• D7.2, Review of VRML and WWW Techniques

Project Number:	ACTS Project N. AC040
Project Title:	COVEN - Collaborative virtual environments
Deliverable Type:	P*

CEC Deliverable Number:	A040-UCL-CS-DS-P-072.b1
Contractual Date of Delivery to the CEC:	12th April, 1997
Actual Date of Delivery to the CEC:	12th April, 1997
Title of Deliverable:	Review of VRML and WWW Techniques
Work package contributing to the Deliverable:	WP7
Nature of the Deliverable:	R**
Author(s):	UCL, IIS, EPFL, Geneva

Abstract:

This deliverable is a report describing the current state of the VRML standard, and associated techniques for supporting collaborative environments over the Internet.

Keyword list:
Collaborative Virtual Environments, Virtual Reality Modelling Language, World Wide Web

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

Executive Summary

COVEN is focused on the development of a computational service for teleworking and virtual presence. The overall objective of the project is to provide the facilities needed to support future co-operative teleworking systems. Groups of geographically remote users will be able to interact with each other within a shared virtual environment. A platform will be developed that can be used as basis for future Collaborative Virtual Environments applications.

The COVEN platform will be an open system that can interact with other simulation and virtual reality facilities. Thus it is important to investigate related technologies that aim to standardise some aspect of distributed virtual environments. The Virtual Reality Modelling Language (VRML) is such a standard that aims to provide a metafile standard for 3D objects. VRML is tightly integrated with WWW techniques, having grown from a desire to provide compelling 3D graphics on consumer-grade computers.

VRML is now in its second version, and is currently being standardised by ISO. It is interesting for COVEN, because it not only describes the static elements of a scene such as geometry and shape, but the dynamics and evolution of the environment. It has been designed to be easily extended and much work is currently being undertaken to support such facilities as large-scale distributed environments, novel interface technologies and data set visualisation.

The results of the COVEN project will be fed back into the ongoing evolution of VRML which is now overseen by the newly formed VRML Consortium. Some initial evaluations of the support for collaborative virtual environments are described. In addition, one of the project members has already been active in the ISO review of VRML, and the results of this contribution are presented here.

Contents

1. INTRODUCTION 4

2. INTRODUCTION TO VRML 5

2.1 History of VRML2.0 5
2.2 Overview of VRML2.0 5
2.3 Current VRML Activity 8

3. VRML97 STANDARDISATION PROCESS 10

3.1 BSI Comments on the Committee Draft Text 10
3.2 CD Ballot Result 11
3.3 Major Changes to the CD Text 11
3.4 Rejected Changes to CD Text 12
3.5 Progress to DIS Text 13
3.6 Future VRML Standards 13

4. VRML EXTENSIONS 14

4.1 VRML Working Groups 14
4.2 Multi-user Extensions 14
4.3 Scripting Support 19
4.4 Avatar Standardisation 23

5. VRML AND WWW TECHNIQUES 26

5.1 MIME Types 26
5.2 Resource Location 26
5.3 WWW Integration 26

6. PRELIMINARY CRITIQUE 28

6.1 Support for Collaborative Virtual Environments 28
6.2 Procedure Issues 30

7. CONCLUSIONS 32

8. BIBLIOGRAPHY 33

1. Introduction

The aim of work package 7 is to contribute to the various standardisation processes related to virtual environments technology. The **Virtual Reality Modelling Language (VRML)** is one such standard that is quite broad in its scope and application. The basic VRML specification provides a 3D metafile format that can be used to describe the appearance, geometry, behaviour and interactions of a 3D scene. Its major advantage over other 3D file formats is that it is tightly integrated with other World Wide Web technologies, and that VRML browsers have been developed for a broad range of machines.

This document reviews the current state of the art in VRML, a field that it changing rapidly and expanding in many directions. The base VRML standard is currently going through an ISO standardisation process, and one of the Coven members has been very active in the review, proposing a number of clarifications and participating in the international editing meeting to review the results of a preliminary ballot on the document. VRML will be submitted as Draft International Standard during April 1997.

Many of the features of VRML of most interest to Coven are currently the subject of discussion and proposals for extensions to the VRML node set. These cover such topics as multi-user worlds and avatar representation, and we present overviews of these proposals.

2. Introduction to VRML

2.1 History of VRML2.0

The VRML2.0 specification was the result of a community effort to produce a 3D scene specification language that could be used to provide animated interactive worlds across the Internet. It was derived from the earlier VRML1.0 specification, which had succeeded in providing a widely used standard for static scene description. VRML1.0 had few intrinsic advantages over other established formats such as those derived from established CAD packages, apart from integration with the World Wide Web through the use of the URL naming scheme (see Section 5).

The development of VRML2.0 was driven by a community process that, in a very short period of time, reviewed design criteria, proposed and reviewed specifications, and then chose from a number of proposals that which would be developed into the current standard. The proposals discussed in early 1996 included ActiveVRML from Microsoft and HoloWeb from Sun Microsystems, but MovingWorlds from Silicon Graphics was selected. Unlike VRML1.0 which was derived from Silicon Graphic's Open Inventor library, Moving Worlds was unlike any existing 3D graphics system and so the implementation of browsers that conform to the specification is a process that has not yet been completed at writing. MovingWorlds was reviewed and refined over the following few months and published on August 6th 1996 at http://vrml.sgi.com/moving-worlds/spec/index.html**.**

The criteria that VRML2.0 was designed to support include that it be:

- easy for content developers to construct worlds
- possible to compose VRML objects together and add new types of object through an instancing and prototyping mechanism
- possible to implement a browser on a wide variety of platforms including relatively low powered personal computers
- possible to implement arbitrary scale worlds with multiple participants

2.2 Overview of VRML2.0

2.2.1 Basic VRML Nodes

The components of a VRML scene can be broken down into the following rough categories: geometric shape, visual and audio properties, scene hierarchy and behaviour. The nodes that describe geometry, visual properties and scene structure are similar to those in VRML1.0. Geometry can be specified by primitives such as Box, Cone and Cylinder, or more general shapes such as IndexedFaceSet, Text and Extrusion. To each geometrical shape, visual properties such as colour, texture and material can be applied. The scene is structured as a transformation hierarchy. That is, nodes can have children that inherit their parent's position transformations.

Additions in VRML2.0 are nodes that can specify audio properties, behaviours and movies, alongside some more minor extensions such as Fog, Background and Billboards. Most importantly behaviours are supported through a data flow mechanism (see Section 2.2.2) and nodes that can perform simple tasks such as interpolation of values or sensing of events. The main component of the data flow is the Script node which allows more complex behaviours to be specified in a number of different programming languages (see Section 4.3.1).

A complete list of the nodes is given in ***Table 2-1***

Grouping nodes	**Sensors**	**Appearance**
Anchor	CylinderSensor	Appearance
Billboard	PlaneSensor	FontStyle
Collision	ProximitySensor	ImageTexture
Group	SphereSensor	Material
Transform	TimeSensor	MovieTexture
Special Groups	TouchSensor	PixelTexture
Inline	VisibilitySensor	TextureTransform
LOD	**Geometry**	**Interpolators**
Switch	Box	ColorInterpolator
Common Nodes	Cone	CoordinateInterpolator
AudioClip	Cylinder	NormalInterpolator
DirectionalLight	ElevationGrid	OrientationInterpolator
PointLight	Extrusion	PositionInterpolator
Script	IndexedFaceSet	ScalarInterpolator
Shape	IndexedLineSet	**Bindable Nodes**
Sound	PointSet	Background
SpotLight	Sphere	Fog
WorldInfo	Text	NavigationInfo
	Geometric Properties	Viewpoint
	Color	
	Coordinate	
	Normal	
	TextureCoordinate	

Table 2-1 Table of VRML Nodes

2.2.2 Data Flow Model

VRML2.0 uses a data flow model to describe transformations of the properties of objects in the scene database, through the use of sensors, interpolators, scripts and routes. Each VRML2.0 node may have one or more inputs, or *eventIn* fields, where data can be received. The reception of data on certain fields triggers the calculation of an internal function, which may subsequently generate values on one or more *eventOut* fields which are propagated to all connected nodes. The types of data that can be passed around include Boolean values, colours, positions, orientations and references to other nodes. In addition nodes might have plain *fields*, which are not exposed to the data flow and are used for static state that is initialised at start up, and *exposedFields* that serve as both eventIn and eventOuts.

