oriented resource deployment and control*.

As we shall see, it is a concern to support production personnel in capturing social action in potentially large scale, mass participation electronic arenas which guides the work we present in this section.

3.2.1. Activity-Oriented Camera Deployment and Control

We seek to support event management in electronic arenas by facilitating camera deployment and control. We wish to do this through elaborating techniques which take into account the ongoing activity in the electronic arena, making this available (i) as a resource to guide personnel in their camera deployment decisions and (ii) to inform autonomous camera movement algorithms designed to seek out 'hot spots' of activity. Exactly how we do this will be discussed later in this section. Both uses of activity information require that data sources be found that can serve as adequate heuristics for activity. The next subsections discuss what these data sources might be.

3.2.1.1. Activity Heuristics

We suggest that activity in an electronic arena might be made available in two basic ways.

- **Activity indicators**. By this we refer to traces of participant-activity which are available to whatever system it is that is maintaining the electronic arena. In a shared virtual environment, for example, it would be possible, in principle, to formulate some measures of communicative activity through, e.g. carrying out appropriate computations over keystrokes (for text communication) or audio-bandwidth usage (for audio communication). Equally, in virtual environments where objects are manipulated, some index of activity could in principle be computed on the basis of the prevalence of these interactions. Finally, for embodied activity in a mixed reality electronic arena, it would be in principle possible to use, for example, video analysis techniques such as those discussed in Deliverable D6.4 and D7a.1 to appraise activity and its locus.
- **Awareness-based activity inferences.** A second way to heuristically determine where the action is in an electronic arena is to infer the patterns of collective awareness that exist within the participant-population and use this to infer, in turn, where action of interest is being or is likely to be realised. Let us explain this in more depth through an example. Imagine an electronic arena where performers are acting out an event in the style of promenade theatre, moving through a virtual environment as they perform. These performers and their actions will be the subject of attentiveness from the audience as the audience maintain an awareness towards what the performers are doing. This attentive awareness is likely to be revealed by their positions and orientations around the performers, the directions of the gaze, and the correlated movements they undertake as they follow the

performers. Naturally, in an example like this, knowing where the performers are at any moment may be resource enough to facilitate camera deployment but, in electronic arenas where interesting action might occur at any place and at any time, being able to make inferences about where this might be on the basis of the patterns of attentiveness and awareness among participants could be a viable approach. A number of virtual reality systems which are of potential use in electronic arenas actually implement an awareness model of some sort which could enable awareness information to be captured much like the activity indicators we have just discussed (e.g. the MASSIVE system used in several of the inhabited TV experiments, Greenhalgh and Benford, 1995). When this is not the case, an awareness model could still be superimposed on participant position and orientation data to infer patterns of awareness (indeed, this is the approach in much of our own work).

3.2.1.2. The 'Spatial Model' of Awareness

The so-called 'Spatial Model' of awareness—largely developed in the ESPRIT project COMIC (1992-1995)—is one of the most ambitious attempts to provide multi-user cooperative systems with a notion of awareness which can shape information display to participants as well as their activities with information and interactions with each other (see Benford et al., 1994). As our own work builds upon this approach, we shall describe it in some depth. The Spatial Model supposes that objects (which might represent people, information or other computer artifacts) can be regarded as situated and manipulable in some space. The notion of space is very generally conceived only subject to the constraint that well-defined metrics for measuring position and orientation across a set of dimensions can be found. In principle, any application where objects can be regarded as distributed along dimensions such that their position and orientation can be measurably determined is amenable to analysis in terms of the Spatial Model though, naturally, virtual reality applications give a ready understanding of space in terms of 3D spatial geometries.

The interaction between objects in space is mediated through the relationships obtaining between up to three subspaces: aura, focus and nimbus. It is assumed that an object will carry with it an aura which, when it sufficiently intersects with the aura of another object, will make it possible for interaction between the objects to take place. On this view, an aura intersection is the pre-condition of further interaction. In many applications (our own included, see below), this helps with the management of scale as further awareness analysis need not always be performed. For objects whose aurae intersect, further computations are carried out to determine the awareness levels the objects have of each other. The subspaces of focus and nimbus are intended as representing the spatial extent of an object's 'attention' and its 'presence' respectively. Thus, "if you are an object in space, a simple formulation might be: the more an object is within your focus, the more aware you are of it; the more an object is within your nimbus, the more aware it is of you," and accordingly, "given that interaction has first been enabled through aura collision: The level of awareness that an object A has of object B in medium M is some function of A's focus in M and B's nimbus in M" (Benford et al., 1994).

It is important to note that in the above definition, awareness-levels are defined per medium. Thus, the 'shape' and 'size' of each of the aura, focus and nimbus subspaces can be different, for example, in the visual (graphical) than in the audio-medium. In this way, I may be aware of the sounds made by another object but without being able to see it. Benford et al. (1994, 1996) go on to show how simple instantiations of this model can have a high degree of expressive power, for example enabling one to distinguish between different intuitively familiar 'modes of mutual awareness' on the basis of A's awareness of B and B's awareness of A. However, perhaps the most important point emphasised in this work is the insistence that awareness is a joint-product of how I direct my attention to you (focus) and how you project your presence or activity to me (nimbus).

3.2.1.3. Mapping Activity and Awareness

In an unpublished paper, Sandor and Jää-Aro propose 'activity maps' as representations of virtual environments based on activity heuristics such as those we have discussed (indicators like text/speech input measures, measures of avatar displacement, object manipulation and so forth). An activity map will be some analogic representation of the virtual spaces enabling a user to (for a visually rendered map) see a depiction of activity levels across a virtual terrain. Sandor and Jää-Aro propose that activity maps are computed by summing activity measures at each locus on the map. The number of loci differentiated in the map is the resolution of the map. A low resolution map (for example) might just show activity levels in North-East, North-West, South-East and South-West quadrants. As we have discussed, activity maps might also be inferred from the *operation* of an awareness model such as the Spatial Model (if a system implements it) or from its *application* (if a system doesn't). Separate awareness related maps could be computed for focus, nimbus or combined awareness measures based on joint focus/nimbus functions. A focus map would show the 'hot-spots' where, in general, the population of participants are directing their collective attention. The loci of promenading performers (see our example above) one can imagine would show high collectively summed focus levels. Conversely, a nimbus map would highlight the loci where participants are projecting their presence. Finally, a combined focus/nimbus awareness map could show the summed combinations of focus and nimbus at each locus to give a general impression of the distribution of collective awareness around the electronic arena.

Maps computed in this way would give an overview of activity and awareness in an electronic arena. Our proposal is that it is such maps which can potentially serve as a resource to production personnel in guiding directorial work—in particular camera deployment and (as discussed in the next major section) sound control. The exact visual form that our activity maps take is discussed shortly.

3.2.2. Algorithms

It is possible that readily interpretable activity maps would suffice to enable a director of an event in an electronic arena to, for example, give verbal instructions to a virtual camera operator ("head South-West and find out what's going on") or make simple manual deployments (e.g. to an approximate location in the South-West). However, we wanted to explore some design possibilities which might enable the algorithmic calculation of more optimal initial deployments which could, in turn, be manually refined by a camera operator. Our camera deployment algorithms are all concerned with finding a location and orientation for a camera in relation to a *group* of participants in an electronic arena. Given an identification of a set of group members, our algorithms return co-ordinates and a view vector for a camera so that a shot of the group can be optimally framed subject to certain constraints.

In our applications, the user can select a group in an explicit fashion by, in desktop implementations, the conventional means of drawing a selection region on the on-screen activity map display or, in RoundTable versions, placing an object on the activity map projected onto the table to select a proximal group.

While we imagine camera angles ultimately have to be refined by a human camera operator, we suggest that they may be aided by software heuristics giving rough starting positions that then can be adjusted. We have explored various algorithms (based on, e.g. centres of gravity, bisecting view directions, centres of viewpoint) which each seem to us to give reasonable initial camera deployments (for formal specifications, see Deliverable D4.3/4.4).

We imagine that personnel deploying cameras would make such deployments in the light of inspecting an activity map or some analogous overview, perhaps by explicitly selecting the group themselves. We have also explored a camera type whose behaviour is more closely related to the nature of a computed activity map. We have experimented with an actively *activity-seeking camera*, which we refer to as ’puppycam’ as its behaviour is in many ways redolent of a young puppy always seeking out matters of momentary local interest. Conceptually, the puppycam will follow the gradient of an activity/awareness function as mapped on the basis of heuristically derived activity indicators, awareness measures or whatever. In our experiments so far we have applied an awareness model onto participant orientation and position data to yield an awareness map. Our puppycam will always move in the direction of increasing awareness. The intention is that it will find the area with the highest activity, as given by the heuristic that high awareness levels also correspond to high activity levels. The actual implementation at every time-step samples the awareness function at twelve points on a unit circle surrounding the camera as well as at the camera position itself and then moves the camera to the position which has the highest awareness value, facing in the direction of increasing awareness.

