Workshop on Distributed System Aspects of Sharing a Virtual Reality, Proceedings of the Sixth Workshops on Enabling-Technologies: Infrastructure for Collaborative Enterprises, Cambridge, MA, June 18-20, 1997, pp. 187-192

# Communication Visibility in Shared Virtual Worlds

Michael Capps Seth Teller
<capps, seth>@graphics.lcs.mit.edu
Computer Graphics Group
MIT Laboratory for Computer Science

## Abstract

*Though the service of shared virtual worlds is an active area of research, little effort has been made to optimize such systems for urban world spaces. Tracking the motion, action, and communication of thousands of users in a city requires a application of visibility for spatial and logical subdivision of updates. We propose herein the City-Level Optimizations for Virtual Environments (CLOVES) substrate for the MIT City Scan (automated urban geometry acquisition) project. CLOVES includes a generalized spatial subdivision optimized for visibility; a Graduated Visibility Set (GVS) generator; associated interest management techniques; and model service to distributed heterogeneous clients.*

Keywords: Virtual Reality, Virtual Worlds, Collaborative Virtual Environments, Interest Management, Data Subscription, Visibility, GVS

## 1. Introduction

The Computer Graphics group at MIT-LCS is engaged in the City Scanning project, a system designed to automatically generate layout, landscape, and geometric building structure using pictures of a city taken by a non-autonomous roving vehicle.

With its vast (virtual) extent, a city model yields additional complexity by allowing larger user populations. Hundreds of human-sized avatars can interact comfortably in a single large building; we envision models with city blocks lined with such buildings and worlds with thousands of interacting heterogeneous clients.

We hope to provide high-fidelity telepresence in virtual urban environments for both the single user and large synchronous communities. For this, we provide CLOVES (City-Level Optimizations for Virtual Environments), which serves as the data, navigation, and communication infrastructure for the MIT City project. CLOVES includes a central database for storing the city information, as well as modules which optimize the database for model traversal, service to remote clients, and messaging.

There are five primary research issues that we will address while implementing this system:

- the annotatable image/geometry database uses a spatial subdivision in which each node can have an arbitrarily-typed subtree
- a specification protocol for heterogeneous clients, to inform the server of client capability and type (e.g., available bandwidth, class of motion, etc.)
- visibility processors that can make incremental refinements to the spatial subdivision of the model, as well as generate specialized information like our novel Graduated Visibility Set (GVS)
- model server algorithms that use client specifications to provide proper data formats, avoid starvation, and incorporate motion prediction for precaching
- interest management techniques including culling based on functional groups, message frequency, and level of visibility

We discuss these ongoing research interests in turn in sections 2 through 6. For each, we give a broader overview, a description of related work in the field, and a discussion of our progress and directions. The focus of this paper is on the distributed aspects of CLOVES, especially as applicable to message transmission and culling in the later sections. We follow these with a short discussion of the implementation of the CLOVES system, and conclude after a look at our future plans in this area.

## 2. Content Creation and Storage

The MIT City Scanning project produces CAD models representing urban geometry by processing photographs. As geometric information is reconstructed from image data, it is checked into the model database, along with a description of the source of the data and any annotations. Annotations are stored in a variety of formats, and often far outstrip the geometric data in storage size. They are primarily used to store the information used to generate the reconstruction, such as camera descriptors and digital images. In this way, it is possible to incrementally refine the geometric information in the database by examining

how it was originally generated and using that information to determine if additional refinement would be useful.

### 2.1 Previous work

Most graphics engines use spatial subdivision to facilitate various functions such as frustum culling, memory management, visibility, and the like. We are unaware of a generalized spatial subdivision which combines the functionality of different tree types.

Object-oriented databases and image processing are well-explored areas of research; we claim only the unique application of OO-annotation as a storage medium for incremental image processing.

### 2.2 Research issues

The ability to keep an annotation associated with a group of polygons current throughout the refinement process is critical to the incremental processing used in CLOVES. Large models can require very complex preprocessing functions, such as model segmentation and visibility determination, and a driving requirement of our system is that models be available without requiring hours or days of initialization.