As an example, the fields of the OrientationInterpolator are given in Figure 2-1. The OrientationInterpolator's purpose is to generate values on the eventOut *value_changed*, whenever the eventIn *set_fraction* is changed. These values are chosen by interpolating between the values defined by the exposedField *keyValue* where the exposedField *key* gives the reference fraction values.

```
OrientationInterpolator {
 eventIn      SFFloat  set_fraction
 exposedField MFFloat  key
 exposedField MFVec4f keyValue
 eventOut     MFVec3f value_changed
}
```

Figure 2-1: Fields of the OrientationInterpolator

Figure 2-2 shows an example VRML2.0 file which uses the OrientationInterpolator, and Figure 2-3 the corresponding scene graph and data flow graph. The behaviour is very simple, when the user clicks upon the cube it starts spinning. The TouchSensor senses the user clicking on the geometry and generates a time value on a data stream. This then flows to the TimeSensor node, which in turn generates a fraction value. This fraction is used by an OrientationInterpolator to start the generation of orientation values which are routed to the position of the cube to provide an animation. The data flow is constructed using a ROUTE statement which gives the names of two nodes and the fields between which a connection is to be made. The nodes are named using the DEF statement (see Section 2.2.3).

```
#VRML V2.0 utf8
Viewpoint {
   position 0 0 5
}
DEF BOX_POS Transform {
  children [
    Shape {
         geometry Box {}
    },
    DEF BOX_TOUCH TouchSensor{},
  ]
}
DEF BOX_TIMER TimeSensor {
  cycleInterval 5
  startTime -1
}
DEF BOX_ENGINE OrientationInterpolator {
  key [ 0, .5, 1]
  keyValue [ 0 1 0 0, 0 1 0 3.14, 0 1 0 6.28]
}

ROUTE BOX_TOUCH.touchTime TO BOX_TIMER.set_startTime
ROUTE BOX_TIMER.fraction TO BOX_ENGINE.set_fraction
ROUTE BOX_ENGINE.value_changed TO BOX_POS.set_rotation
```

Figure 2-2: Example VRML2.0 File

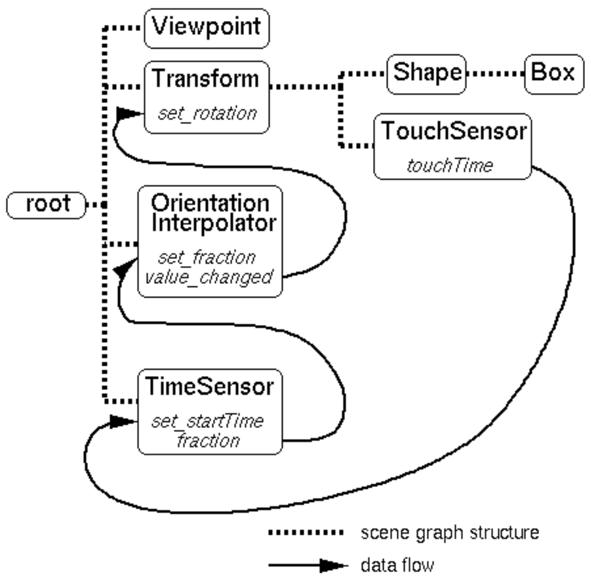

Figure 2-3: Scene and Data Flows Graphs of VRML2.0 Example

2.2.3 Extensibility

Two mechanisms are provided to support scene composition and extension. These are the instancing scheme, which allows a node to be named and re-used within a file, and the prototyping scheme, which allows new nodes to be created that encapsulate complex behaviour and define new properties.

The instancing scheme is simple in application. A node can be named as it is defined using a DEF statement. That node can then be re-used with a USE statement, or can be referenced in the data flow with a ROUTE statement. An example of the use of the DEF and ROUTE statements was given in *Figure 2-2*.

The prototyping scheme allows new nodes to be created, and is a more powerful mechanism than instancing. Two constructs are provided PROTO and EXTERNPROTO. A PROTO gives both the definition of the node and its implementation (in VRML), whilst an EXTERNPROTO gives a node definition and a reference to a PROTO in another file. This mechanism allows libraries to be created and re-used between files. The EXTERNPROTO mechanism also allows browsers to give their own optimised definitions of nodes for behaviours that can not be expressed in VRML without breaking content for other browsers. This is possible since multiple definitions of the node implementation can be referenced in the EXTERNPROTO, and one of these can give a default implementation in VRML for browsers that don’t recognise or support the new node type.

2.3 Current VRML Activity

The process of designing VRML2.0 was overseen by the VRML Architecture Group (VAG), a self-selected body of VRML experts. Since then, the VRML Consortium has been founded (see http://vag.vrml.org/consort), which puts the development of VRML on a more sure, but business led, footing. The VRML consortium is a non-profit corporation whose founder members include all the main Internet and 3D graphics companies. The VAG has now been dissolved, with key members joining the VRML Consortium’s VRML Review Board which will drive the development of VRML standards. The consortium now oversees the progression of VRML through ISO standardisation and is sponsoring the formation of working groups (see Section 4.1) in areas of VRML such as multi-user extensions, business applications and colour modelling.

The VRML community is growing rapidly with the availability of plug-in browsers for Netscape and Internet Explorer, and editing tools such as CosmoWorlds from Silicon Graphics. Two conferences solely devoted to VRML, World Movers and VRML97, have taken place in the USA in 1997 already, and there are tentative plans for a workshop on VRML in Europe later this year.

3. VRML97 Standardisation Process

The decision to standardise VRML2.0 was made in June 1996, two months before the actual publication of the final Moving Worlds document. The committee responsible, the ISO/IEC JTC1 SC24 (Computer Graphics and Image Processing) committee, accepted the document as the basis for a new 3D Metafile standard. The VAG would act as the primary architects of the specification, with the VRML rapporteur group, representing ISO/IEC JTC1 SC24, providing expertise in other graphics standards, and experience of producing standards documents. The VRML2.0 document was published as Committee Draft 14772, and the initial plan was to hold an ballot in November to produce comments to enable the document to be re-written in December so that a Draft International Standard (DIS) could be published in the new year. This would have meant that the DIS ballot could have been completed by May 1997.

Since VRML2.0 was the first version of VRML to be taken to ISO to be standardised, it could not be referred to as VRML2.0. In ISO terms the standard should be referred to as **VRML97.**

3.1 BSI Comments on the Committee Draft Text

The British Standards Institute (BSI) generated a large number of comments on CD 14772 at meeting a held at the University of Leeds at the end of October 1996 [4]. The breadth of comments generated reflected the diverse backgrounds of the delegates which including a few who had been involved in previous related specifications such as PHIGS and GKS, current users of VRML and researchers experienced in the use of other 3D graphics toolkits. The outcome of the meeting was a 70 page comment document detailing editorial and technical issues that needed addressing before the document progressed to DIS.

The official position was that the BSI disapproved the CD for the following reasons:

- overall quality was poor
- inappropriate language
- lack of a concepts section
- lack of a BNF specification
- unusable across SC24 application areas
- no formal extension process

There were of the order of 300 editorial comments, and 45 technical comments about aspects of VRML that appeared to limit its usefulness or deviated from prior standards or established practice. Many of the editorial comments reflected the fact that the document was produced as a set of web pages, this being the first time ISO had agreed to publish a specification in electronic form. It was also apparent that the document was not structured coherently, using many different terminologies for the same concepts and repeating many unnecessary statements.