3.2.3. Implementation and Desktop Interfaces

We have developed a prototype implementation of the principles we have described, called SVEA (Sonification and Visualisation for Electronic Arenas). The new sonification components of SVEA (concerned with sound mixing and so forth) are described in the next section. Deliverable D4.3/4.5 from Year 2 described some demonstrations of sonification techniques to render, in sound, various aspects of participant activity. We do not repeat this account here. In this section, we concentrate on how SVEA implements the Spatial Model, how it computes and displays awareness maps, the support for simple mouse-based interaction we have included, and various other usability features, before describing

We place isosceles triangle shaped markers representing the participants of the electronic arena on a 2D projection of space with the colour of these markers displaying a measure of the activity they show. (Typically we project participant location onto the groundplane and ignore the elevation coordinate in composing the 2D projection. This is a tolerable simplification as very commonly in our experience action is around a groundplane in the events we have conducted – even if ‘flying’ is allowed. On this and other issues to do with 2D simplifications of 3D action, see Deliverable 4.3/4.4.). Areas of high activity will thus be conspicuous as large, brightly-coloured areas. While SVEA can use a variety of different kinds of data to give activity measures, we default to applying the awareness model just described. We give a colour to a participant P's marker which is in relation to the sum of awareness that all other participants have of P—the greater this figure, the brighter the colour.

The sharp apex of the triangle is used to point to the participant's location, with orientation being represented so that the triangle can be understood as an arrow pointing in the direction the participant is facing. This has the consequence that a distinctive 'flower' shape can sometimes be seen as a group of participants aggregate and face inwards.

In the desktop version of SVEA, the 2D display can be magnified up to 16 times by selection of a menu option to zoom in on a selected group of interest with the centre of that group being the centre of the zoom. Zooming rescales the relative separation of markers in screen distances but not the size of the triangles themselves. Zooming, therefore, can clarify the relative positions of participants that are so close to each other as to appear overlapping when in a 'wide angle' view. A different zooming technique is used in the RoundTable implementation of SVEA (see below).

3.2.4. Camera Deployment and Real-Time Interaction with Activity Maps in SVEA

In addition to the markers depicting participants, a set of cameras, representing possible viewpoints in the environment, is displayed. By default, the cameras will be activity-seeking puppycams, i.e., they will move towards areas of high activity. Cameras can be selected, with the intended semantics that the view from that camera is the transmission (TX) view. SVEA can be connected to a DIVE visualiser (see http://www.sics.se/dive/) that will enable a 3D visualisation of the electronic arena to be obtained from the selected camera.

In the desktop version of SVEA, algorithmic camera deployment is actioned by dragging the mouse over the display to select a set of markers. As soon as the mouse is released a camera is deployed to the algorithmically computed optimal location for that group according to whichever algorithm is set as a preference. In this, algorithmically enhanced way, camera deployment can be efficiently actioned by a single interface gesture.

In principle, SVEA can take a real-time stream of data concerning participant position and orientation and visualise inferred awareness levels. In our experimentation to date with SVEA, we have worked with data logs from actual inhabited TV events (e.g. the *Heaven and Hell - Live* data) or with similarly formatted logs from the simulations conducted in Workpackage 5 (see Deliverable D5.3). Jason Morphett at BT Labs has kindly provided these data logs that we read into SVEA via an autonomous thread to simulate the real-time arrival of data at a socket. As an alternative way to control data log input, a 'time slider' is provided at the base of the SVEA display to move backwards and forwards through the data. Thus, we have equipped SVEA in its desktop version with basic tools to support the off-line browsing of activity in an electronic arena as well as real-time action on the basis of it.

Figure 3.1: SVEA - a pre-recorded data file has been read in and is visualised.

SVEA has been implemented in Java 1.2 and Swing for maximal portability, and while this promise has not been fulfilled in all respects, we have run it with acceptable performance under Solaris 5.6, Irix 6.5, Windows 95, 98 and NT.

3.2.5. Implementation of SVEA on the RoundTable

As described in Deliverable D6.4, the RoundTable lets users manipulate a computer display by moving physical blocks on a table surface. As eRENA is concerned with mixed realities, it seems only self-consistent to explore techniques of interaction with computer-displayed environments through the use of physical objects for the behind-the-scenes production work associated with an event alongside exploring mixed reality experiences for public participants to the events themselves.

The RoundTable is a prototype tangible interface consisting of a translucent surface, onto which computer graphics can be projected from below. A video camera with an infrared filter is suspended above the table. Video analysis software can then pick up the identity, position and orientation of plastic blocks covered in IR-reflective foil and forward this information to other processes, commonly affecting the projected display. (The hardware set-up and associated software is described in more depth in Deliverable D6.4.)

Re-implementing SVEA on the RoundTable enables group selection, camera deployment and selection, and display zooming through the manipulation of physical icons (phicons) on the table-top projection surface. Amongst other matters, this facilitates some of those aspects of the interaction with SVEA, which—in its GUI realisation described above—are supported by somewhat cumbersome menu selection operations. Let us give a few details about how this is achieved.

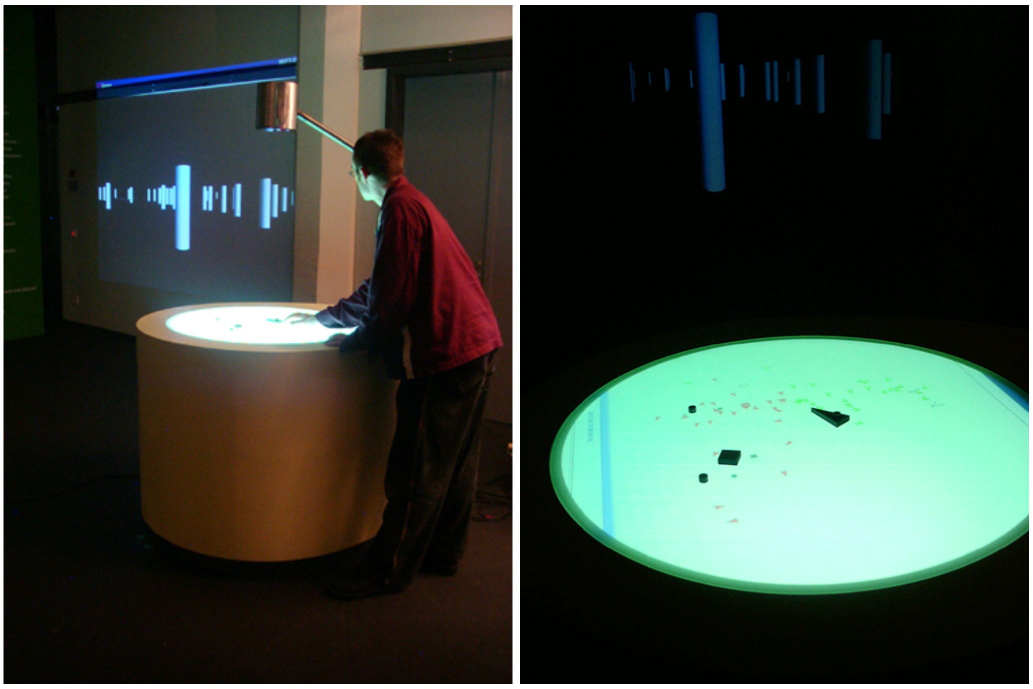


Figure 3.2: (left) Camera control environment with 3D camera view, (right) SVEA visualization and physical icons for manipulation.

We currently use four different physical objects for interactional purposes (see Figure 3.3). The isosceles triangular shaped object is a *camera* phicon and is used to deploy virtual cameras in the electronic arena. As in the sense given to such shapes in the SVEA visualisations (see above), the triangle is to be interpreted as an arrow-like pointer in the direction of the camera's view. Once a new camera phicon is detected by the RoundTable's vision system, RT (see Deliverable D6.4), a virtual camera is assigned to the location indicated and pointing in the direction suggested. The view onto the electronic arena from this virtual camera can be selected for transmission (TX) by placing a small round *camera selector* phicon into the non-reflective hole in the middle of the triangle.

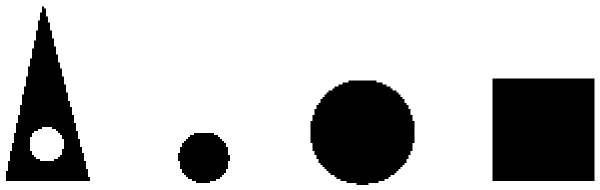


Figure 3.3: Shapes of detected phicons: camera, camera selector, probe, and zoom (showing their relative sizes—the camera phicon being approximately 8cm long)

The *probe* phicon is used to select a group of avatars in the projected visualisation. Rather than attempt—using phicons—to replicate the mouse gesture of clicking and dragging used to select groups in the screen-based SVEA, we decided to exploit and extend the awareness model underlying the visualisation to enable context-sensitive selections to be made. Let us explain this in more depth. Imagine an object in the electronic arena at the position corresponding to a probe placed on the projected visualisation. Assign focus and nimbus (see above) to this object just like other objects and avatars in the electronic environment. Just as we computed the awareness participants in the environment can have of each other (remember this is what is signified by the basic shading of the triangles representing participants), it is possible to determine the awareness the probe-object would have of avatars in proximity to it. The avatars with an awareness level above a given threshold can be returned as a selected group. In this way, the probe can be used to select the group of avatars that the probe would be aware of from the spot where it is deployed. When a new probe phicon is detected by RT, the group of triangular avatars identified in this way is highlighted by darkening their colour.

We describe this use of the probe phicon as being ’context-sensitive’ because exactly which avatars it selects, how many and in which configuration, is dependent upon the avatars’ orientations and proximity with respect to the probe. If the avatars are sparsely distributed in the environs of the probe, maybe only a few will be selected. If the avatars are more densely packed where the probe is deployed, very many may be selected. However, in both cases, the same basic gesture—placing a probe phicon upon the table-top projected visualisation—will be used to make the selection.

