Deliverable 6.3

Individual and Group Interaction

ABSTRACT

This document consists out of: *Tracking Technology* (1.2) which describes the extended virtual environment EVE, the vision based setup and the tracking of people in this environment, *Simple Prototype Setup* (1.3) describes the undersea prototype in EVE, followed by the the work between ZKM and EPFL on interface paradigms for linking real and virtual people (1.4 through 1.6).

Document	eRENA – D 6.3
Type	Deliverable report with video material
Status	Final
Version	1.0
Date	August 1999
Author(s)	ZKM: Michael Hoch, Detlev Schwabe, Jeffrey Shaw, Heike Staff EPFL: Soraia Raupp Musse, Fabien Garat, Daniel Thalmann KTH: Kai-Mikael Jää-Aro, John M. Bowers, Sten-Olof Hellström
Task	6.3

Table of Contents

Preface

Heike Staff

Zentrum für Kunst und Medientechnologie

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

This document comprises Deliverable D6.3 of the eRENA project of the i3 schema of the ESPRIT-IV research action of the European Communities. The deliverable constitutes the output of Task 6.3 of the eRENA workplan and, while its delivery constitutes the completion of the task, several of the collaborative activities which have been initiated in this task will be continued into Task 6.4 (and elsewhere) in Year 3 of the project.

Task 6.3 has been concerned with formulating novel technologies for the support of individual and group interaction within electronic arenas. Like all work in Workpackage 6, it focus on supporting participants to events in electronic arenas through giving intelligible, usable and pleasurable interfaces to use to make their presence felt in an event or to carry out activities within it. In contrast to much orthodox work in interface development and in the research field of virtual reality, we have particularly accented group interaction and its support, and making features of group activity available to guide individual and collective behaviour. Another contribution to the deliverable is concerned with working on simple video analysis technologies which might readily scale between individual and collective use.

Task 6.3 has involved two strong (and novel with Year 2 of eRENA) collaborations to realise its aims. ZKM, EPFL and the University of Geneva have collaborated to produce a series of demonstrations of how interaction between individuals and groups can be supported in electronic arenas in particular where some of the 'participants' are artificial humanoids or other virtual creatures capable of a degree of life-like behaviour. This collaboration has involved work from each institution to make its technologies interwork with those from the others: humanoid representation from Geneva, collective behavioural modelling from EPFL and video tracking systems from ZKM, for example. The collaboration has also found a focus around the EVE dome as an interesting environment for exploring such hybrid interactive phenomena. Clearly, this collaboration also instances a move to the cross-workpackage integration of Workpackages 5 and 6. This collaboration comprises Chapter 2 of the deliverable, forms the bulk of its mature and detailed research, and immediately follows this preface.

A collaboration between KTH and British Telecom has led to the development of some deliberately simple, low-cost readily accessible video interaction technology in a preliminary version. This technology builds on the video based analysis system Wobblespace which is of extreme relevance to Workpackage 6 and this task but, for convenience in maintaining the coherence of the project's account of the *Out Of This World* inhabited television demonstrator, was delivered as part of Deliverable D7a.1. In the time since evaluating Wobblespace, KTH and BT have initiated a small collaboration to realise a generic successor to Wobblespace that could support individual and group interaction. The very early stages of this exploration are reported in Chapter 4. This will lay the ground for further development, more rigorous user-testing and application to performances in events in Year 3.

Between these two instances of cross-site collaboration appears a short paper from KTH describing how the proposals for ‘activity oriented interaction’ essayed in connection with issues to do with event production in Deliverable D4.3/4.4 can be turned to good use in providing participants within an electronic arena with resources to guide their navigation and interactive behaviour. The notions of activity oriented camera control and deployment can be found more fully discussed in Deliverable D4.3/4.4 and we anticipate that these ideas (or a selection of them) will find their way into demonstrations in Year 3 as the project turns its attention, amongst other things to novel concepts of avatar, and their control in electronic arenas.

1. Technology for Group Interaction Between Real and Virtual

Michael Hoch, Jeffrey Shaw, Detlev Schwabe, (ZKM)
Soraia Raupp Musse, Fabien Garat, Daniel Thalmann (EPFL)

1.1 Introduction

The tasks in WP 6 are based on the general notion that people should experience the sense of presence in a virtual space through movements as much as through visual or acoustic impressions. As social beings people are necessarily aware of their (local) position in group situations and they experience the physical space and the social climate of a group situation by simultaneously watching and moving around. The technology development in WP 6.3 aims at this behaviour. The research focuses on navigation for groups with shared displays.

Three steps have been undertaken by ZKM with the help and the support of the technological developments of EFPL and Geneva.

- The first step, was the hardware and software development of a vision system capable of intelligently tracking smaller and larger groups of people. We choose the dome architecture of the Extended Virtual Environment (EVE) as a nearly ideal environment because of its size and it's *window-into-a-world* paradigm that provides unique opportunities for people to imagine the qualities of the projected virtual world. In our vision system we define a group as a simple aggregation of individuals who are detected by determining their position, the direction and the speed of their movements (which constitutes together a "zone of attention").

- In the second step, the vision tracking hardware and the software technology has been applied on a situation in the EVE dome where 5 persons interact with the projected imagery of a virtual deep-sea world in which schools of marine life forms roam the grounds. In our prototype setup we implemented a simple behavioural animation system in which the group behaviour of 40 fishes emerges from a set of well-combined acceleration rules. The interface software we developed related the movement behaviour of the 5 persons to the flock of fishes in terms of attraction, avoidance, fleeing, and following.

- The experiences with this scenario have provided the basis for the third step. Here we defined interface paradigms between real users in real space and virtual groups and crowds. We created a prototype framework for to explore different modes of group modalities of interaction and differing linkage between real and the virtual crowds. Based on events in real space, that got recognized using the vision system, corresponding events in virtual space were triggered using the server-structure for guiding and interacting with crowds from EPFL. The linkage between the real events and the "virtual" events can be stipulated and easily changed using a script-language.

The above steps lay the technical foundation to explore modes of group interaction between real and virtual people in more detail and to also explore the social and artistic perspective of the approach. The defined interface paradigms of step three supply a flexible system that will

be used in a workshop environment (WP 7.b1, Workshop 3: "Mixed reality group and crowd interactions").

1.2 Tracking Technology

This Section consists out of: *Tracking Technology (3.2)* which describes the extended virtual environment EVE, the vision based setup and the tracking of people in this environment, *Simple Prototype Setup (3.3)* describes the undersea prototype in EVE.

The dome architecture of the Extended Virtual Environment (EVE) is an ideal controlled environment for the development of a vision system capable of intelligently tracking smaller and larger groups of people. The dynamics of individual and group behaviour in this environment (e.g. spatial position, spatial relationships, temporal movement patterns, etc.) can then be mapped to effect interactions with the projected imagery. In this WP vision tracking hardware and software technology will be developed enabling such interactions between real and virtual represented people. In the following section we will first describe the EVE environment, thereafter, the vision-based interface and the tracking algorithms involved.

1.2.1 The Extended Virtual Environment EVE

The Extended Virtual Environment [4] EVE is a unique implementation of a *window-into-a-world* paradigm, originally developed in 1992 and presented to the public at IMAGINA 1993. EVE is an enterable 3/4 of a sphere projection dome with a diameter of 12 meters and a height of 9 meters made of soft, inflatable fabric. A constant air supply is responsible for maintaining the dome's shape. The inner part of the skin is used as the projection surface. A rotating door entrance prevents the air from escaping outside. The dome contains a turnable stereo-projection device in the center. Stereo projection is accomplished by using two projectors with polarized lenses. Passive polarized glasses are used for viewing the stereoscopic imagery. In addition to the video projectors, the device contains a loudspeaker system and an infrared camera (see figure 1). In the standard application, an observer stands inside the dome and wears a small head-mounted infrared light pointer.

1.2.1.1 Pan & Tilt Head

The central part of the EVE dome is a stereo-video projection apparatus which can be rotated motor-controlled by 360 degrees around the vertical axis and has a rotation range from approx. –15 degrees (pointing slightly downwards) to 90 degrees (pointing straight up) about the horizontal axis. The projection head is mounted on a tripod so that the center of the projection coincides with the centre of the sphere. The projectors (two Synelec LightMaster [29] utilizing Texas Instruments DLP technology [27], [28]) support an 800 by 600 pixel resolution and have a wide angle lens providing a 60 degree horizontal projection angle. Linear polarized filters are mounted in front of the lenses to separate the stereo images. An audio speaker system with four mid-range speakers is also built into the head. In conjunction with a sub-woofer system at the basement of the tripod, a good sound system is available. All necessary signals for RGB video, power supply, audio, motor control and serial lines for configuring the projectors are brought into the head via a slip-ring unit.

Figure 1-1: The EVE stereo-projection device features two rotational degrees of freedom, two video projectors (with polarized lenses), a stereo audio system and a built-in infrared camera for tracking purposes.

The head automatically follows the movements of one visitors head who carries special polarized glasses with a mounted infrared light pointer. The infrared light spot on the dome surface is tracked by an infrared camera, also mounted inside the head. The camera image is analyzed by a tracking software, running on a PC that is installed in the basement of the tripod. The tracking software determines the position of the light spot in relation to the centre of the camera image (which coincides with the centre of the projected images) and calculates acceleration and deceleration values which are sent to the servo amplifiers via a serial controller. As a consequence the tracking software controls the motors for the horizontal and vertical motion of the head so that the centre of the projected images are coinciding with the viewing direction of the visitor.

1.2.1.2 Application Platform

Basically any computer hardware setup which is capable of producing a synchronized set of two 800 by 600 pixel resolution images can be used as application platform. The current applications are running on a two-processor 150 MHz, R4400 Silicon Graphics Onyx RealityEngine2 with a multi-channel option installed. Out of the three possible 800 by 600 pixel channels only two are used at the single available refresh rate of 60 Hz. In order to use the multi-channel option, the frame buffer must be configured to 2400 by 600 pixels (3 times 800 by 600). Since this resolution cannot be shown on a regular monitor, a VT320 terminal is connected to the machine for administration and control purposes.

To be able to select and start different applications from inside the dome, a simple application chooser utility has been implemented which is fully controllable with only the wireless joystick. A configuration file is used to define which applications are available and how they are started. With the thumb knob one can move through the list of applications and can start

one by pressing one of the buttons. By convention all applications must be able to be terminated by pressing the fire trigger and the other two buttons simultaneously.

1.2.1.3 Image Rectification

When projecting a rectangular image onto a spherical surface the result appears more or less distorted to an observer, depending on his current position[1]. To compensate for this distortion, in many applications the following described pre-warping process can be used to get an image rectification at least for one predefined observer position[2]:

A texture of the actual image content is mapped onto a polygonal, rectangular grid which then is displayed instead. By displacing the grid points, depending on the grid's resolution, arbitrary image warping can be achieved. For the EVE dome projection it turned out that it is sufficient to just displace the border grid points to a bulged-in arc-like shape and interpolate the inner grid points linearly to get an adequate rectification. To use arcs as the shape of the boundaries was motivated by the fact that a plane (in this case the boundary of the projection pyramid) which slices a sphere always results in a circular curve. So the projection of a rectangular image onto a spherical surface results in an actual surface on the sphere which is bounded by arcs.

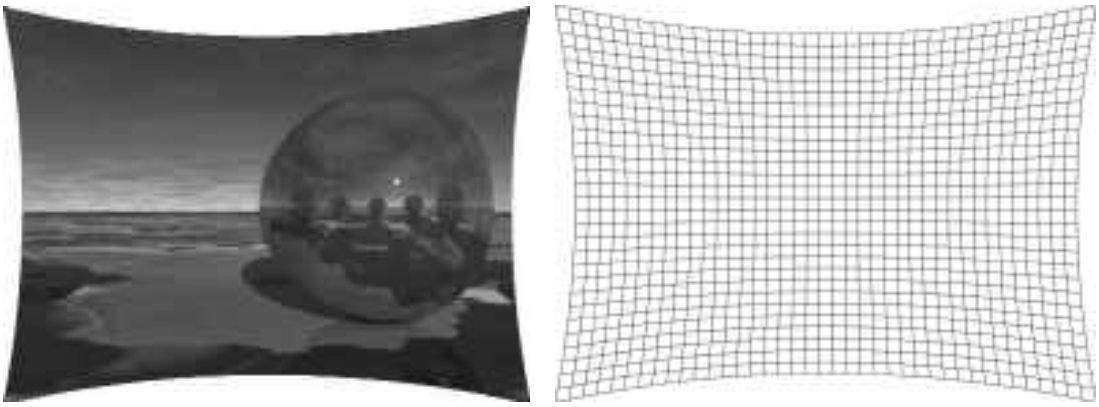


Figure 1-2.: A prewarped image and the underlying deforming grid. The boundaries are arcs and the inner curves are linearly interpolated

The method to get the actual image content as a texture to be mapped onto the prewarped grid, depends on the application. One can say that generally it is a process of multi-pass rendering. First, a virtual scene is rendered in a hidden frame buffer, then the content of that buffer is transferred into texture memory and finally the polygonal grid is rendered and displayed with the texture mapped on it.

The drawback of this method is clearly that it needs a large amount of the available texture memory. Applications which already use textures for their virtual scenery have to make sure that these textures are properly restored after the warping process. This in fact can lead to time consuming reloading of the texture images from the main memory. In the case of the later described iC_inema application (in fact the only application which uses the warping process

[1] Theoretically, the only position which is free of distortion is at the center of the projection which, of course is usally occupied by the projection system itself.

[2] In fact, to get a rectification dynamically for any arbitrary observation position, the head of the observer must be tracked to be able to adjust the rectification algorithm according to position changes.

so far) the situation is quite simple since the texture images are retrieved directly from hard disk.

1.2.1.4 Application Interface

Currently the application interface consists of a small circuitry board that interfaces the two angle sensors of the pan & tilt head as well as a 5-channel wireless joystick with the RS-232 serial port on the application machine. An easy-to-use, shared memory based API serves as the software interface between the actual application and the state of the pan & tilt head and the joystick. A background process running on the application platform, continuously reads the current status of the angle sensors and joystick and writes the values into a C-structure in a shared memory buffer. Any application just has to connect to the same shared memory area at the program start and is then able to access the current values at any time. Additionally, the EVE API also provides a simple event queue for the 10 possible joystick events (5 buttons can be PUSHED or RELEASED) for the application programmers' convenience.

To measure the current orientation of the pan & tilt head, two absolute angle sensors, each with a 12-bit resolution are built into the head. For mechanical reasons both sensors are installed and aligned to the vertical rotation axis. As a consequence one sensor actually measures the sum of the horizontal and the vertical angle, while the other delivers the value for only the horizontal angle. The vertical angle is determined by subtracting the latter from the first, regarding the possible overflow due to the 12-bit limit. To read out the current value of the sensor device, the interface transmits a pulsed signal to each sensor in order to receive the value bit by bit.

The wireless joystick, which has been adopted from the original setup, is a standard PC game joystick, reconstructed for wireless connection, with an integrated thumb knob, one fire trigger and three generic buttons of which one is not activated. A dedicated radio receiver is able to receive five different functions from the joystick, which are used as forwards and backwards on the thumb knob, the fire trigger and two of the buttons. The circuitry is responsible to read out the status of the angle sensors as well as the joystick receiver. Then it sends a complete data block to the application platform over the serial port. The status refresh rate lies at approximately 40 fps, which is currently limited by the used communication speed of 9600 baud.

While the user is turning his head in order to look into different directions, a dedicated PC grabs the images from the infrared camera, tracks the motion of the infrared lightpoint on the projection surface and calculates and transmits the acceleration and deceleration values to the motors which control the orientation of the projection device to keep the projected images centered on the lightpoint. Two absolute-angle sensors measure the current orientation of the projectors and deliver these values to the actual application which can then render an updated view. The application programmers interface is implemented as a shared memory interface between a separate process on the application machine, which continuously reads the current state of the angle sensors and the actual application. For non-standard applications, an external control interface is available to control the turning of the projection device directly from the application program. For the work described here, the projection is completely controlled by the application.

1.2.2 Vision Based Interface

For tracking the movements of individuals and groups in the EVE projection environment in an unencumbered way, we use a vision-based interface that is based on luminance-level segmentation and blob analysis. The image processing system tracks the user via an infrared camera setup that is mounted on the ceiling of the dome. It consists of a standard black and white camera and 10 standard halogen lamps. The lamps are placed in a ring around the camera and are covered by an infrared filter (figure 1-3).

