

D2.8 Part A: Final London Travel Demonstrator

Project Number:	ACTS Project N. AC040
Project Title:	COVEN - Collaborative Virtual Environments
Deliverable Type:	P*

CEC Deliverable Number:	AC040-UCL-CS-DS-P-028
Contractual Date of Delivery to the CEC:	March 31st, 1999
Actual Date of Delivery to the CEC:	July 1st, 1999
Title of Deliverable:	Final London Travel Demonstrator
Workpackage contributing to the Deliverable:	WP2
Nature of the Deliverable:	R**
Author(s):	UCL, SICS, Lancaster

Abstract:

As part of the European ACTS programme, COVEN addresses generic requirements for Collaborative Virtual Environments (CVEs) and aims at providing services to support collaboration within multi-participant virtual environments.

This document describes in detail the final version of the COVEN Business Demonstrator, the *London Traveller* scenario. It covers the proposed scenarios, the exploitation of the COVEN services, application functionality and required platform extensions. The final demonstrator is a compelling and impressive application, and we detail avenues for further work and exploitation that should be followed.

Keyword list:
Collaborative Virtual Environments, COVEN platform, scaleability, London Demonstrator, travel

***Type: P-public, R-restricted, L-limited, I-internal**
****Nature: P-Prototype, R-Report, S-Specification, T-Tool, O-Other**

Executive Summary

This deliverable specifies the final COVEN Business Demonstrator, the *London Traveller* scenario. It details the proposed scope of the application, user scenarios that will be supported, and how the applications will exploit the scaleability extensions made to the COVEN/DIVE platform the final stages of the project.

The overall goal of this application is to demonstrate scaleability both in terms of supporting a large number of users, and supporting complex world models and interactions. A subsidiary goal is to support the publication of these applications on the web (data files and binaries) so as to augment their visibility and, at the same time, allow for further network experiments. These goals will exercise all the extensions to the COVEN platform that result from the work in Work Packages 1 and 8.

The London Traveller demonstrator is situated in a large cityscape and supports three main sets of tools:

- *conferencing facilities* will enable many types of collaborative meeting to take place within the model
- *data visualisation facilities* will allow groups of travellers to plan travel about the city using collaborative visualisations of external data sources
- *simulation facilities* that support travel rehearsal and other

The most notable aspect of the London Traveller demonstrator is that the travel destination is represented with a 16x10km 3D model that comprises several hundred megabytes of data. Supporting such a model in real-time between multiple participants is a key success of the demonstrator and of the project [28][29].

Chapter 1 introduces the demonstrator and Chapter 2 discusses the application domain, example scenarios. Chapter 3 discusses the application functionality. The technical details and platform extensions are given in Chapter 4. In Chapter 5 we discuss how we have utilised the usability guidelines and results from network trials in designing this application. In Chapter 6 we discuss the key successes and general experience of creating this demonstrator within COVEN. Finally we conclude in Chapter 7 with a discussion of possible avenues for future exploitation of the demonstrator.

Table of Contents

1. INTRODUCTION 6

2. SCENARIOS 7

2.1 APPLICATION DOMAIN 7
2.2 GLOBAL SCENARIO 8
2.3 COMPONENT TASK SCENARIOS 9

2.3.1 Remote Conference Planning Meeting 9

2.3.2 Visitor Information Visualisation 10

2.3.3 Group Travel Rehearsal 11

3. FUNCTIONALITY 12

3.1 LONDON MODEL 12
3.2 UCL AREA MODEL 14
3.3 VISUALISATION TOOL 16

3.3.1 World Features 17

3.3.2 Miscellaneous tool features 18

3.4 CONFERENCING 20
3.5 SIMULATIONS 22

3.5.1 London Underground Journey 22

3.5.2 Speech Rehearsal 23

3.5.3 Crowds 24

4. TECHNICAL DETAILS 25

4.1 PLATFORM LEVEL 25

4.1.1 Centre Hierarchy 25

4.1.2 T-meshifying 26

4.1.3 Linearising Pointsets 27

4.1.4 Binary Format 28

4.1.5 Holders 29

4.1.6 Multi-Transform 31

4.1.7 Entity Copy 32

4.1.8 Extendible User-Interface *32*

4.1.9 Audio Groups *32*

4.1.10 Subjective Views *33*

4.1.11 Plugins *34*

4.1.12 Scripting *34*

4.1.13 Speed-up Tricks *35*

4.1.14 Renderer Extensions *36*

4.2 APPLICATION LEVEL 38

4.2.1 Plugins *38*

4.2.2 TCL Functionality *39*

4.2.3 Binary Compilation *39*

4.3 PERFORMANCE 40

5. EXPLOITATION OF COVEN SERVICES AND DESIGN GUIDELINES 43

5.1 COVEN SERVICES 43

5.2 APPLICATION OF THE GUIDELINES FOR BUILDING CVE 45

5.2.1 Modelling Perspective *45*

5.2.2 Developer Perspective *46*

5.2.3 Designer Perspective *47*

5.3 IMPACT OF NETWORK TRIALS 48

6. CONCLUSIONS 50

6.1 OVERALL CONCLUSIONS 50

6.2 COMPARISON WITH SPECIFICATION 50

6.2.1 Specification Document *50*

6.2.2 Extensions *51*

6.2.3 Limitations *51*

6.3 FUTURE EXPLOITATION 52

Table of Figures

Figure 1 We replace the physical planning meeting, exemplified on the picture to the left, with a virtual counterpart.. 9

Figure 2 Overview of London model from the south east 13

Figure 3 Example view of the London model from ground level.....14

Figure 4 Overview of the UCL area model within the larger London model 15

Figure 5 Pearson Building of UCL.....15

Figure 6 DIVE City Visualisation Tool User Interface Display.....16

Figure 7 View of one of the conferencing suites inside the Pearson Building 20

Figure 8 View of two members of a crowd from onboard a London Underground train 23

Figure 9 View of the a seminar crowd that provide feedback upon speech rehearsal..23

Figure 10 Example of constant frame-rate rendering. From left to right and top to bottom: view on Onyx at 10fps, view on Onyx at 20fps, view on O2 at 5fps.....37

Figure 11 Twelve users involved in a network trial of part of the London Demonstrator.....49

1. Introduction

The broad aim of this demonstrator is to present an application to a group of users enabling them to specify and rehearse a meeting of any sort. This includes supporting the selection of features at a given location (Hotels, Conference Venues, etc.), using both abstract and facsimile based information visualisation approaches. The demonstrator provides users with the ability to navigate through a large virtual cityscape (representing a real location). Their navigation is aided by a number of dynamic information visualisation systems. A suite of collaborative features aids the users in constructing, rehearsing and participating in both virtual and real meetings.

A novel feature of this multi-user demonstrator is its grounding in a virtual city that represents a real location, in this case 160 square kilometres of the city of London. Use of cityscapes to portray information (of a city) is a natural progression from the use of rooms [22] to organise data. Although the city of London is used in this demonstrator, the tools detailed in this document may readily be used in any given cityscape, regardless of size.

The London Demonstrator is state of the art with regard to the breadth of collaborative facilities and the degree of scaleability that is supported. With this demonstrator we pull together the development work on novel interaction techniques, scaleable rendering techniques, network architectures and CVE platforms. The facilities are fully integrated into the COVEN/DIVE platform [15] and are generally available for future application development.

We discuss the realisation of the demonstrator in the following stages: scenarios for use in Chapter 2, functionality in Chapter 3 and technical details in Chapter 4. We then discuss how the evaluation work of COVEN has influenced the design of this application in Chapter 5. Chapter 6 discusses the success of the demonstrator with respect to project goals and the original demonstrator specification. Finally in Chapter 7 we conclude by discussing further development that could be performed on the demonstrator and possible avenues for future exploitation.

2. Scenarios

We discuss the scenarios for which the application is planned in three stages:

- Application Domain - to delimit the scope of possible application
- Global Scenario - one possible application that exercises all the provided functionalities. In this scenario the demonstrator is a persistent collaborative resource that is re-visited many times during the proposed scenario.
- Component Task Scenarios - more detailed discussion of task specific scenarios that are supported within the global scenario.

2.1 Application Domain

The demonstrator is based within a large cityscape. Cityscapes provide a natural context in which to place virtual applications, especially those that involve a real location. An advantage with cityscape based on-line virtual systems, such as Alpha-world and Blaxxun [3] is that they promote serendipitous interactions. This promotes spontaneous collaboration and information exchange. The Cityscape used in this demonstrator comprises a model of the whole of the centre of London with particular emphasis based upon certain destination target areas where there is more detail. Currently an example is model of the university buildings of UCL, which provide a focus for users within the large cityscape, thus promoting serendipity.

There are two clear application domains addressed by the demonstrator: conferencing support and travel rehearsal. Conferences and meetings enable users to exchange information and resolve issues. Support for virtual conferences enable distributed users to participate without the need for (normally lengthy) travelling. Here, conference support in a virtual environment can also support a number of users to participate in a normal meeting or conference arrangement. Occasionally, it is necessary for participants to be present at physical meetings. In this case the supporting tools can aid in the preparation of a meeting. For example, an agenda for the real meeting can be prepared virtually. Other cases include ensuring the correct size of room for the number of participants, and if breakout sessions emerge is there sufficient space/rooms for interested parties. Providing the users with a representation of the actual intended meeting space enhances such organisation aspects. To support such a wide range of interactive meetings, a suite of tools is offered, such as a notebook, connection to the World Wide Web, overhead projector, etc.

Travel rehearsal is an emerging facility that is at the core of the COVEN project. This allows users (both individually and in groups) to explore and learn unfamiliar surroundings. It is an obvious advantage to business travellers who wish to travel

between two given locations with minimal fuss. Enabling them to experience (and be guided through) key aspects of the journey beforehand (such as airports and train stations). This rehearsal provides users with travel information that cannot be gained from any other means.

2.2 Global Scenario

A conference is arranged by a number of people in a remote location that represents a compromise for each of the different participants. The market for this form of conference arrangement is significantly growing, with a number of airports boasting purpose built facilities. Common tasks in setting up a conference include specifying the physical arrangement of the conference room or building itself, providing a selection of hotels and local places of interest and providing participants with maps so they can find their way between the different locations.

The London demonstrator supports the whole range of conference management, but focuses on the needs of the conference participant. Participants engage with the demonstrator from their remote (or local) locations, which may already be populated with similar users. Users can then explore the City of London or use any of the many services offered by the demonstrator. An entire suite of supporting services is offered to users enabling them to engage the demonstrator in a wide range of activities. For example, they could use it for travel rehearsal, allowing them to experience the route they will use to arrive at the conference centre. Users are free to explore a range of different arrival paths, enabling them to choose the route that best suits them.

In addition to travel rehearsal the demonstrator helps users to choose a hotel or any other specific venue, such as a public house or a theatre. Not only does the demonstrator provide a powerful and novel visualisation system, but also enables them to rehearse their travel plans to and from these locations. For example, a user may wish to choose a hotel close to the conference centre, this can be simply achieved (additional constraints, such as price and quality can also be included) through the visualisation system. This then enables the user to experience the route from the conference centre to the hotel, or from the nearest metro station to the hotel.

Further services within the demonstrator support synchronous and asynchronous collaborative activities within the virtual space. A suite of tools provides users with the ability to hold a virtual meeting within a virtual meeting room in the City of London. Sufficient support is made available by the demonstrator to enable users to discuss specific professional issues, or their travel plans to the real conference centre.

A major advantage of the London demonstrator is not only that it provides a large range of services available to all users, but that it enables them to *seamlessly* move between the different services. For example, a user can use the visualisation tool to choose a hotel, then wander through the City of London to find the hotel, choose a different route back to the conference centre, than take part in a virtual meeting. All

these different tasks can be achieved seamlessly in any order, without the user explicitly notifying the demonstrator of the change of context.

2.3 Component Task Scenarios

2.3.1 Remote Conference Planning Meeting

Conferences and meetings enable people to exchange information in an optimal way. Virtual conferencing is an interesting application area that allows reducing costs and saving time. However, in some situations, physical contact is an essential human factor that can not be replaced. In the following scenario, we address this issue by allowing users to collaboratively plan a real meeting from their separate geographical locations. The virtual meeting will take place in a model of the building that will host the final meeting, so that future visitors may get acquainted with the physical building. The demonstrator will use a COVEN plenary meeting that would take place at UCL as an example. Ideally, the meeting should take place in the virtual room that represents the targeted meeting room. As a result, users will be able to rehearse their way to the main conference room.