Although different methods are described above for explicit group selection at the interface (i.e. mouse-dragging a selection-box), a virtual camera is algorithmically assigned to a group in the same fashion as soon as the group is identified. The possible algorithms for computing location and angle of view of this camera remain as described before.

Finally, the *zoom* phicon allows one to zoom into the visualisation in a similar context-sensitive fashion. The group of avatars is determined corresponding to those which an object placed in the electronic arena at the location corresponding to the zoom phicon would be aware of. The display is rescaled so that it shows this group plus an area around them. The extent of the extra surrounding area can be set as a preference (we have worked with displays which zoom to an area approximately twice the 'width' of the group, with the centre of gravity of the group at the centre of the zoomed display). This method also demonstrates our principle of introducing context-sensitivity into the interpretation of the deployment of phicons at a physical interface. The level of zoom of the visualisation is dependent on the number of avatars present in the area where the zoom phicon is placed. As described before, when we zoom the display, we do not scale the triangles representing participant-avatars. Deploying the zoom phicon enables the user to 'separate out' densely populated areas where otherwise many avatars might be shown on top of each other. Once the relations between avatars in an electronic arena has been clarified by exploiting the zoom feature, camera phicons can then be positioned to get more appropriate views than would be possible with a uniform unzoomable display.

In the RoundTable production environment we have been discussing, we have had to make important design decisions over issues to do with the relationship between inserting a new camera phicon into RT’s field of view and the 'lifecycle' of a corresponding virtual camera. Does deploying a phicon create a new camera at the designated location? Or does it merely redeploy an available camera? Are there as many camera phicons as virtual cameras (and no more)? Or can virtual cameras be created without upper limit? Does the removal of a camera phicon cause the corresponding virtual camera to pass out of existence? Or does it cause the virtual camera to return to some default behaviour?

Ultimately, we feel that such questions have to be answered with respect to particular applications. A priori, one can argue either way on a number of these issues. It could be that some events in an electronic arena involve action on such a mass distributed scale that it would be unreasonable to set an upper limit on camera number but necessary to extensively use autonomous cameras. It could be that more intimate events would be conducted with a smaller number of cameras and much manual control. It could be that a production crew would find their work very hard to accomplish practically if they could not refer in traditional fashion to "Camera 1... Camera 2..." and so forth, and assign cameras to a fixed number of roles. Indeed, for aesthetic reasons, a director, designer, producer or artist may prefer even a single camera and make no cuts (perhaps in the name of a kind of 'fly on the wall' documentary direction style for electronic arenas!). Even in this extreme case, our RoundTable production suite might be of use for the clues it would give for where the action is.

The fact that one can argue either way on these issues is further fuelled by the fact that *from a viewer's perspective* editing between multiple virtual cameras can be perceptually identical to following a single virtual camera which is capable of teleportation and changes in behaviour. This observation testifies to our point that whether one works with a system that has multiple virtual cameras, or just one, and what relationship the gestures of production staff have to the lifecycle of a camera, are largely matters to do with how best to facilitate the practical work of production of events for electronic arenas.

Our prototype, then, arbitrarily restricts the number of cameras to a user-preferred limit but does so without prejudice to alternative possibilities. Within this set of cameras, the default behaviour is an autonomous one. That is, if a camera is not deployed through activity at the round table, it maintains behaviour which is entirely algorithmically determined. This default behaviour is to rove the space, following the gradient of increasing awareness, while avoiding other cameras (i.e. the puppycam behaviour described earlier). The introduction of a camera phicon will 'claim' the first available camera. Which camera is 'first' is determined in numerical order—thus retaining a sense of 'Camera 1', 'Camera 2' and so forth which could be explicitly referred to by users to individuate cameras. The assignment will pass over already assigned cameras. Amongst other desirable features, this method has the consequence that cameras already selected for TX will not be suddenly and mistakenly cut to another location. The removal of a camera phicon from the RoundTable will 'deassign' any associated virtual camera and return it to its default behaviour.

Let us present some images from the RoundTable showing some closer detail.

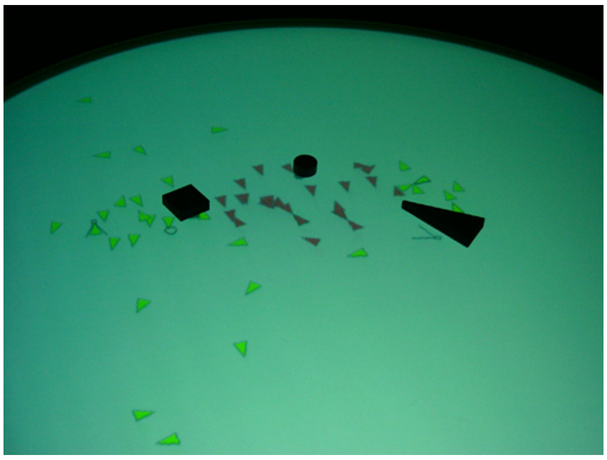


Figure 3.4: SVEA visualization with deployed zoom, probe, and camera phicon (from left to right)

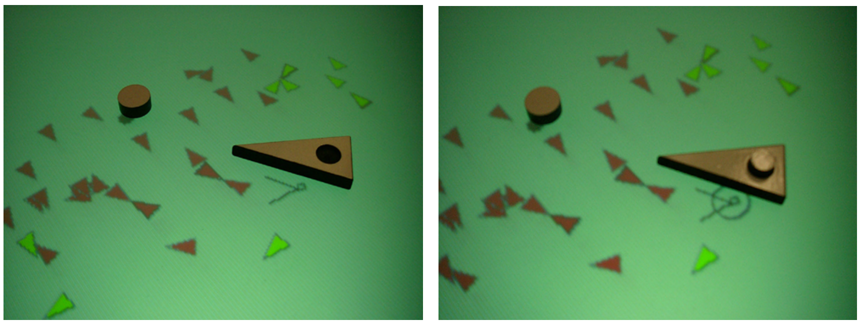


Figure 3.5: Camera selection by placing a small phicon in the hole of the camera phicon: (left) deselected camera, rendered without circle, (right) selected camera, rendered with circle.

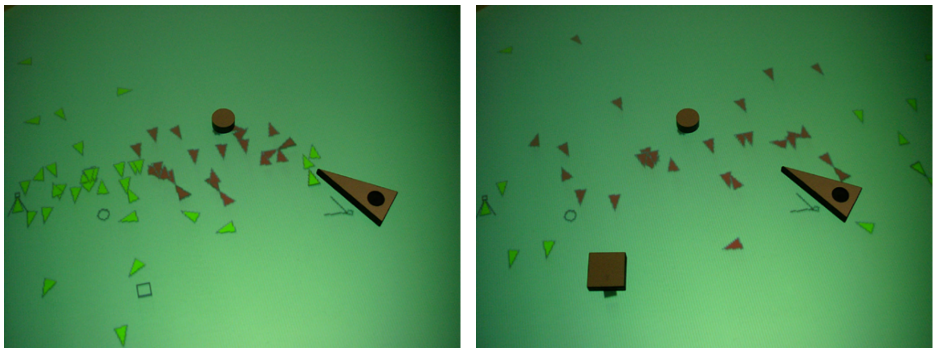


Figure 3.6: Zooming in: (left) Normal view in the visualization, (right) added zoom phicon changes the scale of the SVEA visualization.

3.2.6. Evaluating Our Experience with SVEA on the RoundTable

We have presented the RoundTable to a number of persons independent of the development group so as to evaluate it as a means for presenting production support tools. Users of the RoundTable have included a number of media professionals as well as students and visitors to our lab. Most importantly, several production personnel who were practically involved in the inhabited TV events in eRENA have been presented with the table and the concepts it embodies. These evaluation sessions have been informal and discussive, rather than based around controlled experimentation. This is appropriate given the kind of technology it is and the prototype standing of the applications we have developed for it. The RoundTable can implement any variety of applications and much of a general user's sense of its usability may hinge on particular application design details. On the other hand we are interested in evaluating our applications as well – and each of these can also be realised in desktop environments. Given this interplay between application and interfacing technique, we though it appropriate to run informal conversational evaluation sessions so as to most readily determine the sense of user's comments. What follows is based on our analyses of these sessions.

The main advantages with using the RoundTable that have repeatedly come to light are twofold:

1) Giving both a position and an orientation using a physical block is noticeably faster than using a mouse, since mouse-based positioning requires one operation for indicating the position and a second for indicating the direction – this latter requires dragging and thus takes longer time than just placing an item. (The mouse operations can be combined into

one physical movement, so that the position at mouse-down is taken to be the origin of the camera and the position at mouse-up to indicate the end of a direction vector. However, it is very hard to do this with the same speed and precision as with a phicon.)

2) Several people can stand around the table and easily do adjustments or new placements of cameras, thus supporting mutual awareness among the production personnel as they engage with a shared artefact. Conventional screen-based, desktop solutions are more cumbersome in this regard.

Some of our specific design decisions have raised some interesting issues with users.

Our introduction of a phicon controlled zooming functionality has some important consequences. After zooming, the projection of the virtual space will be transformed such that the position of a camera phicon may no longer correspond to the virtual camera previously associated with it. There are a number of design options here. Zooming could (i) forcibly deassign virtual cameras from phicons or (ii) the phicon and the associated camera become disjoint in their spatial location on the table or (iii) virtual cameras 'snap' to the new relative locations occupied by their phicons. In discussion with users, each of these options have been revealed as having benefits and drawbacks as general solutions, so again ultimately the choice should be made on application-specific grounds. Our choice in our current demonstrator is (i) that has the consequence that cameras have to be reassigned by getting RT to detect a fresh appearance of a phicon. This can be accomplished most simply by covering the phicon to be reassigned with the hand for a brief moment and then revealing it. Some users discovered this for themselves and others had little trouble with the technique once it was suggested to them. (Indeed, the technique of covering a phicon to simulate its removal was spontaneously used by our users in other contexts too, e.g. when the effects of a momentary removal of a phicon were to be investigated, an economical gesture is to cover and then expose the phicon to the camera.)