Figure 1-3: Camera with 10 halogen lamps and infrared filter mounted in a round box

The software system *mTrack* that is being used for tracking people, is divided up into the recognition part consisting of the image processing system, a server program, and the library frontend with the application program (see figure 3-4 and [9], [10]).

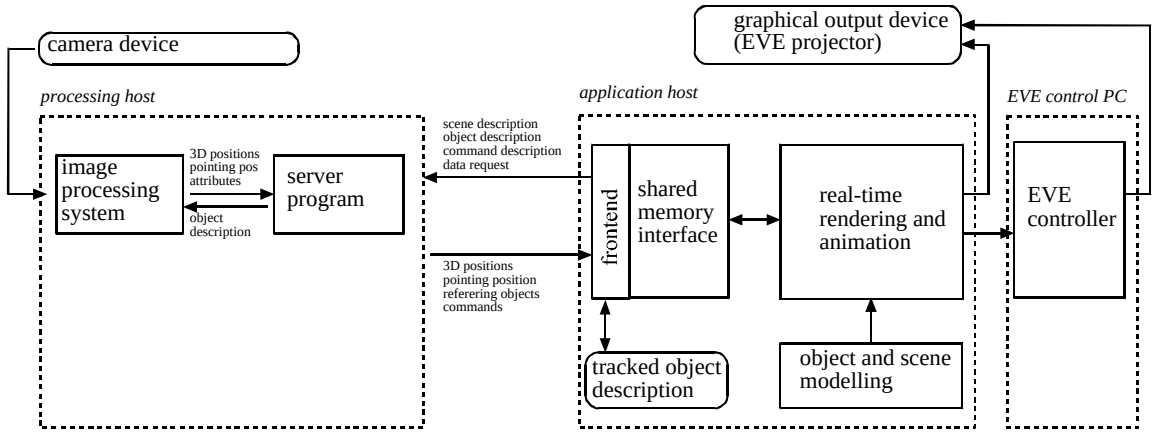


Figure 1-4: Architecture of the mTrack tracking system and application program for real-time rendering and animation within the EVE environment

During initialization the system receives a description of the objects to be tracked. Thereafter, it continuously sends position data and other calculated information of the segmented objects to the server. The server connects the application with the image processing system and updates the current states by an event-driven loop. Upon request it continuously sends data to the application The information that needs to be extracted is: position, velocity, direction of movements, distance to other participants, number of participants, and the possible detection

of interaction between participants. The extraction of the position data is achieved by tracking regions based on luminance and blob segmentation. The other parameters are then calculated with respect to this data.

Tracking regions. For getting update rates higher than 10 frames per second, which is essential for establishing a direct feedback between a user and graphics [21], we had to find a compromise between sophisticated tracking algorithms and simplicity to achieve the desired frame rate on standard PCs. A simple and robust approach is the tracking of coloured or greylevel regions. If the objects to be tracked are of high contrast against the background, a simple segmentation algorithm together with a blob analysis is sufficient to reliably track these objects at a high frame rate. This leads to the algorithm outlined below:

```
PROCEDURE regionTrack

WHILE(true)
  // acquire current image
  current = get_next_image()

  // remove Gaussian noise
  low_pass_filter ( current )

  // perform simple binarization
  binarize ( current, greater_or_equal, 50 )

  // analyze result image
  // exclude small areas and scattered regions
  blob_analysis ( image )
  exclude blobs with area <= 10
  extract_position_data()

  // send data to server
  send_data_to_server()
  ENDWHILE
END
```

Detecting individual behaviour. As basic parameters that are detected for the individuals, we determine the position, the direction of movements, and the speed for each individual by using the tracking algorithms outlined above. Furthermore, it might be of interest to detect the intersection of the direction for two individuals (see figure 1a). As a simple model of the notion of territories and personal zones described in [8], or like the notion of *aura* and *focus* described in [3], a dynamic zone of attention can be defined, that is dependent on the direction and speed of movement of the individual. This zone may later be used to detect interaction between real participants as well as between real and virtual participants (see figure 1-5b).

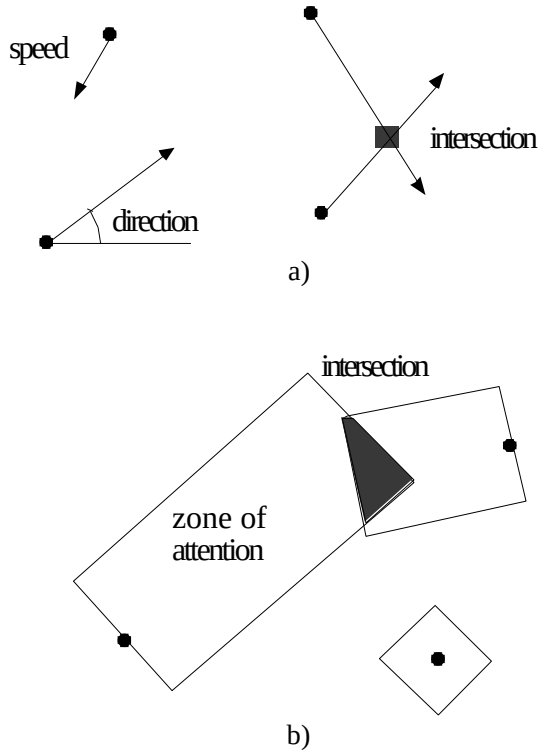


Figure 1-5: a) Detecting position, direction, speed, and intersection of movement. b) Detecting simplified zone of attention.

Detecting group behaviour. In our system the detection of groups in real space is directly mapped onto the spatial attribute of proximity, i.e. it is limited to a simple aggregation of individuals that comply to a certain distance threshold. The threshold is variable and depends on the number of people being present in the interaction area.

1.3 Simple Prototype Setup

This Section describes a setup using the surround screen environment described in the last section (Extended Virtual Environment EVE dome) that we used to explore group interaction in real and virtual space. We have created a prototype framework to explore different modes of group interaction, the position and motion of users in real space are tracked using a vision-based interface that allows the activities of real crowds to be monitored. In the virtual space, we use a simple behavioural animation system that serves as a tested to generate virtual group and crowd behaviour. Exploring different kinds of dynamic relationships between real and virtual groups gives insight to possible directions of group interaction [11].

1.3.1 Introduction

We found the EVE dome to be a perfect analogy to an under-water observatory that resides on the sea bottom. The projection serves as the observation window into the surrounding virtual sea world (see figure 1-6). Schools of virtual fish move through the virtual world, driven by behavioural rules and interacting with the people inside the dome. For realizing this virtual environment a proprietary software programming package for real-time rendering and animation has been developed. In the current state the package mainly supports importing scenes, models and keyframed animation from a commercial 3D modelling and animation software. It further supports the creation of flocks, i.e. large groups of similar individuals and behaviours, which control the basic motion of each of the individuals. The system also

provides a special class of objects that interface with the external tracking system and represent the visitors position and other parameters inside the virtual world.

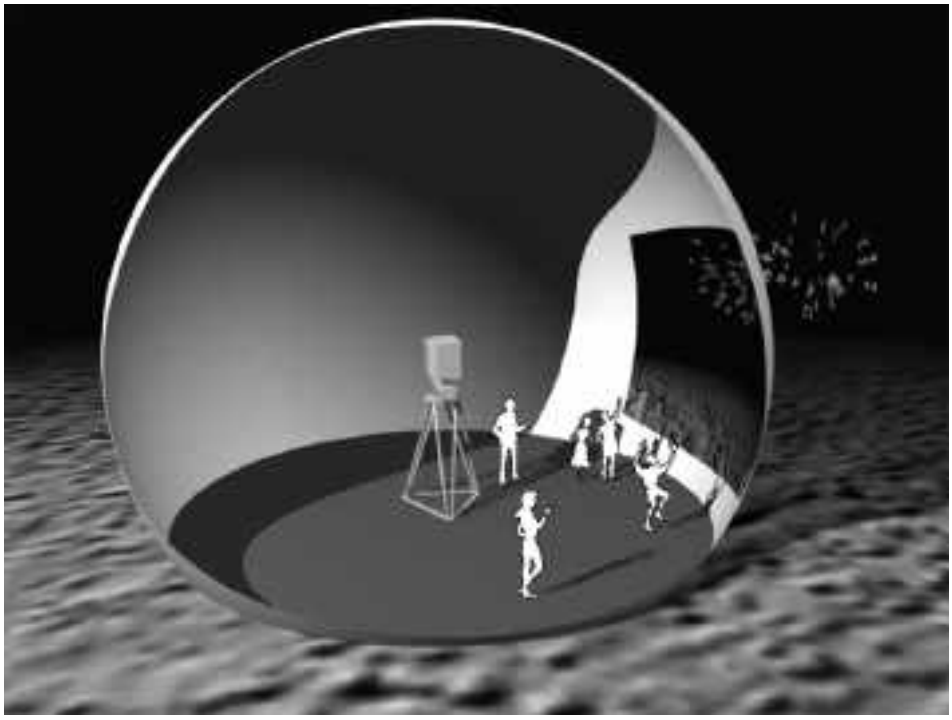


Figure 1-6: The EVE Dome - Extended Virtual Environment. The figure illustrates the dome situated inside a virtual world. The projection acts as a window into the virtual environment.

1.3.2 Behavioural Animation

The implementation of the behavioural system closely follows the work of Reynolds [19], who set up the principals of behavioural modelling for the purpose of animating groups of individuals. In his work he suggested a method for controlling the motion and behaviour of large groups of entities by means of defining a set of simple acceleration rules, which applies to every member of a group. A complex, yet controllable behaviour of the whole group emerges from the application of this local rule set. Many adoptions and applications of the original paradigm of artificial flocking have been seen over the past ten years. Especially in the motion pictures it has become a standard technique to animate fish, birds, wild stampedes, swarms of bats or even armies of martial horsemen galloping into a battle [1] [5] [15] [20] [22]. In computer games it is also a widely used technique for secondary animations to create more interesting and realistic looking virtual worlds.

A behaviour of a flock in our system consists of a set of well-combined acceleration rules. Each rule adds a certain amount to the overall acceleration vector that then drives the individuals into a certain direction. Currently, the package provides the basic behavioural rules for flocking i.e. *aggregation* (staying close to immediate neighbours), *alignment* (maintaining the same speed and heading as the neighbours) and *avoidance* (not bumping into neighbours). Other rules exist for floor and ceiling avoidance as well as for maintaing a certain speed (which is usually used to prevent the individuals from coming to a complete stop). New rules can be added by deriving them from a base rule and implementing a new

acceleration formula (it is planned to have a scripting language like TCL for creating new rules in the future. Currently all extensions must be hard-coded in C++).

A behavior consists of a prioritized list of weighted rules. The most important rules (usually all rules which deal with avoidance or collision) are at the top of the list. When calculating the total acceleration of a behaviour, the list of rules is processed top to bottom. The acceleration of each rule $\vec{a}_i(t)$ is calculated and the resulting acceleration vectors and their magnitudes are summed up separately. This is continued until either all rules have been processed or the sum of the acceleration magnitudes exceeds a predefined maximum acceleration value $a_{\max}$. This process can be formalized as follows: Let a_B be the total magnitude of acceleration and $\vec{a}_B$ the total vectorial acceleration of the behaviour *B*

$a_B := 0;\ \vec{a}_B := \vec{0};$
foreach rule *i* do
  $a_B := a_B + w_i \left|\vec{a}_i(t)\right|;$
  $\vec{a}_B := \vec{a}_B + w_i \vec{a}_i(t);$
  if $a_B \ ? \ a_{\max}$ break;
endfor

By additionally weighting each rule with a factor w_i the overall behaviour can be fine-tuned and new behaviours can be created or adopted to new types of flocks without the need to adjust parameters of individual rules.

1.3.3 Virtual Fish

For our experiments we created a simple virtual scene consisting of a representation of the EVE dome, a flock of 40 fish, five objects serving as stand-ins for the recognized visitors, and a *target fish* continuously moving on a predefined motion path (see figure 1-7). For the behaviour of the fish, the following basic elements have been integrated:

attraction, i.e. the fish should move towards a specific recognized visitor,

avoidance, the fish should maintain a certain distance to the visitors,

fleeing, in the event of fast movements of the visitors or if the visitors are standing close together

following the target fish, in the case that there is no visitor to be attracted to.

To be able to create the intended fish behaviour we had to extend the approach of simply putting elemental acceleration rules into a prioritized list, which does not allow to combine context-sensitive rule selection. To overcome this limitation we created one rather complex acceleration rule which integrates all of the elemental behavior described above and put them into the list along with the standard rules for flocking, floor-avoiding etc. The complex behavioural rule can be described as follows:

```
# There is one big fish moving constantly on a predefined path
# up to 5 persons are tracked by the tracking system

do every time step:
  determine current active group of tracked people, which is:
    the number of currently tracked people which are not in the
      neutral zone
    the center of this group
    the spatial size of the group: the diagonal of the bounding box

    foreach fish do:
      select a person to follow, if
        the fish is not already following a person OR
        the current followed person has been lost by the tracking system OR
        the current followed person has entered the neutral zone
      by
        choosing the nearest person in the active group OR,
        if number of tracked people == 0
          by choosing the big fish

# to prevent the fish to swim "inside" the dome, they are attracted
# by a radially offset position to the actual position of the person,
# they follow, a position which lies outside the dome on a fixed radius
# measured from the center

    start fleeing for a certain amount of time, if
      fish is not already fleeing AND
        close enough to the selected person AND
        the person makes a fast movement towards the fish OR
      if people are forming a group i.e. group size < group size threshold


    turn away from the selected person, if
      distance to person is less than a predefined maximum distance
```

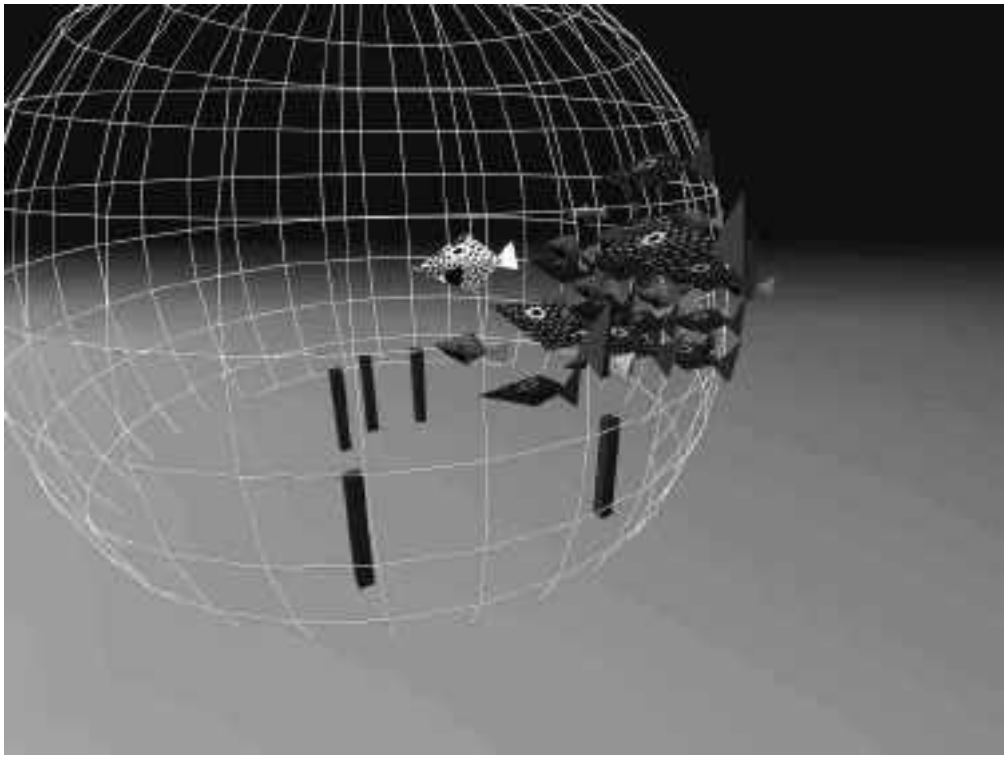


Figure 1-7: External view of the dome, the five stand-in objects, the keyframed fish and the flock of the other fish currently following him

1.3.4 Results

To explore the proposed interaction paradigms of the prototype described in this paper we analysed differing numbers of people in real space and different modes of interaction between the virtual and real world. For the first tests described in this paper, we did not move the projected imagery, i.e. it remained a static window into the virtual world as shown in figure 4. This restriction allowed us to explore single ways of interaction separately without dealing with the complexity of a fully animated virtual world together with a (physically) moving window in the EVE environment. This has the disadvantage that the participants might not get fully immersed in the environment. However, the special setup of the EVE dome still gives a strong effect of immersion. By entering the darkened space, the user as well as the physical space get transported. We tested the environment with one, two, three, four and five people separately. Thereafter, we also made tests with situations of 5 to 10 people being present in the interaction space. The results presented here have been collected in an ad-hoc and intuitive manner.