To speed view-frustum culling, visibility determination, and other functions, we use a hierarchical spatial subdivision to store the geometric data. However, rather than optimizing storage and traversal by simply choosing different split-planes for BSP tree, CLOVES allows any node to be split according to how it can best be stored. Therefore, the first division might be a winged-edge data structure to describe road and city-block layout, as in [SD97]; flat, open areas might use quad-trees; and buildings could use octrees, BSPs, or both. These nodes all share a C++ superclass of a generic node, and specific nodes store the type of their children and have access to virtual methods for their traversal.

The database will be able to serve the geometry in multiple formats for heterogeneous clients; data may be pre-stored in some formats to reduce computation time.

## 3. Heterogeneous Client Specification

A city's inhabitants are diverse, and this should be supported in a virtual urban environment as well. However, most large multi-user virtual environment systems require the use of a proprietary client, available on limited architectures. We wish to make very few assumptions about clients, and expect disparate processing power, display technology, input devices, local memory, portability, and bandwidth. Some VRML researchers have attempted to develop protocols for sharing that information, but none have been accepted by the community. The distributed model management we propose would require us to develop our own such specification. Some systems have included simple descriptions of participating federates; instead, we hope to develop a formal semantic that captures the meaning of these capabilities. Such a formalism would allow greater portability of this specification to other systems, which is one of many important steps on the road to interoperability in shared virtual reality systems.

## 4. Visibility Determination

The city model promises to be larger than can be easily processed by conventional graphics hardware; we reduce the visible set through hierarchical subdivision and occlusion culling. To speed model traversal, we use hierarchical visibility information; to allow for disparate rendering ability in our heterogeneous clients, we store graduated visibility information rather than using simple binary occlusion.

### 4.1 Previous work

Visibility information is now a standard step in rendering complex models at interactive rates. Researchers have exploited the axially-occluded nature of building interiors, for instance, to generate visibility to aid in radiosity computations [TS91]. Berkeley's Soda Hall was split into cells, which generally corresponded to rooms and hallways, and portal-stabbing techniques were used to determine a superset of visible cells from each cell. Researchers at UNC used similar algorithms to accelerate architectural walkthroughs [AR90, LG95].

This visibility preprocessing step is seen as a speed-up technique for other high-fidelity rendering engines. The commercial product Quake by IdSoftware uses binary space partitions to store worlds, and a visibility precomputation is made to speed rendering by complementing view-frustum culling [MA96]. Coorg and Teller of MIT developed a heuristic for determining the largest visual occluders in an environment and culling geometry accordingly [CT96a]. This algorithm dynamically selects these objects; precomputation and storage of the computed occluders is unnecessary and expensive.

Sillion *et al.* store precomputed visibility in urban environments, with the caveat that all viewpoints will be in a narrowly constrained 2-D area, head-height on the streets of the city [SD97]. This information is used to generate textural *impostors* for complex geometric parts of the city model. The city is stored as a winged-edge graph, with edges in the graph representing streets, and faces representing city blocks.

### 4.2 Levels of visibility

With CLOVES we introduce the concept of *levels* of visibility. A binary determination of occlusion ignores many of the issues in rendering and in distributed virtual environments. For example, in Cambridge there are avenues of unobstructed vision which extend many miles in the seaward direction. In that case, binary visibility would cause a boat in that area to be rendered completely, though in reality there is no perception of the object. More importantly to this project, that boat could be a virtual cruise-ship, so a client on the beach would receive updates on hundreds of clients with which there is no chance of interaction.

We propose to enable generation of a Graduated Visibility Set (GVS) in our visibility processor. The GVS consists of n sets of cells; the first cell set is considered completely visible, and the $n^{th}$ set is $1/n^{th}$ as visible the first set. Each cell set contains a list of its members and a pointer to the list in the next more-visible cell set, so each cell set is considered a complete list for that level of visibility. Visibility is decreased according to a set of user-defined rules; we use similar rules to those used in choosing appropriate Level of Detail (LOD) such as in [TF96]. These include using projected screen space of a cell, distance to it, intervening fog, and the like to reduce the visibility detail in a cell.