However, this solution can lead to misunderstandings if users think that triangular phicons always represent the presence of a camera in the electronic arena. For our method of camera allocation, this would be an inappropriate 'user-model'. Rather, the camera phicons should be regarded as *tools with which to deploy virtual cameras*, not representations of those cameras themselves. It is clear, though, that this is a less natural model than expecting a one to one correspondence between virtual camera and camera phicon.

It is important to observe that similar issues arise for most attempts to physically interface to the virtual and are not specific to our application or our particular design decisions. Only in the extreme case of a completely strict coupling between physical activity and consequences in the virtual world (*and vice versa*) would it be possible to think that a physical object could non-problematically represent a virtual one. As soon as the coupling is relaxed the relationship between the physical and the virtual has to be *achieved in users' practical understandings* rather than technically mandated (see Bowers, O'Brien and Pycock, 1996). In a sense, this argument presents a limit case on the use of tangible mixed reality interfaces. When using simple 'inert' physical objects like blocks, many simple user-interface operations will cause problems with the significance of the physical objects. Rescaling and scrolling a display will make the objects disjoint from anything they are to represent. In the absence of remotely controlled motorised blocks (which may be conceivable in some applications), the situation can only be addressed either by disallowing such operations or trying to inculcate a user model that is tolerant of such discrepancies.

In the case of a production tool intended to be shared by personnel, with individuals possibly attending to different parts of the display at the same, there may be good arguments for disallowing operations such as zooming and scrolling. Changing the viewpoint in the display places fairly strict restrictions on the type of collaboration between production personnel as any zoom or scroll would potentially move an individual’s work area out of sight. We conclude therefore that our concept of ‘context sensitive zooming’ supported by a phicon, while of some novelty (we know of no other work which has such an elegant, single phicon solution to such a fundamental interaction issue in tangible interfaces), is probably not of great practical utility in our target domain (production work for electronic arenas). In our domain, on the occasions when zooming etc are required, conventional desktop solutions seem more approriate. Accordingly, our most recent investigations of the RoundTable do not exploit our zooming techniques and treat the relationship between phicon and virtual object (e.g. virtual microphone) in a much more straightforward manner.

Presenting SVEA to users has suggested to us that we should revise our visualization strategies. While the initially adopted coding of the participants’ mutual awareness by colouring the insides of the avatars with a colour increasing in brightness in proportion with increasing awareness has worked fairly well, we consider instead colouring the entire focus space of the participants might give a better feel for what the awareness space around the participants is like. A technique involving the superimposition of partially transparent coloured discs has been effectively used in our sound control work with the RoundTable.

Let us close with some specific remarks about the practical viability of tools like SVEA and means of delivery of applications like the RoundTable. To put the question bluntly, could media professionals envisage using such technologies? Much depends upon the degree to which one regards events in electronic arenas as requiring new practices over familiar TV, film, theatre and performance art techniques, or whether one should merely incrementally develop existing practices and technologies. One of the directors of eRENA’s inhabited TV events envisaged problems in using a device like the RoundTable because it suggested to her that she would have to disengage with inspecting view monitors (both of TX and sources) in order to use it. To be sure, in conventional TV practice, a director will be continually attending to her sources and to TX with, if she is also doing vision mixing, fingers poised on pre-programmed buttons to bring about transitions between sources. SVEA on the RoundTable is a very different setting from this. In our current design, the selection of a camera requires the placing of a small peg into a camera phicon. This is not a faster gesture than pressing a button ones finger is already poised over. Equally, to deliberate on camera deployment one has to look at a display on the table top – and this looking may well momentarily take actual source monitors out of sight.

For our TV director, these arguments were not destructive of SVEA or the RoundTable, though. They merely pointed to a tension between our designs and standard TV directorial practice. A technology like ours could still be used by non-directors. Indeed, typically in a large-scale distributed live real-world event, a director will be assisted by several assistants who submix or pre-select views for her. A RoundTable production suite could well have a specific role in browsing and homing in on action in electronic setting analogous to that one. Selecting a camera with the peg would not necessarily select TX but would sub-edit a shot for directorial consideration. This would be another way in which the technologies of this chapter relate in a complementary fashion to those developed for *Avatar Farm* (see Chapter 2), which allowed the director to retain her engagement with views and with minimal manual reaching to action transitions. Furthermore, many of the media professionals we consulted were happy to imagine media productions quite different from orthodox TV and so forth, which the RoundTable and applications like SVEA could fit in with.

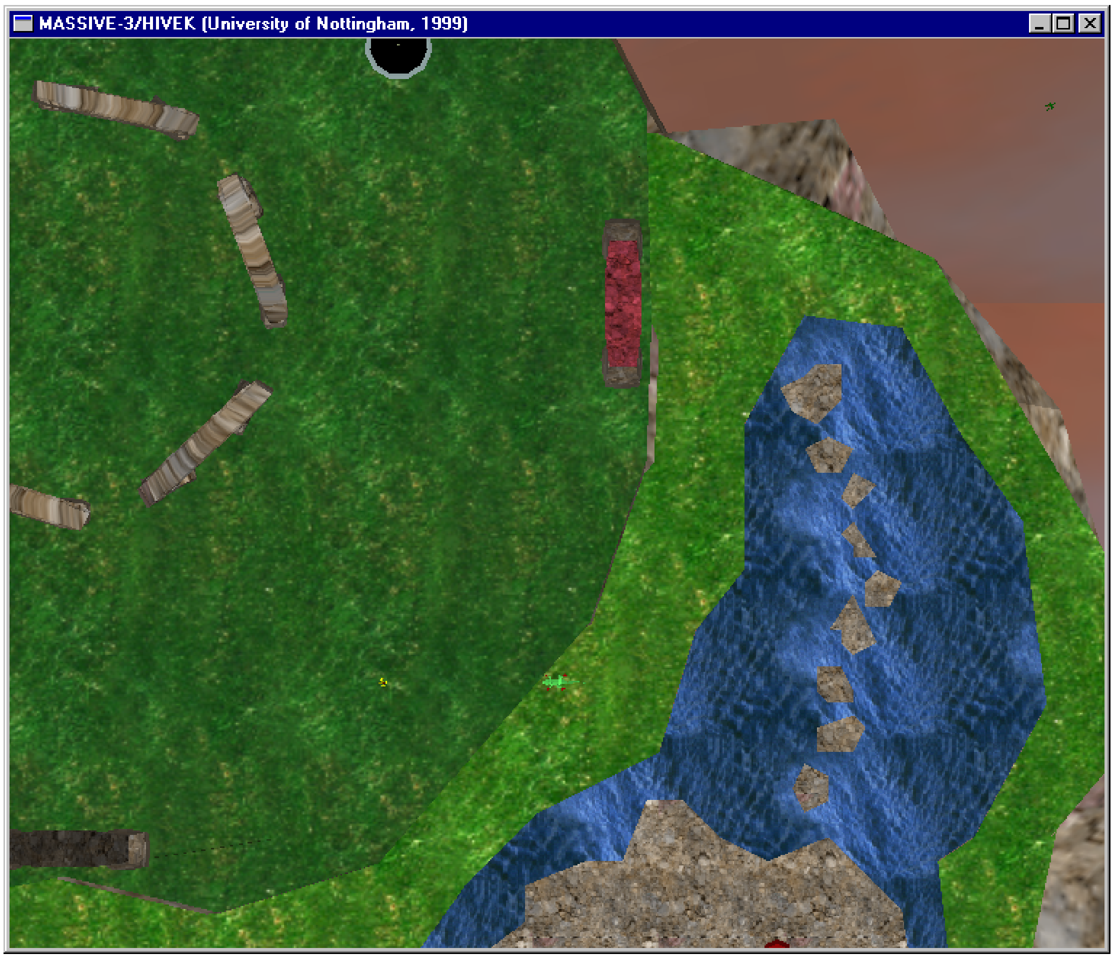


Figure 3.7: A map view on one of the virtual environments from Avatar Farm.

3.2.7.MapView

In parallel with KTH work on SVEA, Nottingham has developed MapView, a Java application which displays a plan view (an orthographic projection from above) of any MASSIVE world and permits the replay of any recorded activity within the environment to be shown (see http://www.netcomuk.co.uk/~gregw/research/index.htm).

In contrast with the simple visualisation strategy in SVEA, MapView displays the full graphics of the world - textures, set props etc. This makes for a more interesting view and also can aid the prediction of where actors are likely to move, since paths and walls are visible in the view.

KTH has added RoundTable interaction functionality to MapView in a manner similar to SVEA. Based on our experiences from SVEA the interface has been considerably simplified. We do not allow scrolling or zooming (even though this is possible in the mouse interface to MapView). We do not utilise nor compute participant awareness information and thus do not use automatic cameras (though these are functionalities that could be added). Instead we track a single phicon, which is considered to be 'on' at all times, and use its position to determine the placement of camera. The view from this camera is displayed as a perspective rendering on another workstation.