3.2 CD Ballot Result

A meeting of the ISO/IEC JTC1 SC24 VRML rapporteur group was held in San Diego in order to consider the results of the CD ballot and recommend changes. The BSI was represented by Ken Brodlie (University of Leeds), Anthony Steed (UCL), Simon Bee (University of Loughborough) and Hugh Steele (Metamorphix). Other attendees included three members of the VAG, three from ANSI, and one from France. The purpose of the meeting was to produce a plan for the editors (Dick Puk and Rick Carey) to update the document, and a response document to answer each comment individually. The CD ballot resulted in six yes votes, three yes with comments (Austria, France, Japan), two no with comments (UK, USA) and one not returned.

The reaction of the VAG delegates was that they appreciated the comments and added approximately one hundred of their own that had arisen on the various VRML mailing lists since August. However whilst they welcomed the proposed editorial changes, no technical changes were made that changed functionality since they believed there would be serious resistance from the community. They backed this up by adding that there were already eight browsers under construction and 20,000 worlds in existence, not to mention a number of books either in press or already on the shelves.

Throughout the meeting the VAG members emphasised that the VRML consortium would encourage extensions to VRML, and that VRML itself was designed with that in mind.

3.3 Major Changes to the CD Text

3.3.1 Editorial

All the BSI's comments were processed with few objections [16]. Significant improvements include the addition of a glossary and the reduction of the number of terms used to refer to similar concepts.

3.3.2 Spelling

International spelling was adopted where possible in line with usual ISO practice. However due to a perceived implementation burden on browsers, international spelling was not adopted for field and node type names.

3.3.3 Scope

The UK requested that the VAG re-draft the scope section of the specification since it was felt that the claims were far too broad and did not give any indication of the usefulness of VRML. The VAG members agreed that they saw VRML as an abstract functional specification of worlds, providing a universal file format for 3D, and would make the scope section more explicit.

3.3.4 Concepts

A large number of the UK's comments were concerned with the poorly structured concepts section. This section was completely re-drafted following the UK's proposal and a number of figures were added to explain the role of a VRML browser as an execution engine. New text was produced or suggested for the concepts behind: time, event processing, lighting, sensors, interaction, navigation and avatars.

3.3.5 Interaction

A particular concern of those representing the BSI who were involved in VR research was the constraints imposed on world authors by the interaction techniques described in the Sensor node descriptions. They implicitly described a desktop metaphor for interaction and many styles of immersed interaction were prevented. The UK re-drafted the sensor text to remove some of the restrictions without changing any of the functionality, but some issues remain.

3.3.6 Grammar

The lack of a formal grammar was thought to be a major drawback especially when testing conformance of browsers. The UK offered to produce a grammar, and the VAG welcomed this, but it was suggested that it might not be possible to rigorously check it before the deadline for the DIS text. It was agreed that a node dependency graph table would be added, and the UK agreed to draft this.

3.3.7 Conformance

The whole of the conformance clause was re-drafted with the aim of making it easy to add a profiling mechanism at a later date. The minimum requirements in the revised clause 7 now effectively define a base profile. All statements pertaining to conformance or allowable differences have now been removed from clauses 4 and 5. The table of minimum support requirements has been changed to give both file limits and browser limits on the number of nodes supported. Basically, file limits define what files the browser must be able to parse properly, and browser limits define how much the browser must render, and what short-cuts may be taken.

3.3.8 Other Technical Issues

There were a large number of technical areas where VRML seemed to deviate from common practice or published standards. The VAG used two main arguments to support the design decisions made: avoidance of platform dependencies and issues of scene composition.

The requirement for cross platform support meant that the level of detail, colour, fog and lighting models were kept relatively simple so that they could be implemented efficiently on current rendering libraries such as Renderware, Direct3D and Cosmo OpenGL. It also meant that nodes requiring complex calculations such as full object pair collisions, be avoided since research on efficient algorithms was required.

The desire for scene composition meant that absolute frames of reference were considered necessary for distance and time. This was largely due to a perceived implementation burden within the execution engine and renderer if components of the same scene graph were allowed to specify their own units system.

A number of comments were made about the Java and JavaScript clauses, though most of these were irrelevant due to the VAG's retraction of those clauses pending a complete re-write.

The editors were keen to stress that the VRML consortium would encourage experimentation with new nodes through the external prototyping mechanism.

3.4 Rejected Changes to CD Text

The UK had suggested that the node reference be kept separate from the specification of the clear text encoding. This was rejected since it was thought that the two new clauses would duplicate a lot of information and make the document less readable.

Another suggestion was that a number of new data types be added that reflected the ranges of fields. For example it rather than having the xSpacing field of the Elevation grid specified as SFFloat, the positive range be enforced by making a new data type, SFFloatPos. This was rejected since the document did define appropriate ranges in the text, but it was agreed that this would be made more explicit by adding ranges in the header for each node description.

3.5 Progress to DIS Text

The time scale proposed in San Diego was for the DIS text to be prepared by mid-January with the possibility of an informal meeting to ratify the changes in February. This schedule slipped a month; a draft of the DIS text was made available on 22nd February, with the editor's expecting comments by 12th March. The BSI's reaction to the draft DIS text was that although most of the changes had been made, the document was not yet ready to progress to balloting. The main issues were:

- Concerns that, in some places, the language did not follow ISO guidelines.
- Non-international spelling of fields and nodes. A convincing argument still needed to be presented that this was a significant burden on browser authors.
- Most of the examples did not appear to work and there were problems with the presentation.
- Lack of explanatory diagrams for the sensor text.
- Inappropriate formatting (hyper-links and colour diagrams) for a document that will be published on paper and well as on the web.

There were some oversights on the part of the editors in that some of the UK's contributions had not been added. Over all though, most people thought the document was a great improvement and the remaining concerns did not require a further editing meeting.

The final DIS text was distributed to those involved in the review on 14th April, 1997. The new text answered most of the remaining concerns. The document should be available publicly in the very near future at http://vrml.sgi.com/moving-worlds/spec.DIS/ .

3.6 Future VRML Standards

The current VRML document is envisaged as the first part of a many part specification. There are three extensions that are almost certain to be added:

- Binary encoding [5]
- Conformance testing [24]
- External Authoring Interface [1]

Each of these is the subject of a VRML Consortium working group (see Section 4.1). The binary specification is considered to be the most important at this time due to large size of VRML worlds when described in plain text. Compression ratios are promised of around 50 to 1. Fortunately it is also the most complete proposal, being ready for submission apart from the issue of the licensing of a proprietary compression algorithm.

4. VRML Extensions

4.1 VRML Working Groups

One of the main purposes of the VRML Consortium is to sponsor working groups that examine the various modifications and elaborations of VRML that are being proposed. These range from groups working on compliance testing, to business infrastructure. The VRML consortium's role is to provide guidance and publicity for the working groups. Although anyone can join a working group, only the full consortium members can decide on whether a proposal is made an official part of the VRML specification. Since the VRML consortium is still in its start up phase, few working groups have formally been created. However, the following nascent working groups were represented at the VRML97 conference: conformance, lighting/rendering conformance, Living Worlds, Open Community, optimising VRML, colour fidelity, external authoring interface, biota A-life, Universal Avatars, binary format specification, Java specification, information visualisation, textures, text, commerce, databases and multi-user infrastructure

4.2 Multi-user Extensions

4.2.1 Approaches

Approaches to providing multi-user support for VRML fall into three categories:

- Customised Browser: as typified by Sony's Community Place.
- Script Node interface: using a the Java networking libraries through the Script node.
- Node Extensions: new prototyped nodes that transparently encapsulate underlying network technology.

The first approach relies on a extended VRML2.0 browser that communicates events that occur within the local copy of the scene database to other participating browsers. This approach means no other party's VRML compliant browser will be able to join the world unless they support the additional propriety interface within their own browser.