For an example of a GVS, consider a world in which all points are divided into cells, and the GVS has 4 levels. From a particular viewpoint, cells at extreme distance would be in the $3^{rd}$ set. Most nearby cells would be in the $1^{st}$ set, though cells occluded by world objects are in the $4^{th}$ (invisible) set. Areas with natural camouflage, such as man-height objects, might be slightly reduced in visibility ($2^{nd}$ set), as could areas slightly obscured by fog or at a medium distance. Underwater, partially-occluded, or farther areas would be in the last ($3^{rd}$) visible set. The $4^{th}$ set is not stored with a cell but rather is computed as those cells not in the visible sets.

The GVS is an annotation just like any other in the database, so it is possible to generate multiple GVSs for different needs. We envision generating sets for media other than vision; for example, an audio GVS might attenuate aural "visibility" by distance or occluder properties. Complex aural GVSs might take into account that glass can be sound-proof and walls can be paper-thin and permit full audio transmission.

Of course the GVS can be used for geometric LOD decisions, and we plan to include that functionality in CLOVES. The primary use for the GVS, though, is to give the CLOVES server additional information to aid in communications interest management.

## 5. Model Service

The initial data for the MIT City Scanning project, a geometric model of some portion of Cambridge, Massachusetts, will likely many gigabytes of storage space. We expect very few participants will have the ability to store such a model locally, much less process it in a timely fashion. Regardless of storage needs, the transfer time for such a model will be prohibitive when joining such a shared urban world.

The problem of navigating a dataset larger than memory is not new, and is being well-explored by the CAD/CAM community. In fact, the Berkeley Walkthrough [TF96] made use of extensive memory management techniques to allow rendering of a very complex architectural model. Precomputed visibility information allowed subdivision of the world into manageable pieces, which we accomplish with more granularity with the GVS. The model server in CLOVES uses the client specification to decide the size of model pieces to send, and at what level of detail.

The Berkeley system and others also use motion prediction to determine areas of the world that may be needed in the near future. Those areas are pre-cached when possible to prevent memory-swap discontinuity when the user moves into a new portion of the model. CLOVES includes class of motion in the client specification, and it uses these classes (e.g walking, bus, helicopter, and a generalized class) to predict possible paths in the virtual world. That, in conjunction with a knowledge of network latency, bandwidth, and available local memory for that client, comprises a complex heuristic for how much of the model to serve to whom and when.

## 6. Interest Management

Simple point-to-point and broadcast protocols exhibit traffic growth polynomially with the number of users in a virtual environment; these protocols are not appropriate for client-to-client communications in our urban setting. CLOVES provides a hierarchical client-server network architecture and protocols designed to scale well for the thousands of users expected in city environments.

Use of a server hierarchy and inter-client visibility can greatly reduce the number of messages required to keep concurrency for all users. When the system updates an avatar's position only for those users who can see it, many unnecessary messages can be culled in an environment with significant occlusion. In such a large system, users may want more, or even less, selective culling options. Users may want to see only members of a logical group, such as a family or team, or a more complex Boolean operation on many logical sets. In fact, a user may always

wish to see a person, regardless of intervening occluders, granting a sort of x-ray vision. This brings to mind that visibility need not refer only to visual occlusion by an object; for instance, aural "visibility" is affected by glass walls, and thick fog can affect visibility as much as an opaque wall.

## 6.1 Previous work

### 6.1.1 Spatial culling

MUDs were the first to use spatial culling techniques; the room metaphor restricts users to interaction with co-located users. There are some exceptions, but they must be explicitly used, such as by *paging* or *telling* a particular user or *shouting* to all users on the system. NPSNET [MZ95] is a notable example of a system which culls communications by separating the virtual world into grid cells; clients receive updates only from those cells in which they have an interest. Low-bandwidth clients and those that should only have a small field of awareness (like foot- soldiers) subscribe only to the grid they occupy; higher-bandwidth clients with greater awareness (for example, radar installations) would subscribe to a greater number of grids. Almost all multi-user VE systems that have interest managers do some spatial filtering; WAVES [RK93] and GreenSpace [AP95] are further examples.

