Building and Supporting a Large-Scale Collaborative Virtual Environment

Anthony Steed
Department of Computer Science, University College London
London, United Kingdom

Emmanuel Frécon
Swedish Institute of Computer Science,
Stockholm, Sweden

Abstract

The COVEN project's London Travel application demonstrates several techniques that enable large-scale virtual environments. By large-scale we mean both geometrically complex environments and support for many simultaneous participants in the world. The demonstrator is concerned with travel to a virtual model of London, where participants can plan routes, investigate tourist information and meet with fellow travellers or guides. In this paper we discuss how the Distributed Interactive Virtual Environment (DIVE) platform has been extended to meet the application requirements. In particular we shall discuss the technical problems involved in maintaining and rendering a scene database many times larger than one might usually encounter.

1 Introduction

Travel can be a stressful experience, and it is an activity that is difficult to prepare for in advance. Although maps, routes and landmarks can be memorised beforehand, travellers do not get much sense of the spatial layout of the city and can easily get confused when they arrive. There is little doubt that virtual reality can assist in such situations by, for example, providing walkthroughs of virtual cityscapes to effect route learning.

For this application we have constructed a model of the centre of London covering 160 km^2. We have added several services that might aid the traveller such as simulations of public transport, data visualisation services and conferencing tools. The underlying

geometric model is automatically generated from 2D data, so although its geometry is roughly correct, the texture information on the side of the buildings is not. What is important for our purposes is that wherever the traveller strays, there is at least some representation of the city and that this representation is at least believable for someone unfamiliar to the area. This is important in navigation training exercises where it should not be immediately apparent to the subject that they are lost because the scenery is petering out.

The application is built on the DIVE system from the Swedish Institute of Computer Science [10]. The development of this platform[1] and the construction of this application[2] were two of the main goals of the COVEN project [16]. COVEN extensions to the DIVE platform include better support for real-time audio and video, broader file format support, collaboration tool development, human representation and scalability extensions for the renderer and database.

2 Related Work

There are many examples of models of urban environments in the computer graphics literature. Such applications have obvious applications in local government (urban planning), training (driving simulators), data visualisation (air pollution data) and so on. Work can be typified by two systems, the UNC Walkthrough environment [3] that supported real-time experience of complex building interiors and the Virtual LA system that support a large urban space with gross building-level detail [1].

Such applications are similar in that they require large amounts of geometric information. They are also similar in that from a particular viewpoint there will be a high depth of complexity and there is usually large amounts of occlusion between regions in the model.

Models of building interiors lend themselves to analytic determination of visibility through identification of cells and portals [3]. Essentially within a building environment it is possible to identify a graph of *cells*, where adjacency in the graph indicates that the two cells are mutually visible, i.e. there is a *portal.* Once such a graph has been drawn, visibility between two cells can be determined by traversing the graph between the cells, checking that all the intervening portals are visible.

Such analytic solutions are difficult to compute in city models, since portals do not exist, and occlusion between regions is generated by an accumulation of intervening buildings. A more appropriate technique in this case is not to pre-compute what is visible, but to pre-compute what is likely to obscure distant objects [7]. In a city model this is appropriate since a few nearby buildings can obscure large sections of the city.

---

[1] An experimental version of DIVE, 3.3x, incorporating some of the techniques described in this paper in available at *http://www.sics.se/dive*

[2] Cut-down versions of the London Travel Demonstrator for DIVE and VRML are available at *http://www.cs.ucl.ac.uk/research/vr/Coven/Model*

The primary use of visibility techniques is to assist us in determining which parts of the models need to be sent to the graphics pipeline. They have a second role when the model of the environment is too large to fit into memory. The basic technique for supporting a large model would then be to partition it into regular tiles, and, for example, only load in the nearest 9 tiles. If a visibility solution is available then it is possible to load only those parts that are visible or likely to become visible [11,12]. In scenes with little occlusion and thus large numbers of visible objects, more sophisticated techniques are required [2].

The London Demonstrator is unique since it combines several constraints:

- In general there are many potentially dynamic objects with high levels of importance to the user. In particular there will be avatars of other users. Many large model rendering systems assume that the user simply wants to view the model.
- Interaction with model and other participants requires a consistent high frame-rate.
- There are a variety of model structures including both dense interiors and sparse exteriors.
- The platform supports VRML1.0 and partial VRML97 formats and thus we must consider models without clean structure and without the luxury of an identification of cells and portals.
- The platform supports multiple users and uses a shared scene graph approach to distribution. One implication of this is that any supporting structures for rendering or collision detection must be parallel to the original scene graph. Memory being a scarce resource, they must be kept fairly lightweight.