Figure 1 We replace the physical planning meeting, exemplified on the picture to the left, with a virtual counterpart. See Figure 7 in Section 3.4 for a virtual representation with similar affordances.

Remote conference planning aims at improving the quality of the final meeting. Efficient planning implies discussing the agenda and the different issues that will be addressed. References to interesting and preparative material might be pointed out. Consequently, people will arrive at the conference with an updated and accurate background. If conference members wish to split up into smaller groups in order to speed up decision making, the demonstrator will allow them to look for an appropriate room that contains the necessary equipment, e.g. video, computers, OHP, whiteboard, etc. Finally, the conference organiser will take charge of physically booking the different rooms and the equipment required by the meeting and all its sub-meetings.

In order to support a broad range of meeting situations, from conference planning, as described above, to fully computer-based meeting, we are specifying a suite of collaborative tools that are based around the metaphor of rooms [13][22]. Our virtual rooms have properties that are similar to physical rooms, namely:

- Rooms are bounded space that are isolated from each other, e.g. by walls and doors, and thus enable privacy.
- Rooms act as object and participants containers. Additionally, items within rooms have a spatial location and are spatially related.

In essence, our aim is to allow the taxonomy of rooms described below [Covi 1998] to be developed by users configuring rooms to their needs:

- Personal offices.
- Shared rooms that are available to several teams. Their use is usually subject to a schedule, and their state is highly persistent. Their content is seldom changed and the spatial arrangement of the furniture and items that they contain is kept from one meeting to the next one.
- Dedicated project rooms in which some members of a team work daily.
- Dedicated project rooms that act as a shared space where members of a project, that sit elsewhere, can leave items or come and work from time to time.
- Public rooms for casual and social interaction, such as coffee rooms or foyers.

In our meeting rooms, our approach stresses a natural meeting point where the participants can get together. This meeting point is generally a table surrounded by chairs. By sitting on a chair, a participant indicates to other members that he agrees to actively participate to the meeting.

2.3.2 Visitor Information Visualisation

Visitors to a new city will normally want to use the most efficient means of transport available and be assured of reaching their destination promptly. In this scenario, users will investigate the travel possibilities within the city using an information visualisation mechanism. This will enable them to select a number of sites to visit based upon their own requirements, using the results of the visualisation and by investigation the information held on each of the objects within the data repository. Given the selected set of sites, the users can request transferral of this information into the cityscape itself. This annotates their selected areas of interest within the large cityscape. To prevent confusion with other groups or individual users, the notion of subjective views, as provided by the COVEN platform will be exploited. Users will be provided with a navigational tool to help guide them around the city. Exploration of the cityscape in this way enables users to gauge the distances between the locations, and how best to travel between them. While exploring the cityscape, users may select locations of interest and view any information held on those locations. Additionally, users may update or enter new information on any of the locations.

2.3.3 Group Travel Rehearsal

A common problem when people try to navigate through an unfamiliar city is that of getting lost. Good maps and mentally running through the main routes help, but a wrong turning is inevitable at some point. These problems are compound when a group of users are travelling together. Generally there are a number of people who believe that they know the correct route, but they all follow slight variations in the route. Commonly people exploring a city must do so within a give time, whether this is between meetings or number of days in a hotel. When the available time is limited, time lost arguing over the correct route not only reduces the available time, but also introduces tensions within the group.

The London demonstrator supports group travel rehearsal, this enables users, from remote locations, to experience the City of London and to collaboratively choose paths to the selected destinations. Rehearsal in this manner increases the user's familiarity with the cityscape, enabling references to be made in the real world to aid in navigation and referential communication. Additionally initial exploration by groups in the virtual cityscape can reduce the time wasted in the real world.

3. Functionality

We discuss the functionality within the London Demonstrator in five stages:

- London Model - generated model of 160sqkm of central London
- UCL Area Model - detailed model of the area local to UCL
- Visualisation Tool - a collaborative data tool
- Conferencing - tools and example collaborative situations
- Simulations - crowds, audiences and transport

3.1 London Model

The basic element of a travel demonstration must be a representation of the destination. Our model of London comprises a 16x10km segment of the centre of London that is generated from 2D vector, building height and 2D image data from Cities Revealed data set[1]. The resulting model consists of about 160MB of VRML data, not including texture map data.

The original maps are stored as 2D DXF files with several layers each containing a different type of building structure. We are primarily interested in building outlines, roads centre and building spot height layers. The conversion stage consists of the following major stages [32]:

- Identifying and repairing building outlines
- Building extrusion and roof-fitting
- Identifying and connecting road centre lines
- Fitting roads between buildings
- Creating pavements

Each building is composed of two or three layers, entrance, middle and optional upper level and each has a roof. The entrance and upper level are tiled laterally, but are

[1]  The Cities Revealed data set is licensed from The GeoInformation Group.

repeated only once vertically and have a fixed height. The centre layer is tiled and/or slightly stretched to fill the intervening gap. Textures are grouped together into matching sets and the set to apply to a particular building is chosen by a few simple heuristics: buildings with a small footprint are more likely to be brick, tall buildings are likely to office blocks, and so on. The resulting model is stored as a VRML1.0 model in 160 1km square tiles, with each tile being built upon a segment of the Cities Revealed aerial photograph for central London. Figure 2 shows an overview of the model and Figure 3 an example view from ground level.

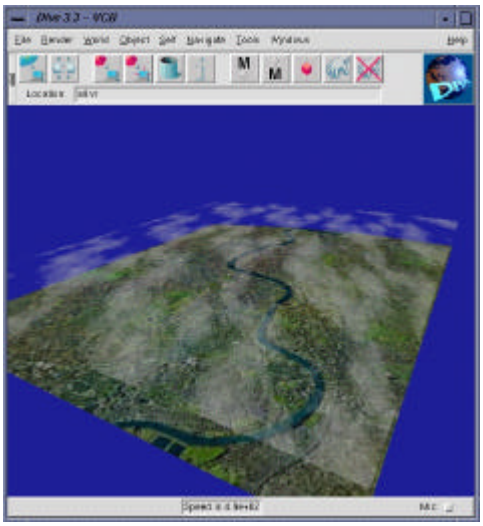


Figure 2 Overview of London model from the south east[2]

[2]The Cities Revealed data set is licensed from The GeoInformation Group

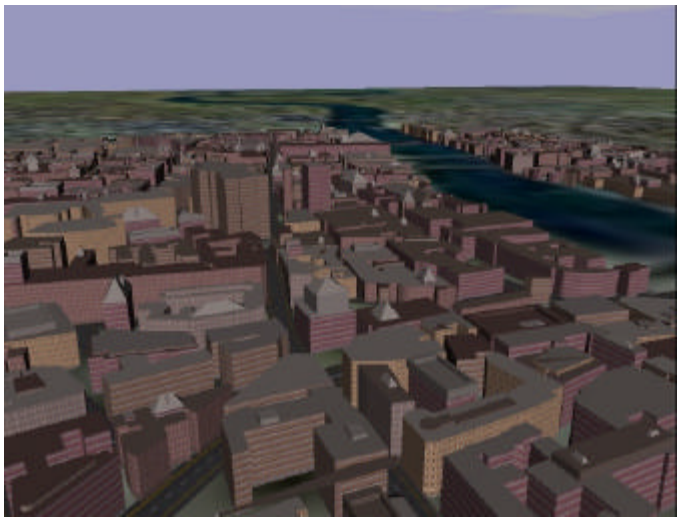


Figure 3 Example view of the London model from ground level[2]

The city is not only important since it can provide travel rehearsal scenarios, but the fact that it is a city with recognisable boundaries provides an easily understood spatial metaphor within which to place other services. If the traveller does not know anything about London, then the layout of the city and the partitions between application components can convey information purely because of abstract relationships between their physical representations. The scale of the city also provides for large numbers of users, since at a human scale, they would normally be far apart. We can leverage such relationships when considering how to make the application scale to large numbers of participants.

3.2 UCL Area Model

The target destination of the global scenario is a conference that would take place at University College London. Thus the first component is necessarily a representation of the relevant campus buildings and local area. In Deliverable 2.5 [9]we detailed the construction of a local area model and the data sources used. Figure 4 and Figure 5 show an overview of the local area in the final model, and some close up views of one of the buildings. In Sections 4.1.13, 4.3 and 5.2 we detail how we have exploited design guidelines and scaleable graphic extensions to speed up the rendering of this section of the model.



Figure 4 Overview of the UCL area model within the larger London model

The Pearson building in this area (see Figure 4, just to the left and below of the centre or Figure 5) provides a place for the situation of the conferencing tools and some of the simulation facilities (see Sections 3.4 and 3.5).

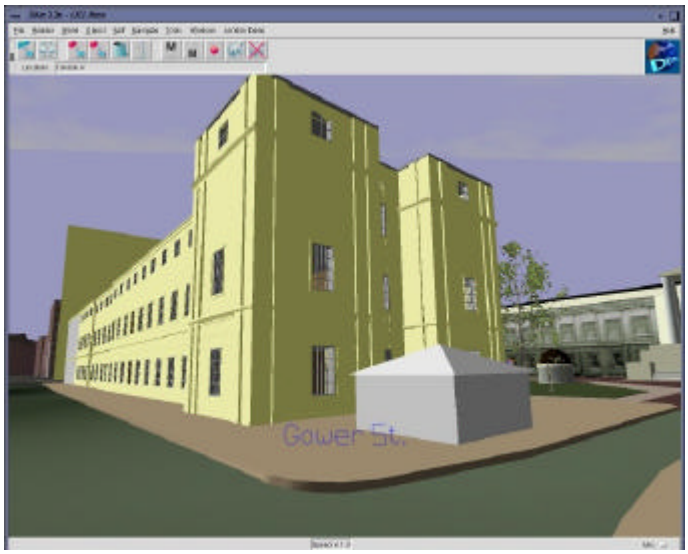


Figure 5 Pearson Building of UCL

Since, in one of the scenarios, we wish to train participants how to reach UCL; there are also representations of distinctive local buildings, and in particular a model of the local London Underground station. This, in combination with soon to be integrated models, allows a traveller to rehearse the trip from London Heathrow Airport to UCL. See Section 3.5.1, for a description of a simulation of public transport.

3.3 Visualisation Tool

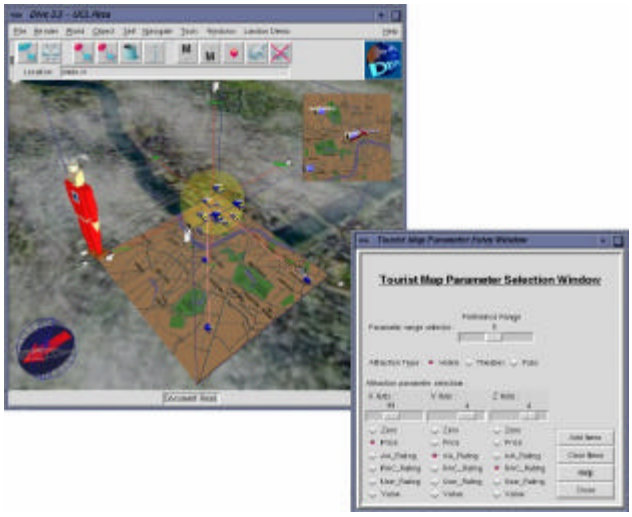


Figure 6 DIVE City Visualisation Tool User Interface Display

The DIVE city visualisation tool has been developed to help users select places of interest according to predefined requirements. To enable this the tool retains a database of attributes (e.g. hotel prices, star ratings, etc.) for each attraction type[3]. Therefore a database entry is maintained for each instance of the attraction containing numerical measures for each attribute. The tool then compares this to user requirements selecting the most appropriate attraction according to the user's selected acceptable range.

Attractions may be chosen from pubs, hotels, and theatres with the tool showing the appropriate attractions' locations on the city map in the base of the visualisation tool. From this users may add attractions that relate to their requirements to a selected list. This is performed after they are happy with the results of the search. Subsequently a user may then decide to go on and look for other types of nearby attraction, the distance between the two affecting their selection of hotel, for example.