The second approach can be used by any VRML compliant browser. The Script nodes required to do the communication must be downloaded along with the world if they have not been cached locally. However, designing worlds within this model means that the author must carefully decide what messages to send between the participants. A number of efforts are being made at providing standard Java scripts that can be used for a wide variety of applications.

The third approach is the subject of much debate and a number of working groups. These plan to provide extension nodes to VRML2.0 that define the shared regions and objects of a collaborative environment. The difference between this approach and the script based approach is that this approach promises to provide better scalability and to release the author from making decisions about message passing. Two complementary proposals have been receiving a lot of attention, Living Worlds and Open Community.

4.2.2 Living Worlds

The Living Worlds (LW) [3] effort aims at defining a set of VRML 2.0 conventions that can be used to build applications which:

- support multi-user presence and interaction,

- can be assembled from libraries of components developed independently by multiple suppliers, and visited by client systems which have nothing more in common than their adherence to the VRML 2.0 standard.

The LW specification includes interfaces for:

- co-ordinating the position and state of shared objects (including avatars),
- information exchange between objects in a scene,
- personal and system security in VRML applications,
- a library of utilities, and some work arounds for VRML 2.0 limitations,
- identifying and integrating at run-time interaction capabilities implemented outside of VRML and its scripts.

The conceptual foundation of LW is based on the following components:

- SharedObject, any set of objects whose state and behaviour are to be synchronised across multiple clients
- Zone, a contiguous portion of a scene acting as a container for SharedObjects
- MUtech, a multi-user technology charged with maintaining multiple instances of a scene in synchronisation on multiple remote clients

SharedObjects can be either pilots or drones where pilots are those objects that emit a change in state or a behaviour and drones are replicating it. The behaviour of a shared object may well include the ability to change its state from drone to pilot. SharedObjects keep their instances in synch by routing all state changes from pilots through the MUtech to the drones. To make things more clear, consider Figure 4-1 where a simple scene in which some objects are local and others are shared, is replicated on Client A and Client B.

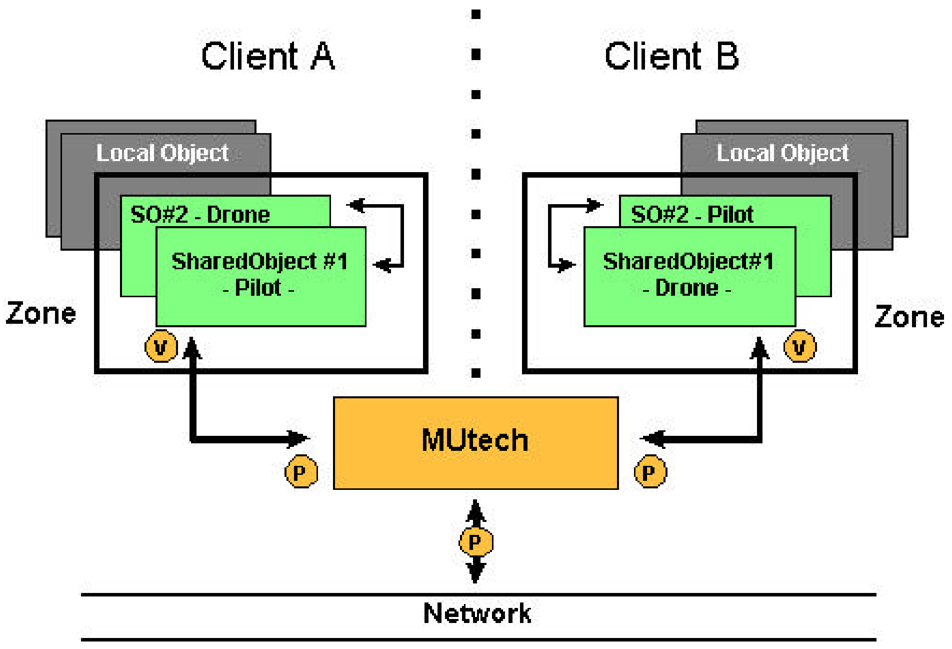

Figure 4-1 Overview of the Living Worlds Architecture

SharedObjects communicate with each other through their drones. Mutechs provide all network communication needed for multi-user interaction beyond that provided by the Browser itself. The Pilot 1 on Client A passes a message to Drone 2, also on A. Drone 2 routes the message through the MUtech back to its Pilot 2 on Client B. If the response from Pilot 2 is intransitive (e.g. a smile), then it is routed like any other state-change back to Drone 2 where the User of Client A can see it. If the response is transitive, e.g. a message, it is given directly to Drone 1, for it to return to Pilot 1 on Client A.

Communication between the MUtech and the scene uses standard VRML 2.0 mechanisms (Protos and ExternProtos linked by routes, events and scripts). Anything that can't be done inside the current standard is declared by definition to be "MUtech specific." This allows MUtech suppliers to experiment with alternative implementation designs, and to access both network data and arbitrary external applications, without having to introduce any non-standard extensions to VRML 2.0. From the standpoint of the world-builder, the MUtech links the local scene graph to a set of external network functions. From the standpoint of the MUtech developer, the MUtech provides a standard interface for tracking and manipulating VRML scenes. This allows world-builders to design a world for multiple uses, each implemented by a different MUtech. It also gives MUtech developers the assurance that they can support avatars and other active objects of all kinds across multiple worlds.

To minimise the impact of differences between MUtechs, all functionality that can be MUtech-specific is split out into separate nodes to be supplied by the MUtech provider. MUtech developers are expected to compete by being able to support different sorts of worlds (e.g. a Myst world with super multimedia, vs. a stock-trading world linked to huge databases) and world developers will select the MUtech that best supports the particular features they want to exploit.

4.2.3 Open Community

Open Community [2] combines a portable multi-user interactive Java API with a VRML prototype layer. Using Open Community, the content developer can create a multi-user virtual world which can operate on any server that supports the Open Community API. The Open Community specification builds on Spline (Scaleable Platform for Large Interactive Network Environments) [13], a distributed multi-user virtual world infrastructure. Open Community includes several desirable features: high scalability, through the use of regions, which is an important feature on the Internet; a streamed media identification system to accommodate new media types; and multicase communication to reduce latency for animation, voice, and chat communication.

4.2.3.1 Infrastructure

The Spline system was originally designed and implemented by Mitsubishi Electric Research Labs to support a virtual world called Diamond Park, where avatars could travel around a large park, talk to others using proximity-based voice chat, ride bicycles in a velodrome, create new world views, and play multi-user games.

Open Community's world model is based upon Splines's and it is implemented through an object-oriented database. The objects have methods associated with them as well as data stored in their instance variables. Applications observe the virtual world by retrieving data from the world model database and affect the virtual world by adding, removing, and modifying objects in the world model. To avoid read/write conflicts, each object in the world model has one process as its owner and only the owning process can modify it. However, the ownership of an object can be transferred from one process to another.

4.2.3.2 Scalability

To provide low latency interaction, the world model is replicated so that a copy resides in each one of the networked application processes. Changes from one world model copy to another are propagated through messages sent over the network.

Two aspects of Open Community support scalability to a large number of users: (i) providing only approximate equality of local world model copies, and (ii) dividing the world model into chunks (referred to as regions) each of which is communicated only to the small group of users that are actually interested in it, rather than to all of the networked users, through the use of user services or multicast groups.

Distributed databases typically require that all local copies of the database must agree exactly on the information in the database. However, this requires object locking and handshaking that is incompatible with real-time interaction if there are a large number of users connected. With Open Community world model copies are only approximately equal in the sense that different users observe events occurring at slightly different times. The time it takes to update a world model running locally in one application process with respect to changes made by a remote process depends on the network `distance' between the two processes. In general, this distance is not more than a couple hundred milliseconds and does not lead to world model differences that are unduly large.