The COMIC model of interaction, which is used in the DIVE [CH93] and MASSIVE [GB95] systems, involves concepts of *focus* and *nimbus* to quantify a client's region of interaction. Focus expresses the area of space in which a client is interested, for a certain medium, and nimbus is the area in which that client can be detected. A client's *aura* is a bounding region containing both the focus and nimbus for purposes of determining interaction; if and only if two clients' auras overlap is there a possibility of interaction between the two.

A natural extension of spatial culling is to filter communications based on visual occlusion. For example, while two clients may be within a few virtual feet of each other, if they are on other sides of an opaque wall there is no need for them to exchange positional updates thirty times a second. Funkhouser used the cell-to-cell visibility generated in the Berkeley Walkthrough for the multi-user RING system; a central server kept track of the current cell for each client, and then would send a client's updates only to other users visible from its cell [TF95].

### 6.1.2 Functional filtering

Functional filtering is the reduction of communications based on functional groupings of clients in a virtual world. This can be performed in both an extrinsic and intrinsic manner, though the focus of prior work in this area has been on filtering communications based on source. Papers on WAVES and NPSNET expressed the importance of enhancing message culling using functional groups. NPSNET researchers suggested that some functional groups, such as tanks or foot-soldiers, could have their messages included or culled as a group—in addition to the standard spatial grid-based filtering of that system [MZ95]. Neither system implemented that functionality.

### 6.1.3 Network topology for interest management

There are three fundamental network topologies used in multi-user virtual environment systems: broadcast, multicast, and centralized. Most distributed VEs employ a form of broadcast [SG93, CH93] for communicating updates, but this means that with each additional client the message-processing load per client grows linearly and the network load grows polynomially. Both of these concerns can prevent a VE architecture from being properly scaleable. Multicasting communications can greatly reduce the load on both the sender and the network, as each sender publishes only a single update instead of one for each other client. Broll's VRML-based VE system [BR97] uses a single multicast group for all members of the virtual world. Since multicast is not a reliable protocol, Broll also uses a message server as a backup to store update messages. NPSNET uses a single multicast group for each grid cell in the simulation [MB95]. Multicast has some drawbacks: it is unreliable (though a reliable version is in development), it is not yet widely implemented, and the effort in modifying groups makes supporting highly dynamic populations difficult.

The third network topology frequently seen in CVE systems is the centralized server approach. A single server can be used for all clients, such as in the Virtual Space Teleconferencing System (VISTEL) [OJ93], to hold all model data and route all messages; this approach of course does not scale well. The STOW-E architecture [ST95] utilizes a standalone interest manager on each LAN. That server gathers the interest expressions of its clients, unions them, and broadcasts them to all other servers. Using that information, it is able to direct its updates only to LANs containing interested parties.

Id Software's popular commercial software Quake uses a central server, to which each of up to 16 clients makes a direct unreliable (UDP/IP) connection. Each client keeps a local version of the virtual environment, and the server relays position updates for only those clients and dynamic objects which are in a user's Potentially Visible Set (PVS). [DK97].

One successful version of a centralized server topology was Thomas Funkhouser's RING system; it reported a

messaging reduction of 98% [TF95] by using the cell-to-cell visibility algorithm of the Berkeley Walkthrough to perform culling. Each client stores a local version of a static virtual environment and communicates its position and motion to the server once every n display updates. The server translates the position into a cell location, and uses the cell's PVS to determine which actors a client can possibly detect. Funkhouser addresses the scalability issue by allowing multiple servers to serve a single intermingled population of clients.

### 6.2 Graduated levels of visibility

While a powerful graphics engine with high bandwidth might always render all visible cells, bandwidth or processing constraints might cause reduced fidelity for other clients with that load. Our message servers can use the CLOVES client specifications, combined with the GVS, to reduce the amount and frequency of updates forwarded to certain clients. As well, if a server experiences a messaging overload, it can use the GVS to degrade outgoing traffic smoothly by lowering the highest visible set that it serves. This reduction need not indicate total quenching of messages from a certain source; the GVS can be used to reduce the frequency of updates for less "visible" areas, lending a fuller spectrum of choices.