For instance, consider a scenario where a visitor to the city requires a hotel and theatre within a reasonable physical distance of each other. The visitor only has a certain amount of money to spend on the hotel, and requires it to have a reasonable AA & RAC "star" rating. Similarly for the theatre they require one with an adequate seating capacity to accommodate their party, and a short distance between the theatre and the hotel. The city visualisation tool can be used to help support this decision process.

[3] We shall assume that that attractions consist of pubs, theatres, or hotels although the tool may be easily extended to include other types.

Using the city visualisation tool in the above scenario the user may specify the attributes that a suitable hotel must posses. For example, the user may select the world's X-axis to reflect the hotels' RAC rating, the Y-axis to reflect the price, and the Z-axis to indicate the AA rating. The acceptable range of values that users are willing to allow the tool to use as a criterion to select hotels should then be given to the tool. Appropriate hotels that meet the user needs, within the chosen selection range, are then added to the visualiser showing their location on the visualiser's base map. These may then be added to the user's selected list of attractions. Next the user may then choose to look at theatres. Again they will select an acceptable selection range for the tool's theatre selection criterion, and input their requirements for the theatre (e.g. price, number of seats, etc.) into the tool setting the attributes to be reflected on the 3D-graph axis shown in the world (see Figure 6). After completing these steps the user should amend the suggested theatres to their selected list of hotels.

Upon updating their selected list the user may then turn on a personal map that they can take away from the visualiser and navigate around the city model to look at a 3D scale representation of their chosen locations (in this case in London). Both the visualiser's base map and the user's personal map show the locations of the attractions in the city. Therefore the visualisation tool has helped the user choose the most appropriate hotel that not only fits their selection requirements (as all of the hotels on the selected list should do), but also one that is within an acceptable distance from the theatre. The user can visually assess these distances from the spatial information provided on the base map. If more information is required about a particular attraction to help the user make their final selection, then the user may click on the attraction's icon on the map of the visualiser to display detailed information from the database relating to the item.

Clicking on an attraction icon in the personal map will transport the user to that location in the city model (the visualiser is itself located above the city). The selected attractions' location in the city are represented via a large "pin" in the city model so that the selected locations stand out while navigating the large city model. The personal map remains with the user no matter where in the city the user is located. A "you are here" pointer on personal map shows the direction and position the user is currently situated in respect to the city map and their selected attractions. Throughout the user may choose to transport to another selected attraction by clicking on icons in their personal map.

The tool, shown in Figure 6 above, has been implemented in DIVE with Tcl/Tk providing the interactive elements and user interface controls integrated into the world. The following section provides more detail about some of the individual features that have been included in the city visualisation world.

3.3.1 World Features

As can be seen from Figure 6 the visualiser consists of two basic elements. A 3D graph of user assignable axes that are set-up to reflect requirements for selecting an attraction in the city. The second main component of the visualiser is the city "base

map" that constructs the tool's floor. Below this is the large-scale model of London. The base map of the visualiser allows users to determine the spatial locations of selected attractions. The 3D-graph also incorporates a "selection sphere" that represents the leeway the user is willing to allow for an item to be selected (i.e. the deviation of the user preference for the selection to be successful and displayed on the base map). The larger the sphere the more deviation allowed for in the selection model.

Each attraction type has a unique icon to represent it, reflecting the attraction type. The nearer to the selection sphere of the 3D-graph that these icons locate themselves, the better they match the user's requirements entered into the user preference dialog. The user preference requirements are set in a dialog generated by DIVE Tcl/Tk (visible in the top left-hand corner of Figure 6). This is accessible by clicking on the floor of the visualiser (i.e. the base map). The dialog allows the user to choose the selection sphere's size (i.e. the selection deviation), and the attraction type. After selecting these a list of attributes associated with the previously selected attraction is displayed in the lower half of the dialog. This is constructed from information contained in the attraction databases, of which there is one for each attraction type. This section of the dialog is where user assignments are made setting the chosen representational attribute to the axis of the 3D-graph drawn in the VR world display (shown in the right hand side of Figure 6). The value sliders allow the user to numerically select values for each attribute. The 3D-graph will then be updated to immediately reflect these changes allowing instant user feedback of the progress of the selection process.

If an attraction is deemed by the tool to meet the users needs the icon is duplicated on the city map (the floor of the visualiser world). This icon remains connected to the 3D-graph icon via a line (as can be seen in Figure 6). The user is therefore presented with the spatial location of the attraction in the VR world in relation to the 3D graph. As previously mentioned the user may select a 3D-graph attraction icon, via a mouse click, to display detailed information about the attraction directly from the database. This feature can be seen operating in the dialog displayed in the bottom right-hand corner of Figure 6.

Adding the chosen items to the selected list will also update the personal map by making the icons visible in their correction spatial locations. This map is fixed in the user's visor and so remains with the user until it is turned on/off with a menu selectable option. So that the personal map does not affect the user's vision of the world a personal map position control panel has been constructed. This allows the orientation of the map to be changed in the visor so that it does not affect the user's view (see the bottom left-hand corner of Figure 6). It also allows the user to zoom-in on the map so that it can be studied in greater detail.

3.3.2 Miscellaneous tool features

To help users navigate around the visualiser while choosing suitable hotels, theatres, etc. the tool provides a number of predefined viewpoints. These include a set of

camera positions in the visualiser that may be selected by clicking on the camera icons. The views include:

- **A plan view** offering a top down view of the world (i.e. looking down at the map). This is useful to help users visualise spatial information presented between selected attractions added to the visualiser's base map.

- **Wide perspective views (placed at each corner of the visualiser)**. These viewpoints are intended to present users with the widest possible amount of information on screen including the 3D selection graph and the base map. One of these viewpoints is shown in Figure 6.

Another additional feature included in the tool is, when the user adds attractions to their selected list the tool indicates via a dialog the number of attractions that lie within the selected range (in relation to the currently defined selection sphere size). This feature enables the user to quickly visualise the results of their search without having to manually count the suggested items added to the map by the tool. For example, if only one hotel was added to the visualiser's base map the user might wish to refine their search to lead to more acceptable possibilities. This may be achieved by either enlarging the selection sphere's size, or changing the axis attributes before settling on the selection.

Finally, the user attribute selection window also contains buttons that allow the user to reset the visualiser, clearing all previous selections from their selected list. This also resets their personal copy of the visualiser's map shown in the visor.

A button is also included in the visualiser to display some simple tips on using the tool, summarising the possible user interactions with the visualiser world. It is hoped though that the user interface is straightforward to understand, providing an intuitive interaction metaphor for attraction selection from a simple flat file attribute database.

3.4 Conferencing

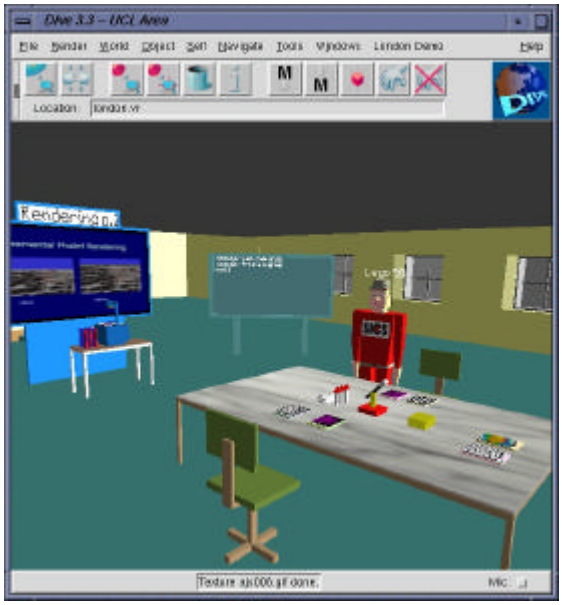


Figure 7 View of one of the conferencing suites inside the Pearson Building

We support a number of different types of meeting room [16]. Figure 7 shows a "conference room" with several media facilities. We also support smaller, more intimate seminar rooms and rooms such as virtual lecture halls, which are a kind of meeting room where there are a few speakers and many attending people. We represent lecture halls as big rooms with a podium and chairs. Again, by sitting on the chairs, participants indicate that they wish to listen to the speakers. Standing in front of the podium ensures that users' communications are broadcast to others.

Most of the virtual tools presented below are simplifications of real-life counterparts. We believe that similarity will make their usage more natural. However, our experience has shown that there is a conflict between realistically modelling the objects appearance so that they are easy to recognise and the lack of an interface that is the same as the real world and that will cause interaction problems [30][31]. Consequently, we need to achieve a balance between resemblance and simplification.

Among the collaborative tools that we have provided are:

- **Notebooks** are used to take notes and to be able to save them to a local file for future usage outside the environment. The three-dimensional representation of a notebook resembles a white sheet covered by horizontal and vertical lines and a recording button in the bottom left corner. To operate on a notebook, participants select them to bring them in front of their virtual eyes. This first interaction enables awareness by indicating to other participants a current focus on a given document. Further user interaction enables operations such as edition or viewing content.
- **WWW icons** embody Web locations inside a virtual environment. Web documents allow all meeting members to have access to a single location. A WWW icon

represents a location by a title, an iconic picture and a link to the location. An icon is represented as a sheet of paper showing the picture associated to the current page and a set of buttons to browse the pages or edit the icon. A number of different operations are supported. The document associated with a page can be viewed within the virtual environment or in a separate application depending on its MIME type. HTML pages are shown in a separate Web browser, sounds are played as if they originated from the icon and three-dimensional description result in new objects being inserted in the environment. It is also possible to create new pages, remove pages, associate another document to a page, and change a picture for a page.

- **Folders** mimic the way that we organise the documents we take with us to a meeting. It is possible to add or remove WWW icons, notebooks and other folders to a folder.
- A virtual **PostIt™ note** can have different information associated to it, i.e. a text message, a URL or a viewpoint. Users choose the information type on note creation and note types are differentiated using colour codes. Depending on the type, different behaviours will result from user interaction:
  - For text notes, a separate window pops up where the text can be read or edited. The text is also shown on the 3D representation of the note.
  - For URL notes, the associated location is shown in a manner similar to WWW icons.
  - For viewpoint notes, the avatar of the participant that has interacted with the note will be transported to the viewpoint that was saved when the note was created.
- **Virtual bookshelves** allow books and reports to be placed on them. Books and reports are associated with links to documents on the WWW. Bookshelves are placeholders for documents that are related to a topic. They act as repositories for interesting things and their content is gradually updated by each project member, each one having his own domain of competence. Consequently, the bookshelves content improves with time.
- **Electronic mail** is a means of involving users who are not connected to the environment. Our interface to electronic mail is a metaphor borrowed from real-life. Documents are inserted in an envelope, an address is written on it and the envelope is posted in a mailbox. A few user interactions are enough to send notebooks, folders and WWW icons to anyone.
- An **overhead projector** is a central tool in a meeting room. In a real meeting, speakers will stand up in front of an overhead projector and show slides as they are talking. In the virtual setting, notebooks and WWW icons can be placed on the overhead projector to display them to others in the room. The screen in front of the projector shows the same three-dimensional representation as the document. A control panel is used to change between slides. On user interaction, the screen acts as an amplifier of the document that is placed on its associated overhead projector. The action result, instead of having local implications, is distributed to all other participants. As an example, if the document placed on the overhead projector is a WWW icon containing a link to an HTML document, the content of the document will be opened in a separate window at all sites. Additionally, each participant can have a personal remote control to the overhead projector. It is useful in interactive

situations where one member of the audience wants to ask a question related to a given slide.

- One feature of virtual rooms is that any **virtual object**, such as a prototype model or a data representation, can be shared between members. Virtual environments have the freedom of being able to represent things that do not exist yet or that will never exist. Objects can be placed anywhere, on tables, on the floor or simply floating in space, depending on their use and size. Participants can get a common point of view on a single entity and use it as the centre point of a discussion.
- When looking at a shared object and discussing its appearance and different parts, it is useful to have an **arrow** to point at specific locations on the object. We will implement virtual pointing arrows for sharing and pointing at objects. Virtual arrows automatically align themselves to the normal to the object at the anchor point, but they can be rotated freely later on.