First, we explored if it is possible to reinforce mutual awareness between human participants by having two participants in real space approaching each other. When the participants came close enough to form a single group in the environment, the fish were fleeing, which had an intensifying impact on the awareness of the people. On the other hand, we found it difficult to reproduce such results when more then three people where present in real space. This might indicate that the interaction paradigm (fleeing fish, attraction) is too simple for a more complex situation. It might as well stem from the environment itself which is merely a dark and empty space with a strong focus on the projected visuals. Adding sound or adding other environmental artefacts, or, having a fully projected surround environment, e.g. by using four or more projectors, might overcome this limitation. Another problem we encountered, is the

virtual scenery: The underwater world with fish influenced the expectations of the participants and limited the variety of interactions that people were willing to try. Either a more abstract, or a more human-to-human-like communication mode might deal with that. An advantage of the familiar underwater seaworld is, that the transportation, i.e. "the extent to which the participants perceive that they have left behind their local space and have entered into some new remote space" [2], did work very well, further enhanced by the influence of the EVE scenario itself as described above.

One limitation of the scenario is that the centre part of the interaction area is covered by the projector, i.e. an area of 1.5 meters diameter is inaccessible and limits the movement to a circular space around that area. We did not encounter this as being a major limitation. However, we do believe that it influences the motion and, with it, the way the user is represented in the virtual world as well as the user's awareness of the environment. But, since we wanted to explore the applicability of different modes of interaction between real and virtual space, we did not consider this as to be a major problem.

As a general observation, we can say that the interactive nature of the environment made the visitors more active, i.e. it was unlikely that people would come in and stand still. Also, we found that feedback is an important issue in our environment: Because the tracking system currently only tracks up to five people, the participants had difficulties in linking their movements to events presented in the virtual scenery in situations where more than five people where present. To overcome this problem we used a simple graphical representation of the participants in virtual space, a thin pole that became visible when being close to the projection screen. Hence, a participant could reassure himself of being tracked by walking towards the screen.

1.3.5 Conclusion

In this section we have presented a prototype for exploring different modes of group interaction in real and virtual space. An exploration of situations with differing numbers of people present in the interaction space has been undertaken using an underwater virtual world. A more thorough investigation and interrogation of participants would give a more profound insight in the applicability of the proposed scenario. However, these first results clearly show some strengths and weaknesses of the approach and give hints for further investigation. For future work we would like to test the scenario with more human-to-human like communication models, which would require a more appealing virtual world as well as more complex behaviour patterns of the virtual creatures.

1.4 Interfacing Real People

To interface the real people we use the approach described in section 1.2 *Tracking Technology*. Here, the real space used for the prototype is an open space in the ZKM | Institute for Visual Media, but, the results apply to the EVE environment as well. To monitor the movements of up to five people in real space we mounted the infrared camera described in section 1.1 on the ceiling above the interaction space. The movements of the people in space and over time are tracked by calculating the position, the direction of movement and the speed of movement of each individual. Tracking of real people in this sense is reduced to tracking of points, i.e. a person is modelled merely as one single region. Other approaches might also use

gesture analysis for complex behaviour and/or interaction paradigms. Here, we focus on the movement in space with respect to the virtual world and the inter relationship of people in this public interaction area. Furthermore, the results of the previous section clearly indicate that the space/time position tracking creates very complex patterns. Hence, detecting more group related events, as described in section 1.2, and supplying a basic repertoire to interface to virtual computer generated scenery has been the intention of this work described in this section.

1.4.1 Creating Events in Real Space

In the following we will first describe the events that get triggered due to the movement of the tracked person in the interaction space. Thereafter, a table will list the events and supplied parameters in detail.

enter	When a person is entering the interaction space this event gets triggered. This event is limited to a 15% edge area of the interaction space preventing *enter* and *exit* events to take place when the tracking algorithm might loose an observed person.
exit	When a person is leaving the interaction space this event gets triggered. As with the *enter* event this event is limited to a 15% edge area of the interaction space.
group_formed	This event gets triggered when a group has been formed due to the algorithm based on distance measures between the individuals outlined in section 1.1.
group_deleted	This event gets triggered when a group has been deleted due to the algorithm based on distance measures between the individuals outlined in section 1.1.
increase	The number of elements in a group has increased.
decrease	The number of elements in a group has decreased. If only two people are in the group the group will be deleted.
pattern_event	As an example of group patterns performed by the observed actors or visitors, we implemented the *circle* and *line* events that get triggered when all people form a circle or a line. For the circle we calculate the sum of distances from an ideal circle and allow a deviation of 0.7 meter. For the line event we use a linear regression algorithm to calculate the maximum axis of the position of all visitors. We allow a deviation of 0.4 meters for all people (sum of deviations). For the circle event there must be at least 4 people present, for the line event the minimum is 3.
fast_movement	If a visitor moves faster than 1.5 meter per second a *fast_movement* event is triggered. If the movement is in a direction along the x- or z-axis a *fast_movement_direction* with an indication of the movement direction (left, right, front, back) is triggered, otherwise a *fast_movement_position* is recognized and the position of the person is supplied.

normal_movement	For movements that exceed a threshold of 0.4 meters a normal_movement event is triggered. If the movement is in a direction along the x- or z-axis a *normal_movement_direction* with an indication of the movement direction (left, right, front, back) is triggered, otherwise a *normal_movement_position* is recognized and the position of the person is supplied.
activity	This events indicates group activity, modeled as the sum of movements of all individuals of a group normalized by the number of individuals in the group. If this sum exceeds 0.1 meters the event is recognized.
spatial_event	As an example of a spatial event we used the location of all people. Hence, if all people are in one of the quadrants of the interaction space (left, right, front, back) this event gets triggered. Furthermore a center area, i.e. a circle of 1.5 meters radius in the center of the interaction space, serves as a location where a spatial event gets triggered when all people are within this area.

Event name	Event ID	Output Parameters	Description
enter	RP_event0	i, (x_min, z_min, x_max, z_max), (x_pos, z_pos)	person number, spatial region, entering position
exit	RP_event1	i, (x_min, z_min, x_max, z_max), (x_pos, z_pos)	person number, spatial region, last position
group_formed	RP_event2	i, (x_min, z_min, x_max, z_max), (x_pos, z_pos)	group number, spatial region, position
group_deleted	RP_event3	(x_min, z_min, x_max, z_max), (x_pos, z_pos)	group number, spatial region, entering
increase	RP_event4	i, n, (x_min, z_min, x_max, z_max), (x_pos, z_pos)	group number, number individuals, spatial region, position
decrease	RP_event5	i, n, (x_min, z_min, x_max, z_max), (x_pos, z_pos)	group number, number individuals, spatial region, position
pattern_events	RP_event6-7	pattern, (x_min, z_min, x_max, z_max), (x_pos, z_pos)	pattern id (circle, line), spatial region, position
fast_movement	RP_event8-9	i, dir, (x_min, z_min, x_max, z_max), (x_pos, z_pos)	person or group number, direction angle, spatial region, position
nomal_movement	RP_event9-10	i, dir, (x_min, z_min, x_max, z_max), (x_pos, z_pos)	person or group number, direction angle, spatial region, position
activity	RP_event11	i, level, (x_min, z_min, x_max, z_max), (x_pos, z_pos)	group number, activity, spatial region, position
spatial_events	RP_event12	location, (x_min, z_min, x_max, z_max), (x_pos, z_pos)	location id (center, left, right, front, back), spatial region, position

Table 1-1: Events detected by RPI-Client and possible output parameters

1.4.2 RPI-Client

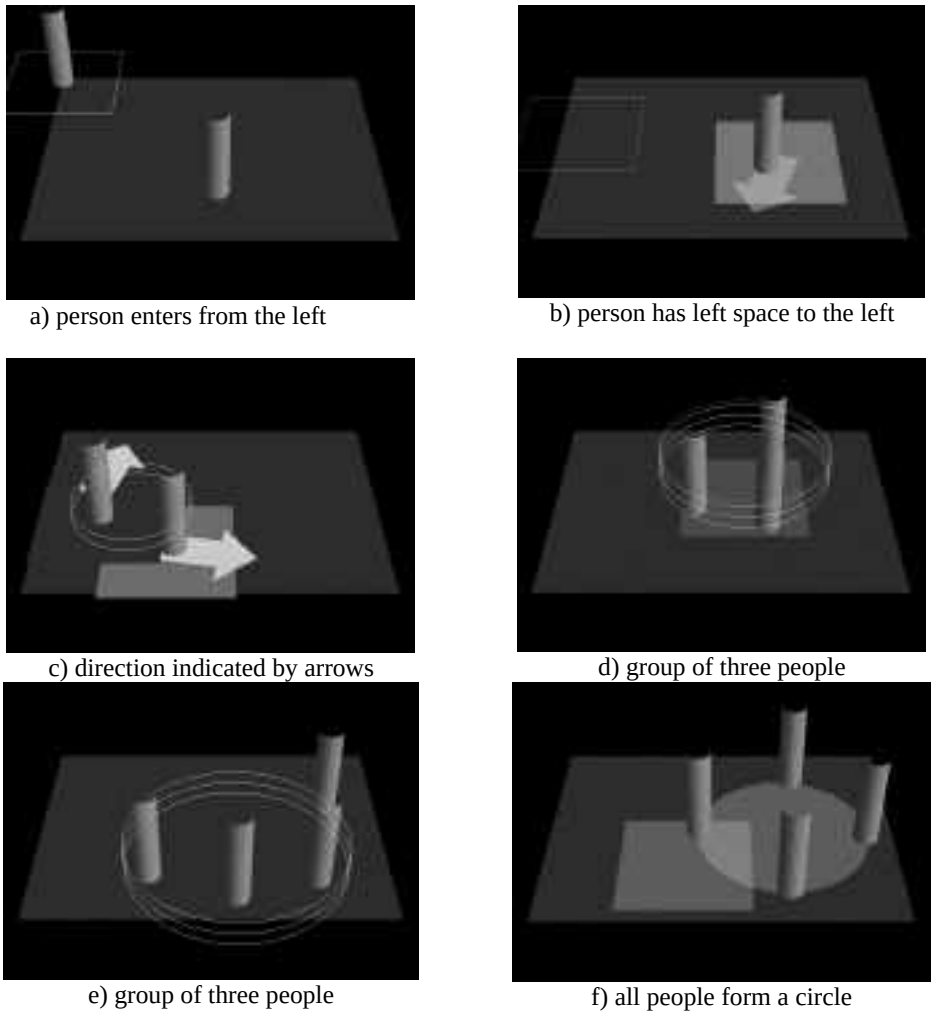
The RPI-client (Real Person Interaction Client) interfaces the detected individual and group behaviours, individual and group states, and the triggered events with the virtual scenery. It currently implements this by sending the generated events to the server program which, in turn, will pass the appropriate events to the ViCrowd Client (see section „Interfacing Virtual Crowds“). As an extension sending also state information or having direct real-time links between states of the observed people and the virtual scenery is possible. The update rate of the state information is limited by the tracking system and currently runs at 12 fps on a Pentium K6 333 MHz. For the prototype described in this chapter, however, the communication between the server and the RPI-Client is limited to sending the events listed in the following table. Also, the rate for creating events had been adapted to the processing rate of the client server architecture, which ignores events if too many would get triggered at one time step (see also discussion in the results section). Hence, the rate of generating events is set to 0.5 seconds. By creating the events we use a spatial sub-division of the real space in order to link to the corresponding spatial section of the virtual world. Hence, each event is linked to a spatial region in real space where the event took place.

Event name	Event ID	Output Parameters	Description
enter	RP_event0	(x_min, z_min, x_max, z_max)	spatial region
exit	RP_event1	(x_min, z_min, x_max, z_max)	spatial region
group_formed	RP_event2	(x_min, z_min, x_max, z_max)	spatial region
group_deleted	RP_event3	(x_min, z_min, x_max, z_max)	spatial region
increase	RP_event4	(x_min, z_min, x_max, z_max)	spatial region
decrease	RP_event5	(x_min, z_min, x_max, z_max)	spatial region
line	RP_event6	(x_min, z_min, x_max, z_max)	spatial region
circle	RP_event7	(x_min, z_min, x_max, z_max)	spatial region
fast_movement_ direction	RP_event8	(x_min, z_min, x_max, z_max), dir	spatial region, direction (left, right, front, back)
fast_movement_ position	RP_event9	(x_min, z_min, x_max, z_max), (x_pos, z_pos)	spatial region, position
nomal_movement direction	RP_event10	(x_min, z_min, x_max, z_max), dir	spatial region, direction (left, right, front, back)
nomal_movement position	RP_event11	(x_min, z_min, x_max, z_max), (x_pos, z_pos)	spatial region, position
activity	RP_event12	(x_min, z_min, x_max, z_max), level	spatial region, activity
spatial_events	RP_event13	(x_min, z_min, x_max, z_max), location	spatial region, location id (center, left, right, front, back)

Table 1-2: Events send to the server by the RPI-Client

1.4.3 Visualization of Events

To visualize the events and to create a feedback mechanism for the visitors we developed a graphics application that visualizes the current states of the tracking program and the events that get triggered. We chose this for providing a feedback to the visitors being present in the interaction space, because the results of the „Simple Prototype Setup“ described in the last chapter showed that this is a critical issue. Without a feedback the user would often not know if he or she is part of the scene or get tracked at all. The program renders the spatial layout of the real space. Visitors are symbolized by cylinders that get updated at 12 fps (tracking rate). Events that get triggered are shown as fading spatial regions (displayed as filled rectangles) on the ground marking the space where the event took place. For some events additional information is presented, e.g. groups are shown as wireframe circular rings with the number of rings corresponding to the number of individuals in a group (figure 1-1d and e). Fast movements are rendered in dark red (all other events are rendered black and white), direction information is supplied by displaying a filled arrow that is linked to the cylinder corresponding to the person and pointing in the direction of movement (figure 3-8c). Enter and exit events are shows as wireframe regions (figure 1-8a and b).



a) person enters from the left
b) person has left space to the left
c) direction indicated by arrows
d) group of three people
e) group of three people
f) all people form a circle

Figure 1-8: Visualization of the events

1.5 Interfacing Virtual Crowds

1.5.1 Introduction

In order to interface virtual crowd and real people, we have used ViCrowd System, a multi-layer system with different levels of autonomy. This section aims at describing ViCrowd concerning the guided crowds (groups of virtual people which can be guided or interact with real participants), as well as the Client/Server architecture used to provide the communication between real and virtual people in this work. Finally, we describe the two scenarios built for the integration with real people as well the events generated in virtual space. Also, some considerations about the space mapping in RPI Client (Section 1.4.2) are presented.

1.5.2 ViCrowd Model

This section aims at presenting some concepts of our crowd model [16] [17] and more explicitly about the guided crowd. The simulation of human crowds for populating virtual worlds provides a more realistic sense of virtual group presence. In some virtual environments, it would be useful to simulate populations in an autonomous way, thus the agents have a kind of environment knowledge and are able to move and interact within this environment. However, depending on the application, more ways of interaction can be required in order to provide a real time communication between participants and virtual agents. We have worked with three levels of autonomy: guided, programmed and autonomous (see Table 1-3) in order to establish the required control of crowds, depending on the application.

BEHAVIOUR CONTROL	GUIDED CROWDS	PROGRAMMED CROWDS	AUTONOMOUS CROWDS
Level of Autonomy	Low	Medium	High
Level of "Intelligence"	Low	Medium	High
Execution frame-rate	Low	Medium	High
Complexity of behaviours	Low	Variable	High
Level of Interaction	High	Variable	Variable

Table 1-3. Characteristics of different types of crowd control.

These three levels of autonomy are represented using two kinds of interface: scripted or guided interface. *Scripted interface* uses a script language where action, motion and behavioural rules are defined in order to specify the crowd behaviours. While action and motion describe explicit behaviours of crowd, called ***programmed crowd*** (see Fig. 1-10 left), the behavioural rules are used to define ***autonomous crowd***. All these information can also be sent by an external process in order to guide crowds explicitly, during the simulation. We called this type of crowd as ***guided crowd*** (see Fig.1-9). The small window on bottom right represents the Textual User Interface (TUI) client where textual commands can be specified. Figure 1-9 shows guided agents reacting according to a textual command: GOTO Station.

