A Network Architecture for Remote Rendering

Gerd Hesina and Dieter Schmalstieg
Vienna University of Technology, Austria
email: [gerd|dieter]@cg.tuwien.ac.at

Abstract

Internet-based virtual environments (VEs) let users explore multiple virtual worlds with many different geometric models which are downloaded rather than pre-distributed. To avoid long download times, we have developed a method that optimally utilizes network bandwidth by downloading only the exact portion of geometry that is necessary for rendering. The solution is based on progressive geometry data structures (smooth levels of detail) and selective download.

1. Introduction

Distributed VEs can bring together a large number of participants in a simulated three-dimensional space. Such a system may be described as a database - a collection of objects - shared over a network. The content and state of the database defines the scene that is presented visually to the human participants. In particular, the geometric descriptions (polygonal models) of the objects visible to the user must be available at the user's workstation for rendering. A simple approach to make these geometric models available is to replicate the information at every host. This approach is feasible for simulations with a fixed set of objects and an environment of restricted size (e. g., a specific terrain for a training simulation or a specific dungeon for a computer game).

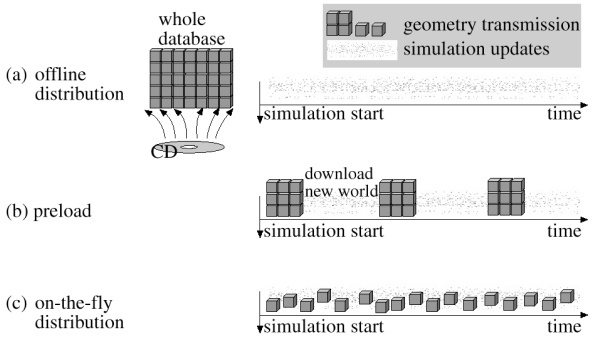

Figure 1: Transmission models for graphical data: off-line, preloading and on-the-fly.

However, with the adoption of the Internet as a medium for distributed VEs, full replication is no longer a viable option. Users connect to Internet servers and rapidly explore multiple VEs populated with a large number of geometric models which are downloaded rather than pre-distributed. Large data sets lead to extensive download times, thus necessitating a new paradigm for the transmission of geometric models.

In this paper, we present a method that optimally utilizes available network bandwidth for the runtime distribution of geometric models in distributed VEs. Our solution exploits selective download and progressive geometry data structures for incremental transmission.

2. Related work

While the efficient transmission of simulation messages has received a lot of interest, e. g., [1, 8, 9], relatively little interest has been paid to the transmission of geometric models in VEs.

The data for the visible objects in the scene must be available to the host to perform the rendering. Off-line distribution of the database (Figure 1a) is often used for simulation [4, 9] or CD-ROM based computer games [7, 10]. Some applications [7] also allow to download the environment before use (Figure 1b). Better use of spatial coherence is made by on-the-fly download of individual objects' geometric description such as NPSNET [9], or individual geometric levels of detail [11] (Figure 1c). In [5], a method for paging geometric data from disk for building interiors is presented. However, their method is designed for indoor environments and does not make use of progressive geometric models.

Recently developed progressive geometry models allow a fine grained transport of "streaming" geometry [2, 6, 12]. However, to our knowledge, a fully distributed system that incorporates this kind of progressive fine-grained geometry transmission has not been demonstrated so far.

3. Remote Rendering

Demand-driven geometry transmission [11] is a method for efficient transmission of geometric models in a distributed VE. A server stores the data for a region of the

VE, composed of objects that are arranged spatially. Some of the objects may be avatars that represent other participants. A client allows the user to display and navigate this VE database. A server stores the database of geometric objects and clients may download individual objects. By requesting and storing only those models that are currently visible to the user, the amount of data that must be transmitted and stored locally is significantly reduced. If the required data can delivered over the network "just in time" for the rendering process, both intractable setup times and elevated storage requirements are resolved without compromising visual appearance. The geometric models are effectively cached in the client's memory, and the cache's content is determined by the behavior of the users and the state of the simulation. We call this process *remote rendering*.