A total of four conferencing suites exist inside the Pearson model. They can be used simultaneously and each one is private, in that only members within the conference space can participant in the meeting, and events and audio generated in the meeting room and not broadcast to all world members.

3.5 Simulations

The demonstrator includes three simulations that enhance the travel scenario. All three simulations use holders (see Section 4.1.5) to limit event distribution to only those interested participants that are close to the simulation.

3.5.1 London Underground Journey

The first simulation is of a journey on the London Underground that arrives at a station close to the UCL campus see Figure 8. This allows a UCL visitor to rehearse the actual route from the station to UCL and through the computer science department. Models of Heathrow Terminal 3, the Heathrow Express train service and Paddington Station are currently being integrated, and this will allow participants to experience the journey from the airport to UCL.

The train is functional in that the participant can enter and leave the train, and the doors open and close to prevent their leaving between stations.

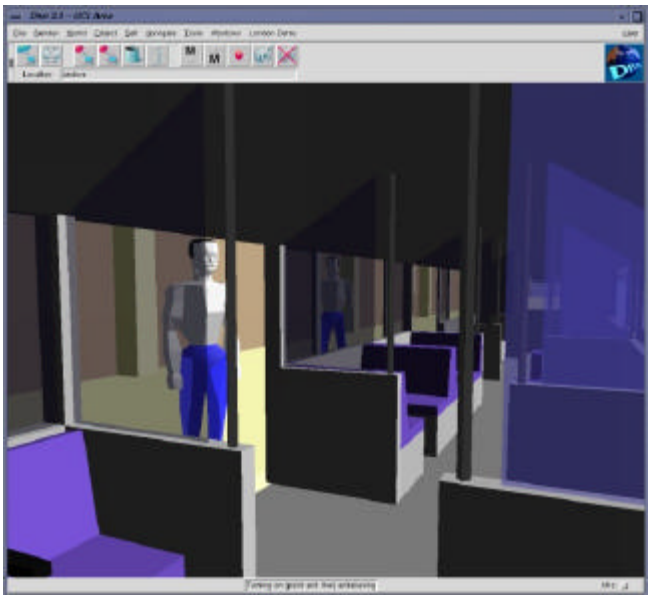


Figure 8 View of two members of a crowd from onboard a London Underground train

3.5.2 Speech Rehearsal

The second simulation component is of an audience in a seminar room [26]. This has been integrated within one of the Pearson building seminar rooms (see Figure 9). The purpose of this simulation is to enable talk rehearsals for prospective conference attendees, or for support of those who suffer from a fear of public speaking.

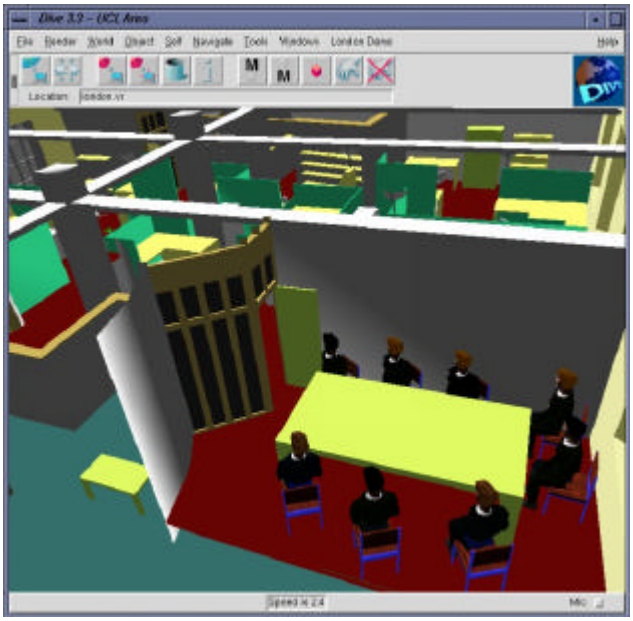


Figure 9 View of the a seminar crowd that provide feedback upon speech rehearsal

3.5.3 Crowds

The final component simulation is of crowds of people in the local area. These can assist the lost traveller since they indicate major routes through the city. Currently the simulation focuses on a small crowd of 20-30 participants who follow routes in around the local London Underground station and UCL campus. A couple of members of a crowd can be seen through the window of the train in Figure 8.

4. Technical Details

We discuss technical details in three stages:

- Description of platform level extensions made in order to enable this demonstrator
- A description of the programming models behind the demonstrator
- A brief discussion of performance issues

4.1 Platform Level

In this section we detail the major platform extensions that have been required by the final demonstrator. Note that none of these facilities changes the core DIVE model or replaces existing functionality. In most cases the extensions are provided through augmented C and TCL APIs to the programmer.

4.1.1 Centre Hierarchy

The graphics pipeline is usually the bottleneck of virtual reality applications and platforms. Similarly to cinema or television, in order to give the user the illusion of a continuous movement, VR platforms seek to render as many frames as possible every second. A crucial point consists then in rendering just as many polygons as required. One, among many, solution to this problem involves culling away the three-dimensional objects that are outside the viewing frustum. To achieve this, VR platforms cache a simplistic approximation of every object of the scene hierarchy in the form of a bounding box or sphere. All vertices of an object are confined within the bounding shape. Since spheres or boxes are simple shapes, it is possible to write optimised algorithms that decide whether an object is outside, inside or crosses the viewing frustum. These algorithms are hierarchical and only the polygonal description of the objects that are inside the frustum will be sent to the rendering engine. It is worth noticing that the bounding shapes can also be used to improve the execution speed of essential functionalities such as collision detection or ray casting through the VR scene.

In its current version, DIVE uses bounding spheres for approximating the extent of objects. The DIVE "scene graph" is composed of objects and views, where objects are the only entities that can carry a transformation in the form of a rotation and a translation (expressed relatively to the object above in the hierarchy). Each object's

position is then the result of the composition of a hierarchical chain of transformations. This absolute position is currently cached in each object. Consequently, it was decided that the only field required to represent the bounding spheres was the radius for the sphere that is centred around an object's absolute position and encloses all the vertices of the visual representation of an object and all its sub-objects. As described above, the rendering, collision and intersection modules make extensive use of this cache as a way of rapidly excluding objects during hierarchy traversals.

A recurrent problem in adapting three-dimensional modelers to DIVE is their tendency in producing models that are not "centred". Modelers are targeted at describing static scenes. Consequently, it will often be the case that the vertices composing the scene contain the transformations within them. In summary, modelers use hierarchies mostly in the purpose of grouping logical entities of the scene together, but the models are flat and the hierarchies do not contain any explicit transformations. Such models usually result in excessively large DIVE bounding spheres. We have developed an algorithm that allows us to compute a probable centre for each object of a hierarchy by recursively calculating probable transformations. The algorithm recursively computes the bounding boxes of each object of the hierarchy and arrange so that its absolute position will be placed at the centre of the bounding box by balancing each object's transformation and its visual representations. This is achieved by modifying directly the vertex co-ordinates in representations that allow it or by adding intermediate objects for views that have an implicit centre, e.g. cylinder views in DIVE are always centred around their local origin, and placed along the Z axis. Intermediate objects are also added for entities of the hierarchy that are not leaves but cannot carry any transformation, e.g. level of details, switches, billboards, etc.

This facility is particularly useful when viewing a scene from a position situated around its centre, since most of the objects are then culled away and not sent to the rendering engine. When visualising from a "corner" and most of the objects are within the field of view, the algorithm is of no particular use. However, this is a situation that seldom occurs, especially in the case of the London application. In practice, we have applied this algorithm to most of the static scenery of the London application, i.e. to each tile that composes the London model, as well as the UCL area and the Pearson building.

4.1.2 T-meshifying

Another recurrent problem when integrating models designed in external modelers in to a DIVE application involves finding an accurate exchange format. In the early stages of the project, VRML was elected as the format of choice when exchanging models between partners. This choice was guided by the fact that VRML is a recognised international standard, supported by a growing number of three-

dimensional applications and toolkits. However, an unexpected flaw of VRML is that it does not support triangular meshes. For simplification reasons, most rendering engines use triangles as their lowest-level primitive. All "complex" primitives that they offer in their API start by triangulating the polygons. Consequently, triangles are primitive of choice when it comes to outputting the results of modelling, since it will allow the platform to avoid the triangulation step and save some precious milliseconds. One particular optimisation case of rendering triangles is triangle strips. The idea simply consists in rendering continuous series of triangles by defining each new triangle using the two previous vertices and a new vertex. Consequently, to render n consecutive triangles, only n+2 vertices will be sent to the rendering engine.

By exchanging models using formats such as VRML, the continuity information is easily lost. Intelligently reconstructing triangle strips from a group of triangles is a complex problem and an active area of research. We have written a simple, greedy algorithm in order to test what performance improvements could be achieved using a naive approach.

Multi-polygons are represented in DIVE using pointsets. Pointsets contain a table of vertices, together with normals, texture coordinates, etc. and polygons and triangular strips are build by indexing the vertices from the table. Our algorithm only considers indices when regrouping the triangles into strips. This is good enough as long as vertices are not duplicated in the vertex table (to remedy such badly formed polygons, we have developed yet another algorithm, see below). The algorithm starts with the first triangle and looks for another triangle that shares an edge with that triangle, i.e. that shares two similar indices with the first triangle. Then, the algorithm continues its look up for yet another triangle that shares an edge with the second triangle, and so on until no other triangle is able to complete the strip. Triangles that have already been used are marked, so that the algorithm does not reuse them. Once a strip has been finished, the algorithm restarts with the next unused triangle of the original pointset.

4.1.3 Linearising Pointsets

Seeking to reduce file and memory size, three-dimensional formats such as VRML supports yet another level of indirection when describing the vertex normals of multi-polygons. This is also supported in DIVE pointsets. In a manner similar as for the vertices above, a table of normals is indexed when describing each vertex within each surface. However, this is maladapted to triangular strips, for which one vertex within a strip can only have one normal, despite that it is shared between two surfaces (triangles), and thus could have two different normals within each surface according to the level of indirection described above.

We have implemented an algorithm that linearises pointsets with such indirections. The algorithm suppresses the normal (and texture coordinates) indirection tables and will recreate linearised tables where vertices are considered to be similar only if they share the same normal and texture coordinate. The algorithm totally recreates the pointset: for each vertex indexed within the polygon definition table, the algorithm fetches the appropriate normal and texture coordinate from the old pointset and add them only if another identical vertex does not already exists, otherwise, the index of that vertex is used instead. Testing vertex, normal and texture co-ordinates equality is done with an approximation (currently a 10^{th} of a millimetre). This allows us to correct the mathematical errors that are often introduced by the modelers when they output files. Another side effect of this algorithm is that it gets rid of all unused vertices from a pointset.

4.1.4 Binary Format

Many three-dimensional formats, e.g. VRML, the AC3D file format, DIVE's internal file format, are textual. This allows for human readability and intervention using a regular text editor. However, textual file parsing is traditionally slow since the input methods have to constantly understand string tokens and translate them into the compact internal structures of the software. Furthermore, textual parsing usually requires an additional initialisation step once each structure has been parsed, so as to initialise the internal fields that can be completely deduced from the parsed tokens. An example is VRML, which allows to avoid describing all the vertex normals by simply describing one field, known as the crease angle, and which standardises an algorithm that uses this angle to compute normals.

The initialisation procedure will have to understand the crease angle, and once all the vertices for the multi-polygon have been read, it will have to generate the adequate vertex and surface normals.

Another drawback of textual file formats is their tendency to produce long files. For example, a floating-point value can be binary coded in 4 bytes, while it will probably require more characters as a string.

In order to improve file-loading time and to save disk space, the DIVE platform was enhanced with a binary file format. Two interface functions have been added, so as to be able to save a given part of the hierarchy as a binary file, and to load it at a later time. The implementation sits directly on top of the mechanisms for serialising hierarchies for socket communication. Since these mechanisms are essential to DIVE networking communication, the file format will evolve simultaneously to the rest of the DIVE platform. However, this also has the drawback of making binary files generated

with a given version of DIVE incompatible with later versions. Binary files are marked with a magic number that contains the version number in order to avoid loading files that have been created with an older or newer version.

Because of its complexity and its multi-partner development, the London demonstrator requires many different files. We maintain master copies of all files as textual files, but have written scripts that translates these files into binary files. This is described below in more details.

