

Data Management for Visualizing Large Virtual Environments

Tsai-Yen Li and Chih-Wei Chiang

Computer Science Department, National Chengchi University
64, Sec.2, Chih-Nan Road, Taipei, Taiwan 11623, ROC
e-mail: {li, s8409}@cs.nccu.edu.tw

Abstract

Effective data management is crucial for interactive multimedia applications due to the large amount of data that need to be processed on-line. When the size of a 3D virtual world is larger than the available physical memory of a display client, we need to proactively maintain useful data in the memory for immediate or future graphics rendering. Traditional page-based data management scheme becomes insufficient for applications with strong spatial and temporal coherence. In this paper we describe a 3D navigation system for visualizing large virtual environments. In the system we propose novel data management techniques based on discrete visibility pre-computation, hybrid cache model, and prioritized spatial prefetching. Our preliminary experimental results show that these techniques can improve response time and navigation smoothness.

1. Introduction

Interactive 3D graphics were only possible on expensive workstations equipped with special graphics hardware a few years ago. However, as the computing power of personal computers increase and 3D acceleration cards become more affordable, one can now experience Virtual Reality (VR) on a regular desktop PC. For example, with the development of Virtual Reality Modeling Language (VRML)[15], one can use a regular web browser on a PC to remotely navigate a virtual scene downloaded from the network. However, as the contents in cyberspace become richer, it is likely that a whole virtual scene does not fit into available physical memory any more. Therefore, how to manage a large virtual scene effectively while maintaining interactive visualization becomes an emerging issue that needs to be addressed.

Data management is crucial in many multimedia applications due to the amount of data that needs to be processed on-line. A 3D virtual world consists of 3D models of objects at their designated configurations. For large virtual worlds, these geometric models are typically organized in a database management system

on a remote server. When we navigate in these worlds, we can only see a small portion of the objects in the world under a first-person view model. Therefore, most virtual worlds on the network today partition a world into manageable subspaces connected through portal locations [3]. Only the objects in the current partition are loaded for graphics rendering. However, such a partitioning scheme is adequate only for spaces with appropriate natural partitions such as large occluders. In addition, the delay and discontinuity introduced in scene swapping is usually very disturbing. Therefore, it is more desirable for the system to incorporate a flexible object management scheme that can provide continuous and smooth navigation in a large-scale virtual environment.

In this paper we address the issue of managing spatial data effectively for interactive visualization of a large virtual environment. Specially, we first pre-compute the set of visible objects at discrete viewpoint configurations. When the viewpoint moves, we use this information to quickly identify the objects that need to be loaded at run-time. Secondly, the objects are loaded through a cache mechanism such that more relevant object models can be held in local memory for faster accesses. In addition, the system proactively predicts the possible future configurations of the viewpoint based on the current direction and velocity. Our preliminary experimental results show that these data management techniques improve the usability of the system and make the visualization of large virtual world more effective.

2. Related work

The problem of managing large amounts of data for interactive building walkthroughs has been studied since the early 90's[1][14][10]. Several systems have been built to demonstrate their proposed data management schemes. The topics that have been addressed include visibility preprocessing, levels of detail, data caching, and object prefetching.[4][6][8][10][14] However, most of these experimental systems require expensive multiple-processor hardware for parallel processing of data retrieval and graphics rendering. For example, in [2], a powerful multi-processor workstation is used to visualize a

large power plant.

As the computing power on PC's is increasing, the desire to populate these research results onto PC's also increase. Recently more and more virtual environment researches start to address the scalability problem on the PC platform.[4][11][12] Most of these systems focus on reducing communication complexity when the number of clients increases. They utilize inter-object spatial relations and more appropriate communication protocols (such as multicast) to reduce the amount of data for transmission[12]. Some other studies consider the data management problem in a large-scale world from a client's point of view[5][6][9]. This is also the focus of this paper.

Determining the set of visible objects at the current viewpoint in a large 3D virtual world is usually carried out in a preprocessing step due to the computation complexity involved. In [14], a k -D tree is used to partition the world into *cells* connected through *portals*. The visibility between these cells is then analyzed. Several previous works in computational geometry also focuses on representing and computing global visibility information in a virtual world. [7][8][13]

Caching is a common technique that retains more relevant data for faster accesses. Many replacement policies for caching have been proposed but no universally best policy is found. The well-known Least-Recently-Used (LRU) policy is a good one in general for data with temporal locality. However, it does not perform as well for spatial data[9]. In [5], an effective cache replacement policy, called Most Required Models (MRM), was introduced, based on multiple resolutions of an object that can be transmitted progressively.

A good prefetching strategy can help system performance on the smoothness of user navigation. Most previous systems prefetch only the next predicted viewpoint configuration.[5][14] However, the system may suffer from noticeable delays when the visible set of objects change dramatically as the user changes the viewpoint motion. On the other hand, although prefetching data for more than a single configuration may increase the hit ratio in the future frames, it also could introduce unnecessary false retrievals.