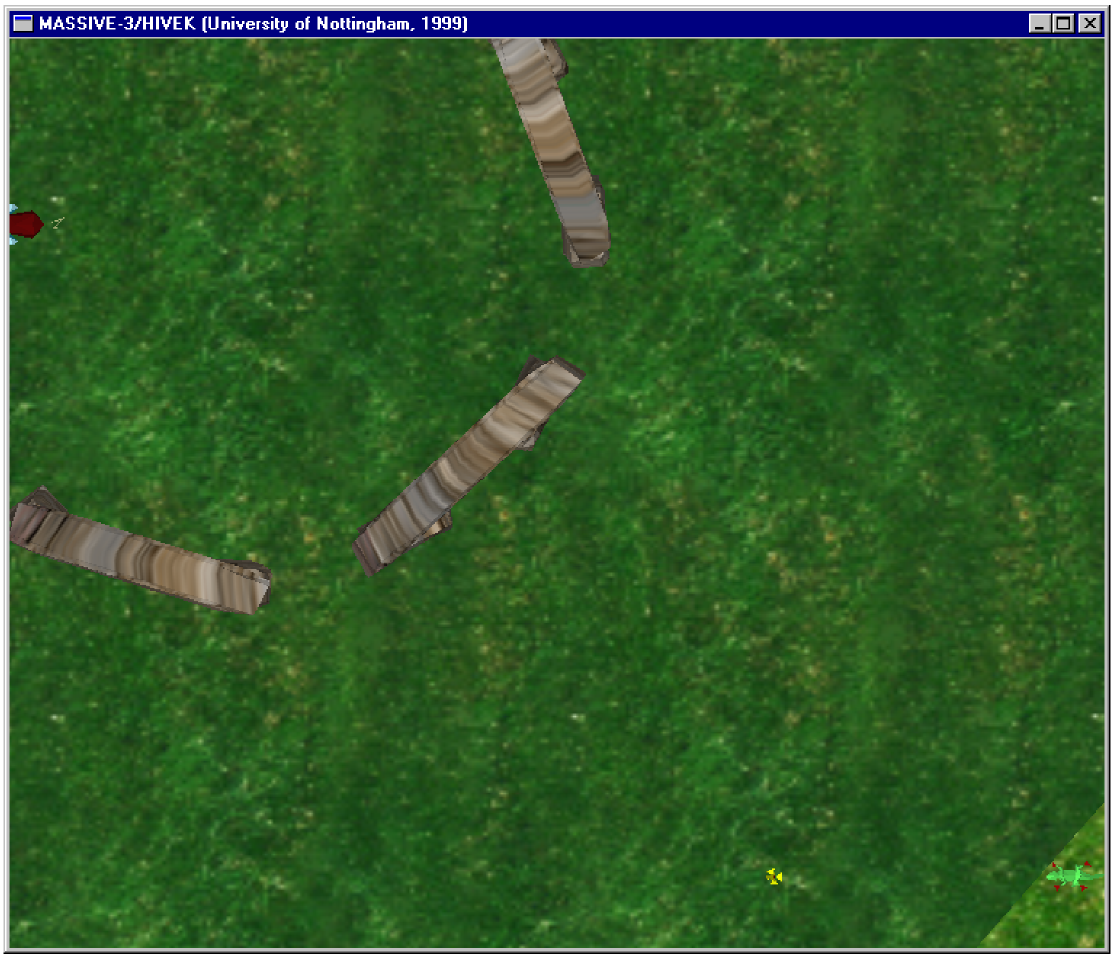


Figure 3.8: A closer view of the top left corner of the previous figure. The small yellow forms to the bottom right are overhead views of avatars.

3.2.8 Evaluating our experience with MapView implemented on the Round Table

At a technical level, this work demonstrates the integration of two sets of important technologies in eRENA: the record/replay functionality of MASSIVE (see Chapter 2) and the RoundTable tangible interface. We have not had time in the final year of eRENA to go much beyond this technical demonstration. That is, we have not yet constructed a detailed application combining these two technologies in a way which independent users could sensibly pass judgement on. We do, nevertheless, have some pointers to important evaluation issues.

Naturally, MapView enables geographical and architectural cues to be available to inform production decisions. A MapView background to a SVEA activity visualisation looks promising. However, our experience suggests that MapView would need to be used with sparser levels of detail to make such a combination workable. There are several issues here. It is sometimes a little unclear from overhead what an object is - especially if has been designed with viewing from the groundplane in mind. Representations that depart from strict orthographic projection of the geometry and the textures may need to be considered. Avatars especially are hard to see (see Figure 3.8) and this is somewhat ironic as it is their positions in relationship to the scenery that we might particularly wish to be informed about. Although MapView gives architectural cues to understanding the action, at the same time, roofs will obscure avatars beneath them. Computationally, it is however not entirely trivial to excise objects that potentially may obscure avatars.

It seems then that more work needs to be done to make such overviews useful in enabling production work. Interestingly, much will be gained from making them *less* veridical and more schematic and, perhaps, having a different sense of scale for different objects (e.g. showing avatars larger than ‘life sized’). Nevertheless, the possibility that an abstract representation (like those in SVEA) might be used in tandem with a more visually detailed one, richer in geographical and architectural detail (like those generated by MapView), remains a promising line of future development.

3.3. Sound Control

We now turn to our sound mixing and related applications developed for use on the RoundTable as production tools for events in an electronic arena. Two of these have been developed in prototype form:

- the *Virtual Sound Mixer* application at KTH
- the *Small Fish* application at ZKM as a preliminary investigations of an *Interactive Compositional Machines.*

3.3.1. The Virtual Sound Mixer

As a complement to our explorations of the RoundTable as a means for supporting camera deployment in a virtual environment, we have also investigated techniques for supporting sound mixing using this method of physical interaction in reference to a visualisation.

Existing explorations of sound for virtual environments have tended to fall into two classes. First there are those which concentrate on techniques for 3D sound localisation using methods such as binaural sound synthesis employing head-related transfer functions (HRTFs) and the like.

The aim here is typically to give an accurate as is possible simulation of the aural experience of a listener in a real-world environment. The cues which listeners use are taken account of in the sound spatialisation algorithms along with, in the case of HRTFs, the use of functions based on measurements taken from real listeners. Begault (1994) presents a review of the most well-known techniques. To increase the sense of a sound being diffused in a determinate environment reverberation techniques such as convolution may also be used (see the discussion in Deliverable D7b.1-2).

A second line of work is much simpler conceptually and technically and this aims to give an approximate listener-relative spatialisation of a sound source through a small number of very simple cues, e.g. panning across a stereo sound image. This is the technique in many established VR systems. MASSIVE, for example, computes, for each listening position, an amplitude and a left-right location as a function of the distance and angle between the source and the position. (For further discussion of sound support in MASSIVE, see Deliverable D7b.1-2.)

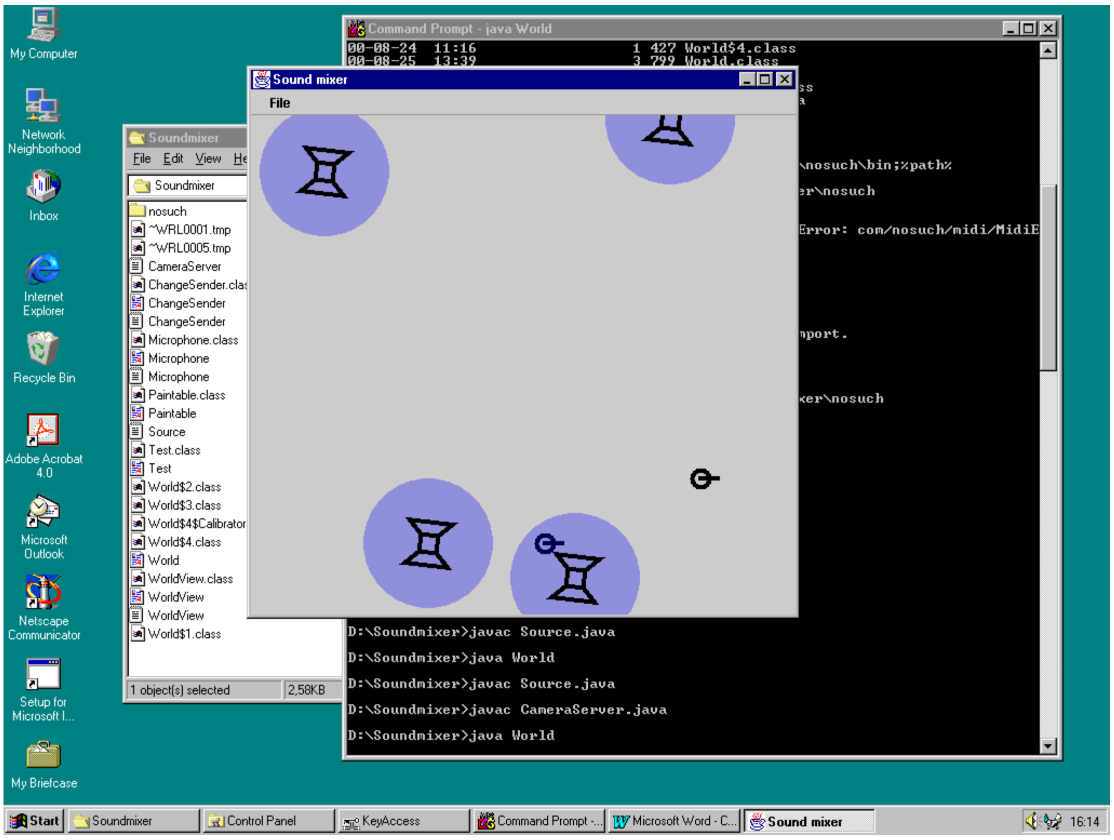
Such techniques have a number of features worth pulling out. The former line of research, in its aim for perceptual 'correctness', commonly leads to computationally expensive solutions. Binaural sound synthesis and convolution are notoriously intensive techniques and, without data reduction, are barely real-time on commonly available computing machinery. In contrast, simple panning techniques, while eminently implementable for real-time operation, give a very gross perceptual impression of sound in space, hardly enough to give a listener a sense of 'presence' in a sound world. In MASSIVE as in many VR systems, the sound server can only handle mono input, for example as might be associated with a microphone placed close to a person speaking. In this way, any spatial cues which one might be able to gain from a stereo, two channel (or more channels for that matter) input cannot be capitalised upon.