4.1.5 Holders

One of the goals of COVEN is to build a framework for controlling the large data flows that are generated by CVEs. Indeed, while creators of multi-user virtual environments often wish them to be spatially extended (if not infinite), to contain many detailed and interactive objects and to allow many simultaneous participants to interact with each other and with the environment, this would not be possible without the use of specific techniques.

In order to achieve rich, interactive, extended and lively environments, populated by many users and applications, we have added a new world structuring technique to the DIVE platform called "holder". Holders are a mechanism that encapsulates part of the database hierarchy under audio and data multicast addresses and associates this particular branch of the hierarchy with an initial file description. Holders require explicit connection to get hold of the branch that they encapsulate. Consequently, they allow the CVE application to control which user process will connect, and thus will held in reducing network traffic and increasing scaleability. In this regard they can be seen as a flexible alternative to explicit partitioning of worlds [23] or awareness models [21]. Holders can be used in two different major configurations, which both are used in the London traveller demonstrator:

- Local holders introduce total locality, i.e. all participating processes will separately read the URL which is associated to a holder. Results of file parsing will be visually and logically similar at all sites, but not shared. In local holders, any modification to the sub-hierarchy will not generate network messages. Local holders are especially useful for:

    - Describing parts of environments that are guaranteed to never change, such as the walls of a building, most of the houses in a town, the soil of an environment, etc. They reduce the size of the dynamic part of the environment that has to be exchanged when new participants enter an environment. In the London traveller application, most of the scenery is encapsulated in local

holder, this includes the different UCL buildings, together with some of their content, as well as the immediate UCL surroundings and the generated London model. The "local" unshared part of the demonstrator consists, at the time of writing, of up to 350 Mb of uncompressed data.

- ♦ A particular application of holders is distributed script execution, i.e. the execution of parallel, independent and synchronised scripts, triggered by high-level events [17]. High-level events are distributed to all connected sites by being sent to the holders themselves, while low-level events are kept beneath holders in the database hierarchy and, thus, not distributed [7]. For example, consider the case of an avatar, which simulates a doorman outside a virtual hotel. High-level events such as "open the door" or "wave right hand" can trigger very complex behaviours that will be executed simultaneously at all connected peers, without generating the network traffic that would generally result from the successive position and orientation updates.

- Distributed holders offer a framework for world nesting. E.G. rooms within buildings, buildings within towns, etc. They nicely restrict network traffic only to the processes that are members, typically a sub-set of the total number of connected sites. This is an option for the different conference rooms that have been integrated in the Pearson building. One side-effect is that holders aid in ensuring privacy since connection is subject to an algorithm that could refuse the entry to some undesirable members.

Holders are essential to the achievement of scaleable worlds. Indeed, they allow the platform to bring into memory only the parts of the database that are necessary to the user. A direct consequence of this is reduced network traffic[7].

We have divided the general London model into 160 tiles. Each tile is associated with a holder and a separate binary file. All binary files are automatically generated using hand-made TCL scripts. This ensures that the master copies always reside in the various VRML files that compose the models.

Each holder is associated with an invisible sphere and a script that describes its run-time behaviour. On collision with the sphere, user processes will connect to the holder, which will result in the local binary file to be loaded in the virtual environment. Since the radii of the enclosing spheres are bigger in extent than the tile and all its elements, an acceptable visual continuity is achieved. There is generally enough time for the whole content of the holder to be read before it enters the immediate focus of the user. The script also disconnects from the holder when an avatar stops colliding

with the sphere. It is also possible to bring up a two dimensional window on interaction, the windows allows to manually connect and disconnect from the tile.

One important aspect of virtual environments is visual continuity and the fact that users should get regular feedback from the state of the system. In the London application, an instanciation of this general principle consists in showing the user that the different tiles are getting loaded. When parsing files, DIVE makes visible each top object as soon as it has been parsed (a top object is an object which is at the top of the hierarchy contained in the file). To benefit from that feature and give system feedback, we have added specific operations to the generating TCL scripts so that they produce an adequate number of top objects in the binary files. To remedy the modelling flaws, these scripts centre the hierarchy and linearise and T-meshify the pointsets. The scripts contain heuristics so that each roof is associated to the walls of its building within the same top level hierarchy. Additionally, road segments are also moved to the top of the hierarchy. As DIVE introduces a slight overload for each loaded top object, future work will divide each tile into smaller logical tiles; each of which will become a top object and group all buildings and road segments of its neighbourhood.

The London traveller application acts as a test-bed for the COVEN scaleable rendering techniques developed at UCL. However, we also wanted the regular DIVE renderer to be able to cope with the complexity by using the more traditional rendering primitives described in D1.4a. Consequently, we have added an option to the generating scripts so that each top object can become a level of detail which arranges for the object to be completely invisible from a reasonable distance. This impairs quality when flying through the model but achieve excellent results when navigating at the ground level.

4.1.6 Multi-Transform

DIVE supports application driven network message aggregation. This feature reduces the number of messages necessary to describe database modifications. For example, several transformations can be grouped into a single message. This is particularly useful when animating the limbs of virtual humans.

In the London application, we have used this feature within the visualisation tool. All transformations that occur within the abstract representation when the point of focus is modified are aggregated within a single operation. Additionally to an incredible gain at the networking level (there may be several dozens of icons to move), this aggregation improves the user interface by giving better feedback. Indeed, all icons appear to move at the exact same time, which makes clearer that the movement is a direct result of user interaction in the two dimensional controlling window.

4.1.7 Entity Copy

DIVE allows dynamically introduced objects within its virtual environments using two different techniques: object description can be read from files and strings in several different formats, or they can be programmatically generated. For simplicity, only the first technique is available from the scripting language. However, this interface allows objects to be modifies once they have been created.

Additionally, we have developed a simplistic technique that consists in being able to duplicate any branch of the hierarchy. This is particularly useful in applications that require many almost identical copies of an object. Such applications can introduce a template by reading a file and copy and tweak it on demand later on. The visualisation tool uses such templates for the different icons representing the hotels, theatres and pubs.

4.1.8 Extendible User-Interface

One recognised quality of interpreted scripting language is their run-time extendibility. From one script, this makes it possible to run code into another script and extend its capabilities at run-time. Since the DIVE user interface is written using TCL and its companion TK, this quality instances itself in letting application objects sending code to the scripts that implements the user interface. An example of such code extension are opening an application dependent window from which input can be gained or extending the regular widget-based interface of the browser.

Our application uses both examples in many places. Worth to be noticed in the specific "London Demo" menu that is added to the user interface without requiring any modification of the standard DIVE browser. From the menu, various application specific objects can be added or removed on demand.

4.1.9 Audio Groups

Within the frame of COVEN, DIVE has been extended to support so-called lightweight audio groups [ref. to D1.4a]. These groups make it possible to partition the sound scape and confine sound communication to their members only. Such groups are being used in several aspects of the London demonstrator. All conference rooms are associated with such groups, this ensure privacy to their members and has beneficial aspects on network traffic. Additionally, the UCL yard contains several talking booths that aim at embedding audio groups with a more natural metaphor.

4.1.10 Subjective Views

In traditional virtual environments each user is present in the same virtual world, albeit from a different viewpoint. Users cannot tailor their representation of the virtual scene or the degree to which they are aware of other user's activities. This is somewhat analogous to early 2D shared multi-user interfaces where users were each presented with the same application views. However, research in 2D interfaces has shown a strong trend to support individual tailoring of the shared views to reflect user demands. These lessons from 2D interfaces have been integrated into DIVE, allowing individual users to have more control over their view of the virtual environment. This provides users with the ability to tailor the environment to suit their working needs, and enables applications to address users on a more individual basis.

The appearance of an object is used to describe its 3D geometry. This may be anything from a simple cube to a more complex building. The subjective views implementation in DIVE allows to associate user and group modifier to any representation [27]. Modifiers are independent effects that may be applied to any of the appearances. Current modifiers are:

- Normal. View the object normally, i.e. no modifications.
- Invisible. The object should not be visible to this user.
- Transparent. Make the object transparent, this modifier is associated with a specific transparency value that ranges from 0 (totally opaque) to 1 (totally invisible).
- Wireframe. Show the object in wireframe mode.
- Dim. The object should be visible but less obvious, such as made darker.
- Bright. The object should be presented to the user and emphasised within their visualisation by means such as increasing its luminosity.

In the London demonstrator we have used subjective views in a way that differs from their original goal: we use subjective views to help the renderer not to draw objects that are totally occluded. Parts of the UCL area are associated with invisible boundaries that react to collision with the users. These boundaries are most generally regular boxes:

- One visibility boundary encloses the main UCL yard. Once inside this zone, the rest of the London City model is made subjectively invisible, since we are ensured that none of it can be seen because of the walls.

- Many other visibility boundaries enclose different parts of the Pearson building. The idea involves making all inner content invisible when viewing it from the outside, and making only the necessary floors and rooms visible when necessary: when visiting one floor, all other floors can be turned off; rooms content only need to be turned on from the inside.

4.1.11 Plugins

DIVE has been extended to offer a basic plugin interface. This interface allows dynamic loading and executing of separately compiled libraries into a DIVE process. The plugin interface fills in the gap between traditional C application writing and the scripting language facilities. Without the plugin interface, writing applications in C would result in running a separate user process that controls part of the environment. This is often a heavy solution. Plugins allow to dynamically load the compiled code at run-time within any user process, such as the regular DIVE browser. The London demonstrator integrates two different plugins (see Sections 4.1.11): one for facial expression and another one for crowd animation.

4.1.12 Scripting

The ability to associate behavioural scripts with objects in the virtual environment adds life to otherwise static graphical scenes. With such scripts, objects can act autonomously, react to external stimuli and give feedback to user actions in a multitude of scenarios, including user interfaces and agent-based systems. These considerations have led to the implementation of a behaviour description language, based on the Tool Command Language (TCL) [25], as part of DIVE platform. In this scripting language (called DIVE/TCL), any entity in the environment can be associated with a TCL script. These scripts, triggered by events such as user interaction or object collision, are distributed among participating peers across the Internet and can be executed at any site. TCL-based scripts offer a powerful interface for application prototyping and nicely fill in the gap between traditional low-level language interfaces to virtual environments, typically in languages such as C or C++, and high-level animation languages.

Most of the London demonstrator is programmed using the DIVE/TCL interface, except for two plugins that had to privilege speed of execution. The demonstrator

would have required a considerably bigger effort to write if it had been developed using the DIVE traditional C interface.

4.1.13 Speed-up Tricks

Additionally, we would like to describe the different techniques and features that we have used in order to speed up vital operations such as rendering, ray casting or object collision. These techniques have mostly consisted in applying simplifications and modifications to the existing models, as produced by the modellers employed by UCL. We would like to stress here that it is the combination of all these improvements that allow the demonstrator to scale down to typical desktop machines with medium range 3D graphics accelerators. We list each item in the following lines, in no particular order:

- In general, all models have been cleaned up from all unnecessary polygons, such as polygons that are too small to introduce any noticeable visual difference or polygons that are totally enclosed into shapes. Additionally, all polygons have been marked as one-sided whenever applicable. The graphics routine will skip polygons for which only the back-faces are visible.

- As explained above, all models have been re-centred in order to help all the renderer's culling algorithm.

- All models have been linearised and multi-polygons that exclusively contained polygons with three vertices have been translated into triangular meshed so as to reduce the number of vertices sent to the graphics pipeline.

- The number of world lights has been reduced. Indeed, when computing the final colour at each vertex, the graphics hardware/software needs to compute the combined effect of all lights. Reducing the number of lights improves therefore rendering speed. In our case, we have been able to increase the ambient factor so as to replace the effect of the original lights without noticeable effect[14].

- We have tried to use Gouraud shaded polygons only when that was necessary [14]. While Gouraud shading requires sending a normal for each vertex, flat shading only requires a normal for each plane surface. The London model contains many surfaces that are inherently flat, e.g. walls, roofs, roads, etc.

- We have reduced the resolution of the different textures used throughout the model. Most graphics hardware uses a texture memory cache. Once the cached is filled in, the rendering performance degrades. We were able to sub-sample most

textures of the London demonstrator without any difference for low and medium distance visualisation.