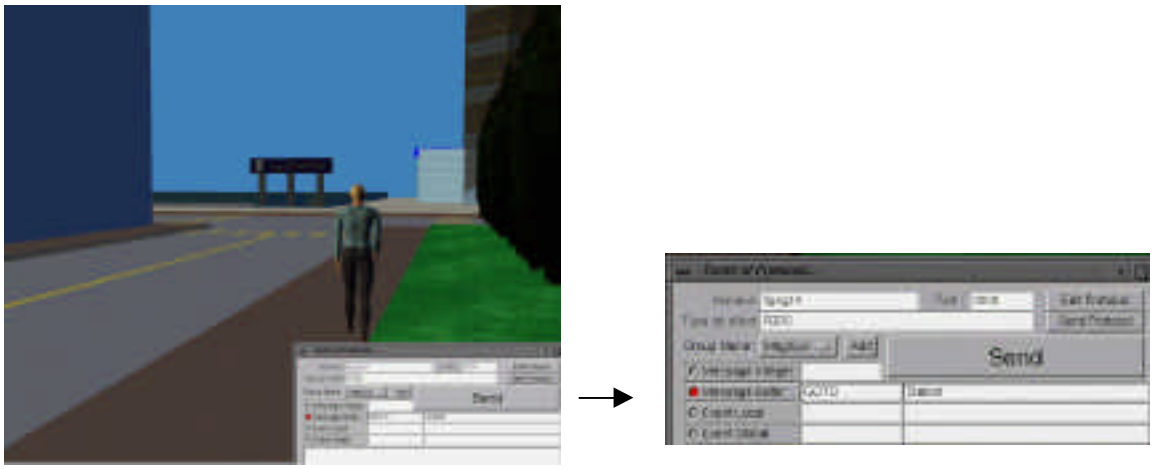


Figure 1-9: A guided agent going to the train station as specified in the Textual User Interface Client (TUI).

The autonomous crowd (see Fig. 1-10 right) is able to act according to the inherent behaviour (seeking goal, walking, avoiding collision, etc), recognising the presence of behavioural rules and also obeying programmed behaviours of groups in the script language. In order to mix

different behaviour natures, we have defined a priority of behaviours, then synchronising different kinds of control: (from high to low priority) guided, autonomous, programmed and inherent. Some examples of mixed control are:

· A group of agents walk on the virtual city (programmed behaviour) avoiding collision with other agents and declared obstacles (inherent behaviour).

· A panic event that was time-defined in the script language occurs (e.g., pre-specified to occurs in frame 1000). The reactions specified in the behavioural rules are activated (the reactive behaviour has more priority than the programmed motions and actions). Then, agents loose their programmed behaviour and employ their programmed reaction (e.g., look for the exit doors in the virtual environment).

· The user/external controller gives a new order to the agents, which form the crowd. For instance, exit in a specific door (external control). Afterwards, the agents stop to react as a function of pre-programmed events and follow the external control.

More details about our crowd model have been published by Musse and Thalmann [16][17]. Although the multi-autonomy level of our crowd approach, this paper is more focused on discussion about the ways of interacting with guided crowds.

Figure 1-10: (left) Programmed crowd walking on the city [7]. (right) Autonomous crowd reacting as a function of a rule based system.

1.5.2.1 Guided Crowds

The guided crowd represents groups of virtual agents, which can be externally guided. As the crowd in our approach is goal-based (the groups of agents always try to seek goals), the guided crowd receives dynamic goals in order to reach during the simulation. The goals concerning the information entities that can be controlled, they are:

· Motion (go to a specific position respecting collision avoidance with obstacles)

· Action (apply a specific posture, interact with an object)

· Events (can be triggered as a function of a matched condition or time-based generated)

· Reaction (can be: an action; a motion; attach to the motion of another object; change the internal status of a group/agent; activate one or more events)

· Internal status of groups (e.g., the group emotion)

· Environment information (used to declare obstacles that have to be avoided and regions where the crowd can walk)

Some of these entities information (specified during simulation by an external controller) are independent of script language information (defined before simulation starts). However, some of them need to be referenced or declared before simulation, because it can also affect autonomous crowds. For example, the events and reactions: an event can be a pre-determined fact (time-based defined in the script, e.g., frame 1000), or still can be activated through an external controller. In both cases, this event has to be declared and its parameters defined in

the script as well as the associated reactions. Table 3-4 describes the inter-dependence existing between these entities:

GUIDED ENTITIES	SCRIPT DEPENDENCE	GUIDED CONTROL
Actions	Not existing	Can just be applied by the guided crowd
Motion	Not existing	Can just be applied by the guided crowd
Events	Have to be declared in the script.	Can be activated by the external control. Can influence the autonomous crowd
Reactions	Have to be declared in the script.	Some parameters of reactions can be sent via external controller. Can influence the autonomous crowd
Status	A status changing can match other events	Can influence autonomous crowd
Environment Information	The guided crowd considers the parameters declared in the script.	The autonomous crowd considers the parameters declared in the script

Table 1-4: Inter-dependence between guided entities and scripted information.

Figure 1-11 shows some images of a crowd evacuation from a museum during a panic situation caused by a statue that becomes alive. Indeed, the statue is externally controlled, and the crowd initially obey the programmed behaviour: walk and visit the museum. Afterwards, when the external controller applies a motion to the statue, the crowd reacts according to the pre-programmed reaction in the script, so, exiting the museum through the two doors.

Figure 1-11: Scenes of simulation of evacuation due to a panic situation. Up left and right: Before the event, the crowd walks. Down, left and right: crowd reacts. The statue motion and action are externally controlled.

1.5.2.2 Interaction Paradigms

The interaction paradigms set different ways to interact or guide crowds. The exchanged information is basically classified in 8 types:

1. Selection: Selection of a group/agent to interact.
2. Motion: Defines new motion paradigm to the selected entity.
3. Action: Selection of an action to be applied.
4. State: Changes the internal status of group or agents.
5. Density: Increases or decreases the number of agents in the output area (camera's view).
6. Events/reactions: activates pre-programmed events and change event/reaction parameters.
7. Request: Requires about the selected entity.

8. Knowledge: Defines environmental data.

Each one of the paradigms presents different information that can be dealt in order to interact with a group or agent of the crowd. The next sections show more information about each interaction paradigm.

Selection paradigm

To manipulate with the groups/agents of crowd, it is necessary to select which group should be dealt with. This paradigm includes functions to select agents or groups. Afterwards, the selected entity might be manipulated with the others paradigms. The way to get information about the entities depending on the interface developed. For instance, a graphic interface can allow graphic functions like pick an agent, in order to select a group near to an object, or more placed in the right side of the output field of view. Considering this paradigm, we can also select agents depending on some conditions, for example, agents near to a specific object (location function), or agents which emotional status is HAPPY (emotional status function).

Selection paradigm:

Select an AGENT
Select a GROUP

Conditional functions to select entities:
~ Group/Agent status
~ Group/Agent location
~ Group/Agent goal

This paradigm defines a specific motion to be applied by the selected entity. Yet, the information defined in the script to inform autonomous crowds is also regarded by guided crowds in order to avoid collision with declared objects. Yet, locations and dimension of obstacles can be dynamically defined in the environment paradigm. If there is no selected entity, this paradigm is applied to everybody.

Motion paradigm:
~ Go to a specific location

Action paradigm

This paradigm sends actions to be applied by the crowd. These actions can be a body posture or interactions with objects [12]. Also, the action can be applied exactly at the moment sent by the external controller or be synchronised with the motion paradigm. For instance, send a motion task in order to reach a counter and after an action paradigm in order to buy a ticket. This paradigm is applied to the selected entity (agent or group), if there is not a selected entity, so, the action is applied to every agent of crowd.

Action paradigm:
~ Apply a body posture
~ Interact with an object

State Paradigm

This paradigm is responsible for setting parameters to change the internal state of agents and/or groups. Various parameters can be manipulated within this paradigm in order to set or change entity's state.

State paradigm (divided into three types of information)
~ Behavior data: group behaviors (flocking, following, adaptability, collision avoidance, repulsion, attraction and split) [MUS98]
~ Quantitative data: number of agents and list of agents.
~ Internal status data: emotional status, individual level of domination, way of walk, relationship with other groups/agents, etc. [MUS97]

Density Paradigm

During the simulation, it is possible to know the location of camera's field vision in order to control which agents and groups could be seen by the application and apply the selection paradigm. So, the density paradigm aims at changing the number of agents located in the output field.

Density paradigm:
~ Increase or decrease.

Events/Reactions Paradigm

This paradigm is responsible for the activation of crowd events and reactions. Events and reactions in our model consist of behavioural rules to be applied depending on the matched conditions. For instance, an event can be activated as a function of a request paradigm, status paradigm, or an autonomous behaviour specified in the script language. Events and reactions have to be declared in the script language (see Table 1-4), anyway some data can be generated in real time during the simulation as well as the activation of events. Normally, we have modelled events that are activated during running time through the external interface. For instance, a panic situation (Fig. 1-11) is declared in the script language as well as the consequent reactions. Anyway, the information about the fine moment when the panic situation occurs can be specified in real time using the event/reaction paradigm.

Events/reactions paradigm:
~ Activate or desactivate
~ Send specific data

Some information is needed to be accessible by the user in order to achieve the interaction paradigms with crowds in ViCrowd. Some examples of information of crowd can be: position, orientation (for crowd, group or agent), nature of group, emotion of group/agent, if group/agent is reacting to some matched event, goal of group, the current group behaviours, etc.

Environment Paradigm

Information about regions where we can walk, obstacles to be avoided and etc, can be defined in the script language before starting the simulation. The guided interface can define complementary information as well as re-define already described information.

1.5.3 Client/Server Architecture

Considering the many possibilities of interaction with ViCrowd, we decided to propose a multi-client architecture in order to provide several manners to communicate with the virtual crowd.

First of all, the protocol we used to provide the communication between the clients and the server in the context of real time animation is described. We have defined a server as an application responsible for sharing and distributing messages among the clients. One client is a specific application that can have more than one connection with the server. For instance, in Fig. 1-12, ViCrowd client has 3 connections (Group1, Group2, Group3) sharing the same protocol in order to communicate with a specific client (*RBBS client – Rule-based behaviour system*). An advantage of a Client/Server architecture is that we can distribute independent process on several computers as well as use other processes as "black boxes". As example, we can separate the knowledge about the virtual environment where the simulation occurs from the rendering client. These processes can run on different computers, and share information via the server. Each client does not have information about other clients except their input and output, so, they can be considered as black box.

As we have oriented the server for managing crowds, we assumed that each client requires a connection for each guided group to be manipulated. At the same time, each client sends information to the server in order to present the input and output that can be understood. This information is described using the protocol to inform the server about the instructions to follow (*responses for each stimulus*) when clients send messages (*stimulus*). Afterwards, the server is capable to understand from which client the message came, to which client it has to be redirected and which responses have to be employed in order to fit the stimulus.

Figure 1-11 shows an example of architecture formed by 6 different types of clients. Each client can have several connections (e.g. ViCrowd client has 6 connections) that are able to send and receive messages from the server in order to drive the guided groups. There are two types of connection: *request* and *group_name* connections. The first one represents connections accessible by all the connected clients. The *group_name* connection deals with information for specific groups, e.g., group 1 in ViCrowd Client.

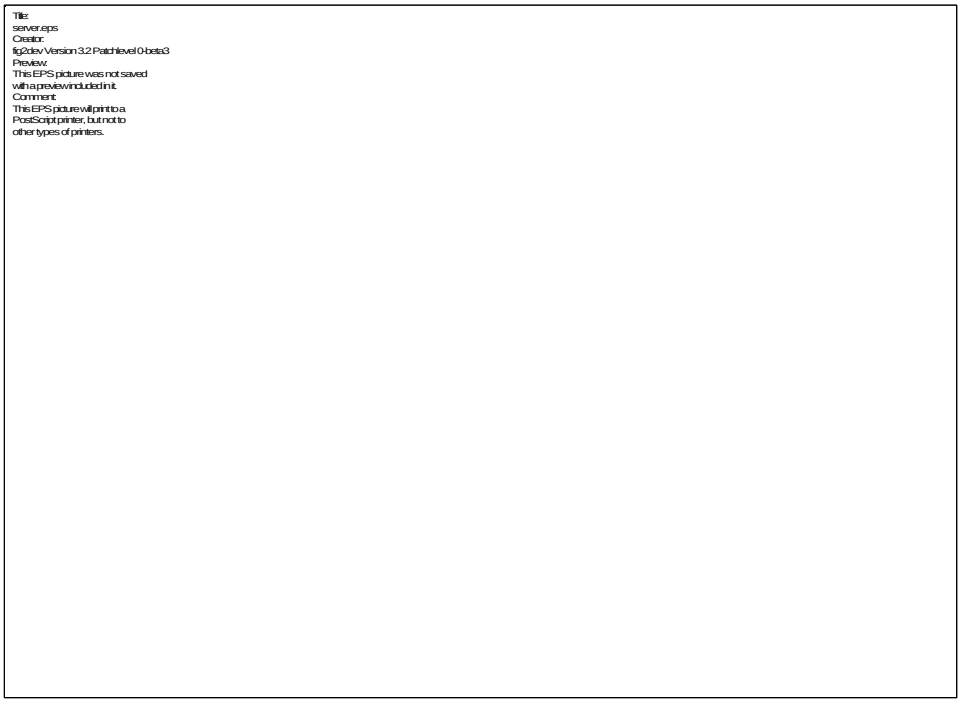


Figure 1-12: Various clients and connections within the architecture.

1.5.3.1 Protocol Information

As mentioned before, some information is sent to the server at the beginning of the simulation, in order to present each client. These information are classified in two different parts: *stimuli* and *responses*. Afterwards, using this definitions, the server is able to recognise the client from which the message is coming and what have to be performed for each message: redirect to other clients, translate some data, etc.

The parameters of the protocol are sent by each client in order to describe the following items:
1. Who is the client, describing the *type of client name* and the *group name* which has to be affected by the interaction (e.g., ViCrowd Client, group_1). In the case of generic connections (information to be accessed by all other clients), a special *group name* is sent in order to be identified as a *request* connection.
2. Which *stimuli* can be received from this client;
3. Which *responses* have to be applied by server or clients, in response of a specified stimulus. This protocol can be changed and re-loaded at any moment during the simulation, allowing a re-management of complex clients in real time.

One client can deal with more than one pre-defined stimulus. For each stimulus, the server receives associated instructions to be applied (*responses*).

Examples of stimulus/responses using the interaction paradigms

Example 1 shows the manipulation of a variable representing the emotional status of a specific group, using the *state interaction paradigm* (see Section 1.5).

Client Name: Textual User Interface	*(identifier of client)*
Group Name: Group1	*(identifier of group name)*
Stimulus: BE > $Emotion	*(Format of Stimulus sent by Client)*
Response: ViCrowd APPLIES STATE BE <$Emotion	*(ViCrowd changes emotional status of Group1)*
EndResponse: Stimulus)	(End of response associated to

For instance, when the Textual User Interface (TUI) client send stimulus: "BE HAPPY", “HAPPY” is stored in the variable $Emotion and afterwards sent to ViCrowd Client that changes the internal status of Group1.

The next example uses two sorts of interaction paradigms: *motion* and *request*.

Stimulus: GOTO SPECIFIC_LOCATION	*(Format of Stimulus sent by Client)*
Response: ViCrowd APPLIES **MOTION**_GOTO <$NAME AT 63150 -5000 -12540 1 0 0	*(ViCrowd activates Group 1 to walk until the specified position using Motion paradigm)*
Response: ViCrowd WAIT_FOR (**REQUEST**_GOALREACHED FOR <$NAME) TRUE	*(Request paradigm is used in order to verify the group location)*
EndResponse.	*(End of response)*

In this case, when TUI client sends “GOTO SPECIFIC_LOCATION” using the connection with group "Group1", the server sends to ViCrowd client: "**MOTION**_GOTO Group1 AT 63150 -5000 -12540 1 0 0 ", which uses a motion interaction paradigm using parameters to define position (63150 -5000 –12540) and orientation (1 0 0). The second response waits until ViCrowd client update the field "**REQUEST**_GOALREACHED FOR Group1". It only occurs when the group has finished the previous response.

1.5.4 Creating events in Virtual Space

We use the Events/Reactions paradigms (section 1.4.2.2) in order to provide the interaction of real and virtual people. In fact, we have defined an event-based architecture where RPI Client communicates events generated by real people and ViCrowd client receives this information from the Server in order to apply virtual events. So, each time RPI client sends an event to the server, this event is sent to ViCrowd client which treat this event, activating the appropriated reactions as programmed in the script.