Furthermore, most existing spatialisation techniques in VR research are based on the goal of giving a 'sonic rendering' at a particular locus corresponding to that of an avatar's position. To use the terminology of Cohen (2000), they are concerned with 'egocentric' sonic projections. As we have argued in a number of places (e.g. Deliverable D4.3/4.4), much of the interest in research on electronic arenas lies in multi-participant environments where an experience can be, in some sense, 'staged' – a staging requiring an 'exocentric' sonic projection (Cohen, 2000). That is, one does not necessarily have to produce simulations of the listening experience of a listener present in an environment so much as mixes which are appropriate given how the action is staged and designed to be experienced. When listening to a rock band concert, one does not hear a mix designed to simulate the experience of anyone on stage (or anywhere else for that matter). Rather, the mix is designed to be appropriate to the conventions of rock music and technically adequate to the characteristics of the exact venue. Our research in inhabited TV clearly exemplifies areas where 'exocentric' graphical projections are often required, and indeed, means for editing between them. A TV camera's viewpoint does not necessarily simulate the visual experience of any particular participant, nor do the virtual cameras we have developed in eRENA. A similar logic applies to sound mixing for electronic arenas. We need to be able to compute sound mixes that are right for the aesthetic, dramatic or whatever aims of the production and be able to cut between them. It is our belief that mixes which give *some* impression of sounds being spatially situated or moving in relation to each other is appropriate for many tasks – without the need for the perceptual accuracy that more simulation-oriented techniques require.

Cohen (2000) presents a general framework and some novel terminology that is a useful contribution towards what is required. He speaks of sound ‘sources’ (a logic sound emitter) and ‘sinks’ (a virtual listener) and generalises the functions found on most mixer-desks of ‘mute’ and ‘solo’ to audio ‘include’ and ‘exclude’ to (respectively) enable and disable sinks in a mix. He notes that a single individual user might be represented by multiple sinks across multiple environments and switch between them for selective attendance. One may also be represented by multiple sinks in a single environment for ‘selective multipresence’ (i.e. by analogy with switching points of view, one could switch listening positions). Cohen (2000) presents some simple demonstrations of these concepts in action applied to musical and conferencing applications.

The most important conceptual move made by Cohen is how he disassociates the concepts of user (or avatar) from the concept of listening position (or sink). The relationship of users to sinks can be many-many. Indeed, there can be sinks which conceptually have the status of ‘virtual microphones’ – listening positions without any (or any fixed) association with a user.

We have extended these ideas in our designs for a virtual sound mixer for the RoundTable as a demonstration of an innovative production tool for electronic arenas. Just as SVEA was concerned with visualising areas of potential interest, we visualise in the Virtual Sound Mixer (VSM) areas of potential sonic interest. Also, just as SVEA was concerned with the deployment of virtual cameras and the selection between them, so is VSM concerned with the positioning of virtual microphones (we call them ‘vmikes’) and the cutting or crossfading from one to another. A microphone can be selected and the audio mix at that spot computed – together with a spatial rendering of the sound – using efficient real-time mixing algorithms. Let us give some more details.



```
00-08-24  11:16                    1 427 World$4.class
00-08-25  13:39                    3 799 World.class
ss
a
\nosuch\bin;%path%
er\nosuch
Error: com/nosuch/midi/MidiE
nport.
ker\nosuch

D:\Soundmixer>javac Source.java

D:\Soundmixer>java World

D:\Soundmixer>javac Source.java

D:\Soundmixer>javac CameraServer.java

D:\Soundmixer>java World
```

Figure 3.9: Four mobile audio sources and two mobile virtual microphones in a sonic space. For clarity, we show a desktop screen capture. The top window would customarily be projected onto the display surface of the RoundTable.

3.3.1.1. Visualising Sonic Activity

Just as there are many different ways in which we could have visualised activity in SVEA, different possibilities present themselves for visualising sonic activity here. For example, we could show a small icon to signify the position of each sound source and colour each according to the momentary (or smoothed over time) intensity of the sound it is producing. Other possibilities suggest themselves when we associates concepts like focus and nimbus (see the discussion of The Spatial Model above) with sound sources and/or vmikes. For example, imagine that each sound source has an 'audio nimbus' associated with it which specifies the area within which it can be heard. Areas of audio nimbus intersection would then indicate areas where more than one sound source can be heard. Particularly interesting is to visualise intersections as, not only do they change in an interesting fashion with mobile sound sources, but they also give information that it is hard to get from visualising the virtual positions of sources alone. Naturally, if one wishes to deploy a vmike to capture the sound from a single source, one can place it close to that source. However, if one wanted to pick up multiple sources simultaneously, the correct single position might be hard to judge from seeing positions alone especially if sounds are directed in their projection within the virtual environment.

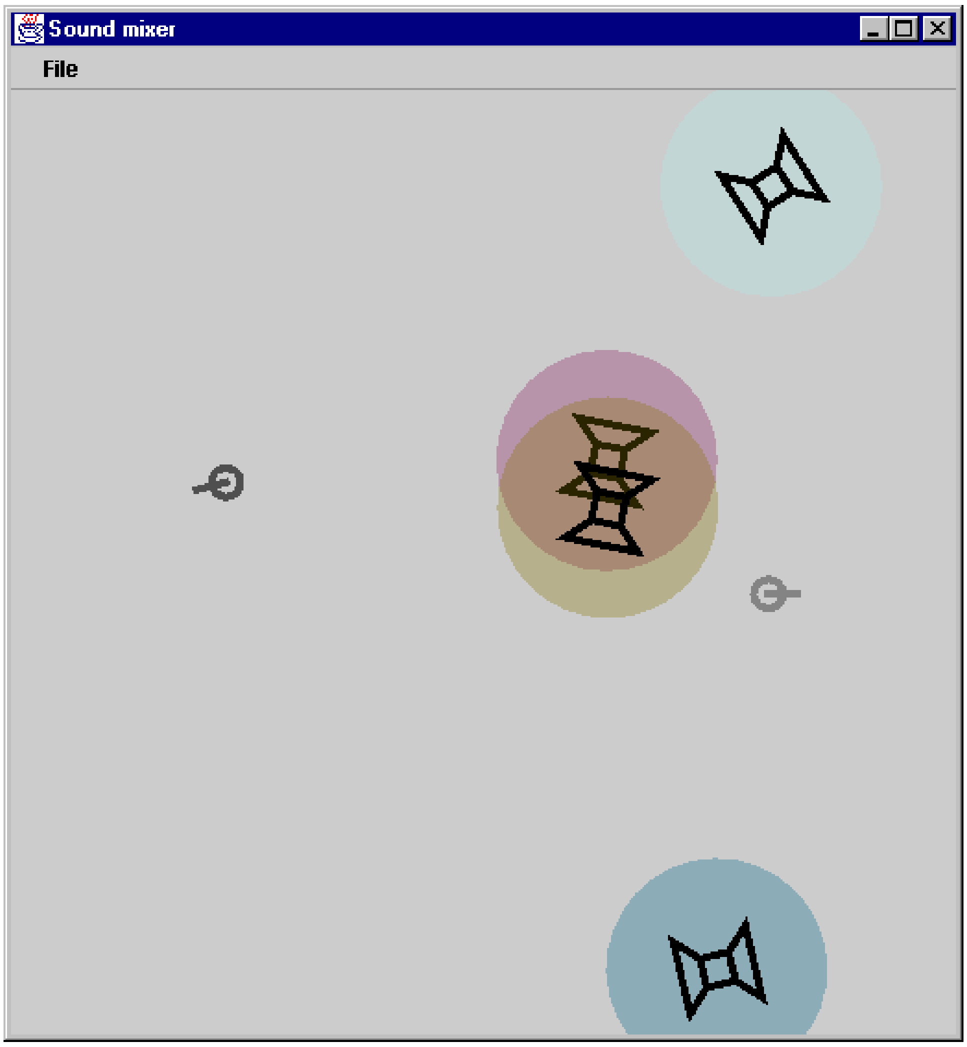


Figure 3.10: Virtual sound sources and vmikes in a sound space. Note the shading effects of superimposing the discs representing the subspaces in which sound can be detected (referred to in the text as 'audio nimbus intersection').

3.3.1.2. Virtual Sound Sources and Virtual Microphones

To introduce a degree of generality to our approach, and to investigate sources that are somewhat richer in their intrinsic spatial characteristics than are commonly investigated, we have devised mixing algorithms for stereo virtual sound sources. Our sound sources are assumed to have a position but also an orientation. Our vmikes are also stereo. That is, our design involves multiple directional sound sources, each of which is stereo, and our task is to calculate a stereo mix at a given location, the location of a stereo vmike. In other words, we have to calculate - given the relative positions and orientations of sources and vmikes - the 'projection' of each source onto the left-right sound image that the vmike is capable of capturing.