3 Building the Application

3.1 Map 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 The GeoInformation Group's, Cities Heights data set[3]. The resulting model consists of about 160 MB of VRML data, not including texture map data.

The original maps are stored as DXF files with several layers. We are primarily interested in building outlines, roads centre and building spot height layers. The conversion stage consists of the following major steps [19]:

- Identifying and repairing building outlines

[3] Cities Revealed, The GeoInformation Group, *http://www.crworld.co.uk*, 

- 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 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 be office blocks, and so on. The resulting model is stored as a VRML1.0 [4] model in 160 1km square tiles, with each tile being built upon a segment of the Cities Revealed aerial photograph for central London. Figure 1 shows an example view of the model and Figure 2 shows an overview of the model extent.

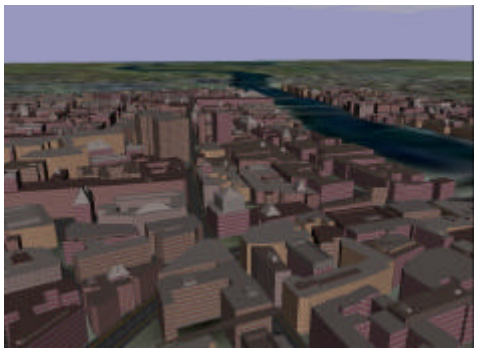

Figure 1: Typical snapshot of the model.

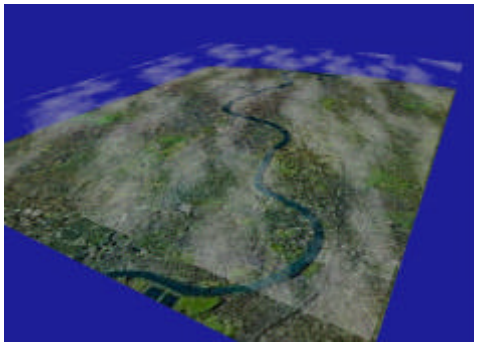

Figure 2: London from the south-west.

3.2 Other Databases

Given that the application is for travel training, there are a few “target” destinations that have been modelled to a higher degree of accuracy by hand. One of our target destinations is the UCL Computer Science Department, and there are models of the buildings on campus, and a detailed model of the interior of computer science. Dealing with this range of scale is obviously an issue when composing models.

3.3 Other Services

The main services we have added to make this a useful travel service include simulation of transportation services such as the London Underground and Heathrow Express, and multimedia and presentation services. The conference and multimedia facilities include a number of collaboration tools such as virtual whiteboards, overhead projectors, virtual notepads and calendar facilities [8]. These are represented in a realistic manner within conference rooms and individual conferences can be made public or private with DIVE's audio and event scoping capabilities.

The final service, the visualisation service, comprises all the tools that extend the realistic metaphor of the city with more abstract data visualisation tools. The main tool is a tourist information graphing tool, that allows multiple participants to view data about hotels, pubs and theatres and create a personal annotated city map which they can then carry around the city model, or even print out and take with them on their actual visit.

The London Demonstrator serves as a comprehensive demo of the advanced collaborative virtual environment platform and services that the COVEN project has been developing. With this demonstrator we show both an interesting and useful CVE application and a mature CVE platform that can provide the level of service required for tomorrow's demanding applications.

4 Supporting the Application

During the construction of this application we have striven to build scalability techniques that are quite general and will apply equally well across different applications and different platforms. In particular we have not assumed anything about the layout of the model, nor have we tuned algorithms for a particular platform. The application as built runs at an acceptable speed on many platforms, from SGI O2s through to Onyx, though of course, the average amount of visual detail is greater for the latter. This has meant that we have had to avoid certain techniques such as static level of detail, and certain types of visibility pre-processing that can only be optimised for a particular platform.

4.1 Database Support

4.1.1 Tile Paging

Given the size of the base model, only a small portion can be stored in memory or rendered at one time so it must be stored in paged tiles. The tiling provides a natural scoping of world events since tile proximity can be used as a gross indication of mutual awareness. Also, the tiles themselves are static objects in the environment since we do not expect the buildings themselves to be changed, or at least not very frequently.

These types of consideration lead to the development of a new world structuring technique called *holders*. A holder is an unsynchronised portion of the scene graph that is not loaded through the world database mechanism but must be loaded explicitly by each process from a URL or file. This enables scaleable virtual worlds since events that are generated within the scene-graph below the holder node are not distributed to other participants. This does not mean that these portions of the scene are static, since, as detailed in [9], events can still be distributed at a higher level of the scene graph and local changes can be propagated down into the holder. Such high-level events can encapsulate complex behaviour and we have used them for this purpose in a number of situations.