To minimise the computation required to maintain each local world model copy, Open Community breaks the virtual world into many small regions, and communicates information about a given region only to the small number of users that are near enough to that region to be interested in it. Each region is associated with a separate user service or multicast communication channel so that processes that are not interested in a region do not have to expend any processing. This allows Open Community to scale based solely on the maximum number of users that are present in any one region, rather than on the total number of users in the virtual world.

4.2.3.3 Communication

Open Community is based on a hybrid communication model. To minimise latency, and prevent bottlenecks, the primary communication used by the system is multicast rather than passing through centralised processes. However, centralised server processes are used for five key services:

(i) A session manager handles the connection/disconnection of users to an on-going session. Its workload is proportional only to the number of users that enter or leave the session in a given time and not to the total number of connected users.

(ii) User servers are used to support users with low network connections (e.g. modems). When acting in this role, a server intercepts all communication to and from the user. The message traffic is compressed in order to take maximum advantage of the bandwidth available (e.g. audio streams are combined and localised before propagated to the user). Servers are replicated as needed so that no one server has to accommodate more users than it can handle.

(iii) A region server maintains a record of everything in a given region. When a process enters a new region, the appropriate region server is queried in order to obtain initial information about the state of objects in the region. After this initial download, the user process obtains further incremental information by multicast communication. Responsibility for regions is distributed among a set of servers, which is made large enough that no one server is responsible for a larger piece of the virtual world than it can easily handle.

(iv) Because the world model copy in an Open Community process only contains information about the objects in the regions it is attending to, there has to be an explicit mechanism for locating far away objects in the virtual world. This is done by having the Open Community servers provide a name service that makes it possible to locate specialised objects called beacons by name no matter where they are in the virtual world. A process can use this name service to rapidly locate any named object through the appropriate beacon server.

(v) To simplify the communication between user processes and the various servers it has to interact with, each process has a server assigned to it that acts as a sole contact point for the process. Every message from a user process (or user server acting on its behalf) that requests a service, is sent to the contact point. This allows the user process to always operate as if there was only one server. The various complexities that arise when servers are replicated are handled by the contact point, which decides where to route the messages it receives from user processes.

Depending on their type, data in the world model can be communicated using various messaging approaches :

(1) Changes in small objects can be sent using multicast or unicast UDP or TCP packets. (For UDP, the objects must be small enough that a message describing them fits in a single UDP packet.)

(2) Graphic models, recorded sounds, and behaviours are represented using large objects. These objects are identified by URLs and communicated using standard Web protocols. Standard formats are used so that standard tools can be used to create models, sounds, and behaviours. In Open Community, the primary formats are VRML, MIDI, and Java respectively. Voice and general sound communication in Open Community may require the use of proprietary compression formats, to accommodate low speed connections. Since these objects change infrequently, this latency time can generally be masked by pre-loading the objects before they are needed.

(3) The final kind of object in the world model corresponds to continuous streams of data such as sound captured by a microphone. These streams are communicated in small chunks using multicast or unicast UDP messages. (Video streams are not yet supported).

4.2.3.4 Open Community API

The Open Community API consists primarily of operations for creating and deleting objects in the world model and reading and writing their instance variables. The application support module contains various tools that facilitate interaction between an application and the local world model copy. Java is the primary high level interface for Open Community. Objects are defined using Java with a few small extensions. The extended Java syntax is supported by a pre-processor called SPOT. In general, SPOT takes in a Java file containing a shared class definition and produces: a new Java file using standard syntax that adds the shared class to the Java API and a class descriptor file that is used by the system core.

Also, Open Community provides a general API for terrain following algorithms. Given any 3D point, an Open Community implementation could calculate whether a collision with a fixed obstacle is occurring and the height of the `ground' below the input point.

Visual and audio rendering are performed by browser plug-ins running on top of Open Community. They do not have to be tightly coupled with the main application or the Open Community core. By default, Open Community operates with VRML 2.0 renderers, but demanding applications may switch to renderers tuned to specific applications.

4.2.3.5 Summary

Open Community provides a Java/VRML 2.0 API that allows content developers to create multi-user worlds that run on any server supporting the Open Community API. This work complements other specifications under development, such as Universal Avatars and Living Worlds, by providing interacting behaviours between avatars, an object ownership model, efficient communication mechanisms, and scalability through the use of regions.

4.2.4 Living Worlds vs. Open Community

Living Worlds and Open Communities both address the same question: what aspects of the technology for shared virtual worlds should be standardised and what left open for innovation?

Living Worlds assumes that VRML will be the language of choice for implementing multi-user virtual environments. LW focuses clearly on technology that can be accomplished using VRML 2.0. Anything that can't be done inside the current standard is declared by definition to be "MUtech specific," i.e. an implementation matter outside the boundary of the interface standard.

By contrast, the Open Community work makes no assumptions about the use of VRML. Instead of starting with objects in a virtual space, they start with a "world model" that is a pure semantic abstraction. Their goal is to insure that behaviours in different worlds, however the behaving entities may be rendered, can be co-ordinated among multiple participants running on distributed platforms. Thus, they offer a standard set of multi-user functionality, and make this available by APIs in Java and C. Thus Open Community proposes standardising precisely those interfaces which LW proposes to keep open.

4.3 Scripting Support

There are two main ways to provide complex interactions and behaviours: through the use of Script nodes that operate upon data flow streams within the VRML scene graph; and through the External Authoring Interface (EAI) [1] which allows an external process to control the VRML scene browser.

In the current VRML specification the languages that can be used in the Script Node are described in two annexes, one for Java and one for JavaScript. Since they are normative annexes a browser does not have to support either in order to be compliant with the standard.

The VRML specification does not preclude additional language bindings, though in the opinion of some correspondents on the VRML mailing lists, there is already one too many. Bindings for compiled languages such a C++ are certainly possible, but there would be a problem of operating system dependence of the resulting byte codes.

The EAI is currently the subject of a proposal for an extension of the VRML specification.

4.3.1 Script Nodes

The Script nodes allow the manipulation of data flow through the VRML scene graph to enable more interesting behaviours to be developed. The definition of the Script node is shown in ***Figure 4-2***.

```
Script {
  exposedField MFString url           []
  field        SFBool   directOutput  FALSE
  field        SFBool   mustEvaluate  FALSE
  # And any number of:
  eventIn    eventTypeName eventName
  field      fieldTypeName    fieldName   initialValue
  eventOut   eventTypeName eventName
}
```

Figure 4-2 Fields of the Script Node

The first three fields define the code that will be executed when the Script node is evaluated and give hints to the browser execution engine about the side effects of this script. Any further fields form the interface to the action of this script. These further fields can be of any type (including SFNode and MFNode), **eventIn** and **field** values can be read by the code associated with the Script, and events can be generated on the **eventOut** fields.

The manner in which the code reads and sends events is language dependent. The bindings for Java and JavaScript are described in Sections 4.3.1.1 and 4.3.1.2. The code is referenced by a URL given in the Script's **url** field. This URL can be multi-valued allowing implementations to be given in a number of different languages, so that a trade-off can be made between, say, speed and platform independence.

4.3.1.1 Java Binding

Java is an object-oriented, platform-independent, multi-threaded, general-purpose programming environment developed at Sun Microsystems, Inc [11].

In this case the **url** field of the Script node defines the name and location of a file containing the Java bytecode that 'manages' this Script node. ***Figure 4-3*** shows an example Script node definition, that uses a Java class.

```
Script {
  url "http://foo.bar.com/Example.class"
  eventIn SFBool my_input
  eventOut SFBool my_output
}
```

Figure 4-3 Example Script node using Java

When events are delivered to the Script node the browser calls one or other of the Java methods processEvent or processEvents that must be defined in the Java class referenced by the script node.