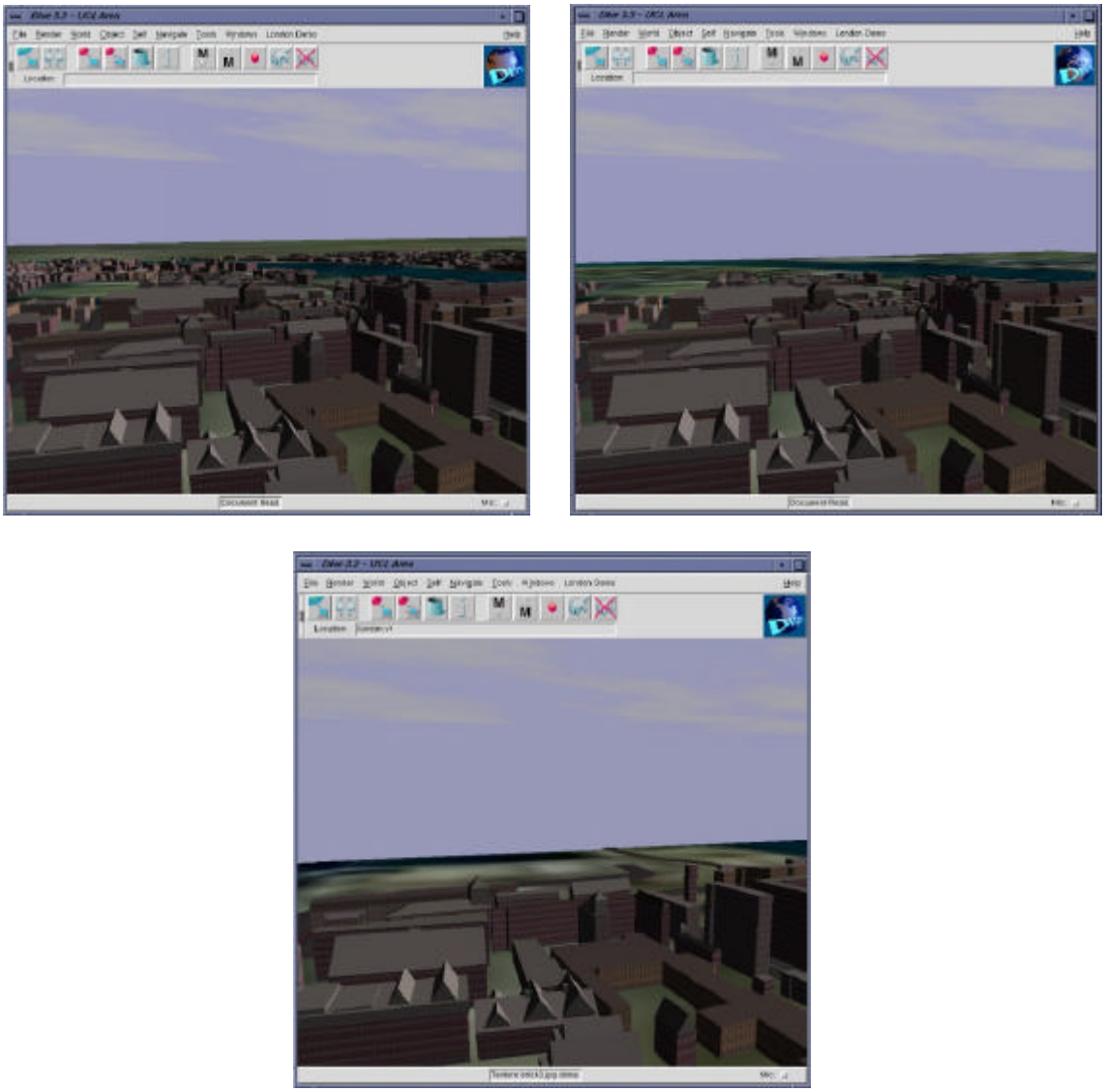
- We have generated simplifications of some parts of the models. These simplifications are used as medium level of details in many cases. Most of the simplifications that we performed removed detail from existing models. This has been possible by a close collaboration between the modelers and the developers. Modelers were asked to correctly name the different logical entities of the models so that the developers would be able to easily detect and operate on these.

- Whenever possible, we have tried to use the same material for the maximum number of polygons. This allows the rendering routines to change material as less as possible, since this is generally an "expensive" operation. Our goal was to achieve an appropriate balance between the number of material changes and the size of the objects so as to keep the benefit from object culling.

4.1.14 Renderer Extensions

The London model is massive and from any viewpoint many hundreds of thousands of polygons maybe seen. We have experimented with renderers that aim to give a constant frame-rate experience so that visual continuity can be maintained. This can be done very simply by optimising the depth of the far clipping plane so that a frame-rate target is met When the rendering target is over-run the far clipping plane is rapidly brought nearer to bring the frame-rate under control. When the target is met within the required the clipping plane is slowly moved outwards. The change in depth is damped so that the frame-time does not oscillate and objects do not pop on and off. Figure 10 shows three example of the frame-rate limiting in action. Figure 10(a) and (b) were taken on a R10000 Onyx with 196M of ram and Infinite Reality graphics system. Figure 10(c) was taken on an R5000 O2 with 128M ram.

For the densely populated interior areas of the model, specifically the Pearson building of UCL, we have integrated a few crude visibility techniques. The model is not suitable for computed visibility solutions such as those developed by Funkhouser [19][20] since the model is not structured with the required identification of cells and portals. However it is very easy to identify large areas of mutual occlusion such as separate floors and groups of rooms by hand. We have integrated tools into DIVE itself that allow objects to be grouped together by hand and identified as "cells". These cells can be turned on or off depending on the user's location using the subjective views capability within DIVE (see Section 4.1.10). Although the resulting visibility relationships are much more conservative than an analytic solution, they are simple for the scene author to describe, and since this facility is available through DIVE/TCL, the types of visibility relationship can be based on complex predicates.



The challenge we are now facing is incorporating other extensions such as incremental rendering, occlusion culling for the streets, visibility graphs for building interiors, image based rendering and incremental level of detail control into a single renderer in a general way.

Figure 10 Example of constant frame-rate rendering. From left to right and top to bottom: view on Onyx at 10fps, view on Onyx at 20fps, view on O2 at 5fps

4.2 Application Level

At an application level the London Traveller is a relatively complicated composition of several smaller application areas. It has been structured to allow other applications to be integrated with ease and also to allow publication of all resources on the World Wide Web without any difficulty[4].

The application breaks down into three main programming layers:

- Model geometry
- Plugin facilities
- TCL scripting functionality

The latter two levels depend on correct structuring of the model geometry in order to provide scaleability. We make extensive use of holders in the application both the delineate areas of the model for logical and semantic structure. For example, each map tile is a both a logical and natural physical boundary and these are based upon existing map boundaries. A single map tile is a useful model "unit" since one kilometre provides a reasonable viewing distance and loading one square kilometre generates a tolerable load on the platforms. Any smaller, and tiles would be constantly loaded, and any larger, tiles would take up too much memory and memory management would be much less efficient.

This level of programming requires correct management of holders and scene graph. There is a little TCL code involved in managing holders through the DIVE/TCL API. This provides much more flexible loading schemes than schemes based upon pre-determined tile or region loading metrics such as ActiveWorlds [1] or SPLINE [2].

4.2.1 Plugins

Plugins allow the DIVE platform to be dynamically extended with efficient native code (see Section 4.1.11). Plugins are necessary when there is a large amount of computational code to perform in order to respond or generate events inside a world. For example, it is preferable to use a plugin over TCL when there is a large amount of vector math or ray intersections to be performed.

[4] Note that early versions of the London Traveller application have been published on the Web in VRML and DIVE formats since 1997. Licensing restrictions limit the publication of more extensive data sets.

There are two main plugins in the London Traveller applications. Firstly the face plugin that animates the faces of the audience in the speech rehearsal segment of the application. This plugin takes simple facial expression parameters (either keywords such as "happy" or "frown" or facial muscle parameter strings) and animates the geometry of a point set mesh to convey that expression. The actual input (a string) and output (a point set array) are quite compact events, but the computation to go from muscle parameters to skin vertices is complex and is thus performed in native code.

The second main plugin, a crowd plugin, animates a set of objects in the environment as a constrained crowd. The operation of the plugin is tightly integrated with a set of in-scene TCL tools that may be used to interactively describe crowd behaviours. A crowd follows a set of regions in a world and the actual path they follow is computed so as to avoid obstacles and other crowd members. These computations require a significant amount of vector math, so are performed in native code through a plugin.

Both these plugins and other minor plugins we have written publish their services by extending the capabilities of the TCL interpreter of the actor they run within. Thus it is possible to check whether a function exists within an actor and program an object reaction to utilise those extra capabilities or intelligently work around them.

4.2.2 TCL Functionality

The majority of the functionality with the London Traveller demonstration has been programmed using the DIVE/TCL API. This allows for flexible programming structures, rapid prototyping, dynamic code extensions and a mature set of distribution services. In a few cases, namely speech rehearsal and crowd animation a TCL function interacts with a native code plugin. We properly cover the cases where the plugin does not load, by providing a clear separation between local application state and global event mechanism.

A number of the demonstration components provide a secondary set of TCL tools that allow in-scene data. For example, the Crowd plugin has an associated TCL tool set that allows the regions for crowd motion to be defined interactively. We also use TCL tools to provide data entry for the tourist information visualisation.

4.2.3 Binary Compilation

The DIVE/TCL APIs cover a very large subset of the internal C APIs of DIVE. As such, quite complex functions can be called, such as object centring and mesh optimisation (see Sections 4.1.1, 4.1.2, 4.1.3). Objects sourced from VRML models or other similar formats are commonly badly structured for real-time performance. In addition loading such file formats is time consuming since they are ASCII based.

We used the DIVE binary file format extensively throughout this demo to reduce load and wait times. However these binary files must first be created and we use DIVE itself to do this, since the file load and save functions may be called through

DIVE/TCL. Thus we provide scripts that load a segment of VRML or plain DIVE .vr file, optimise them using the built in functionality and then save them. This process allows us to re-organise the whole model along more efficient lines. This process takes approximately 2 hours for the whole model. Needless to say, this time is best spent off-line as it improves the overall experience at run-time considerably.

4.3 Performance

The final demonstrator is obviously more complicated than previous COVEN business demonstrator applications, both in terms of model and functionality, so a direct comparison in terms of frame-rate would be impossible. What we can do is test in isolation a part of the model that has remained fairly static between versions. The model of the Pearson building in UCL is used for these purposes since it stresses the graphics capabilities of the platform.

The Pearson model was revised extensively for the final demonstrator following the COVEN guidelines for CVE development [10]. In addition many small errors were fixed and the ordering of the objects was altered in order to utilise the new graphics extensions more efficiently.

In the following tables four viewpoints are used:

v1 view across narrowest part of model looking at a stairwell

v2 long view down corridor

v3 medium view down corridor

v4 view of whole floor or building from head height

The first table compares the efficiency of rendering of a single floor of the model. Three comparisons are made, the old model running without the new graphics acceleration techniques, and the new model with and without the new graphics acceleration techniques.

(Time in ms)	V1	V2	V3	V4
Old model Floor 0 (no graphics acceleration)	90	103	108	95
New model Floor 0 (no graphics acceleration)	56	93	87	70

New model Floor 0 (full graphics acceleration)	22	44	41	36

The speed up between the old model and the new model is due to the new model's having fewer polygons and a far better model structure. The new model structure allows for better view-volume culling since large objects have been broken up and adjacent objects are located proximally in the scene-graph.

There is an obvious trade-off to be made between the following three policies in constructing the scene graph:

- Many small objects. Each objects broken down into connected components. This allows excellent view-volume culling, especially in dense, highly occluded environments. However there is little scope for optimising long triangle strips (see Section 4.1.2), and since many small objects are used there are potentially a large number of material swaps to be made which can slow down rendering.

- Fewer large objects. This is the optimal solution if the objects are totally visible or invisible at all times. It allows the triangle mesh to be fully optimised and the number of material transitions to be minimised. However if the object is partially visible some times, as is certainly the case with a building model were comparatively little of the model can be seen at anyone time, then this is a disastrous option since all the triangles are rendered each frame.

- Balanced objects. Objects are chosen to minimise material swaps, but the triangles themselves are partitioned up into mid-sized blocks such that each block represents all polygons of one material in a certain area. The areas might be rooms or the like that represent likely areas of mutual visibility.