For this demonstration, each map tile is placed under a separate holder. The holders are loaded when the user collides with a bounding sphere. Since this behaviour is scripted in DIVE/TCL more sophisticated look-ahead paging could be performed. Individual tiles are loaded incrementally in order to maintain a constant frame-rate by not locking out the rendering process. Holders are also used in within buildings where large areas of occlusion occur as a first, crude visibility partitioning (see Section 4.2).

4.1.2 Geometric Optimization

VRML1.0 is a verbose ASCII file format, and parsing and constructing an object hierarchy is thus a fairly slow process. We have tackled this by exploiting a new facility within DIVE for serialising the internal object store[4]. A binary DIVE file is approximately 30% of the size of the equivalent VRML1.0 file, plus it has the advantage of parsing straight into the native object format, so loading is approximately 10 times faster.

During the binary conversion process, we can perform other transformations on the scene objects. For example, VRML1.0 describes polygons with the IndexedFaceSet node. This node supports very general unstructured polygon sets, and it is usually necessary to optimise the resulting polygon set to get best performance within the renderer. Many OpenGL accelerators are optimised to support very quick rendering of long triangle strips [17]. A VRML browser would have to analyse the IndexedFaceSet

[4] Binary compilation is not compulsory, but since it does speed up loading considerably, we have created a build procedure to convert the whole London model. This takes approximately 90 minutes on a SGI High Impact workstation.

node to check whether the triangles could be described as strips, and then re-arrange its internal representation of those polygons. Such analysis takes considerable time to do reasonably, so we perform it at the binary conversion stage, and convert the polygon sets to triangle strip sets wherever possible. During this stage we also do some hierarchy re-arrangement in order to support culling techniques.

4.2 Rendering Support

The standard DIVE renderer follows a very simple scheme (see Figure 3). The hierarchy is traversed in depth first order using spherical bounding volume culling. When an entity containing a level of detail (LOD) is encountered a simple range decision is made and then the geometry is passed to OpenGL.

Figure 3: Standard DIVE Renderer.

With recent renderer and model optimisations we are already seeing significant (1.5-2.5 times) improvement in frame rate in the COVEN version of DIVE over previous publicly available versions (DIVE 3.2 and earlier). This is effected by better culling techniques, triangle-strip objects and on demand OpenGL list compilation. These are supported by the off-line methods for hierarchy re-arrangement and mesh and triangle strip optimisation (see Section 4.1.2).

However the London model itself is massive, and given that from any viewpoint many hundreds of thousands of polygons maybe seen, the standard renderer can take several seconds to render a frame[5]. The use of static LODs is inadequate in order to provide a constant frame-rate. LOD is usually described in terms of a set of possibly distance ranges from which a selection can be made at run-time. Although a selection can correspond to a null geometry, the ranges can not be created so that from an arbitrary viewpoint on an arbitrary machine a target rendering time is met. In general practice a minimum platform and minimum frame-rate would be decided, and the LOD ranged tuned to that platform. However in our case, with a faster machine we do not want faster frame-rates, we want more of the model to be rendered.

We have experimented with renderers that aim to give a constant frame-rate experience so that visual continuity can be maintained. In the simplest version, we optimise the depth of a software far clipping plane so that a frame-rate target is met. This

[5] In fact it takes a SGI Onyx2 with Infinite Reality 2 graphics and sufficient memory around 8 seconds to draw the complete 160 km^2 model.

is done by simply computing the distance to objects during scene traversal. When the rendering target is over-run the clipping plane is rapidly brought nearer to bring the frame-rate under control. When the target is met 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. In a more advanced version, we incrementally render the scene if the viewer is not moving, as shown in Figure 4.

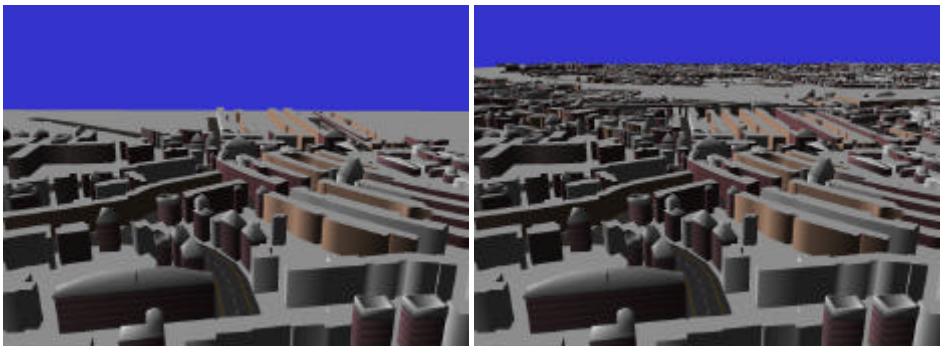

Figure 4: Incremental rendering of the model (after 100ms and 1s respectively).