To provide the integration between virtual crowd and real people, we have worked with two scenarios. Note that only one of these was evaluated in the public demonstration as discussed in section 1.5.3.

1.5.4.1 Scenario 1

In this scenario, we have included 20 agents distributed in a virtual museum. The virtual people are randomly distributed in the room and react differently according to the events generated by RPI client. Then, in this scenario, there are two types of mapping:

1. Mapping of space: depending on the region where the real events are generated in the real space, the virtual people that should react is mapped into the correspondent region in the virtual space. (see further details about mapping space in Section 1.5.4.3).

2. Mapping of events and activation of reactions: depending on the event generated in the real space (Section 1.4.1), virtual events are triggered and reactions are activated in virtual space.

Mapping of Events

Table presents the parameters handled with in the script language in order to activate reactions and mapping who have to react to triggered events. The event ID correspond to event name already presented in Table 1-1.

Event ID	Parameters in the Script Language	Programmed Reaction
RP_event0	WHO=ALL or EXT_INFO (R)	Welcome postures
RP_event1	WHO=ALL or EXT_INFO (R)	Waving postures
RP_event2	WHO=EXT_INFO (R)	Jump motion
RP_event3	WHO=EXT_INFO (R)	Reject postures
RP_event4	WHO=EXT_INFO (R)	Happy postures
RP_event5	WHO=EXT_INFO (R)	Sad postures
RP_event6	WHO=EXT_INFO (R)	Affraid postures
RP_event7	WHO=EXT_INFO (R)	Attacking postures
RP_event8	WHO= EXT_INFO (R) + direction of movement	Lookat <direction of movement> in a higher speed to reach the head motion (20 FPS)
RP_event9	WHO= EXT_INFO (R) + position of movement	Lookat <position of movement> in a higher speed to reach the head motion (20 FPS)
RP_event10	WHO= EXT_INFO (R) + direction of movement	Lookat <direction of movement> in a normal speed (50 FPS)
RP_event11	WHO= EXT_INFO (R) + position of movement	Lookat <position of movement> in a normal speed (50 FPS)
RP_event12	WHO= EXT_INFO (R) + variable	Positive reaction
RP_event13	WHO= ALL/EXT_INFO (R)	Lookat <position of movement> in a normal speed (50 FPS) AND/OR Dancing postures

Table 1-5: Script language information to deal with the triggered events

The <lookat> deals with a behaviour in order to create head motions in specific directions (FRONT, LEFT,RIGHT, UP, DOWN) correspondent of camera's position. The direction is given by the RPI client related to the region where real people events are generated. Following images show two real people configurations and the respective <lookat> behaviours.

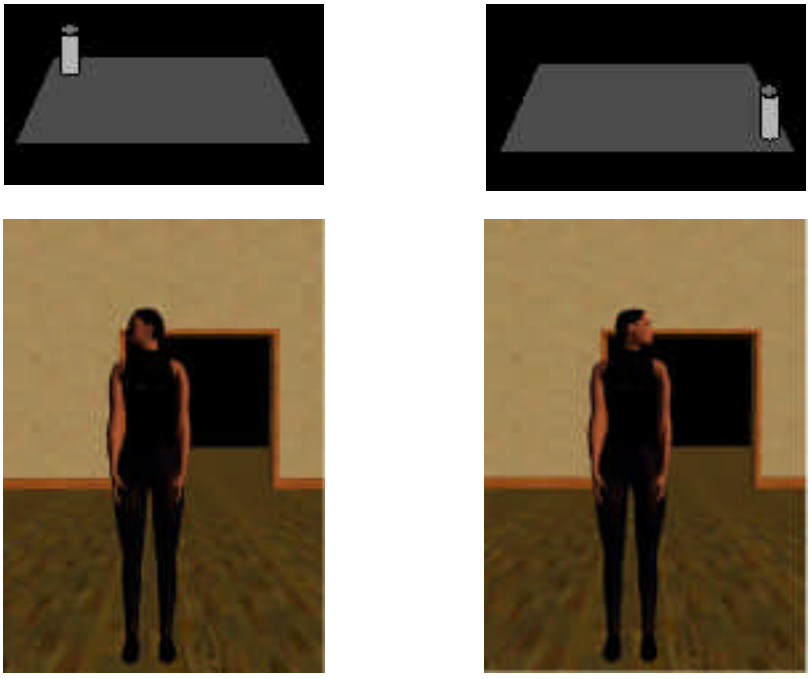
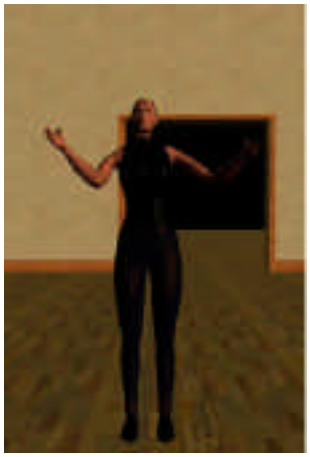
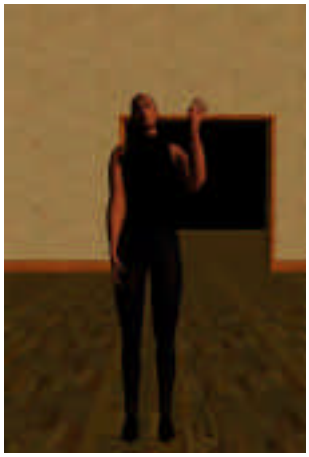


Fig. 1-13: (left-up) real people on the left of real space, (left down) virtual people reacting. (right up) real people on the right of real space, (right down) virtual people reacting.

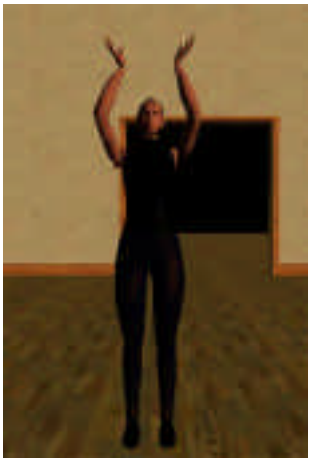
Concerning the reactions of virtual events, the current prototype included postures played in sequence and being interrupted if a new event is triggered. We have chosen the postures based on the meaning of events. For instance, RP_event0, which is triggered when a new person enters in the interaction space, activates the “welcome” animation. The sets of images 1-13 show some images of animation activated as a function of triggered events.



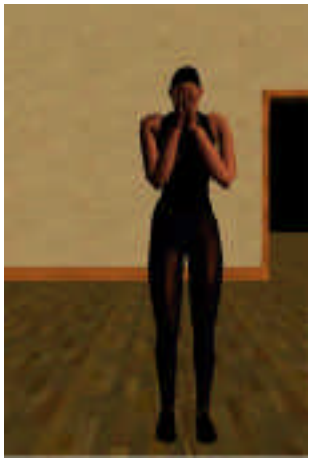
RP_event0: New person enters the interaction space



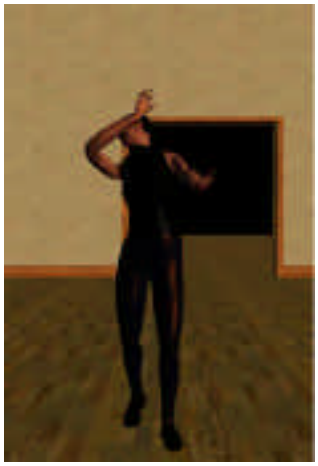
RP_event1: A person leaves the interaction space



RP_event2: Group is formed



RP_event3: Group is deleted



RP_event6: circle pattern formation
(Virtual agents are afraid)

RP_event7: line pattern formation
(Virtual agents attack)

Fig. 1-14: Sequences of animation according to various events

Mapping of Spaces

Considering the mapping of spaces, we considered the same dimensional universe in order to transform RPI client locations to ViCrowd client. Thus, the correlation between the spaces are direct as showing in next figure.

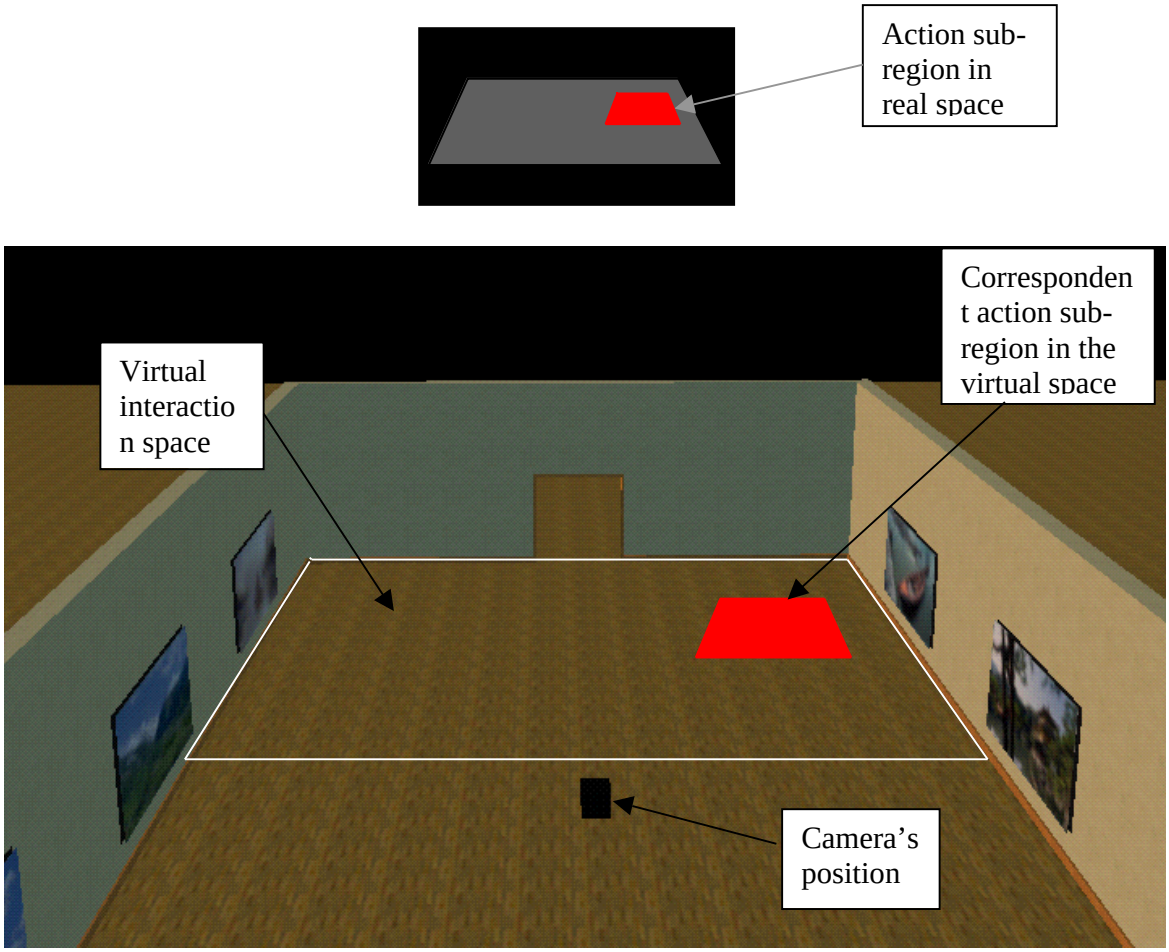


Fig. 1-15: Mapping of spaces in scenario 1

Some images of Scenario 1 simulation are presented in Fig. 1-16 and Fig. 1-17.

Fig. 1-16: A circle pattern is occurred in the interaction space. The virtual group in the correspondent region (marked with a square) in virtual space reacts playing a keyframe animation

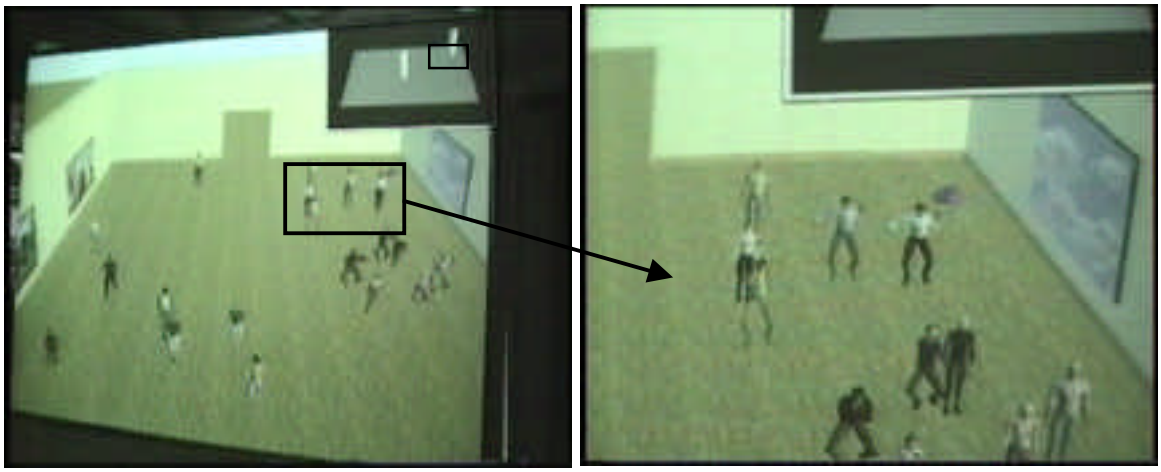


Fig. 1-17: One person is left the interaction space and the people react playing a waving keyframe sequence.

Because each tracked person in the real space could generate various events in different locations of real interaction space, and these events generate immediate response of virtual people, the feeling of interaction was not evident. Yet, the frame rate of simulation with 20 agents in a Reality Engine machine (256 MB RAM, 16 TRAM and 2 processors R4400) does not achieve real time processing, but 12 FPS. Added these difficulties, we decided to try another scenario in order to provide a better feeling of group interaction between real and virtual spaces.

Next section presents the changes made in the script language of Scenario 1 in order to apply Scenario 2.

1.5.4.2 Scenario 2

While Scenario 1 deals with 21 agents, Scenario 2 handles only 3 agents. Fig. 1-17 shows some images of this scenario.

Fig. 1-18: Some images of simulation of Scenario 2. Left, up: nobody is inside the real space. Right, up: a person enters the real space. Left, down: a group formed a line pattern. Right, down: a person exits the space.

This scenario provides the same events mapping as presented in Section 1.5.4.1. However, there is no space mapping in this scenario, then the three agents react in the same way according to the events generated by RPI client.

Following there are some images of real and virtual group interaction prototype running at ZKM.

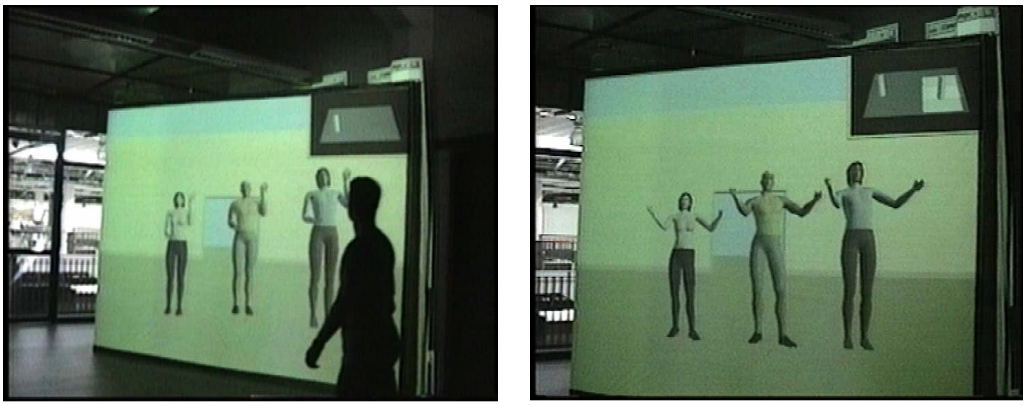


Fig. 1-19: (left) A person enters in the interaction space (the people are reacting because another one has just left). (right) People react because the person entered playing a welcome keyframe.