In the old model there were a few very large objects (e.g. one object for all interior walls, another for all exterior walls, etc.), and in the new model we have chosen the balanced approach. We believe that in this situation this is best choice. However there is a definite case for having another level of detail, with a new level of detail with the large object structure for high-detailed views of the building where the whole building is visible. In effect this is not a real "level of detail" since the polygons are exactly the same in both situations, they are just ordered differently so that they can be drawn more quickly in the appropriate situation. Of course, such an approach would double the memory requirements for the model.

If we compare the rendering times for the new model with and without graphics acceleration we see a rough improvement of a factor of two. Overall we see a factor of improvement of 2 - 4.5. This is an excellent improvement and makes the application

much more responsive and useable on the SGI Impact that was chosen as the target platform.

The following table confirms this opinion. With the new model and rendering extensions, the whole building can be rendered in at most 86 ms which provides for a steady 10 frames per second, with a realistic expected frame rate of 15-30Hz.

(Time in ms)	V1	V2	V3	V4
New model All Floors no graphics acceleration	70	213	212	194
New model All Floor full graphics acceleration	36	74	75	86

5. Exploitation of COVEN Services and Design Guidelines

We address the exploitation of previous COVEN design guidance in three stages:

- COVEN Services - overarching design of collaborative systems
- Design Guidance - reactions to guidelines developed from previous experience in CVE design within COVEN
- Network trials - resolving issues raised during large scale trials

5.1 COVEN Services

The COVEN services were first out-lined in D2.1 and then refined in D2.3 through the experience in the development of the Initial Application (D2.2 and D2.3). These services have once again been refined to take into account the output of several other COVEN Activities: Usage Evaluation of the Initial Applications[5], Collaborative Interaction Techniques [6] and Multimodal Interaction Techniques[4] (see [24] for an overview):

F1 Support for *mutual awareness*, with possibly varying degrees. This involves awareness of the presence of other participants, but also recognition of the identity, role, perception of the current activity of the other participants. Different degrees of awareness may be needed, in particular depending on the distance between the participants, the task context and possibly also in relationship to the notion of 'subjective view' introduced below.

F2 Support for *communication* between participants. The users must be able to communicate with each other using audio and possibly other media including text, video and gestures. Both direct communication - i.e. between speaker and explicit addressee(s) - and indirect communication - with participants able to 'overhear' a discussion (oral or not) - should be addressed.

F3 Support for *resource management*. Two persons cannot manipulate an object simultaneously. For instance a microphone object in a conference room can only be used by one person at a time. The person who has access to the microphone is able to speak. Resource management also includes that each user has a fair chance of accessing an object. This can be achieved by introducing some kind of access procedure.

F4 Support for *participants' roles and rights*. The right to manipulate certain objects can be appointed to users. For instance, one group of users is not allowed to do anything with an object representing a book, a second group may only pick it up and move it to a different position, while a third group is also allowed to read the book. It should be possible to give rights to individuals, to groups of people or to everybody at once. The appearance of users can be used to reflect their rights to manipulate certain objects. For instance, a user dressed as a porter is allowed to give people access to a building.

F5 Support for *object manipulation*. Object manipulation includes geometric and edit operations as well as object transfers. Geometric operations include translation

(possibly extended with ‘grid-snapping’), rotation. Edit operations are create (possibly subject to a maximum), copy, remove, etc. Object manipulation includes facilities to hand over objects to another user or a group of users. It should be possible to hand over information using objects as data carriers or *icons*, i.e., objects may represent documents, images, video, etc.

F6 Support for *group navigation* within the environment. A group of users follows a leader. The group may be attached to the leader so that no one can stay behind. Techniques to keep a group together or to reunite a group are being developed.

F7 Support for *human actors*. Human actors are objects with complex behaviour that represent humans. These objects are also referred to as *agents*. Examples are:
- *Receptionists* to provide information to users.
- *Pedestrians* who walk around to simulate a populated environment.
- An *automated guide* to direct a user to a given location within the virtual environment.

F8 Support for a *global map*. Such an item can be used to see what happens in other rooms or places. It can also be used as a search function to find certain people or as navigation tool to support ‘teleporting’.

F9 Support for *subjective views* on the environment, possibly in relationship to a participant's role. The representations of the virtual world may be tailored for particular users within the environment. For example, the communication cables (normally hidden from view) may be visible to electricians and communication engineers. This both prevents unnecessary detail in a number of user’s displays and augments specialist’s displays with relevant data (task-related views).

We briefly list how each of these services is supported in the platform and is exploited in the London Demonstrator or how we have implemented the service within the application layer:

F1 *Mutual Awareness*. This is a core provision of the DIVE platform. We enhance it here with various avatar facilities derived from studies on collaborative work within these systems. Notably, directional audio assists, as well as better tools for locating and transporting oneself to another avatar

F2 *Communication.* There are many communication services within the London Travel demonstration. Audio is the primary communication media. Text is also supported both the private and public groups (see Section 3.4). Privacy is supported within meeting rooms (see Sections 3.4, 4.1.9), and these have very flexible access policies. Finally we support scalable communication through the use of holders. Potentially thousands of users could use the London Demonstrator given a fairly even distribution of users around the scene.

F3 *Resource Management.* Dive now support a set of ownership locks that allow one to create private objects or temporarily make an object private. Certain more sophisticated types of resource management can be provided through application layer functions using the DIVE/TCL APIs.

F4 *Participants Roles and Rights.* Within the demo there are several possible roles, and we the primary use of these is to limit access to certain functionality areas such as meeting or rehearsal rooms.

F5 *Object Manipulation.* A very complete range of object and material edit copy operations are supported in the core DIVE platform. Certain facilities have been extended, to say, add and remove objects from the heads up display.

F6 *Group Navigation.* This is loosely supported since in trials experience such group functionality is not well utilised. We prefer an informal approach, using group communication facilities to plan travel rather than be constrained to follow another. The latter is certainly possible, and is actually very easy to program into the demonstrator, but it is not appropriate in many situations in the model (for example, when inside a building, or walking down the narrow London streets where a group formation might be a hindrance to progress.)

F7 *Human Actors.* This is widely supported. See the sections on audiences (3.5.2) and crowds (3.5.3).

F8 *Global Map.* This is supported and has greatly extended functionality through the visualisation tool (see Section 3.3).

F9 *Subjective Views.* These are supported in the core DIVE platform and are utilised in a number of places within the demo.

5.2 Application of the Guidelines for Building CVE

In this Section we report on the influence of the COVEN Deliverable 2.6 "Guidelines for Building CVE Applications", on the design and construction of this application. There are three key perspectives described in D2.6, though we discuss them in reverse order:

- Modeller
- Developer
- Designer

We address each in turn.

5.2.1 Modelling Perspective

The demonstrator has very large and details sets of models. We have applied all the theories of model construction from D2.6 in order to reduce file size and improve rendering speed and overall image quality.

All the models have been optimised to remove redundant vertices, and to create shared vertex sets wherever possible. This reduces file size and improves rendering speed. Each polygon set has been pre-converted to a triangle set, thus replacing concave polygons, and these triangle strips are then optimised into long fans and strips. This gives good OpenGL rendering performance (see Sections 4.1.2, 4.1.3 and 4.1.13 for more details). Every appropriate object has been centred in the hierarchy to enable efficient view volume culling and collision detection (see Section 4.3). These optimisations alone are mostly responsible for the considerable speed up in model rendering reported in Section 4.3.

We have reduced the number of global lights in the model, and have altered material settings to compensate. This generally requires a higher ambient value in the material. A single directional light is used to give surfaces varying shading. This helps with picking out flat geometry that would otherwise appear featureless.

Almost all buildings in the model are texture mapped. This gives a good sense of complexity and realism to the model. Where possible these are based on real photographs and these photos often show shadows. We have also utilised billboards where appropriate, in particular in the description of trees. The DIVE platform itself now supports anti-aliasing. We have taken further measure not in D2.6 to use more advanced mip-mapped textures to increase image fidelity without a loss of speed.

The overall scene hierarchy is designed to allow efficient spatial sorting across large numbers (tens of thousands) of objects. At a city scale, there is a regular partitioning of the scene with a well-balanced tree. All objects are arranged in a spatially coherent order, with the common parts of buildings forming the leaves of the trees. Within the UCL area, the hierarchy follows a building/floor/room/furniture, partition. This is also spatially coherent, so affords efficient culling.

All detailed building model has replacement level of detail with defined ranges. The buildings in the large London model do not have individual levels of detail. Partly this is because this would greatly increase the storage required, but also because we are using frame limited rendering (see Section 4.1.14). This gives a coherent rendering per frame, and the trade-offs between object LOD and distance would not give much greater depth complexity than currently seen.

Many of the optimisations described here are applied at the binary compilation stage (see Section 4.1.4). This is much more efficient than hand editing the source models since these are difficult to maintain. We apply algorithmic methods with excellent results. Included in the binary compilation stage are hierarchy re-arrangement, colouring, triangle strip optimisation, point set creation and object centring.

5.2.2 Developer Perspective

We have used the developer guidelines extensively in the production of this demonstrator and much discussion of this can be found in the text of Sections 3 and 4. Many of the developer implementation suggestions are core facilities of the DIVE platform (e.g. subjective views), and as such they need not be developed explicitly.

We have implemented human animation for the audience and crowds in the manner described in D2.6 Section 4.1. Avatars have expressive gestures, with arm pointing indicating objects of interest and eye gaze locking between avatars and the user. These convey a powerful sense of participation in the applications.

We have used subjective views for several purposes as described in Section 4.1.10 and 4.1.13. Mutual awareness is supported through co-location, expressive avatars and in-scene reaction to interaction. Most notable of these are the reactions of objects in the

conferencing suite, with objects and folders animating opening and shutting and appearing in front of the users that are reading them. Similarly communication is supported and implemented within the platform. What we have done is create private groups for conference suites, and used holders and high-level events where possible to facilitate efficient communication at a network level.

There a few instances in the demonstration where resource conflicts are a problem. The only place this might arise is in the conference rooms where there are many shared objects to manipulate. In this case conflicts are resolved at a DIVE/TCL level with individual objects associating themselves with owners when conflicts might arise. Other than this, we have avoided the overheads that would be incurred with explicit ownership mechanisms. We also note that the vast majority of actual model, the city, is not subject to change and is thus marked completely static. If a user does manage to change a segment of the city model, those changes are not propagated to the others. This is useful for persistent models and we can do this because we do not support any scenarios where the fabric of the city is changed. We can make exceptions at the individual building level.

We implement roles and right, object manipulations, global map and group functionality as noted in Section 5.1.

5.2.3 Designer Perspective

As mentioned before the application has been designed with a literal city metaphor with a detailed city model underlying the application. This has been done for a number of reasons primarily that the city provides an encompassing spatial metaphor for groups of people to interact it.

The city provides natural boundaries between interaction spaces - external areas are public, building interiors usually contain private areas. Flying above ground implies a "browsing" context and consequently the user is visible at a great distance. People who are co-located are likely to have similar interests, and so on.

We have taken the literal metaphor implied in this model and extended it to representation of building scale services, such as the conference suite. Although abstract buildings or spaces could be utilised, using realistic building models allows an easily comprehensible "route" to the space and many such spaces can be integrated without complex "hyperlink" style connections.

Where we have provided "magical" functionality, such as the visualisation tool, it is grounded in a non-real scale environment, and is purposely placed high above ground so as not to interfere with understandings of spatial layout or impinge upon or obscure other parts of the model.

We have taken onboard the suggestions for design of usability throughout. All demonstration specific tools are either tied tightly to and activate upon spatial location, or have been integrated into a dedicated menu bar that is added to the main window

bar (see Section 4.1.8). Nothing in the scene is keyboard activated, and all functionality can be accessed through in scene or on screen menus as appropriate.

An important, general HCI, goal is that the user interface should be consistent. This requires the actions of the interface to be consistent with the interface itself (such as push-button widgets all looking and acting identically) and with other user interfaces within the same system (such as applications having a file menu which contains the commands ‘open’, ‘close’ and ‘save’). The London demonstrator strives to meet these goals by providing a consistent interface to the user and be consistent among the common range of 2D and 3D applications.