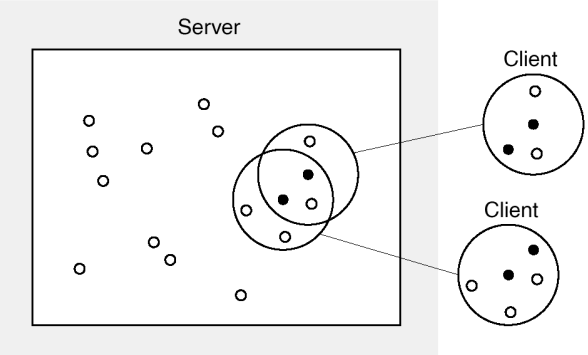

Figure 2: Clients maintain an area of interest (large circles) of the server's database. The geometric description of objects (white dots) and avatars (black dots) is transmitted on demand.

For remote rendering, it is sufficient if the client has the data for those objects that are contained in its area of interest (AOI), as shown in Figure 2. We use circular AOIs, a practical choice for outdoor environments. Rapid changes in the direction of gaze are possible for both immersive head-mounted display setups and desktop browsers, while the rate of translational movement is constrained. Therefore, the content of a circular AOI typically has a high degree of spatial coherence.

The geometric descriptions of the objects contained in the client's AOI must be available to the client. Any such description must be downloaded from the server by using the demand-driven geometry transmission protocol. This protocol between client and server also incorporates messages that let the client inform the server if the user has moved, and let the server inform the client if movement of other objects or avatars has occurred in the client's AOI (e. g., a new object has entered the AOI). For details regarding the protocol, refer to [11].

4. Rendering with Smooth Levels of Detail

While conventional methods use a small set of discrete levels of detail (LODs), our system uses a new class of polygonal simplification called *smooth levels of detail* [12]. A very large number of object approximations with increasing fidelity is encoded in a data stream for progressive refinement of the object from a very coarse approximation to the original high quality representation. This data structure allows incremental transmission, which is exploited by our application protocol (Figure 3). While switching between discrete LODs usually leads to disturbing "popping" behavior, smooth LODs allow a continuous choice of fidelity and avoid such artifacts.

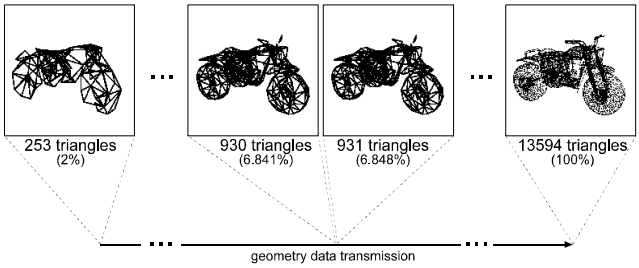

Figure 3: Smooth levels of detail allow progressive transmission of geometry data and display at any desired resolution

As the polygon budget of the client's image generator is limited, there is no point in wasting bandwidth by downloading geometry that will never be rendered; the geometry stream is simply cut off when the desired fidelity has been received. The fine-grained resolution of the progressive data structure makes all received data immediately useful: Whenever a small portion of the object geometry has been received, it is used to refine the visual representation. Thus notable temporal and visual discontinuities that would occur when dealing with a large geometric object as a whole are avoided.

5. Smooth level of detail selection strategy

The content of the client's AOI can be interpreted as a cache which has to be kept consistent with the server's database. The content of this cache is defined by the spatial distribution of the database, but the amount of data for each object in the AOI needs to be determined by a separate selection scheme.

For the selection of discrete LODs, a predictive method was presented by Funkhouser and Sequin [3]: The level of detail selection is formulated as a discrete optimization problem: Given a set of objects available at multiple levels of detail, select those that contribute most to the final image while keeping the cost for rendering these objects below a given threshold (*polygon budget*). The solution for

this optimization (knapsack) problem is approximated by a greedy algorithm.

Using smooth levels of detail, a continuous choice of detail is available for each object. Solving the continuous optimization problem that is equivalent to the aforementioned knapsack problem, we have derived a formula that is both simple (and hence fast) to evaluate and geometrically intuitive.