For example, the Example.class file referenced in ***Figure 4-3*** was produced from the Java source code in ***Figure 4-4*** which implements a Boolean NOT function.

```
 import vrml.*;
 import vrml.field.*;
 import vrml.node.*;

public class Example extends Script {
  SFBool my_output;

  public void initialize()   {
     my_output = (SFBool)getEventOut("my_output");
  }
     public void processEvent (Event e) {
       SFBool result;
        if (e.getName().equals(“my_input”)) {
           result =((ConstSFBool)event.getValue()).getValue();
           my_output.setValue(!result);
     }
 }
```

Figure 4-4 Example Java class

The file illustrates both how events are received from the data flow, and how events are generated. For the input, the method *processEvent* is automatically called and passed a structure that include the name of the eventIn and the associated value. In order for output to take place, a link must first be made between a variable local to the script code, and an eventOut field in the definition of the script node. This is done in an *initialize* method, which is called when script node is created. Once this link has been established, setting the value of the variable results in the propagation of that value from the corresponding eventOut.

The Java file includes three packages: vrml, vrml.node and vrml.field which contain the Java classes that support the Java API. The vrml package contains methods a script can execute that manipulate the browser or scene. It includes methods for getting the current frame rate, speed of navigation of the user, loading and replacing vrml scene graphs and adding and deleting routes. The vrml.node and vrml.field packages define methods for accessing and manipulating vrml nodes and fields. Once a Java script has access to a VRML node, by its being passed by reference to a SFNode or MFNode eventIn, it can manipulate that Node using the methods in vrml.node. This allows it, amongst other things, to access that node’s eventIns and eventOuts, and to write to them directly bypassing the data flow mechanism.

4.3.1.2 JavaScript Binding

```
Script {
  eventIn SFBool my_input
  eventOut SFBool my_output
   url ["vrmlscript:
           function my_input(value)  {
        my_output = !(value);
           }
        “]
}
```

Figure 4-5 Example Script Node using JavaScript

The JavaScript [12] binding grew out of the need for a lightweight script language in VRML. The major differences between the JavaScript and Java bindings can be seen in ***Figure 4-5*** which gives a JavaScript version of the example from the previous section.

From this example we can see that the browser calls a JavaScript method that has the same name as the eventIn, rather than an event processing function, and that eventOuts can automatically be accessed with the same name as given in the Script node definition.

The advantages of JavaScript over Java can be summarised so:

- Scripts can be included in source form within the VRML file as well as being referenced by a URL.
- All VRML 2.0 data types are supported directly. Some, like the vector types have built in methods (such as cross product and normalise) to simplify working with them.
- Receiving eventIns is handled with separate functions to ease development and to speed processing.
- Sending eventOuts is done with simple assignment.
- Scalar data (SFTime, SFInt32, SFFloat, SFBool) can be used directly in expressions. The JavaScript number object converts directly to any of the four scalar data types. For instance, you can add 3 seconds to an SFTime value with a = time + 3;.
- Constructors are available for most data types to ease creation and conversion of data. Constructors typically take a flexible set of parameters to allow construction of objects with any initial value.
- A full set of JavaScript compatible math, date and string objects is available.
- The full set of JavaScript string methods and properties are available. Scalar values automatically convert to strings when concatenated. This makes construction of URLs and VRML strings (for use in createVRMLFrom-String) easy.

Whilst this makes JavaScript very good for simple tasks, Java has the advantages that it is faster since JavaScript in an interpreted language, and the full power of the Java libraries is available. For example, this means that a Script node written in Java can use the widget and networking extensions.

4.3.2 External Authoring Interface

Unlike the Java and JavaScript annexes, the proposed EAI [1] bypasses the data flow mechanism and allows a process external to the browser to control the scene. For example, it allows a program running within a HTML browser frame to obtain a reference to a vrml browser, and use this to control the vrml browser by calling its methods. Implementations of the EAI already exist, though they only support the Netscape Navigator 3.0 and the LiveConnect interface (an alternative would be to use an ActiveX interface in Internet Explorer).

The EAI offers two main facilities: browser control and node event watching.

Browser control features allow the external program to change the description of the world, load new objects into the scene graph, replace the current world, retrieve the current frame rate and addition and removal of routes. In many ways these functions are very similar to those found the vrml package specified within the Java Scripting Reference (see Section 4.3.1.1).

Node event watching allows the program to send events to node eventIn fields, or access or register call-backs on eventOut fields being updated. This is possible since the EAI contains function to obtain references to nodes by looking up the names defined using the DEF construct (see Section 2.2.3).

4.4 Avatar Standardisation

Two main efforts are aimed at avatar standardisation, the Universal Avatars and the VRML Humanoid Animation working groups. The Universal Avatars proposal has applications broader than VRML, as it proposes a standard for encapsulating identity in various network applications. This includes personal information and avatar representation and behaviours. The avatar representation will be that proposed by the Humanoid Animation group, and we discuss their proposal in the following sections.

4.4.1 Specification of a Standard VRML Human Figure

Within the VRML Consortium, an official working group has been formed for specification of a standard VRML humanoid. This group, called "VRML Humanoid Animation" working group (H-ANIM), has a mandate to provide a minimal definition which will allow authoring tools (both VRML and non-VRML) to export a humanoid in a form which may be used by a wide range of applications.

Although the group is in an early stage, it has already prepared an initial draft specification of a standard virtual human. The specification is divided into the following sections:

a) Modelling the Humanoid

This section defines a standard transformation hierarchy in the VRML 2.0 file format. The approach is to define the joints as a series of nested Transform nodes in VRML. For example ***Figure 4-6*** shows a possible definition of the left arm.

```
DEF l_shoulder Transform {
  center     0.167 1.36 -0.05
  children   [
    DEF l_elbow Transform {
      center 0.196 1.07 -0.0518
      children [
         DEF l_hand Transform {
            center 0.213 0.811 -0.0338
            children [
               DEF left_hand Shape {
                  ...
               }
            ]
         }
         DEF l_left_lowerarm Shape {
            ...
         }
      ]
      DEF left_upperarm Shape {
        ...
      }
    }
  ]
}
```

Figure 4-6 VRML Definition of the Left Arm

b) Describing the Humanoid

This section provides a standard way provide more information of the virtual human figure than its geometry, appearance and transformation hierarchy. This information could typically include such things as copyright, creation date, usage restrictions, etc. The information could also contain dimensions of various body parts, as well as mass and centre of gravity of each segment of the body.

An example info field is shown in ***Figure 4-7***.

```
DEF HumanoidInfo WorldInfo {
  info {
     "copyright=EPFL LIG and MIRALab University of Geneva, 1997"
     "bodyHeight=1.7"
  }
}
```

Figure 4-7 Example Info Field

The whole list of information tags is not defined yet.

c) Run Time Joint Binding and Extensibility

For some applications, a dynamic routing might be necessary among the nodes of the humanoid instead of static parent-child relationships among the joints. In addition, it will be useful for the person to attach additional joints to the body. for this person, a new node is added to the VRML file for the virtual human. This node is defined by a PROTO, used to associate a joint name (in the form of SFString) with a joint (a reference to a Transform node).

d) Additional Useful Features

A set of additional features is included, such as ViewPoint nodes in order to display the virtual human from different angles. In addition, a navigation info is added to put the VRML browser into predefined mode (e.g. Examine).

4.4.2 Typical Uses for Standard VRML Human Figures

In addition to the specification, a separate document illustrates example uses of VRML Humanoids. The set of examples includes:

- *Simple, repetitive keyframe animations*. This example might be used for "background characters", "extras" in the virtual world. They keyframe animation is converted into a series of interpolator nodes which are driven by a TimeSensor with its loop field set to TRUE. The outputs of the interpolators are ROUTEd to the body parts of the humanoid.

- *Multiple actions, triggered by sensors*. This example is for humanoids with multiple actions, for example walking and sitting. The approach is based on creating a keyframe file for each action, and then playing them exclusively. The actions are driven by independent TimeSensor nodes triggered by sensors. Note that the actions are not played concurrently.