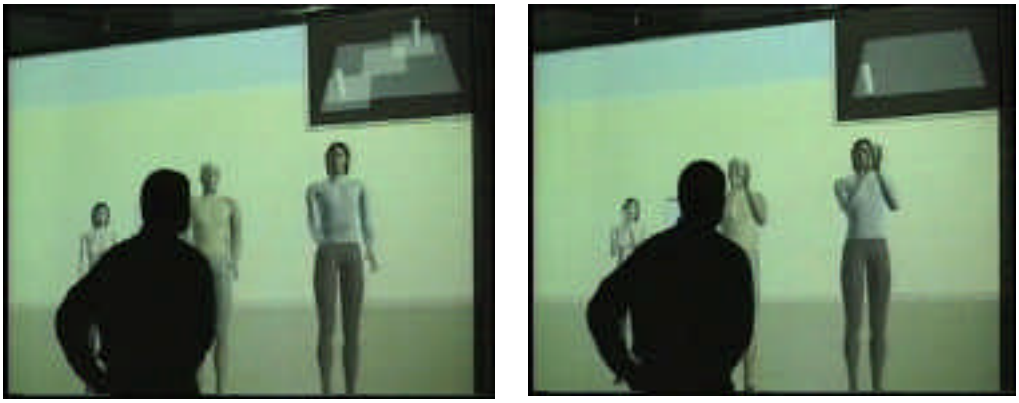


Fig 1-20: (left) A group is deleted from the interaction space, because someone is left. (right) Virtual people react negatively.

1.6 The Prototype Setup

1.6.1 Physical Setup

The prototype has been set up in an open area of about 8 by 8 meters in the ZKM | Institute for Visual Media. The camera was mounted on the ceiling above the interaction area shown in figure 1-21. People could enter from three sides: either through the hallway behind the projection screen or through one of two doors at each side of the interaction area. For the public presentation only the front door on the right side of the interaction area was used. By entering, a visitor would not get tracked immediately, but, would see the screen with the figures first. By approaching the screen the user steps into the interaction area where he or she gets tracked by the vision system.

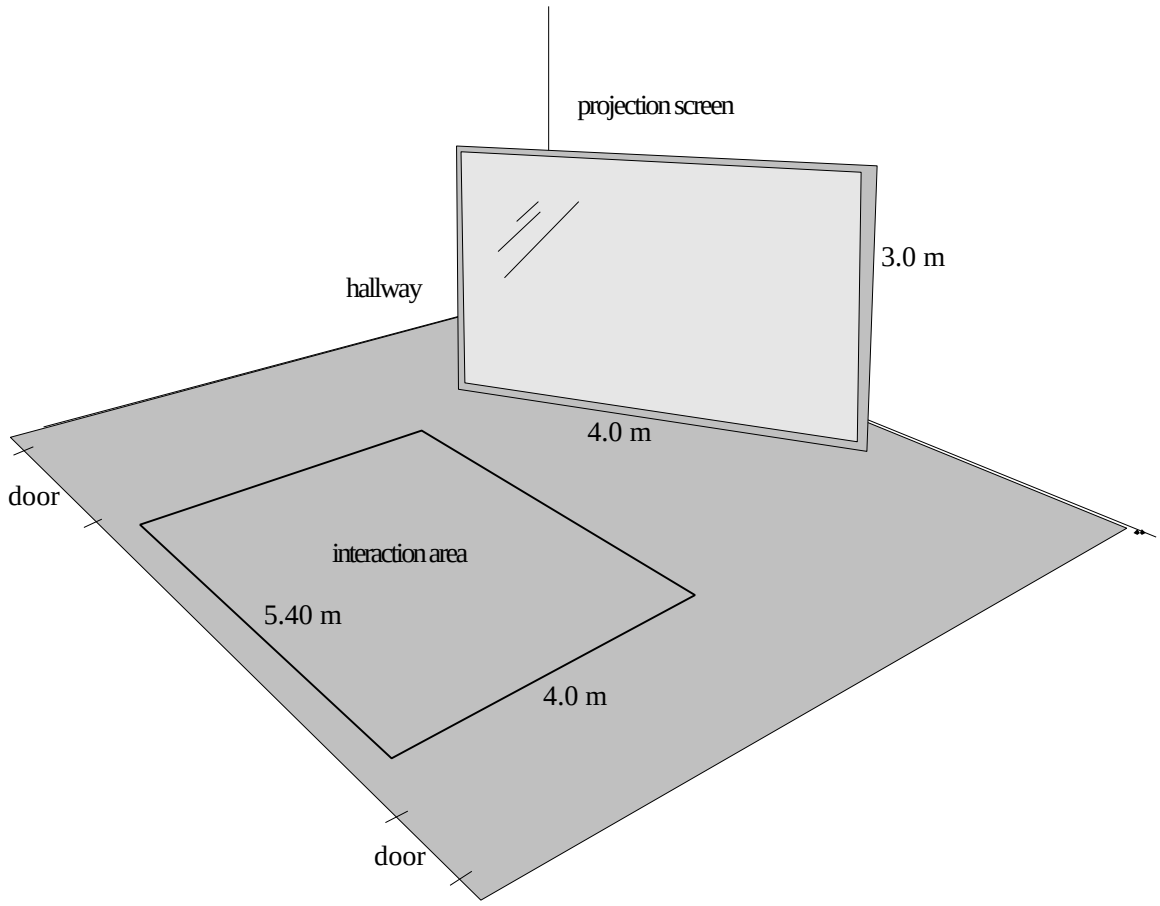


Figure 1-21: layout of the physical setup used for the public presentation

1.6.2 Software Setup

The software setup involves three main parts:

- Software to tracking people movement and events generated by real people (Mtrack and RPI)
- Client/Server in order to provide communication between modules
- ViCrowd client which treats events, define crowd behaviours and provide the display

Following image shows the architecture of the prototype.

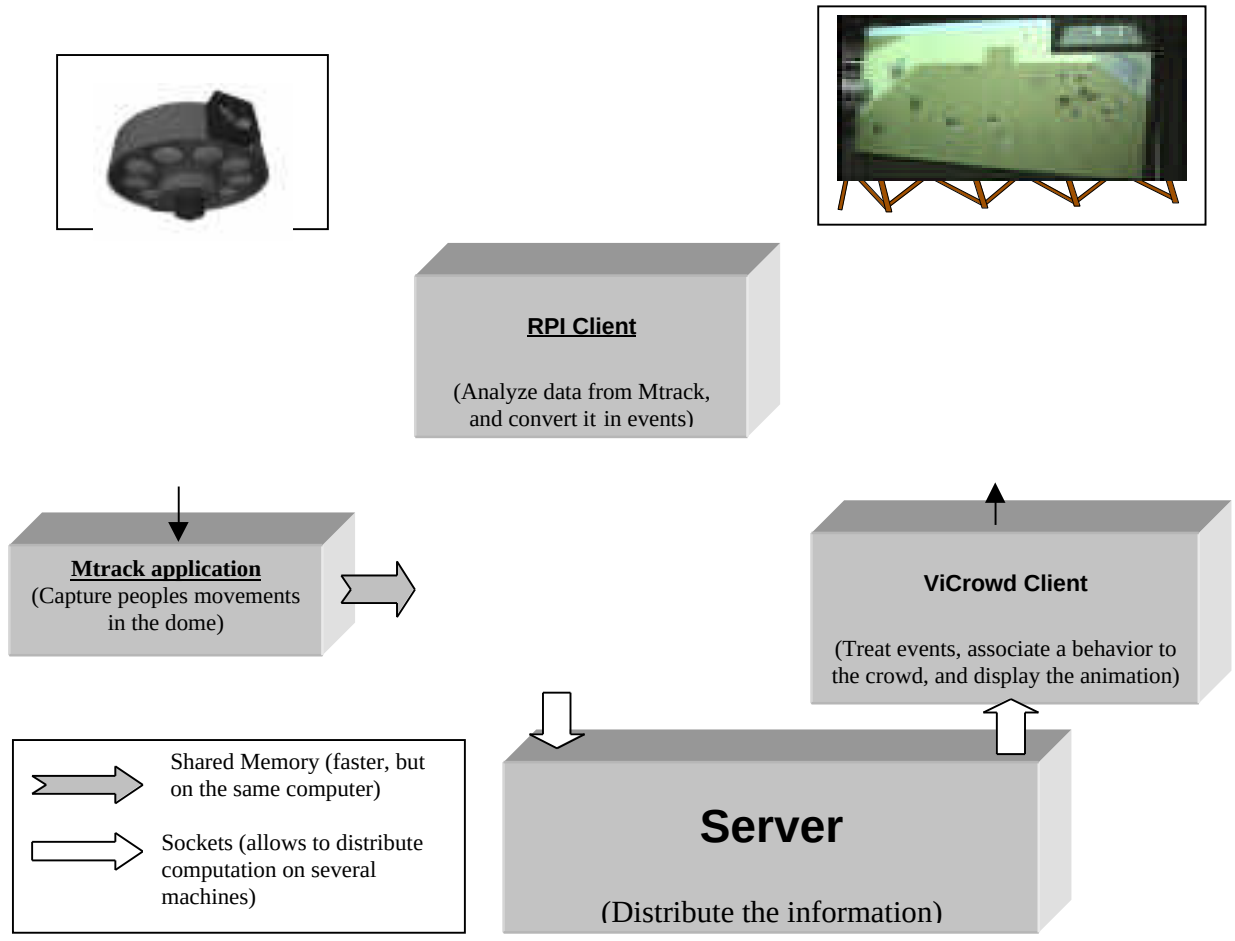
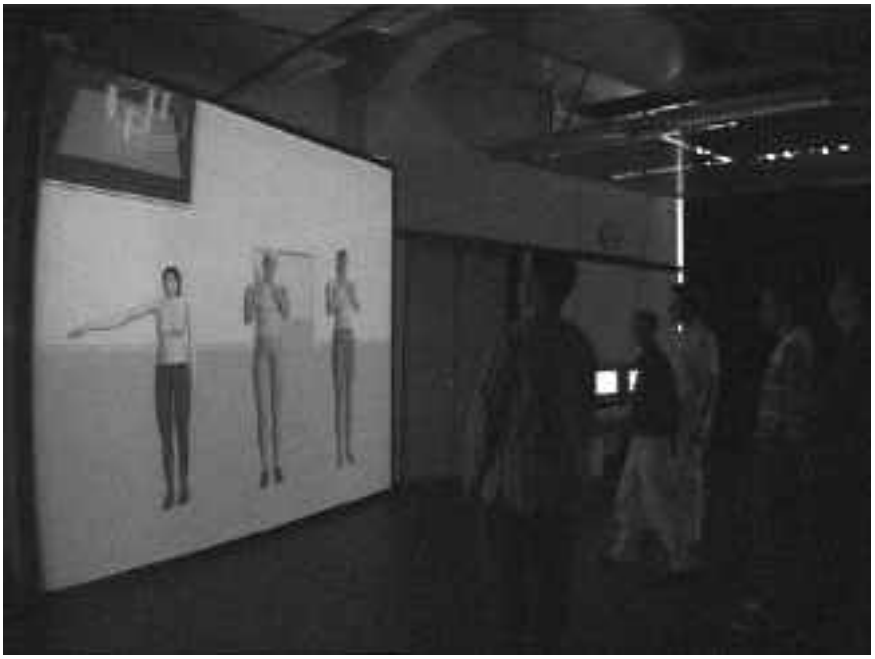


Fig 1-22: Architecture of prototype for communication of real and virtual people

1.6.3 Results

For getting first results from the prototype setup we made a public presentation for two days and observed and interrogated the people about how they could relate to the scenery. For the public presentation we set up a virtual scenery consisting out of three virtual human figures (see figure 1-23). The events in real space triggered animations of the three figures.



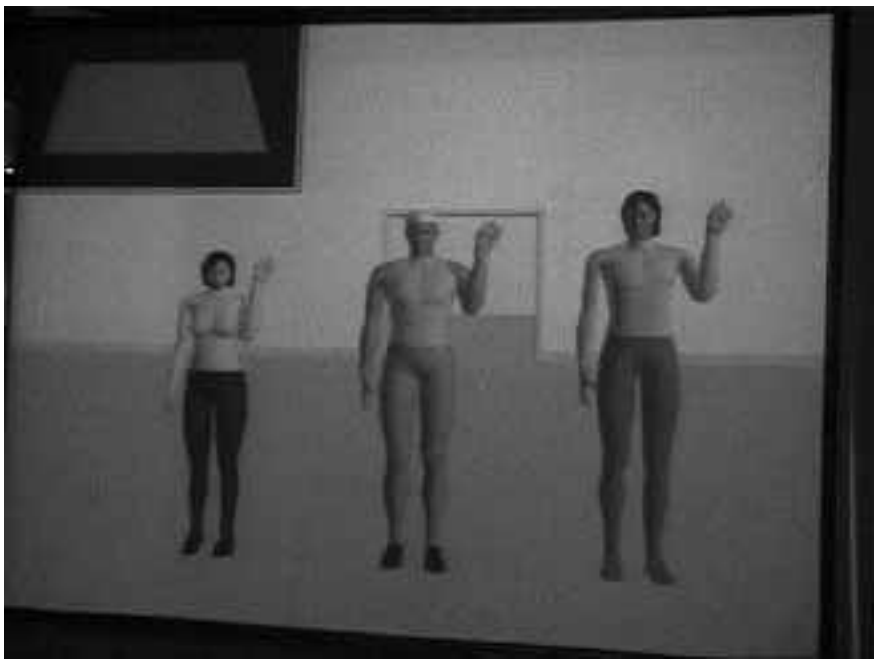


Figure 1-23: a) Prototype setup at public presentation,
b) virtual scenery with three human figures

The first reactions of almost all visitors showed that the interaction paradigm that uses an event- reaction method to trigger the animation does work very well. This is a clear indication that the contact paradigm might in work. We, however, encountered some limitations in the special setup, e.g. the physical interaction space was rather limited due to the height of the ceiling and the angle of view we could achieve with the camera. We also encountered that quite often people would try to approach the screen and, thereby, leave the interaction area (see figure 1-21). As the prototype was more of a technical demonstration there was not so much focus on either the communication side of the scenery, or on the interaction side between real and virtual, or on a kind of storyline. This, however, seems to be a critical issue, specially when presenting virtual humans. Some people disliked the fact that there was a random-like mapping between events in real space and triggered animations. They also had difficulties to relate to the scenery or to one of the virtual humans, because the animations that where triggered where performed by all virtual figures almost synchronously. Some visitors proposed a linking between tracked people in real space and one of the figures that will then play a partner role throughout the interaction. Also, some people had problems relating their movements in real space to the reactions. This is due to the fact that we did not supply any clues, e.g. marking the interaction space or subspaces in the interaction space that would have a corresponding marked area in virtual space that relates to it. Another suggestion was, that sound output or speech input that drives the graphics could enhance the paradigm.

The virtual scenery itself has been considered problematic because of the compromise we had to achieve between realism and fast enough real time graphics. Some people proposed to use more stylized figures or even stick figures that would be easier to relate to. On the other hand almost all visitors agreed that the confrontation with a virtual human is interesting situation with a certain amount of tension that also stipulates expectations on the visitor site. This is a contrast to the „Simple Prototype Setup“ were virtual fish formed the scene and this kind of tension was not mentioned. We clearly interpret these findings that the proposed paradigm would work, if the figures would have a greater variety of reactions and also some kind of autonomous behaviour. Also, using more than three people and having a linkage between a figure and a tracked individual either through a certain amount of time or using the spatial mapping technique between real and virtual space described above would overcome some of the biggest problems the visitors had. Adding a storyline and/or creating a more compelling background scene, e.g. adapting the background to the physical space the screen is located in or creating a kind of arena setup, was also mentioned by some people to be important.

For future work we like to overcome some timing and delay problems that arose due to the fact that events are only triggered at certain time steps. Furthermore, specially when more than two people are present it often happens that combination of events are triggered, i.e. an enter event and a movement event at the same time or a group formed event and a movement event at the same time. An ordering of the events in a prioritized list and adding some direct links between states of the RPI-client and the ViCrowd-client is planned. Another possibility is the combination of different modes of perception, i.e. adding sound output, speech input and/or adding other interfaces or objects in the interaction space and paradigm could be considered. Improving on the variety of the animation repertoire and adding autonomous behaviours will be considered in the next version.

In conclusion the prototype framework clearly showed the strength and weakness of the proposed approach for linking real and virtual people. It leaves some work for future work but already shows that the event-reaction paradigm works really well and a contact situation

between real people and virtual people might lead to an pleasing environment if some of the limitations of the simple scenery used for the prototype can be overcome.

1.7 Future Work

The research in year 2 has focussed on the possible interactions that can be generated between groups of real people and groups of virtual humans (and organisms). This has been done in an immersive visualisation environment where the real persons have been spatially tracked and the virtual objects have been algorithmically controlled to respond their movements.