3. System architecture

In this section, we describe the techniques that we propose for effective data management. We assume the current configuration (q_c) of the viewpoint is represented by three parameters: (x, y, \mathbf{q}) . A user uses a 2D mouse to control the virtual world. The horizontal and vertical components of the vector dragged out by the user represent the linear velocity (v) along the viewing direction and the angular velocity (\mathbf{w}) of the

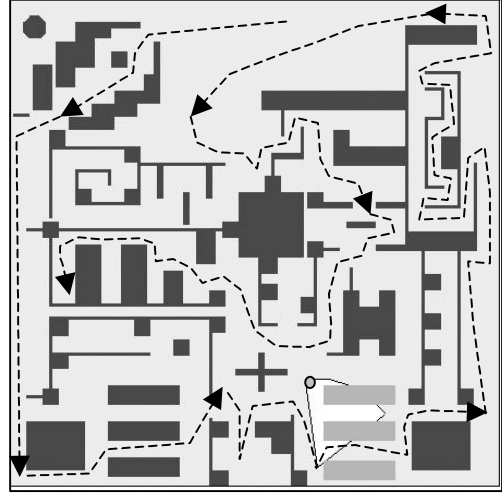


Figure 1: an example of visible objects from a viewpoint configuration in a sample world

viewpoint, respectively. These velocities are used to update viewpoint configurations. The next predicted configuration is denoted by q_p .

Since no multi-processor capability is assumed, object retrieval and graphics rendering need to be performed in each frame update. We assume that the time for each frame update, denoted by t , is composed of three components: the time for retrieving objects for the current frame (t_c), the time for graphics rendering (t_g) and the time for retrieving the predicted objects in future frames (t_p). For a specified frame rate, the system first ensures that the information for the current frame can be updated correctly and then use the remaining time to retrieve object models for future frames.

3.1. Visibility pre-computation

We assume that the virtual world does not change during the course of user navigation. The visible region, the cone depicted in Figure 1, for a viewpoint configuration q is denoted by $R(q)$. We pre-compute the distinct sets of visible objects for all possible discrete q 's at a certain resolution. The resolution is chosen such that the R 's for neighboring q 's overlap. The set of visible objects, denoted by s_q , is called *visible set* for a given q . Since s_q does not necessarily change as q changes, we compute and store these distinct s_q 's in a list, referenced by all q 's.

At run-time, we use these pre-computed visible sets to quickly identify the set of objects that need to be retrieved. However, a general q does not necessarily fall onto the grid points of our resolution. Snapping a configuration into the nearest grid point may not give us a correct visible set. Therefore, we take a more conservative approach by retrieving objects in

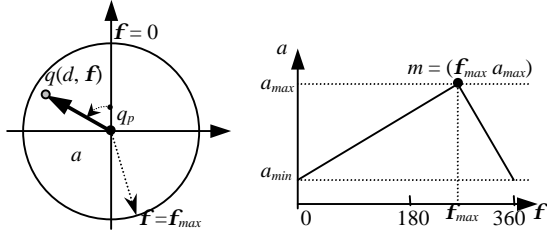


Figure 2: determining the parameter a in eq. (1) for the relevance function

the union of s_{q^i} 's for all of q^i 's 1-neighbors on the grid. Since the resolution is chosen such that neighboring R^i 's overlap, no objects are missed with such an approach.

3.2. Object prefetching

In an ideal navigation system, we desire a consistent and high frame rate. High frame rate makes a system more responsive, while a consistent frame rate makes a system more predictable and easy to control. For each frame update, we are allowed to spend some time t_p to retrieve future object models. Although this allowable time may vary for each frame, it is important to have a flexible prediction strategy that can make good use of the remaining time to retrieve the most relevant objects for the near future.

A good prefetching strategy directly affects the usefulness of the retrieved models. Instead of retrieving only the objects for a single predicted configuration q_p , we retrieve the objects for a list of configurations in the close neighborhood of q_p . In the allowable period of time, objects are retrieved according to the order of their relevance to q_p . The relevance of a configuration is defined according to four factors: (1) the distance from the current viewpoint location (d), (2) the viewing direction (\mathbf{q}), and (3) the linear velocity (v) and (4) angular velocity (\mathbf{w}) of the viewpoint. We use a continuous quadratic surface to define a quantitative relevance value for a configuration with respect to q_p .

Assume that the neighboring points of a given configuration are defined in the form of polar coordinate system: (d, \mathbf{f}) with the origin at q_p , as depicted in Figure 2. The parameter d is the Euclidean distance from the origin, and \mathbf{f} is defined as the angle from the view direction. We use a parabolic function:

$$w(a, d) = -ad^2 + c, \quad (c: \text{constant}) \quad (1)$$

along a ray emanating from q_p to formulate the relevance value (w) of any q^i 's. The shape of the surface mainly depends on the absolute value and the change rate of the drop-down parameter a . Larger a means faster dropping rate along a ray.