- *Multiple concurrent animations*. This example has been given for playing multiple actions at the same time, for example walking and waving. This involves triggering more than one TimeSensor simultaneously. The problem of multiple actions controlling the same body part has not been addressed in the document, therefore it is relatively simple.

- *Compressed animation data*. This is a proposal for the future to use compressed animation data. This is done typically using the Script node that decompresses the data in remote file and uses it to drive the rotations of joints. The approach for compression is not defined, but most probably the MPEG-4 SNHC parameters will be used.

- *Inverse kinematics operations*. This example is to ask the humanoid to reach a particular point. This is done again by a Script which allows the inverse kinematics script to access body dimension information.

- *High-level behaviours in a multi-user environment*. This is an example of high level behaviours such as walk, wave and smile, which have very low bit rate requirements.

- *Live data from a performance animation system*. This is for virtual humans controlled by motion capture, using a special Script to connect to the remote machine on a given port.

- *MPEG-4 data*. This is for virtual humans controlled by compressed parameters.

5. VRML and WWW Techniques

VRML was designed as a 3D object standard for use over the Internet, so it is tightly integrated with WWW technologies. For instance, VRML browsers are usually provided as WWW browsers plug-ins, and VRML has taken the concept of hyper-links and applied it to 3D objects that can act as “anchors” to other worlds. In this section we examine the connection between WWW Techniques and VRML.

5.1 MIME Types

A Multipurpose Internet Mail Extension (MIME) [6] type specifies the type of media and format of a file. Specifying a MIME type allows a browser to apply filters and start other browsers when particular types of data are retrieved. VRML2.0 is identified with the MIME type *model/vrml*. For compatibility with VRML1.0 the type *x-world/x-vrml* must also be supported. The MIME type of a file is usually defined in a protocol header when the file is transferred, but generally VRML files can be identified by the *.wrl* suffix.

Since a VRML scene can define properties by reference to external files, such as images for texture maps, or code for scripts, these too have known MIME types, though they are not defined by the VRML document. For example the MIME type for a JavaScript file is *application/x-javascript* and for Java bytecode file is *application/x-java*.

5.2 Resource Location

The components of a VRML scene are specified using Uniform Resource Locators (URLs) [10]. These refer to files available on named servers via a named protocol (usually http or ftp). In VRML, their use is slightly extended, in that multiple URLs may be given whenever a resource has to be specified (for example the *url* field of the Script node). The list of URLs gives an order of preference for the version of the resource to be loaded. This is used in two manners. Firstly as a guarantee against the first resource being unavailable or secondly, when Scripts are being defined, to allow a behaviour to be defined in a number of languages so that browser dependencies are avoided.

Two future developments are required to be supported once they are standardised. These are Uniform Resource Names (URNs) [7], and the IETF’s proposed Data Protocol [9].

URNs are location and protocol independent identifiers for Internet resources. A typical URN gives a name for a resource, and a URN aware browser would then be able to determine the nearest site with that resource and the manner in which to retrieve it. Although there is no standard for URNs at the moment, they can be assigned on domain names.

The Data Protocol allows binary data to be in-lined within other files using a base-64 encoding. It will allow, for instance, a JPEG file to be included with the main body of a VRML file.

5.3 WWW Integration

VRML scenes may contain Anchor nodes that act as the 3D version of 2D hyper-links. Essentially these define a group of geometry, that once clicked upon by the user causes the loading of a URL. Since this URL might indicate a resource of any MIME type, this allows integration with standard WWW resources. There are two manners in which this might happen: if the VRML browser is a plug-in to a scene, the 3D view would probably be replaced with the new presentation, but if the VRML browser is a helper application, it sends the URL to a standard WWW browser for loading.

There are two other manners in which 2D and 3D resources can be integrated: through the scripting language APIs (see Section 4.3.1), and through the External Authoring Interface (see Section 4.3.2).

For the scripting languages, both the browser APIs support a call similar to

void **loadUrl**(String [] url, String [] parameter)

which allows the loading of URLs to be determined by more complex events within the VRML environment.

The EAI provides a similar call, but as described in Section 4.3.2, it also provides more broad communication between applets within the browser.

6. Preliminary Critique

The state of the art in VRML is moving very quickly at the moment, so many objections to current VRML standards or practices may well be obsolete within a few months. However there are issues to address about the manner in which VRML will progress and some gross design issues that might become problems later.

6.1 Support for Collaborative Virtual Environments

A set of requirements to support collaborative virtual environments were given as part of the COVEN review of DIS and HLA [17] derived from a study at Nottingham University [19]. They are reproduced here to give an overview of the areas which VRML currently addresses and which are currently within the scope of VRML working groups.

General design requirements of future distributed virtual environment systems can roughly be classified in three broad categories:

1. requirements associated with the general characteristics of virtual environments;
2. requirements due to distributed systems aspects;
3. requirements related to scalability and heterogeneity issues.

6.1.1 General Characteristics

Requirements from the first category include the following:

- Rendering - not only visual, but also auditory, haptic, and tactile feedback;
- Low lag, high update rate - with the highly interactive, immersive VEs requiring very low latencies;
- Collision detection - not just at the bounding box level, but at much more detailed levels for effective interaction;
- Navigation and viewpoint control - provide different “vehicles” to move around in the virtual environment;
- World construction facilities - i.e., support for geometric and behavioural modelling *within* the VE;
- Persistence - maintaining the state of the VE across different instantiations;
- Exported behavioural models - remote simulation of object behaviour instead of frequent transmission of state updates.

VRML certainly supports both auditory and visual properties of objects, but not haptic nor tactile. Audio support is rather limited in that sounds can only be played back rather than streamed from external sources. Video support is considerably better, MPEG movies are supported directly, and video can be streamed from external sources through the use of dynamic PixelTexture nodes.

VRML was designed to run across many different platforms, so the rendering and audio specifications are fairly simple in order to give high update rates (see Section 3.4). There are hints within the scene to enable the browser to make scene rendering optimisations (e.g. level of detail nodes, and the ability to explicitly define bounding boxes in grouping nodes). However there are no explicit mechanisms by which to guarantee a frame rate, though it would be possible for a script to monitor the present frame rate and make alterations to the scene accordingly. Lag is only addressed in the basic specification through the ability to lazily evaluate the data flow.

Collision detection of the viewer and the scene is supported, but collision detection between pairs of objects is not. The latter was not addressed in the base standard because of the computational overheads that it would incur, though it is expected that the appropriate nodes will available as prototype nodes in the not too distant future.

Navigation metaphors are supported within the specification and can be scripted in VRML if the default behaviours do not suffice. However as mentioned in Section 3.3.5, there is a definite bias towards desktop metaphors and several authors have noted the difficulty of supporting immersive viewing [22,21,20].

No world construction facilities are directly provided, but tools to accomplish this can be scripted. For example, although still in preliminary development VermelGen [23] is a set of Java classes that use the External Authoring Interface, to provide editing tools for many of the VRML nodes.

With VRML files being specified by name, and usually loaded via the WWW, worlds are not explicitly persistent. This is one of the issues being addressed within the Living Worlds working group.

Remote simulation is possible using scripting, though the mechanisms that might used to specify the transfer and execution of behaviours are not defined.

6.1.2 Distributed Systems Aspects

The distributed nature of future large scale CVEs generate a number of specific requirements:

- Naming - facilities for identifying large numbers of users, systems, objects, etc. on a potentially global scale;
- Trading - managing the exchange of services between system components;
- Resource discovery - locating objects and services of interest;
- Distributed storage - with all its related issues of data duplication, concurrent access, fault tolerance, etc.
- Security - authentication and authorisation;
- Communication - i.e., point-to-point, broadcast, multi-cast mechanisms;
- Continuous media support - facilities that allow not only the exchange of discrete events between entities, but also data streams such as audio and video.