It quickly became clear that the dynamics of group interaction is a specific problematic, quite different to the communication models that occur in the interaction between individual persons (or between just a few persons). Individual interactions are above all a psychological dynamic where (as well as speech) facial expressions and gesture play a dominant role. Needless to say the current state of virtual human technology does not yet permit a deep going interaction on the level that is equivalent to our human to human complexities of expression and communication. Group interaction on the other hand is a simpler more anonymous dialectic whereby the location of one's body in space, its proximity to other bodies, the patterns of group accession and their changing formations over time are the determining factors. In this domain of interaction the current state of tracking technology and virtual human generation seems to be remarkably able to articulate complex and satisfying scenarios of interaction. In fact the research in year 2 has arrived at the point where clear objectives can now be formulated for the year 3 workshop on this theme (WP 7b.1), which is mainly concerned with the programming of various scenarios of virtual group behaviour that can be meaningfully linked to various scenarios of real group behaviour. One of the more interesting challenges will be to discover methodologies whereby the virtual humans/organisms can actually inspire specific desired behaviours amongst the real people, and where the real people can inspire emergent behaviours amongst the virtual organisms whose social organisation and comportment may turn out to be governed by rules that are quite different to our own! This will remind us that the virtual domain is not necessarily a clone of our real world, in fact more interestingly it can be a foreign (even alien) territory in which we are given the opportunity to discover how to communicate with its indigenous life forms, and so, as in all such adventures into the unknown, learn something about ourselves. The paradoxical identity of the virtual humans, beings that are both familiar yet distant, make them ideally suited for this undertaking.

1.8 References

[1] Batman Returns (1992), Motion Picture, Warner Bros., 1992

[2] Benford, SD, Brown CC, Reynard GT, et. al. (1996), "Shared Spaces: Transportation, Artificiality and Spatiality", Proc. ACM Conference on Computer Supported Cooperative Work (CSCW'96), Boston, pp. 77-86, ACM Press, 1996.

[3] Benford SD, Greenhalgh C.M, and Lloyd D (1997), "Crowded Collaborative Virtual environments". Proc. ACM Conference on Human Factors in Computing Systems (CHI'97), Atlanta, Georgia, US, March 22-27.

[4] Blase C, Morse M, Blunk A, et. al. (1997), ”Hardware, Software, Artware : Confluence of Art and Technology. Art Practice at the ZKM Institute for Visual Media 1992 - 1997”, Cantz Verlag, 200 pages + CD-ROM.

[5] Cliffhanger (1993), Motion Picture, TriStar Pictures, 1993

[6] Cruz-Neira C, Sandin D J, DeFanti T A, ”Surround-screen projection-based virtual reality: The Design and Implementation of the CAVE”, Computer Graphics Proceedings, Annual Conference Series, volume 27, August 1993, pp 135-142

[7] Farenc, N; et al. “A Paradigm for Controlling Virtual Humans in Urban Environment Simulation”. Applied Artificial Intelligence Journal. Special issue on Artificial Intelligence. 1999 (to appear).

[8] Goffman E (1971), “Relations in Public: Microstudies of the Pulic Order”, Basic Books.

[9] Hoch M (1997), "Object Oriented Design of the Intuitive Interface", 3D Image Analysis and Synthesis, Proceedings, Erlangen November 17-19, Infix 1997, pp 161-167.

[10] Hoch M (1998), "A Prototype System for Intuitive Film Planning", Third IEEE International Conference on Automatic Face and Gesture Recognition (FG'98), April 14-16, Nara, Japan 1998, pp 504-509.

[11] Hoch M, Schwabe D (1999), "Group Interaction in a Sourround Screen Environment", Proc. IEEE Computer Animation '99, Geneva, Switzerland, MAY 26-28, 1999.

[12] Kallmann, M. and Thalmann, D. “Modeling Objects for Interaction Tasks”. Proc. Eurographics Workshop on Animation and Simulation, 1998

[13] Krüger W, Fröhlich B (1994), ”The responsive workbench”, IEEE Computer Graphics and Applications, May 1994, pp 12-15.

[14] Merleau-Ponty M (1992), “Phenomenology of Perception”, Routledge.

[15] Mulan (1998), Animated Motion Picture, Walt Disney Productions, 1998

[16] Musse, S. R., Thalmann, D. (1997), “A Model of Human Crowd Behaviour: Group Inter-Relationship and Collision Detection Analysis”. Proceedings of Workshop Eurographics Computer Animation and Simulation. Budapest, Hungary, 1997.

[17] Musse S. R., Babski C, Capin T, Thalmann D (1998), “Crowd Modelling in Collaborative Virtual Environments”, ACM VRST‘98, Taiwan.

[18] Renault O, Magnenant-Thalmann N, Thalmann D, “A vision-based approach to behavioural animation”, Visualization and Computer Animation 1, 1990 pp 18-21

[19] Reynolds C W (1987), Flocks, Herds, and Schools: A Distributed Behavioural Model, Computer Graphics, Volume 21, Number 4, July 1987

[20] Stanley and Stella (1987), Computer Animation, Electronic Theater, SIGGRAPH 1987

[21] Stary C (1996), “Interaktive Systeme. Software-Entwicklung und Software-Ergonomie”, Vieweg Informatik/Wirtschaftsinformatik 1996.

[22] The Lion King (1994), Animated Motion Picture, Walt Disney Productions, 1994

[23] Tu X, Terzopoulos D, “Perceptual Modeling for Behavioural animation of fishes”, In Proc. 2nd Pacific Conf. on Computer Graphics, Bejing, China, 1994

[24] Tu X, Terzopoulos D, “Artificial Fishes: Physics, locomotion, perception, behaviour, Computer Graphics Proceedings, Annual Conference Series, volume 28, August 1994, pp 43-50s

[25] Frécon E., Stenius M., DIVE: A scaleable network architecture for distributed virtual environments, Distributed Systems Engineering Journal (DSEJ), 5 (1998), pp 91-100, Special Issue on Distributed Virtual Environments

[26] Hagsand O., *Interactive MultiUser VEs in the DIVE System*, IEEE Multimedia Magazine, Vol 3, Number 1, 1996

[27] Yoder L., *The Digital Display Technology of the Future*, INFOCOMM ´97, June 1997, Los Angeles

[28] see http://www.ti.com/dlp

[29] see http://www.synelec.com

[30] see http://www.sics.se

2. Activity-Oriented Navigation

Kai-Mikael Jää-Aro (KTH), John M. Bowers (KTH), Sten-Olof Hellström (KTH)

Conventionally, navigation and manipulation in virtual environments is based on geometry, i.e. the user moves his avatar through space by somehow indicating the direction and speed of the desired motion. In some systems this functionality has been extended with the ability to approach given objects in the scene [36][37].

We propose that navigation would benefit from *activity-based* interaction, in which higher-level information about the behaviour of the inhabitants in the environment is used to guide the motion of the user('s avatar) through space. The emphasis is shifted from commands of the type "Move me three units forward!" to "Take me somewhere interesting", where "interest" is not defined in terms of predefined properties of landmarks or objects in the environment, but on the real-time behaviour and activities of the other inhabitants in that space.

We can see a trend of *agency* in these navigational methods:

- The most direct is full six-degrees-of-freedom navigation – the user has total control over his navigation, but also has to perform the navigation every step of the way to get anywhere.

- More commonly, some kind of *vehicle* is used to constrain the motion of the avatar, adapted to the environment such that "topologically meaningless" navigation is prevented and/or "meaningful" navigation encouraged, e.g. the avatar may be restricted to keep its feet on the ground plane.

- The user may indicate an object in the scene (either by selecting it directly with a pointing device, or by typing its identifier in a dialog box) and let the system take care of the actual transport to that object.

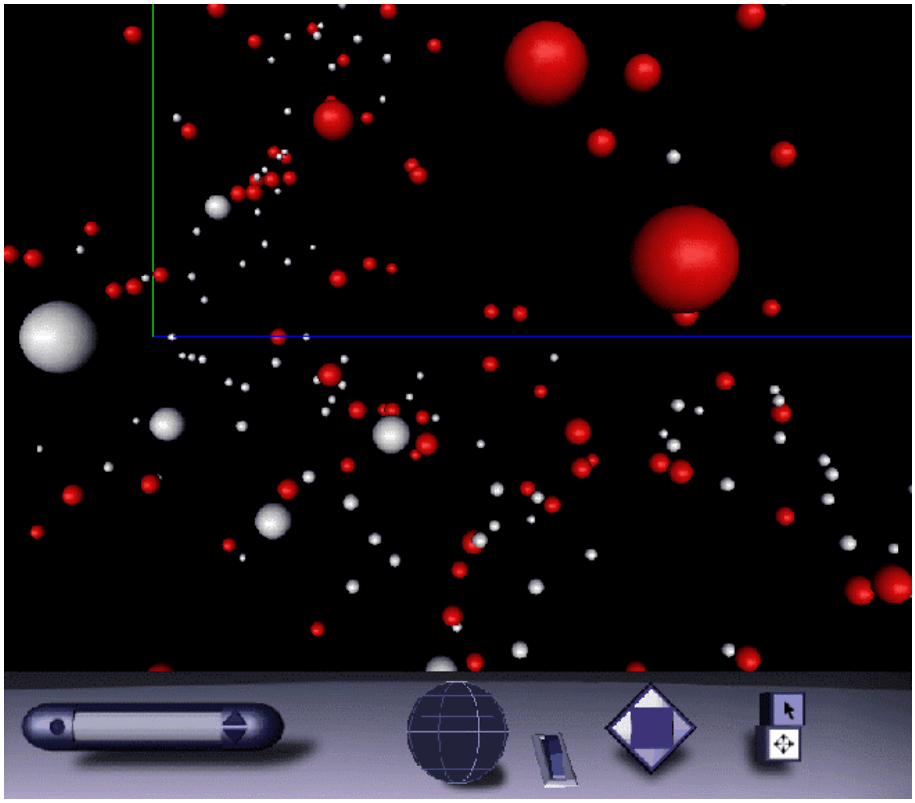


Figure 1: The CosmoPlayer VRML browser, currently owned by Viewpoint Digital. The user has at the bottom a set of navigational devices, from left: A list of fixed positions in the space which the user can be transported to (none available in this world); a sphere which lets the user rotate the world around two axes; a wheel which lets the user move towards or away from the centre of the world; a panning device that lets the user translate the world along two axes and, at the far right, a mode toggle which interprets clicks on objects as either directions to load the hyperlink embedded in that object, or as a command to approach that object.
In a menu not visible in this image, the user can chose to be either teleported to the selected target or transported along a smooth path. That the navigational degrees of freedom are broken up on several separate widgets is of course necessitated by the mouse as an interaction device, but it is often perceived as simplifying navigation, at least when the world is projected on a 2D surface.

Our proposed navigational method would go even further in indirection, in that the user does not necessarily have to know where the system will take him when giving a command. However, we hasten to point out that we acknowledge the need for all the other forms of navigation – we do not intend to replace them, but to complement them with methods that are more task-oriented. All the arguments against agents apply.

To begin with we need to distinguish between activity-oriented navigation *in the large* and *in the small*. The former is coarse-grained navigation, with the intention of finding some interesting place, the latter is motion of the avatar with social significance within a group.

2.1 Some Definitions of Activity

We have spoken without qualifications of *activity*, since a more precise definition of what activity *is*, is dependent in the actual application, we may even suspect that an inhabitant of a shared VE may have needs of different activity measures at different times. We can still suggest a few activity measures, which we have used in our own work:

Aggregations of participants

In the Spatial Interaction Model, all participants (as well as certain artefacts) extend a *nimbus* function into the environment, which represents their presence in that space. All participants also have a *focus* function, which represents their observational ability of space. Areas where the sum of participants' focus functions is high are places where many people want to make themselves known. Areas where the sum of nimbus functions is high are places closely observed by many participants. Either of these criteria could be used as a definition of "activity". Nimbus computations could be done for undifferentiated space as a whole, but we may also perform *awareness* computations, i.e. compute the product of the focus of one participant/artefact and the nimbus of another, in which case we compute these values only for the participants.

Communication

As we mentioned before, social VEs are for the express purpose of supporting communication between people. A fair indicator of activity is then the amount of communication measured in, say, the number of bytes transmitted per time unit, or participants simultaneously talking in a given region. ("Talking" may of course be through typed text in VEs thus limited.)

Manipulation of the environment

In VEs which contain objects which can be interacted with—clicked on, turned, read, etc – we can find the regions/objects with the highest number of such interactions, either per time unit, or in total.

2.2 Navigation in the Large

For the purpose of the present project, shared virtual environments are *social* environments, i.e. the inhabitants can be presumed to be explicitly interested in getting in contact with each other [34]. Therefore it makes sense to add functionality to the VE interface to allow people to find gathering places. These may be explicitly created, as the market squares and town halls of the VE, as it were, but we also want to be able to detect the impromptu meetings, the rave parties, the illegal demonstrations and other activities that occur without a predetermined time and place.

The Spatial Interaction Model [31][38] gives us a framework for detecting such aggregations of activity, as has been detailed in Deliverable 4.3. In addition, the *Æther* extensions of the Spatial Model [39] that support diffusion of awareness over time might let us find places where there *has been* activity – useful for virtual can-and-bottle collectors?

We will suggest a few different ways in which activity oriented navigation can be supported in shared virtual environments.

2.2.1 Direct Activity Displays

Many computer games require situation awareness and thus display maps or "radarscope" displays of the activity of the other players, i.e. projections of the space from above with (even further) simplified representations of the participants and possibly also artefacts in the environment and the participants' actions on these.

Figure 2: A scene from Super Maze Wars, by Callisto Corporation. The "radarscope" below the main window lets the player see where the others are and take appropriate action.

As described in Deliverable 4.3, we have ourselves done similar visualisations, directly aimed at displaying the activity of the population of a shared VE, in this case for the purpose of deploying virtual camera crews to presumed interesting scenes in inhabited TV settings.

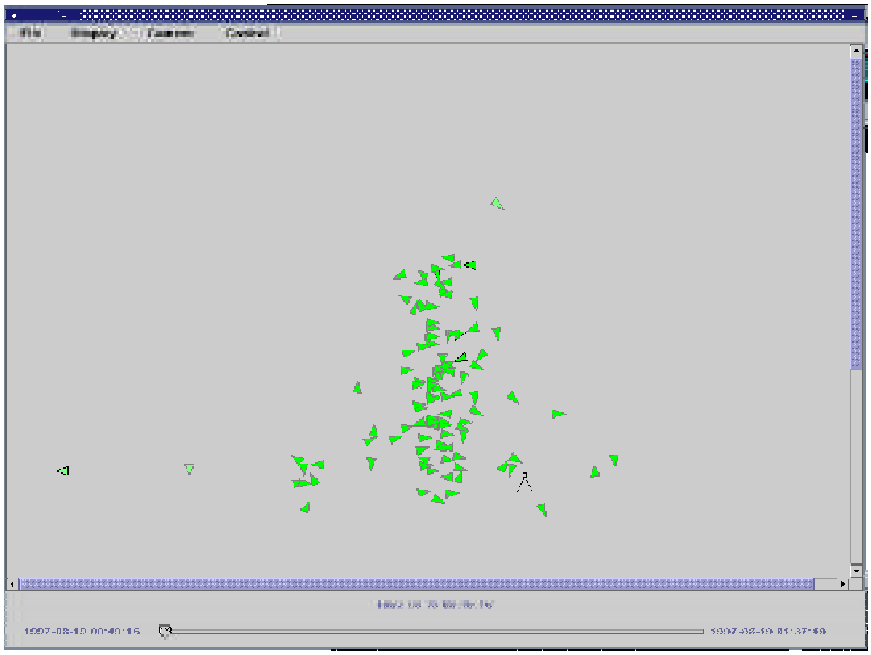


Figure 3: An overview of a virtual environment.

There are a number of constraints on such radarscope displays:

- They have to be inconspicuous, either by being fairly small or by being easy to hide and retrieve when needed.
- If small, it must nevertheless be possible to see enough detail that activity will not be undetectable.
- They must be easy for the participant to orient himself in, regardless of rotations and translations.

For large VEs, it can be difficult to display the entire space in sufficient resolution to be able to discern activity. One can then use "fish-eye" presentations of the space, so that the immediate surroundings of the participant are rendered in high detail, while areas further away have more aggregated views of activity [35][40]. Fish-eye displays of this type are uncommon in VEs, but Mel Slater and others have used the technique in order to simulate on small-field-of-view displays so as to give users a better view of their immediate surroundings [42]. While it is necessary to let the participant scroll this display in order to find interesting areas, this must be done in such a way that he immediately can tell his own position in relation to the high-resolution area of the display.