To assist with the incremental renderer, the DIVE database is structured in a k-D tree [5] so that a very rapid front to back ordering can be derived. In order to create a consistent picture it is necessary to render in three stages per frame: first the background is progressively painted in with all static objects and the scene image is stored. Secondly depth buffer writing is disabled and all moving objects are drawn and the resulting image is shown to the user. Finally the image without the moving objects is restored to the off-screen buffer in preparation for another frame. In this manner it is possible for moving objects including other avatars to be correctly depth buffered against the rest of the scene.

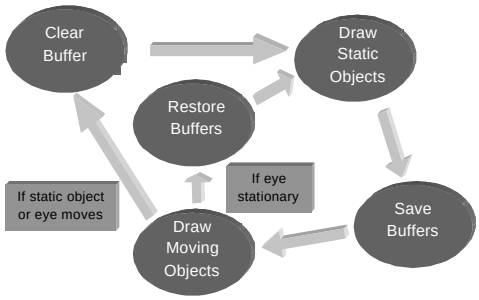

Figure 5: Approach for Incremental Rendering.

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 as created is not suitable for computed visibility solutions such as those developed by Airey [3] 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 view capability within DIVE [18]. Although the resulting visibility relationships are much more conservative than an analytic solution, it is quick and simple for the scene author to describe, and is more generally applicable to unstructured geometric models.

5 Discussion

5.1 Scalability

The two primary extensions described in this paper, holders and constant-frame rate rendering have been essential in enabling the London Travel application. Previously it was not possible to run such a large and complex application within DIVE and it is our belief that no other general CVE platform would support such an application and still retain the ability to support groups of participants.

We have successfully used the application in a series of network trials involving UCL, SICS, University of Nottingham and Lancaster University. Results have been encouraging. Whereas in previous trials we had to radically cut down the amount of geometry in the scene in order to allow it to be shared, we can now run with the city model, the full interior model and the full set of simulations and tools[6]. Furthermore, data is not distributed before the trial and is loaded as required through the standard DIVE database, or through holders as required.

In trials with eight people, two at each site, the application has been usable on SGI O2s with 128 MB of memory despite the high volume of position event traffic and accompanying audio. The frame rate has usually been tied to between 8-10 frames a second.

[6] Note that since not all COVEN partners have licensed the full city model data, not all trial participants can see the complete city. We have hand created a local area model for public distribution.

5.2 Extensions

5.2.1 Level of Detail

Currently we only use LOD as an indication of acceptable visible error and not as a technique to balance the frame rate. We have been experimenting on two aspects of LOD and plan to incorporate these into the base platform in the next few months.

The most common level of detail technique simply allows the renderer to choose between a fixed number of representations of an object with differing rendering costs. One major problem of such a technique is that the switch from one level to another might be noticeable. One way around this is to use continuous level of detail, where a model can be smoothly morphed between low and high levels of detail [13]. We have an implementation of such a scheme that allows for compact storage.

The second aspect that we have been looking at is level of detail control. In a situation where there is a constant frame rate, the problem can be considered as distributing a rendering "budget" over a set of objects with differing importances to the final visible image [15,14]. Such a technique will remove the need of a moveable far clipping plane that culls all objects beyond a certain distance.

5.2.2 Visibility

Given the nature of the models we are encountering, an obvious extension would be to support one or more visibility culling techniques. However there are two main problems with these for a general system: variation in model structure and pre-processing requirements.

The two most relevant visibility approaches are cells, portals and pre-computed visibility graphs for interior scenes, and occluder selection for exterior scenes (see Section 2). Although we have incorporated simple versions of such visibility techniques in a explicit manner (see Section 4.2) neither of these general approaches is uniformly applicable in our models since we have both interior and exterior space, and, perhaps more importantly, we have boundaries between the two types of space. Although the use of one or the other algorithm could easily be scoped to regions within the model, it would be difficult to support visibility from, say, a building interior out through a window to an exterior street scene. Pre-processing requirements for such a model would be quite onerous. In such a large model, storing a visibility graph would be problematical given its size. It is also not obvious how one would store such a graph in a tiled manner so that it could be paged in and out as the relevant part of model was paged in and out. Pre-selection of occluders is a promising technique since it is locally determined.