As observed by Funkhouser and Sequin, the relative importance of the object should be proportional to the size of the object in the final image: objects that appear larger should also be drawn with more detail (more polygons). Consequently, the number of polygons drawn for each object in a given frame is made proportional to the screen-size of the object:

$$p(x) = \frac{A(x)^{\alpha} \cdot P}{\sum_i A(i)^{\alpha}} \qquad (1)$$

where:

$p(x)$ …selected number of polygons for object $O(x)$
P …polygon budget
$A(i)$ …screen-size of $O(i)$
α …correction factor ($0 < \alpha \leq 1$)

With α set to 1, the screen size determines the number of polygons for each object. The screen size of polygons will be constant over all objects. However, this does not take into account that visual quality is not linearly dependent on the number of polygons: Adding more polygons yields diminishing returns in visual quality. Adding 100 polygons to an object displayed with 200 polygons is much more important that adding the same amount to an object displayed with 20000 polygons. Therefore α should be chosen < 1.

For $\alpha = 1/3$, it can be shown that equation (1) gives the exact solution to the continuous version of the problem formulated in [3]. Other criteria that contribute to the importance of the object such as screen position (focus) or user-defined importance are easily incorporated into the computation.

6. Prefetching strategy

To make geometric data available in a timely fashion, downloads must be initiated some time before the data is actually needed. Objects that are assumed to be rendered at a certain level of detail in the near future are partially *prefetched*. Should some geometry data not become available in time, the object can be displayed at a lower fidelity (*graceful degradation*) and be refined when the missing data becomes available (*progressive refinement*).

As the client must know in advance which objects to download, a larger AOI radius is considered for prefetching than for rendering. For each object in this larger AOI, the amount of geometry (number of polygons) that should be downloaded must be determined.

It does not make sense to request more data for download than the available network bandwidth allows. Therefore, data requests should be prioritized, where priority is defined by some measure of (anticipated) importance for future images (similar to the LOD selection described in section 5).

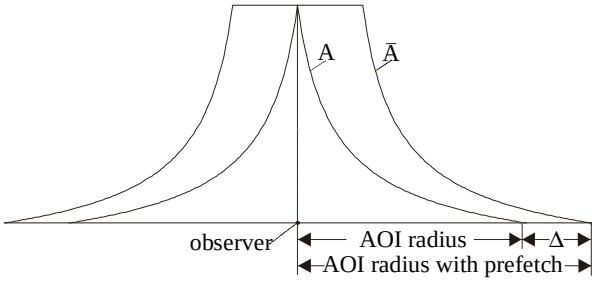

Figure 4: For objects of equal size, the importance is proportional to the screen-size A. By assuming a movement of Δ towards the object, the amount of prefetching is determined.

Prefetching assumes that the user has traveled towards the object by a constant distance. From the increased screen size the desired number of polygons is computed:

$$\bar{p}(x) = \frac{\bar{A}(x)^{k} \cdot P}{\sum_i A(i)^{k}} \qquad (2)$$

where:

$\bar{p}(x)$ …desired number of polygons for object $O(x)$
$\bar{A}(x)$ …increased screen-size of object $O(x)$

The difference between desired and available number of polygons for each object gives the amount of geometry that should be downloaded. This amount is weighted by the available bandwidth (polygon budget for download).

$$DL(x) = \frac{\left(\bar{p}(x) - avail(x)\right) \cdot \bar{P}}{\sum_i \left(\bar{p}(i) - avail(i)\right)} \qquad (3)$$

where:

$DL(x)$ …number of polygons to be downloaded for $O(x)$
$avail(x)$ …number of locally available polygons for $O(x)$
$\bar{P}$ …polygon budget for download

To avoid excessive overhead, downloads must have a minimum size or they will not be considered.

7. Implementation and evaluation

We were interested in assessing the performance of remote rendering. For our tests we used an entry-level workstation (SGI O2) as the client. Our selection algorithm as detailed in section 5 was always capable of staying within the polygon budget, so the O2 was capable of delivering a

steady 15 frames per second for a polygon budget of 10000 triangles. Network capacity was intentionally limited to 64kbps, which characterizes consumer level network bandwidth.

Experiment procedure. The experiment consisted of a client connecting to the server and moving through the VE while the content of its AOI was monitored. The path of the client through the VE was prerecorded and played back for the experiments to be able to recreate exactly the same user behavior. For prefetching, the AOI for download was 10 percent larger than the AOI for rendering.