With such an overview display, the user can move to an interesting area, simply by indicating it in the display. This motion could be either animated or a "teleport", an immediate translation to the indicated position. Bowman *et al* [33] caution that teleportation leads to disorientation, but their research was done in an immersed environment, where the users could see only their immediate surroundings and had no overview of the environment. In addition we note that it often is fairly easy to get disoriented in most virtual environments anyway, due to the impoverished location cues and identifying features of spaces – teleporting between familiar locations in a virtual environment should lead to less disorientation. The

map display we have posited should further minimise disorientation, as start and ending points are clearly visible on the map.

Teleportation is clearly faster than an animation. For some applications environments a relatively slow animation may be useful if it allows the user to interrupt the motion if something turns up along the way. A further point we wish to make is that there are certain social problems with suddenly appearing, uninvited, in the centre of a group; an animation is easily brought to a halt at a respectful distance, whereas a teleport preferrably should include a "shimmer-in" period, which makes the group aware of a new arrival slightly before full communication is established. This will of course offset the speed advantages of teleportation to some extent.

2.2.2 Gradients

Another possibility is the use of *gradients*, i.e. the user receives an indication of the *direction* to higher activity. This can be as simple as an arrow indicating the direction of (higher) activity; but we could also use aural means of indicating activity—in an environment in which the sounds of the participants are audible, the user can steer towards increasing sound levels, this can be further enhanced by stereo or three-dimensional sound 0.

With the appropriate controls, the user can move towards—or away from—an activity centre at adjustable speed.

Now, since activity changes over time, both global and local maxima may change over time. Therefore it may be more appropriate to define these maxima in terms of *groups*, the disjoint sets of participants that contribute to that activity maximum, and as long as these groups remain coherent (according to some suitable metric), the navigational controls should steer towards these groups instead of the exact maxima in the activity map. There may be several local maxima, and a navigation tool should allow the user to test these groups in order. The user may thus instruct the navigation to go to the next lower-activity group.

2.2.3 Time

The *Æther* extensions of the Spatial Interaction Model [39], among other things, incorporate diffusion of nimbus over time in addition to space. This gives the user the possibility to search for activity in the (recent) past as well as in the current time, by extending his focus in time. In this we may look for places where there *has been* activity, perhaps by looking at the left-behind traces of those who have left. We note that time, as a dimension, does not have quite the same properties as space: the user cannot move arbitrarily through time in a real-time environment—we can define environments where motion backwards in time is possible, but those seem unlikely to overlap with the type of virtual environments for social interaction we discuss here. Neither can the user extend his focus into positive time, i.e. into the future, unless we use some kind of predictive system, which however, as is a property of such, would be subject to correction over time, and therefore could not be trusted unconditionally. (Of course, since the entire idea of activity-oriented navigation is based on heuristics, corrections to the automatic functions would be commonplace anyway.)

2.3 Navigation in the Small

Once the user has arrived at an activity hotspot, whether by the methods outlined above, or conventional spatial-oriented navigation, he will likely be interested in communicating with those present there. Apart from the pure verbal (or textual) communication, this will also contain elements of avatar motion. As shown by previous studies [32] this is a quite important factor, even with quite impoverished avatars.

One of the most important behaviours we have perceived is facing another avatar. Two persons intending to speak to each other, almost invariably precede this by turning their avatars to face each other. Events in the environment may also induce users to glance at each others' avatar, i.e. turning to face an avatar, then immediately returning to the original orientation, in order to ascertain the effect of the event on the others in the group. Thos faced are not picked randomly they are normally the friends, team mates, or people otherwise important to me.

The functions we feel can be automatically supported are

- Turning towards the person currently speaking in the group.
- Glancing towards other members of the group to check their attentiveness.

2.4 References

[31] Benford, Steve, Bowers, John, Fahlén, Lennart E., and Rodden, Tom (1994).
A spatial model of awareness.
In Björn Pehrson and Eva Skarbäck, editors, *Proceedings of 6th ERCIM Workshops*. ERCIM, June 1994.

[32] Bowers, John, Pycock, James, and O'Brien, Jon (1996).
Talk and embodiment in collaborative virtual environments.
In *CHI '96: Human Factors in Computing Systems: Common Ground*, pages 58–65.
`http://www.acm.org/pubs/articles/proceedings/chi/238386/p58-bowers/p58-bowers.html`.

[33] Bowman, Doug A., Koller, David, and Hodges, Larry F. (1997).
Travel in immersive virtual environments: An evaluation of viewpoint motion control techniques.
In *Proceedings VRAIS '97*, pages 45–52, 1997.

[34] Electric Communities (1995).
Commerce and society in cyberspace, 1995.
`http://www.communities.com/company/papers/commerce_n_society/commerce_society.html`.

[35] Furnas, George W. (1986).
Generalized fisheye views.
In *CHI '86: Human factors in computing systems*, pages 16-23. SIGCHI, April 1986.

[36] Mackinlay, Jock D., Card, Stuart K., and Robertson, George G. (1990).
Rapid controlled movement through a virtual 3D workspace.
Computer Graphics, 24(4):171–176, August 1990.
Proceedings SIGGRAPH '90.

[37] Mohageg, Mike, Myers, Rob, Marrin, Chris, Kent, Jim, Mott, David, and Isaacs, Paul (1996).
A user interface for accessing 3D content on the World Wide Web.
In *CHI '96: Human Factors in Computing Systems: Common Ground*.
`http://www.acm.org/pubs/contents/proceedings/chi/238386/index.html`.

[38] Rodden ,Tom (1996).
Populating the application: A model of awareness for cooperative applications.
In Mark S. Ackerman, editor, *CSCW '96: Cooperating Communities*, pages 87–96, November 1996.
`http://www.acm.org/pubs/articles/proceedings/cscw/240080/p87-rodden/p87-rodden.pdf`.

[39] Sandor, Ovidiu, Bogdan, Christian, and Bowers, John (1997).
Aether: An awareness engine for CSCW.
In John A Hughes, Wolfgang Prinz, Tom Rodden, and Kjeld Schmidt, editors, *Proceedings of the Fifth European Conference on Computer-Supported Cooperative Work*, pages 221–236, Dordrecht, Boston, London, September 1997. Kluwer Academic Publishers.

[40] Sarkar, Manojit and Brown, Marc H. (1992).
Graphical fisheye views of graphs.
In *CHI '92: Human Factors in Computing Systems: Striking A Balance*, pages 83–91. SIGCHI, May 1992.

[41] SIGCHI (1996).
CHI '96: Human Factors in Computing Systems: Common Ground, April 1996.
`http://www.acm.org/pubs/contents/proceedings/chi/238386/index.html`.

[42] Slater, Mel and Usoh, Martin (1993).
Simulating peripheral vision in immersive virtual environments.
Computers and Graphics, 17(6):643–653, November/December 1993.

Wenzel, Elizabeth M. (1992).
Localization in virtual acoustic displays.
Presence: Teleoperators and Virtual Environments, 1(1):80–107, 1992.

3.VIDI:
A Proposal for a Simple Technology to Make Video Supportive of Individual and Group Interaction in Electronic Arenas, Together with some Initial Development Experience

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

Jason Morphett and John Odlin,
BT Labs, Martlesham, UK

3.1. Introduction

This document briefly discusses VIDI, a simple technology to support using video input as a means for making individual and group interaction available in an electronic arena. Much of what exists in this document concerns proposed features of the application we are collaborating on, though we have developed a preliminary version of VIDI which has undergone some testing in a performance setting.

Our interest in using video as a means to build interactive technologies arises from a number of considerations. First, Deliverable D7.a1 presents an extensive discussion of the development of 'Wobblespace' and its various incarnations. In *Out Of This World*, for example, Wobblespace was used at two junctures to enable audience interaction with some features of the events in the electronic arena. As described in that deliverable, in the main part of the show, Wobblespace made use of analyses of luminance change in the image from a video camera directed at the audience to determine the progress and outcome of a voting process. We wanted to build on this application by simplifying it for general purpose uses. Indeed, we intend our VIDI application to be usable by a very broad range of persons in different settings where some form of unencumbered interaction with events is required.

Second, we are attracted to the fact that video input 'naturally scales'. A camera can be placed in front of a small object or a large one. A camera can be responsive to a finger or whole body movement. A camera can pick up on activity in one individual or in a group. As such, it seemed that video mediated interaction might enable a seamless pathway between individual and group interaction.

Third, we wanted to develop a technology that would complement the work on mTrack at the ZKM and, the version of it in use at GMD, marsTrack. These systems work towards object tacking by convolving the image in various ways and carrying out computations on the basis of the pattern of luminance in the image. As an alternative, we wanted to work with simple luminance analysis with tracking. The reason for this is our belief that tracking systems are often unnecessarily powerful for the tasks they are sometimes put. If, for example, one merely needs to detect activity in a portion of the image and trigger some event consequent upon that, it may well not be necessary to actually track objects in that region. Simple luminance analysis may suffice.

Fourth, putting all these intentions together, we felt that a simple luminance analysis program which makes its results available in readily distributable form and that is targeted on

simple ‘domestic’ hardware requirements (a conventional PC with a desktop-style, low resolution ‘webcamera’) would give greatest access to video mediated interaction.

3.2. Specifying VIDI

To prototype our development of Wobblespace, we decided to construct VIDI which conducts elementary luminance analyses and normalises these for transmission as MIDI (Musical Instrument Digital Interface) controller and note event data. This enables experiments on interaction with sound to be readily performed in development. We also envisage that the control of sound through unencumbered performer gesture is a likely destination for VIDI. We will not leave VIDI only able to speak MIDI. The ultimate intention is develop VIDI as a luminance analysis server which can communicate analysis results remotely, perhaps via tcp/ip, to any application which has need of such data. In this way, the load of some kinds of elementary peripheral image processing will be removed from the client. The remote delivery of information about interactive behaviour is something which we envisage will need to be supported in electronic arenas.

Wobblespace as developed for *OOTW* performed a series of luminance analyses by different colours while conducting the analysis over the entire image. To calibrate Wobblespace in *OOTW* proved troublesome in some respects as documented in Deliverable D7a.1. This was in an environment where a reasonable degree of control could be exercised over ambient lighting and the colours of interaction devices such as sheets of paper. In a uncontrolled domestic environment, for example, it is hard to imagine that the remote coordinator of an electronic event would have more control over ambient lighting conditions, almost certainly less. Accordingly, we simplified the design of VIDI still further so that only a global luminance value (light/dark) forms the subject of analysis. So effectively we analyse for greyscale levels and not by separate colours.

To give VIDI flexibility in the face of these severe restrictions, we allow the user to define a set of regions in the video frame, and define streams of reports on luminance data to be output from the application, one for each region. At the moment regions are constrained to be rectangular but we could implement more freely drawn regions if a clear need arises. Regions can show any pattern of interrelations between them. They can be discrete or overlap. They can include only a few pixels or they might include the whole image. A ‘background’ region is defined as a default consisting of all the video frame not included in the other defined regions.

VIDI should be able to report on either normalised luminance level for any region (computed by averaging the luminance over the whole region) or normalised luminance change for any region (computed by measuring frame on frame changes and normalising them as a proportion of the maximum change possible in the region). Each region should be set to either report on luminance change or luminance level. It ought to be possible to momentarily ‘mute’ or ‘solo’ by analogy with these facilities on sound mixing desks.

These normalised values should be subject to further rescaling so that, for example, low level fluctuations might still lead to big changes in the output if that was more meaningful to the client application. Our current idea for a rescaling function allows the normalised luminance or luminance change level to be clamped by setting a range between two values on the function’s ‘input side’ and mapping this range to an output range which is defined (in our ‘musical’ version of VIDI) in terms of the desired MIDI range the output (i.e. it is mapped to the standard 7-bit resolution of MIDI).

Output can be naturally expressed as MIDI controller data and facilities should exist for setting the controller number and output MIDI channel. Another useful facility is a ‘frame skip’ value which thins the data analysis so that, for example, MIDI is only output every so many frames of video input and not at the maximum rate of a new value per frame.

We also implement MIDI note data to be input. A MIDI note event can be of two sorts: a note on to designate the start of a note, a note off to designate its end. A note on event has a note number (conventionally interpreted as a pitch value on an equal tempered chromatic scale), a velocity between 1 and 127 (a representation of how fast the note’s onset is), and a MIDI channel number. Velocity equal to zero is conventionally used to signal a note off. In our implementation we propose that a note number can be associated with a region analysed in the video frame. A note on is only triggered if normalised luminance or luminance change goes above a user-defined threshold. A note off is triggered when the luminance or luminance change falls back below that threshold. We are experimenting with a way of setting a note’s onset velocity which takes the local rate of change as the luminance or luminance change crosses the threshold and normalises this as before. This value can then be clamped and rescaled also as described before. Velocity for note offs are, as we mentioned above, conventionally represented as zero so no calculation is needed when the threshold is passed by diminishing luminance or luminance change values. A simpler variant is to create a note event with every frame of the input (or as often as a frame skip value allows) and use normalised and rescaled luminance or luminance change to give onset velocity direct. This gives can give attractive pulsing output but allows the user to only control the velocity of the notes, not the velocity and duration.

In this way, the contents of a video image can be analysed by region and the results of this analysis expressed in terms of a continuous steam of values or as a set of events. The analysis can take place in terms of luminance or luminance change. While we have prioritised MIDI out so as to experiment with sound control, it should be easy to see how our principles could generalise to other protocols one might want to design.

The results of video analysis are shown visually in terms of shading added to a video window on the desktop. This image can also be set to full-screen so that a post-analysis image can be projected to give the user feedback deemed valuable according to the analyses presented in Deliverable D7a.1, or inspected in detail.

3.3. Experimenting with VIDI

To date our experiments with VIDI have concerned a number of basic explorations of a reduced set of functions. At the time of writing VIDI only supports MIDI output (but this can be routed internally to a soundcard or externally to a MIDI port). We have not yet fully implemented the luminance analysis server concept. We have not yet implemented the clamping/rescaling as a user-interface feature. Currently, VIDI simply implements luminance analysis output by normalising direct to MIDI’s 7-bit range. We have concentrated on implementing luminance change analyses (as these were used most extensively in Wobblespace and are most suited to detecting movement in the image). MIDI controllers can be output but only the continuous pulsing of note event data is supported. Finally, we are only able to transmit data at the frame rate, we do not skip frames.

Our current version is implemented in C++ and runs under Windows 95, 98 and 2000. Windows NT4 is also supported but the option to display full-screen is not supported due to NT restrictions on support for the DirectX libraries we use.

This pre-prototype version has been tested informally by routing in image from a live camera, from video tape and, via a Hauppauge WinTV card, live televsion. We are developing the following highly preliminary impression of VIDI's capabilities. First, it is readily possible to define regions that show the requisite sensitivity to changes in the image. This can lead to some interesting effects if, for example, the region where a speakers' mouth is on the frame sends controller data to a speech simulation sound model in an external synthesiser. It is readily possible to identify different regions and use them to trigger different sounds or different influences upon a sound. Interestingly, we have some preliminary positive results capitalising on orthodox shot compositions in broadcast TV or conventional video material. It is possible to define regions which are jointly sensitive to say a two party shot or a three party shot. Also interestingly, static shots lead to very different results than cuts (where instantaneously there tends to be great luminance change across all regions) than pans (where the luminance change, as it were, percolates across the regions in relation to the pan). In other words, in this fashion, one could potentially musically 'play' with shot composition and camera movement. All of these results have been tentatively demonstrated using conventional 'General MIDI' (GM) or specifically designed synthesised sound. It is somewhat unlikely that VIDI will give satisfactory results directly by unreflectively connecting it to just any sound (and certainly not sounds of any synthesis sophistication). Sounds have to specifically selected or designed to work well with VIDI and the changes in the image that its is picking up on. A preliminary experiment with VIDI as part of a music performance was unsuccessful because the exact 'tuning' of sound synthesis models to VIDI regions had not been carefully considered.

3.4 Related and Future Work

As a possibility for a generic simple application, the possibility of VIDI-like simplicity has been largely ignored. Roads (1996) cites a number of video analysis systems for use in music and sound control but, on closer inspection, most of these fcator in added complexity in how the image is analysed. Singer's (n.d.) videoIn object for use within Opcode's Max music and multimedia environment is the closest we can find but videoIn only allows contiguous regions on a grid (as well as requiring that the user own a licensed copy of Max, a highly sophisticated programming environment). Although there is little innovation in our image analysis methods, we feel we have a simple application which occupies a potential popular and accessible niche (only PC, soundcard and consumer webcam required to begin to explore video-mediated interaction). As such we feel the further development of VIDI is justified as a low-cost accessible alternative to more sophisticated tracking solutions for video-based interaction and, as such, might facilitate accessibility to broad participation events in electronic arenas.