6 Conclusions

We have been successful in supporting a large-scale application on a platform that supports groups of users. The London Travel demonstrator is a complex application, both in terms of geometry and functionality, and this complexity constrains the use of established techniques for visibility pre-processing and model management. What we have done is migrate the control of model management and visibility into the scene scripting language, which has given us a degree of flexibility and generality that enables scalable applications.

The challenge we are now facing is incorporating other extensions such as occlusion culling for the streets, visibility graphs for building interiors, image based rendering and continuous level of detail control into a single renderer in a general way so that they can be utilised by other applications.

7 Acknowledgments

This work is sponsored by the EU ACTS Programme COVEN (ACTS N. AC040) and we would like to thank the members of that project that have contributed to this demonstrator.

8 References

1. Chan R., Jepson W., Friedman S. Urban Simulation: An Innovative Tool for Interactive Planning and Consensus Building. Proceedings of the 1998 American Planning Association National Conference, Boston, MA, April 1998, pp. 43-50. See also *http://www.ust.ucla.edu/ustweb/ust.html*
2. Aliaga D. Cohen J., Wilson A., *et al.* MMR: An Interactive Massive Model Rendering System Using Geometric and Image-Based Acceleration. ACM Symposium on Interactive 3D Graphics, 1999, pp. 199-206.
3. Airey J., Rohlf J., Brooks F. Towards Image Realism with Interactive Update Rates in Complex Building Environments. ACM Symposium on Interactive 3D Graphics, 1990, pp. 41-50.
4. Bell G., Parisi A., Pesce M. The Virtual Reality Modeling Language Version 1.0, November 1995. Available at *http://www.web3d.org/VRML1.0/vrml10c.html*
5. Bentley J.L., Multidimensional Binary Search Trees Used for Associative Searching. Communications of the ACM, 18(9), September 1975.
6. Bourdakis V. The Future of VRML on Large Urban Models, UKVRSIG97, Brunel University, 1st November, 1997. Electronic version available at *http://www.brunel.ac.uk/depts/mes/Research/Groups/vvr/vrsig97/proc.htm*
7. Coorg S., Teller S. Real-Time Occlusion Culling for Models with Large Occluders. 1997 Symposium on Interactive 3D Graphics, ACM SIGGRAPH, 1997, pp. 83-90.

8. Frécon E., Avatare-Nöu A. Building Distributed Virtual Environments to Support Collaborative Work" ACM Symposium on Virtual Reality Software and Technology (VRST'98), Taiwan, 1998
9. Frécon E., Smith G. Semantic Behaviours in Collaborative Virtual Environments, Proceedings of Eurographics Workshop on Virtual Environments (EGVE'99), Vienna, Austria, 31st May-1st June 1999, Springer Computer Science, pp. 95-104.
10. Frécon E., Stenius M., DIVE: A Scaleable network architecture for distributed virtual environments, Distributed Systems Engineering Journal (special issue on Distributed Virtual Environments), Vol. 5, No. 3, Sept. 1998, pp. 91-100. See also *http://www.sics.se/dive*
11. Funkhouser, T.A., Sequin, C.H., Teller, S.J. Management of Large Amounts of Data in Interactive Building Walkthroughs, Proceedings of 1992 Symposium on Interactive 3D Graphics, Computer Graphics, pp. 11-20, ACM Press.
12. Funkhouser T. Database Management for Interactive Display of Large Architectural Models, Graphics Interface '96, Canadian Human-Computer Communications Society.
13. Hoppe H. Progressive Meshes. Proceedings of SIGGRAPH 96. ACM SIGGRAPH pp. 99-108
14. Howell J., Chrysanthou C., Steed A., Slater M. A Market Model for Level of Detail Control. Accepted for Virtual Reality Software and Technology (VRST), 1999.
15. Mason A., Blake E. Hierarchical Level of Detail in Computer Animation. In Computer Graphics Forum 16(3), Proceedings of Eurographics '97, pp. 191-199.
16. Normand, V. *et al.* The COVEN Project: Exploring Applicative, Technical and Usage Dimensions of Collaborative Virtual Environment, Presence: Teleoperators and Virtual Environments, 8(2), 1999, MIT Press.
17. Silicon Graphics. OpenGL on Silicon Graphics Systems, Insight Guide
18. Smith G. and Mariani J. Using Subjective Views to Enhance 3D Applications. ACM Symposium on Virtual Reality Software and Technology (VRST '97). ACM Press. Swiss federal Institute of Technology (EPFL), Lausanne, Switzerland, September 1997, pp. 139-146.
19. West J. MSc Thesis, Department of Computer Science, University College London.