To indicate our solution, let us consider some canonical cases. Let us imagine a source 'facing' a vmike. Here we would wish for the vmike to 'pick up' the source with left-right channels reversed (remember: the source and the vmike face each other). A vmike behind the same source would pick up (assuming an omnidirectional source) the left-right image intrinsic to the source. A source placed to the left (right) of the vmike should mix biased to the left (right) side of the vmike's stereo image. To more precisely determine the sound projection in such cases, a representation of the source is used in terms, not just of a left and a right channel, but also in terms of the sum of the left and right channels (a so-called 'centre' channel as it preferentially picks out those sound components which are common to both left and right) and in terms of the difference between left and right channels (a so-called 'surround' channel). The relative mixes between the left, right, centre and surround channels allow much greater flexibility in spatialising sound than conventional pan or balance controls. Indeed, in this fashion, we independently can control the centre and width of any spatialisation of a two channel source. The source can be 're-centred' by conventionally panning the centre channel. Its width can be controlled by then mixing in quantities of the left, right and surround channels. A stereo source can be remixed, for example, so that its centre appears half-left and the sound image is heard between hard-left and centre. In this way, an impression of a source to the left of the stereo vmike can be given. The relative orientation of the source to the vmike can be determined by positioning the left, right and surround channels within a given width and with a given centre.

These techniques of remixing stereo sources in terms of left, right, centre and surround channels are well-known in the audio engineering world. Our application of them to calculating mixes and spatialisation in virtual environments is, as far as we know, novel. We use some simple vector geometry to give us the width and centre of the sonic projections onto the vmike. Our solutions produce changes in the mix whenever a sound is moved, whenever a vmike is moved, and whenever either change their orientation.

3.3.1.3. VSMRT: The Virtual Sound Mixer on the RoundTable

We tested our spatialisation algorithms first independently from the RoundTable and the visualisation techniques we mention above. An application was built in the MIDI and audio processing/programming environment MAX/MSP which took four spatialised dynamically varying stereo sources and moved a virtual microphone along eliptical paths between them. This application was tested in two ways. First, the improvising electro-acoustic music performance group *The Zapruda Trio* performed using this simplified VSM in two performances. We wanted to see if the sound mixing and spatialisation algorithms we had devised were usable in such situations. The group reported considerable satisfaction with the results, finding that the solutions computed by our application superior to those based on conventional panning and rebalancing of a stereo mix. Also, by controlling the path of a vmike, it was possible to control the relative mix and spatialisation of four stereo sources simultaneously – something which cannot be done dextrously with a conventional mixer. Secondly, we subjected some specimen dynamic mixes to critical listening by four electro-acoustic music composers associated with an institution independent of the eRENA project: the Department of Music at the University of East Anglia, UK. Uniformly, the composers were pleased with the 'smoothness' of the spatial changes and of the relative loudness attenuation we were supporting. Our application did give an impression of 'movement through space' even though we were not implementing a number of well-known movement cues (e.g. Doppler shift) – a point we shall return to below.

Pleased with these evaluations, we implemented our application with the RoundTable as an interface. The virtual space consists of four stereophonic sound sources which move around in the space (a simple random walk algorithm has been used for this purpose). Two stereophonic microphones are controlled by directional phicons. The positions of all these are transmitted as MIDI signals using the Nosuch MIDI Java library (see http://nosuch.com/nosuchmidi/) to the MAX/MSP program performing the spatialisation and mixing. The microphones can be in two states, selected or deselected. Deseclected microphones are muted. Selection and deselection is done through the insertion and retraction of small cylinders in the triangular phicons.

The sound sources are visually represented as little dumbbells (indicating their stereophonic nature) surrounded by a translucent disk-shaped cloud outlining their aural nimbus. Areas where the clouds overlap are drawn darker, indicating a higher aural density in that space.

When a single vmike is selected, the mix and spatialisation at that point is computed along the lines we have described. If this vmike is deselected and the other selected a cut to the other vmike is made. If both vmikes are selected we cyclically interpolate between the positions and orientations of the two vmikes. In this way, a crossfade can be elegantly supported, where the crossfade is not merely a mixing up of one vmike while the other is mixed down but a virtual spatial interpolation. This gives much more smooth results and allows users to experiment with interesting transitions as well as techniques for using the two phicons together to produce novel effects.

3.3.1.4. Evaluating Our Experience of VSMRT

In the lifetime of the eRENA project we have not been able to give VSMRT the opportunity to be evaluated by members of the public in a public setting, though the exhibition of VSMRT is planned. Rather, we have been able to obtain ‘expert evaluation’ from a number of musicians and media professionals. The following is a digest of the most commonly mentioned points.

The idea of using a sound visualisation and interacting with a tangible interface is very promising and suggestive for sound mixing applications. Presenting multiple sources in a virtual environment and mixing by manipulating a vmike is a very economical way to support mixing and spatialisation. A number of users, however, have remarked that they would like to be able to deploy microphones of different characteristics. At the moment all our virtual microphones are ‘omnidirectional’ in that they are equally sensitive to sound sources in front as behind. Directional and ‘gunshot’ alternatives are thinkable and a combination of different vmikes would also support interesting effects. In the terms of the Spatial Model, this would involve investigating different focus shapes for vmikes.

Even in our current design, supporting just two microphones enables a number of interesting transition effects and, as in other applications of the RoundTable, users have soon discovered that they can deselect a phicon by placing their hand over it. In the current application this enables users to make patterned gestures over the two microphones to, in a crude but suggestive way, ‘cut-up’ the sonic material.

The presence of two phicons also enables users to collaborate on the mix and the existence of a qualitatively new behaviour (the crossfade/spatial interpolation described above) can act as a motivation for their joint interaction.

Most users have again been impressed by the smoothness of our sound spatialisation and mixing algorithms. Our solutions do seem to produce much smoother effects than orthodox panning and balancing.

The visualisation does enable users to predictably associate sounds with regions on the table. However, it is questionable the degree to which listeners can imaginatively project themselves to be present inside a sonic space at the locus designated by the vmike phicon. Clearly, we do not yet implement a number of cues that would enhance impressions of sonic spatialisation. In the future, and in response to 'expert user' feedback, we plan on experimenting with Doppler shifts and phase delays between channels, supplementary reverberation, amongst other techniques.

Interaction supported by phicons works well provided that users do not make sudden gestures and expect those to have smooth effects. The framerate that the RT vision software is capable of working at fluctuates and has been observed to be as slow as 3 frames per second. This does not support the immediate detection of movement which would be necessary for a user engaged in, say, swiftly inter-cutting between vmikes. Slow graceful gestures, e.g. slow steady movement of a vmike, can be tracked adequately at such slow rates. However, users are entitled to expect that a variety of gestures, at a variety of tempi, can be equally supported. Indeed, interesting dramatic effects should be possible by suddenly changing the dynamics of the mix. We are limited in our ability to support this with our current hardware. Analysis framerates also place constraints on how tight the coupling between sound and image can be in production settings where, for example, sound mixing personnel are endeavouring to complement a vision mix.

Overall, we believe that VSMRT demonstrates an interesting concept of relevance to the support of electronic arenas: a tangible interface to a virtual sound mixer. There are important limits on the practical viability of our approach due to hardware and processing limitations, though this does not gainsay the validity of the concept. The existing version of VSMRT is a clear enough demonstration to act as a springboard for further design alternatives (e.g. more varied vmikes, a greater variety of transition effects, further sound spatialisation cues, the ability to mix more sources, including combinations of pre-recorded and live ones).

3.3.2. Interactive Compositional Machines on the RoundTable: *Small Fish*

A number of users of our virtual sound mixing applications have remarked that working with them is akin to composing a piece of music. Although our applications work with a small number of input sound files, the flexibility with which these can be remixed enables some novel combinations of sounds to emerge in the mix. As a direction for possible future research, therefore, perhaps we can identify a concept of 'interactive compositional machines' and explore tangible interfaces to them. While it is not new to investigate the automated composition of music (indeed this has a long history, see Roads, 1996, chapter 18 for a survey), examining real-time interaction with algorithmic systems supported by a physical interface like the RoundTable is rather more novel.

Our preliminary investigations of this are concerned to use activity at the RoundTable to influence features of sound in addition to the mixing and spatialisation aspects we report on above. In this way, the nature of sonic material could be altered by activity at the RoundTable to a more radical extent than just the remixing of prepared files or externally generated sound streams.

As a first invetsigation, we ported one piece of the CD-ROM *Small Fish* that has been created by Kiyoshi Furukawa, Masaki Fujihata and Wolfgang Münch at the ZKM in Karlsruhe to the RoundTable. *Small Fish* allows the user to manipulate a predefined set of musical algorithms, that stipulate what kind of relationships between visual graphics and musical time exist and how the player is able to influence the sounds via the screen. As a CD-ROM, *Small Fish* uses a traditional interface with mouse. For porting the application to the RoundTable we modified the mouse interface to actually get driven by the table output. We allowed three phicons of the same shape to stipulate the position of three different graphical objects. Hence, allowing three users to simultaneously interact with the system (see Figure 3.11).