Influence of spatial coherence. This experiment was designed to examine the influence of spatial coherence on the caching behavior at the client. To evaluate the quality of the cache, we defined a *quality* metric as the number of polygons that *could* be displayed (were available) over the number of polygons that *should* be displayed (should be available), both according to the choice of the selection algorithm presented in section 5. With a perfect caching strategy, this rate should be 1. Figure 5 shows the quality over time for increasing velocities (% of AOI change/sec). After the initial cache fill, the quality remains close to 1 for small velocities (solid line), while larger velocities (exceeding the bandwidth) lead to degraded quality (dashed line).

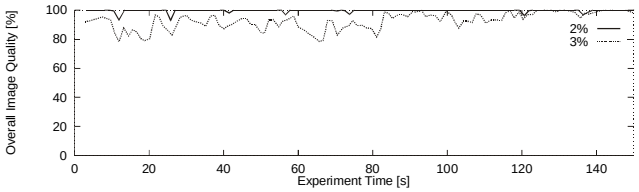

Figure 5: Influence of spatial coherence on the availability of geometric data

Influence of selective download. This experiment was intended to demonstrate the benefits of selective smooth LOD downloads. We compared our strategy to a simple scheme that fully downloads all objects in the AOI.

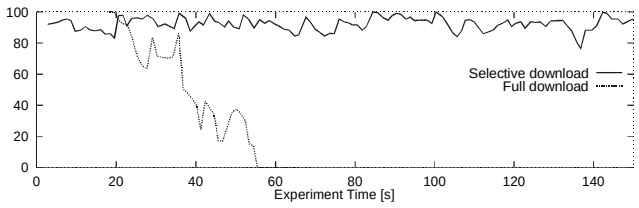

Figure 6: Influence of the selection scheme on the availability of geometric data

Figure 6 shows that for the same velocity, our selective smooth LOD downloading strategy produces a good image quality (solid line), while a naïve scheme (full download of all objects in AOI) by far exceeds the bandwidth and leads to unacceptable results (dashed line).

8. Conclusions

We have demonstrated a distributed virtual environment that combines selective demand-driven transmission of geometric objects with smooth levels of detail. This approach avoids long download times for Internet-based virtual worlds and improves the network utilization by adaptive use of both rendering and network capacity.

Acknowledgments. This project was sponsored by the Austrian Science Foundation (*FWF*) under contract number P11392-MAT. Many thanks to A. Fuhrmann, H. Hey, G. Schaufler, Zs. Szalavári, and M. Wimmer for their help.

References

1. Carlsson C., O. Hagsand. DIVE - A platform for multi-user virtual environments. Computers & Graphics, 17(6), pp. 663-669, 1993.
2. Deering M. Geometry Compression. Proceedings of SIGGRAPH'95, pp. 13-20, 1995.
3. Funkhouser T., C. Sequin. Adaptive Display Algorithm for Interactive Frame Rates During Visualisation of Complex Virtual Environments. Proceedings of SIGGRAPH'93, pp. 247-254, 1993.
4. Funkhouser T. Network Topologies for Scalable Multi-User Virtual Environments. Proceedings of VRAIS'96, Santa Clara CA, pp. 222-229, 1996.
5. Funkhouser T., C. Sequin, and S. Teller. Management of Large Amounts of Data in Interactive Building Walkthroughs. SIGGRAPH Symposium on Interactive 3D Graphics, pp. 11-20, 1992.
6. Hoppe H. Progressive meshes. Proceedings of SIGGRAPH '96, pp. 99-108, 1996.
7. Id Software. Quake. Computer game, information available at http://www.idsoftware.com/quake/, 1996.
8. Lea R., Y. Honda, K. Matsuda, S. Matsuda. Community Place: Architecture and Performance. Proceedings of ACM VRML'97, pp. 41-50, 1997.
9. Macedonia M., M. Zyda, D. Pratt, D. Brutzman, and P. Barham. Exploiting Reality with Multicast Groups: A Network Architecture for Large-scale Virtual Environments. Proceedings of VRAIS'95, 1995.
10. Origin. Ultima Online. Computer game, information available at http://www.owo.com/, 1997.
11. Schmalstieg D. and M. Gervautz. Demand-Driven Geometry Transmission for Distributed Virtual Environments. Computer Graphics Forum (Proc. EUROGRAPHICS'96), 15(3), pp. 421-433, 1996.
12. Schmalstieg D., G. Schaufler: Smooth Levels of Detail. Proceedings of IEEE VRAIS'97, pp. 12-19, Albuquerque, New Mexico, March 1-5, 1997.