### 6.3 Functional culling

In addition to standard walkthrough functionality, the CLOVES client provides the user with the ability to specifiy interest in functional groups. A separate application can be launched that lets the user join and leave functional groups (see screenshot), specify viewing interest in groups, and create new groups. Group creation and modification is accomplished either by selecting users from the population or through more sophisticated Boolean expressions consisting of client specifications, user information, and existing groups. These membership descriptions are included in the client specifications mentioned previously.

The viewing interest value is a tiered value similar to that used in the LVS. The highest values allow the user to receive updates on group members regardless of their position or occlusion status. Our client is equipped with an additional ability to display such groups at all times; certain objects, including avatars, can be placed in an *x-ray visibility* group such that they are always shown. This function would not be active in competitive scenarios, but would for example allow a parent to find a child regardless of intervening objects or avatars.

We envision these functional groups to be extremely useful in well-populated virtual environments. Most users will likely only wish to interact with a small subset of the urban population at any one time, so functional culling should contribute significantly to scalability. Even a simple binary grouping of two teams in a game would generally reduce communications by 50%. In a competitive scenario, a user could reduce the scope of awareness to the opposing team, but then easily switch to fuller representations when coordination with teammates was desired.

## 7. Implementation

The CLOVES system consists of four main parts: the database, processors, the client, and the servers. The image processors of the City project interface with the database through a C++ library, storing geometry and annotations. The database allows for incremental refinement by these processors, as well as incremental adjustments to the general-tree spatial subdivision and to multiple visibility sets (PVS, GVS, etc.). The visibility processors use a complex set of heuristic to optimize node depth, balance, and the like for rendering and message culling.

We currently use a modified version of the Berkeley-Princeton Radiosity Walkthrough, a later version of the Berkeley Walkthrough, as our primary client for visualizing the city database [FT96]. The client uses the SGI IrisGL libraries, and is currently being ported to OpenGL. We plan to support SGI and WindowsNT as our main platforms, but look forward to greater diversity as the main portions of CLOVES are completed.


*Figure 1 : City client with group selection*

We chose the central server topology as the most successful trade-off of latency and scalability vs. centralized message filtering and visibility processing. We have implemented a messaging substrate and server software very similar to that of the RING system; all clients connect directly to a central server, or servers. The server sends a client only pertinent update messages; depending on a simple heuristic or administrator commands, the server culls first by visibility, and then by

groupings, or vice-versa. Multiple servers can work in tandem either by an arranged spatial division of the world or by simply load-sharing; each organization offers advantages, which have been examined in [TF96a].

The model service topology is a single centralized server, which we will use as we investigate the complex heuristics for minimizing starvation and model delay across an entire population. We expect in the future to further distribute the model service task to increase availability, reduce network latency, and of course to support larger populations.

## 8. Conclusion and Future Work

The CLOVES architecture, and the overall City Project, are in an ongoing design and implementation cycle. As of this writing, we have completed the hierarchical server construction and the city client with group subscription. In the short term we will complete general-tree, GVS generator, and a specialized Quake server as a proof-of-concept for our culling techniques. The remainder of the project has an expected completion of 2 years.

Our plans for future additions and refinements include GVS processors for other media, and for use in rendering LOD decisions; more complex motion prediction supporting classes of motion; and quantitative utility testing of the general-tree spatial subdivision.

This paper presented the CLOVES architecture and its role in the MIT City project, both currently works in progress. The network architecture chosen enables scalability for potentially populous urban environments, and the interest management system implements and improves upon ideas from military, academic, and commercial systems presently in use. It is our hope that the CLOVES system will prove effective for our problem domain, and that many of the techniques explored can be leveraged effectively into more traditional collaborative virtual environments.

## 9. Works cited


[AP95] Pulkka, Aaron. "Spatial Culling of Interpersonal Communication within Large-Scale Multi-User Virtual Environments." MS Thesis, University of Washington, 1995.

[AR90] Airey, J., J. Rohlf, and F. Brooks, Jr. "Towards image realism with interactive update rates in complex virtual building environments." *Symposium on Interactive 3D Graphics*, 24(2):41-50, March 1990.