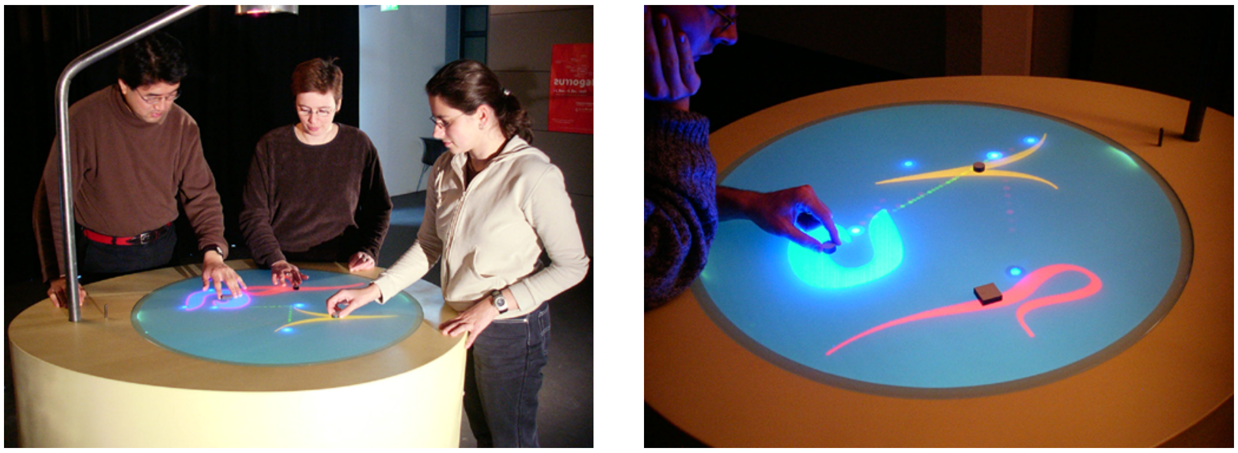


Figure 3.11: RoundTable running Small Fish, *(left) three users interacting simultaneously, (right) detail view of the graphical objects.*

By modifying the position of the phicons on the table surface, a graphical object's position is stipulated (see Figure 3.11, right). Music is generated by producing a MIDI stream of notes that gets influenced by the position of smaller circular objects that stick to the larger graphical objects (see small blue points in Figure 3.11, right). By moving the larger objects, the user can 'catch' or remove the small objects from other objects. The other two small circular objects are special, they move around autonomously, touching the smaller objects in a predefined order. Whenever they touch a small (blue) object a MIDI note is produced. The tone quality and the note itself is stipulated by the y-position of the 'collision' (i.e. from lower left top upper right in Figure 3.11, right). Hence, notes and speed can be stipulated by the user by modifying the position of the objects in y-direction and spatially arranging the objects on the 2D display surface.

3.3.2.1. Evaluating Our Experience with the Small Fish Interactive Compositional Machine

Our first results from showing this application to some 'players' were really positively. All users liked the tangible interface to the application. We noted that users did interact in an immediate and intuitive way. The interface allowed a single user to most easily change the position of the graphical objects at different spatial locations on the table. There is no more a point and click operation needed, no positioning of one interface handler to different graphical objects, but more a direct manipulation of graphical objects with the aid of the physical objects on the table surface. Even when more than three people where present, we noted that interaction did switch between users, i.e. showing collaborative behavior. Also users reported it being "fun" to interact with other users simultaneously. In the future we like to further pursue this approach. To do this, we would like to create different physical icons for this application and modify the graphical objects on the display to better match the size and style of the phicons. Also, the application was inherently created for a rectangular computer screen, hence, occasionally objects disappeared from the RoundTable display because they moved to the corners of the underlying rectangular display. We believe by making these modifications we can successfully improve this application.

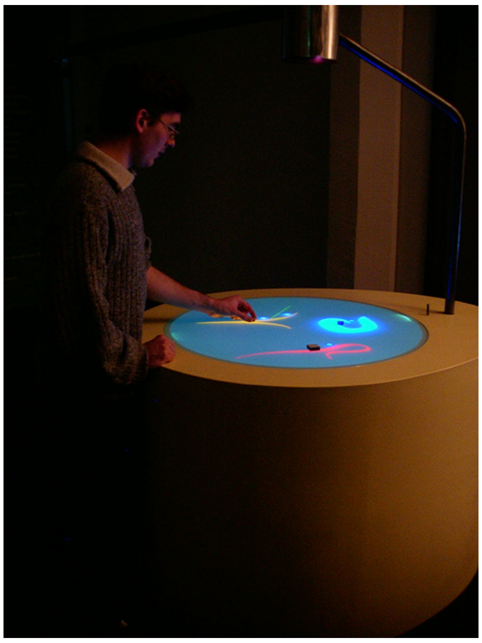


Figure 3.12: Small Fish *application with a single user interacting*

3.4. Conclusions

This has been an ambitious chapter presenting a series of concepts for the support of production in electronic arenas, together with applications and novel interface techniques to help realise them. Let us summarise what we think we have achieved, outline the limits of our accomplishments, and lay down some challenges for future work.

We have argued for a general orientation to the support of production in electronic arenas in terms of the phrase ‘activity-oriented resource deployment’. Our idea is that production personnel should be supported in deploying production resources by means of real-time information about the activity in an electronic arena. Virtual camera control and microphone deployment should be informed by a sense of where the action is. This requirement derived from our empirical studies of production work in Year 2 of eRENA and has enabled us to approach image and sound control in innovative ways. We facilitate production work through visualisations of activity and enable personnel to deploy production resources through interaction with these visualisations. In Deliverable D4.3/4.4 in Year 2, we also experimented with and successfully evaluated some sonification techniques as well. In addition, we have proposed novel virtual camera types that automatically seek out activity, as well as a general design philosophy that emphasises the ready combination of automatic and manual techniques.

In Year 3, we have been particularly devoted to four lines of work. (1) The refinement of our existing camera control applications. (2) The extension of our approach to sound control. (3) The migration of our applications to the RoundTable tangible interface. And (4) the evaluation of our efforts. We believe that we have successfully worked on all fronts and have gained a knowledge of the advantages and limits of our approach.

The general orientation of supporting production activities through giving information about activity within an electronic arena has always been endorsed by the media professionals we have consulted with. In a live, large-scale event in a mixed reality environment, it has often proved problematic to follow the action and resources to support this are gratefully received. There are many specific questions, though, which need to be addressed if our approach is to be workable in a particular context. Appropriate indices of activity have to be defined. Appropriate visualisation (and sonification) strategies are required. Appropriate interfaces need to be designed. Our attempts provide reference points for future practical experiences with this approach.

While we have demonstrated the basic feasibility of extensions of our techniques to sound mixing (and also suggested a further extension to the interactive composition of music), we do not yet have experience of working with live sound or with managing a very large number of sound sources. Our work is promising but limited at this stage.

All of the applications we report in this chapter have had dual realisations in both conventional desktop interface environments and on the RoundTable. This puts us in a strong position to make comparisons between the two approaches to interface design – a matter of considerable contemporary concern in the field of human-computer interaction. Our overall view is that a tangible physical interface (like the RoundTable) enables some interaction capabilities while inhibiting others. And the converse is true for conventional desktop interfaces. The RoundTable enables some kinds of gestures to be made much more swiftly than others (e.g. the simultaneous specification of an orientation and a position). Other gestures may be slowed (e.g. those which require an object to be visually searched for on the table before an action, like selection with a peg, can be initiated). The RoundTable permits multiple viewers to experience its information visualisation more readily than a screen on a personal workstation. This facilitates certain patterns of collaboration and inter-working between personnel. Providing multiple interface tools (rather than the single focus for action provided by a conventional mouse-pointer) also enables simultaneous working by a number of individuals on a shared display. However, conventional desktop interface techniques have considerable merits when it comes to the number of different actions that can be performed. Commands can be actioned from the keyboard, by mouse-gesture, by menu selection and so forth. On the RoundTable, we have tried to avoid simulating or reproducing such widgets, preferring a consistent physical interaction approach throughout. Accordingly, we have a limited number of blocks with basic functionality. This works well. Our attempts to go beyond this and trying to support classic operations such as zooming and scrolling have introduced tensions and inconsistencies in our design approach and given users difficulties in finding the appropriate ‘model’ for understanding what’s going on. In short, the RoundTable is an elegant interface for applications that need a small repertoire of commands to be supported. This small repertoire may still be very expressive and accomplish important outcomes: even our simplest implementation of SVEA on the RoundTable enables its users to control and edit between multiple cameras in a virtual environment (no mean feat). However, when a large command repertoire needs support, the RoundTable as we have worked with it is not the best solution. (Naturally, modifications could be considered – like bar-coding multiple phicons – but that would be quite another story.)

As a final word of caution, we must note that our technologies remain as prototypes and demonstrators. This is fine for a long-term research project like eRENA but it has to be admitted that our production technologies have not been tested in anger in the same way that, for example, the technologies reported in Chapter 2 have been. In project wide terms, this is a satisfactory situation. We have approaches to the production of events in electronic arenas which have been tested in the 'here and now' and have been shown to be workable in terms of the existing competencies of media professionals. We also have clear instantiations of technologies which could 'push the envelope' a degree further, involve novel approaches to understanding events in electronic arenas and novel interface techniques. These may require the reshaping of some of the practices of media professionals but we do not have evidence that any of our work does violence to how such personnel could see their activities developing in the future. In short, at the end of Year 3, eRENA in Workpackage 4 is able to offer an array of applications, approaches and interface techniques of varying levels of risk and ambition. Whether an event in an electronic arena is conceived of as a re-instantiation of an existing format or of a radical experimental nature, we have production techniques and experience that can be of assistance.