For example, interaction with the visualisation tool exploits commonly identifiable 2D interaction techniques such as push-buttons and slider bars. Interaction with the interface is instantaneous and provides instant feedback of the underlying state of the system. For example, is a user selects ‘hotels’ to choose from the hotel toggle button highlights; also if a selection priority for a user is that the hotel must be a 5 star rating this is also instantly recognisable in the interface. There is also a strong and efficient like between the 2D-control panel and 3D visualisation, where user’s interactions with the control panel are instantly reflected in the visualisation. This enables users to rapidly produce their query through a number of short, quick, interactions. The benefits of this approach as synonymous with that of rapid prototyping in application development.

It is important to note that many user-interface components are automatically provided as a function of place. For example, text chat windows are started on entry to a conference room. A particular case of this is the rehearsal audience. On approach the audience appears to "wake-up" - the avatars start moving and open their eyes. When the user departs they then close their eyes again. This simple function both conveys that the application is ready and is enabled and focused on the participant.

5.3 Impact of Network Trials

For a large part, the experience of the network trials has been distilled into the design guidelines. We note a few new recent lessons learnt from recent trials.

Figure 11 shows a part of the London Demonstrator being used in the 1998 wide area trials [18]. Twelve participants are present, ten are visible in the Figure and the viewpoint is that of the eleventh participant. It is obvious from the Figure that the participants are co-located and mutually aware from the positions and orientations of their avatars.



Figure 11 Twelve users involved in a network trial of part of the London Demonstrator

We summarise the major findings from the trials, more details can be found in Deliverable 3.5, Section 6.4.

In the trial we found it very useful to have sites represented by avatar colour, with each member of a site distinguished by some "accessories" in different colours. Nametags are just above visible above the avatars, and it was important that these be real names rather than abstract "handles" or usernames. Spatialised audio was extremely useful to indicate avatar location. In the trials no overview map was provided, but people were able to get an overview by flying upwards. The audio waves that occur on speech were also useful, as was the red interaction ray that is activated when a user selects an object (both of these are visible in the Figure). Users reported that the slow frame-rate was a hindrance. Navigation around building interiors was complex, though this is to be expected from analogy with real-life spaces.

All these factors have been addressed in the final demonstrator. Frame-rates have been improved considerably (by a factor of two for building interiors), and we now support constant frame-rate rendering. Avatars are more expressive, with eye gaze and body posture. A global map is provided as well as other collaboration facilities. Finally the user-interface has been simplified in many areas, and the overall rendering quality is enhanced.

6. Conclusions

We conclude with the following discussions:

- Overall conclusions on the success of the demonstrator
- Comparison of the implementation with specification
- Routes for exploitation

6.1 Overall Conclusions

The London Travel demonstrator is, to our knowledge, one of the most complex CVE applications currently being demonstrated. The demonstrator's novelty comes from the fact that it supports scaleable models, complex functionality and multiple users simultaneously. Each of these aspects had been demonstrated in isolation before, but rarely together. The demonstrator and platform are both very general and unlike some CVE applications, they can be dynamically extended and reconfigured for specialist purposes. Finally, the demonstrator has been constructed for distribution on the World Wide Web, once data sources are publicly available.

The London travel demonstrator utilises all the services that COVEN has developed, and it and the DIVE platform together prove the approach that COVEN has taken to network, service and application layers. We have integrated many of the research themes on scaleability and novel interaction.

6.2 Comparison with Specification

6.2.1 Specification Document

Firstly we compare the specified application with its realisation. In all areas the specification was met or exceeded. We detail the proposed functionality in this section, and the extensions in the following sections.

The following table lists the specified functionality with its associated priority and technical difficulty as specified in Deliverable 2.7a. We assigned priority and difficulty in terms on *High, Medium* and *Low*.

Function	Difficulty	Priority	Reference
Extension of communication facilities	*Medium*	*High*	*3.4*
Extended awareness facilities	*Medium*	*High*	*3.4, 4.1.9, 4.1.10*
Document sharing	*High*	*Medium*	*3.4,*
Text to graphics translator	*Medium*	*Low*	*3.4*
3D collaborative interactive Data entry tool	*Medium*	*Medium*	*(see note)*
Interactive data visualisation	*High*	*High*	*3.3*
Personal navigator	*Medium*	*Medium*	*3.3.2*
Creation of large London model	*Medium*	*High*	*3.1, 3.2*

All High Priority items were completed. Of the Medium Priority items, Document sharing and Personal Navigator were completed, see below for discussion of the Data Entry Tool. The Low Priority item was also completed. The right most column in the table indicates a primary reference to discussion of this item within this Deliverable.

The *3D collaborative interactive Data entry tool* was specified as a tool to allows users to enter tourist data about into the system using a 3D front end. Although this is possible to do, and a very early prototype was built, this functionality was dropped from the completed demonstrator since it is not obvious that a 3D tool is the way to go. There are vast repositories of such information available in proprietary databases, and inputting such data by hand would be tedious and wasteful. Further projects will have to investigate efficient methods of distributing such data source around a 3D model, and allowing visitors to manipulate extrinsic properties of data items (such as their position) but not intrinsic properties. The biggest problem is simply how does one place, say web page links, in the 3D model since the 3D-position information is simply not known? Here CVE tools can help, but unfortunately such data sources were not available, so the prototype tools could not be pursued.

6.2.2 Extensions

The primary areas where we extended the demonstrator beyond that described are in the provision of simulations of transport, crowd and audiences. Several other platform extensions have been made to enable the large London model and scaleability. These are detailed in Section 4.1.

6.2.3 Limitations

There are two primary limitations in the production of this demonstrator: model description and real data entry.

There is an obvious limitation to the number of buildings that can be modelled realistically in a reasonable amount of time. The Pearson Building model took three to four person months of effort to make useable despite the fact that electronic architects plan were already available. The main source of building geometry in this demonstrator was models automatically generated from the Ordance Survey data (see Section 3.1).

These models, although they are roughly correct in geometry, only convey a rough sense of the spatial layout of the city. We have previously called this technique *impression* that is trying to convey continuity of experience in the model, without giving away the fact that there is varying degree of accuracy in the description of the model. In addition the texture maps are extracted from a small set based upon a few heuristics so any particular building will not look like its real counter part. Creating realistic external representations for the buildings in this model would take hundreds of person year's worth of effort and tera-bytes of disk store for the images. Creating realistic internal models of buildings would take perhaps three orders of magnitude longer. Capturing the required data is also an extremely difficult process and one that can probably not be carried out until further research in vision techniques for grabbing complete scenes from passive camera views.

The second problem is related: as it is time-consuming to describe buildings, it is also time-consuming to input other types of real-world information into the database, especially data such as that used by the tourist information visualiser. A second problem is that this data is often proprietary and difficult to integrate due to the vast number of data sources. It is unlikely that, for example, a hotel's web page would have enough information on it to locate a Hotel Pin in the model with any degree of accuracy. Again this relates to the problem of passive data capture in the real world.

6.3 Future Exploitation

The demonstrator is quite broad in functionality, and various aspects lend themselves to exploitation in a variety of application areas.

Obviously the support of a large city model is useful for many civil engineering, tourist service, city planning, disaster management and heritage applications. The data visualisation tool has obvious application in real-time data management for city scale data such as GIS. The conferencing services would be useful in any collaborative application where document sharing or meeting management is required.

References

[1] ActiveWorlds, http://www.activeworlds.com

[2] Barrus, J., Waters, R. and Anderson, D. Locales: Supporting Large Multiuser Virtual Environments, IEEE Computer Graphics & Applications, Vol. 16, No. 6, 1996, 50-57.

[3] Blaxxun http://www.blaxxun.com/

[4] Multi-Model Interaction Techniques for Collaborative Virtual Environments, COVEN Deliverable 4.2, 1996.

[5] Usage Evaluation of the Initial Applications, COVEN Deliverable 3.3, 1997

[6] Collaborative Interaction Techniques, COVEN Deliverable 4.3

[7] The Final COVEN Multi-User VR Platform, COVEN Deliverable 1.4a, 1998

[8] Online Citizen Application, COVEN Deliverable 2.4, 1998

[9] Online Business Application, COVEN Deliverable 2.5, 1998

[10] Guidelines for Developing CVE , COVEN Deliverable 2.6, 1998

[11] Specification of London Travel Demonstrator, COVEN Deliverable 2.7, 1998

[12] Usage Evaluation of the Online Applications (Part A), COVEN Deliverable 3.5

[13] Covi, L. , Olsson, J., Rocco, E., A room of your own: What do we learn about support of teamwork from assessing teams in dedicated project rooms? Proceedings of CoBuild'98, First International Workshop on Cooperative Buildings: Integrating Information, Organisation and Architecture, GMD, Darmstadt, Germany, ISBN 3-540-64237-4, February 1998.

[14] Foley, J., van Dam, A., Feiner, S. and Hughes J. Computer Graphics Principles and Practice, 2nd Edition. Addison-Wesley

[15] Frécon, E., and Stenius, M. DIVE: A scaleable network architecture for distributed virtual environments, Distributed Systems Engineering Journal, Vol. 5, No. 3, 1998, 91-100.

[16] Frécon, E., Avatare-Nöu, A. Building Distributed Virtual Environments to Support Collaborative Work. 1998 ACM Symposium on Virtual Reality Software and Technology (VRST'98), Taiwan.

[17] Frécon, E., and Smith, G. Semantic Behaviours in Collaborative Virtual Environments, Proceedings of 5th Eurographics Workshop on Virtual Environments, 1999.

[18] Frécon, E., Greenhalgh, C. and Stenius, M. The DIVEBONE - An Application-Level Network Architecture for Internet-Based CVEs, submitted to VRST99.

[19] Funkhouser, T.A., Sequin, C.H. and Teller, S.J. Management of Large Amounts of Data in Interactive Building Walkthroughs, Proceedings of 1992 Symposium on Interactive 3D Graphics, Computer Graphics, 11-20, ACM Press

[20] Funkhouser, T. Database Management for Interactive Display of Large Architectural Models, Graphics Interface '96, Canadian Human-Computer Communications Society.

[21] Greenhalgh, C. Large Scale Collaborative Virtual Environments, to be published spring 1999, London, Springer-Verlag.

[22] Henderson, A.J., Card, S.A. Rooms: The Use of Multiple Virtual Workspaces to Reduce Space Contention, ACM Trans. on Graphics, 5(3), 1985.

[23] Macedonia, M., Zyda, M., Pratt, D., Brutzman, D. and Barham, P. Exploiting Reality with Multicast Groups: A Network Architecture for Large-scale Virtual Environments, in Proceedings of IEEE Computer Graphics & Applications (1995), 38-45.

[24] Normand, V. *et al*. The COVEN Project: Exploring Applicative, Technical and Usage Dimensions of Collaborative Virtual Environment, Presence: Teleoperators and Virtual Environments, 8(2), in press.

[25] Ousterhout, J. Tcl and the Tk toolkit. Addison-Wesley, Reading, MA, 1994, ISBN 0-201-63337-X.

[26] Slater, M., Pertaub, D. and Steed A. Public Speaking in Virtual Reality: Facing an Audience of Avatars, IEEE Computer Graphics and Applications, 19(2), March/April 1999, p6-9.

[27] Smith, G. and Mariani, J. Using Subjective Views to Enhace 3D Applications. ACM Symposium on Virtual Reality Software and Technology (VRST '97). ACM Press. Swiss federal Institute of Technology (EPFL), Lausanne, Switzerland. September 1997. Pages 139-146.

[28] Steed, A. and Frécon E. Building and Supporting Large-Scale Collaborative Virtual Environments, submitted for publication.

[29] Steed, A., Frécon, E., Pemberton D. and Smith G., The London Demonstrator, submitted for publication.

[30] Steed, A., Tromp, J.G., Experiences with the Evaluation of CVE Applications, in: Proceedings of Collaborative Virtual Environments Conference (CVE'98), 17-19 June, Manchester.

[31] Tromp, J., Steed, A., Frécon, E., Bullock, A., Sadagic, A. and Slater M., Small Group Behavior Experiments in the COVEN Project, IEEE Computer Graphics and Applications, Vol. 18, No.6, Nov/Dec 1998, 53-63.

[32] West, J. MSc Thesis, Department of Computer Science, University College London, 1998.