[BR97] Broll, Wolfgang. "Distributed Virtual Reality for Everyone--a Framework for Networked VR on the Internet." *VRAIS '97*, IEEE, Albuquerque, NM, March 1997.

[CH93] Carlsson, C., and Hagsand, O. "DIVE-A Platform for Multi-User Virtual Environments. *Computer & Graphics*, Vol. 17, No. 6 (1993), pp. 663-9.

[CT96a] Coorg, S. and S. Teller. "Temporally Coherent Conservative Visibility." *Proceedings of the 12th ACM Symp. on Computational Geometry*, Philedelphia, PA, May 1996.

[DK97] Kirsch, Dave, ThreeWave Software, responsible for QuakeWorld server. Personal Communication, June 1997.

[FT96] Funkhouser, T., S. Teller, *et al.* "The UC Berkeley System for Interactive Visualization of Large Architectural Models." *Presence*, 5 (1), January 1996.

[GB95] Greenhalgh, C. and S. Benford. "MASSIVE: a Collaborative Virtual Environment for Teleconferencing." *ACM TOCHI*, Vol. 2, No. 3, Sept. 1995.

[GS96] Smith, Gareth. "Cooperative Virtual Environments: lessons from 2D multi user interfaces." *Proceedings of* CSCW '96, November 16-20, Boston, MA, pp. 390-398.

[LG95] Luebke, D. and C. Georges. "Portals and mirrors: Simple, fast evaluation of potentially visible sets." *Symp. on Interactive 3D Graphics*, pp. 105-6, April 1995.

[MA96] Abrash, Michael. "The Quake Graphics Engine." Compuer Game Developers Conference, April 2, 1996.

[MB95] Macedonia, M., D. Brutzman, M. Zyda, D. Pratt, P. Barham, J. Falby, and J. Locke. "NPSNET: A Multi-player 3D Virtual Environment over the Internet." *Symposium on Interactive 3-D Graphics*, (Monterey, CA), 1995.

[MZ95] Macedonia, M., M. Zyda, D. Pratt, D. Brutzman, and P. Barham. "Exploiting Reality with Multicast Groups: A Network Architecture for Large-scale Virtual Environments." *VRAIS 1995*, IEEE, pp. 2-10.

[OJ93] Ohya, Jun, Kitamura, Yasuichi, Takemura, Haruo, et. Al. "Real-time Reproduction of 3D Human Images in Virtual Space Teleconferencing," *VRAIS '93*, pp. 408-414.

[RK93] Kazman, R. "Making WAVES: On the Design of Architectures for Low-end Distributed Virtual Environments." *VRAIS '93*, pp. 443-449.

[SD97] Sillion, F., G. Drettakis, and B. Bodelet. "Efficient Impostor Manipulation for Real-Time Visualization of Urban Scenery." *Eurographics '97*, Volume 16, Number 3

[SG93] Shaw, C. and M. Green. "The MR Toolkit Peers Package and Experiment." *VRAIS 1993*, IEEE, pp. 463-9.

[SO97] Sandin, D., G. Olson, and M. Macedonia. "Distributed, Interactive Collaboration--Where is it?" *Symp. on Interactive 3D Graphics*, Providence, RI, April 29, 1997.

[ST95] RDT&E Division and Advanced Telecommunications Inc. "Synthetic Theater of War-Europe Technical Analysis." Naval Command, Control and Ocean Surveillance Center.

[TF95] Funkhouser, T. "RING: A Client-Server System for Multi-User Virtual Environments." *1995 Symposium on Interactive 3-D Graphics*, (Monterey, CA), 1995, 85-92.

[TF96] Funkhouser, T. "Database Management for Interactive Display of Large Architectural Models." *Graphics Interface '96*, Toronto, Ontario, May 1996, pp. 1-8.

[TF96a] Funkhouser, T. "Network Topologies for Scaleable Multi-User Virtual Environments." *VRAIS '96*, IEEE.

[TS91] Teller, S. and C. Séquin. "Visibility Preprocessing for Interactive Walkthroughs." *SIGGRAPH '91*, 25(4):61-69.