VRML was designed so that it would be easy to distribute worlds, but the base specification does not define any networking support layers. The discussion of networking extensions in Section 4.2 indicates that these issues are being addressed, but since implementations of the technologies are not yet freely available, it is difficult to describe how the proposals satisfy the above requirements. A few points can be made though. Open Community describes both point to point and multicast connections, the issue of resource naming, and the distribution of events and real-time streams. Issues of security and distributed storage are slightly higher level and more within the realm of the Living Worlds specification, though there is overlap in the two specifications. Services for Trading and Resource Discovery are not explicitly defined, but could be built upon Living Worlds either system.

6.1.3 Scalability and Heterogeneity Issues

Finally, the scale and heterogeneity of future distributed CVEs lead to a number of high level requirements, some of which are intimately related to the requirements mentioned above:

- Multiple users - possibly huge numbers of them;
- Co-operation support - promoting mutual awareness of and communication between users;
- Multiple applications - support for running many different applications concurrently;
- Multiple worlds - different applications and/or user groups may participate in different virtual environments at the same time, possibly requiring connections between the different worlds;
- Subjectivity - that is, how are objects represented on different VE platforms with widely differing capabilities;
- Negotiating capabilities - to resolve conflicts between users of heterogeneous systems with incompatible capabilities.

The Living Worlds proposal defines mechanisms for allowing multiple zones, and rules for accessing these zones, so we presume that subjective views, mutual awareness and multiple worlds can be supported. It is not clear whether negotiating capabilities can be supported, or whether this is within the scope of Living Worlds or left to the multi-user technology.

6.2 Procedure Issues

6.2.1 Standardisation and Conformance

Many common concerns about the usefulness of VRML derive from the experience with VRML1.0 where worlds could not be expected to appear identically on different browsers. Much effort has been expended in the design of VRML to remove such browser dependencies, and the ISO process made the conformance section much more concrete, see Section 3.3.7.

However a major issue remaining is that a VRML browser does not have to support either scripting reference nor the external authoring interface in order to comply with the specification. The current effect of this is that authors are using the multi-valued url fields of Scripts to give a behaviour in both Java and JavaScript in order to support as many browsers as possible. This is exacerbated by the face that implementations of Java do differ despite its being a standard. This is a particular concern of the VRML Java working group [25].

Other issues of conformance are being dealt with by a conformance working group [24]. This will focus initially on the colour and lighting models, which were a source of problems for VRML1.0.

6.2.2 Working Groups

The working group process is designed to ensure that all issues relating to the development of VRML are dealt with in a fair and open manner. Anyone may join a working group so there is scope for both corporations and individuals to contribute at a useful level, before proposals go to the VRML Review Board for ratification. However there are no requirements on groups to follow established standards or procedures apart from their internal considerations of the likely hood of their proposal being accepted. An example where a liaison is taking place is within the Humanoid Animation working group [14] who are using the body parameters proposed for MPEG-4 [18]. Another is the announcement of a possible working group applying the Distributed Interaction Simulation protocols to multi-user VRML worlds [15]. However it is too early in the working group process to tell whether such dialogues will be sufficient to ensure the ease of interoperation of VRML with other systems.

7. Conclusions

We have introduced VRML2.0 and some of the many areas of associated activity. The VRML specification offers a degree of standardisation in the areas of describing dynamic and interactive virtual environments, and is well suited to WWW integration. Through the extension mechanisms, it appears possible to support large scale collaborative worlds, though the required infrastructure is only just appearing. It seems likely that VRML2.0 will use existing standards and techniques, especially in the areas of human modelling and networking support.

Some reservations have been expressed about a few aspects of VRML, but these minor problems that are being addressed. The success of VRML as a platform for CVEs now depends upon the flexibility of the networking APIs and the application of techniques for scalability. The work of COVEN in work packages 1 and 8 will be highly relevant to this area, and we expect to participate in the relevant working group processes.

8. Bibliography

[1] Chris Marrin, *Proposal for a VRML 2.0 Informative Annex, External Authoring Interface Reference*, http://www.graphcomp.com/info/specs/eai.html

[2] David Anderson, Dan Greening, Moses Ma, Maclen Marvit, Richard Waters, *Open Community*, http://atlantic.merl.com/opencom/

[3] Yasuaki Honda, Mitra, Bob Rockwell, Bernie Roehl (editors), *Living Worlds*, http://www.livingworlds.com/draft_1/index.htm

[4] British Standards Institute (various), *BSI Comments on CD 14772 - VRML*, November 1996.

[5] Gabriel Taubin, Bill Horn, Francis Lazarus, Mitra, Fabio Pettinati, *VRML 2.0 Compressed Binary Format*, http://www.rs6000.ibm.com/vrml/binary/

[6] David Anderson, John Barrus, John Howard, Charles Rich, Chia Shen, Richard Waters, *Building Multi-User Interactive Mutlimedia Environments at MERL*, Technical Report TR-95-17m Mitsubishi Electric Research Laboratories. Available at http://www.merl.com/

[7] IETF work-in-progress, *Universal Resource Name*,. ftp://ftp.ietf.org/Internet-drafts/draft-ietf-urn-syntax-02.txt, http://services.bunyip.com:8000/research/ietf/urn-ietf/

[8] IETF work-in-progress, *The Model Primary Content Type for Multipurpose Internet Mail Extensions*, ftp://ftp.internic.net/rfc/rfc2077.txt

[9] IETF work-in-progress, *The Data: URL scheme*, Internet Draft, http://www.internic.net/Internet-drafts/draft-masinter-url-data-02.txt

[10] IETF, *RFC 1738 Uniform Resource Locator*, http://ds.internic.net/rfc/rfc1738.txt

[11] Ken Arnold and James Gosling *The Java Programming Language*, Addison Wesley, Reading Massachusetts, 1996.

[12] Netscape Communications, *JavaScript Language Specification*, Version 1.1, 18 November 1996, http://home.netscape.com/eng/javascript/

[13] John Barrus, Richard Waters and David Anderson, *Locales: Supporting Large Multiuser Virtual Environments*, IEEE Computer Graphics and Application, November 1996.

[14] VRML Humanoid Animation Working Group, *Draft Specification for a Standard VRML Humanoid*, http://ece.uwaterloo.ca:80/~h-anim/spec.html

[15] Don Brutzman, message to www-vrml mailing list, Saturday 8th March, 1997.

[16] Anthony Steed, *The New VRML2.0 Document*, ERCIM VRML Workshop 29/30 January 1997, edited by D. Duce, K. Kansy and M. Wilson, ERCIM Research Report Series.

[17] *Review of DIS and HLA techniques for COVEN*, COVEN Project Report AC040-TNO-FEL-DS-P-071.b0

[18] *MPEG-4 SNHC Verification Model*, http://drogo.cselt.stet.it/mpeg/documents/w1545.zip

[19] D. Snowdon, C. Greenhalgh, S. Benford, A. Bullock and C. Brown, *A Review of Distributed Architectures for Networked Virtual Reality*, to be published in VR Research Applications.

[20] Anthony Steed, *Data Flow Languages for Immersive Virtual Environments*, From Desktop to Webtop: Virtual Environments on the Internet, WWW and Networks, British Computer Society, 14-17 April 1997

[21] John Edwards and Chris Hand, *MaPS: Movement and Planning Support for Navigation in an Immersive VRML Browser*, VRML97, The Second Annual Symposium on the Virtual Reality Modelling Language, Monterey February 24-26 1997.

[22] Randy Stiles, Sandeep Tewari and Mihir Mehta, *Adapting VRML2.0 for Immersive Use*, VRML97, The Second Annual Symposium on the Virtual Reality Modelling Language, Monterey February 24-26 1997.

[23] Justin Couch, *VermlGen*, available at http://ww.vlc.com.au/VermelGen

[24] *NIST VRML Testing*, http://www.nist.gov/vrml.html

[25] *Java-VRML Working Group*, http://www.sea.co.il/Java-Vrml/