The value of a , as a function of \mathbf{f} , increases line-

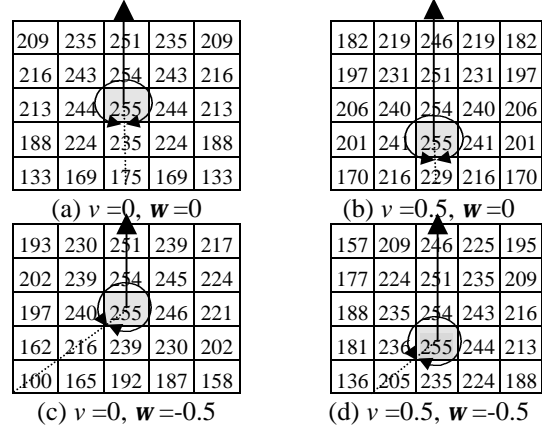


Figure 3: sample relevance masks for different linear and angular velocities

arly from the given viewing orientation (\mathbf{q} or $\mathbf{f}=0$) toward both clockwise and counter-clockwise directions (possibly with different decreasing rates), as shown on the right of Figure 2. Two parameters affect the shape of the relevance function: the maximal value of a (a_{max}) and its corresponding orientation (\mathbf{f}_{max}). We adjust these two parameters according to the linear speed (v) and angular speed (\mathbf{w}) of the viewpoint motion. As v increases, a_{max} should also be set to a larger value to make the configurations behind the origin less relevant. As \mathbf{w} increases counter-clockwise, the apex point \mathbf{f}_{max} should also be shifted toward the positive direction from the center (π).

In our current system, we use a 5-by-5 *relevance mask* to define the relevance value of neighboring configurations. The configuration q_p is located at the center of the mask or is offset by one cell for large v to accommodate more relevant configurations. To find the relevance value for a configuration q in the mask, we first determine the apex point ($\mathbf{f}_{max}, a_{max}$) in Figure 2 according to the current v and \mathbf{w} and then compute a according to the orientation (\mathbf{f}) of q under consideration. We then use this value a and eq.(1) to compute the relevance value for q . Examples of computed relevance masks are shown in Figure 3. Configurations of higher relevance values will have higher priorities in the prefetching queue for retrieving their visible sets.

3.3. Cache model

The LRU replacement policy takes advantages of the temporal locality to speed up page-based data retrieval. However, the sequence of data retrieved in a virtual world also possesses high degree of spatial locality. Therefore, we propose to take a hybrid approach of using both the access time and the relevance value of the q^i 's to account for both the temporal and

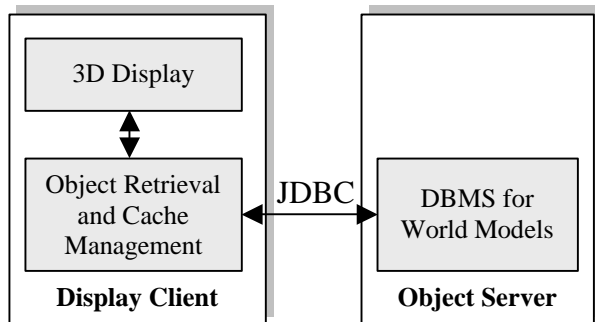


Figure 4: system architecture

spatial localities. For objects retrieved at different frame updates, we adopt the traditional LRU policy at the frame level. For objects retrieved for the same frame (possibly due to prefetching), we use the relevance values of the associated viewpoint configurations to order the replacement queue for caching. Therefore, the cache model is consistent with the prefetching strategies used above while the advantages of the LRU policy is also partially kept.

4. Experiments

The system architecture is shown in Figure 4. The object retrieval and cache management module is the main concern of this paper. Geometric models are retrieved from the object server through the JDBC interface. The 3D-display module is a VRML browser although the type of geometry representation does not limit the underlining data management scheme. The experiments are conducted in a 2D simulated environment where the object retrieval time and graphics rendering time are estimated according to the size of object models. The system frame rate is partially determined by the allowable prefetching time, which is adjustable for the tradeoff of display smoothness. The data management module is currently implemented in the Java language and provides a 2D graphical interface such as the one shown in Figure 5, to assist user navigation.

4.1. Results

The experiments are conducted on a sample collision-free navigation path generated by a path planner. The viewpoint along the same path is simulated with various experimental parameters for comparisons. As shown in Table 1, statistic data including average time for a frame update (t_{ave}), standard deviation time (\mathbf{s}), and cache hit ratio (\mathbf{a}) are collected in each experiment. The time for updating a frame includes both graphics rendering time and data retrieval time. The sample viewpoint path consists of 1068 steps as shown in Figure 5. When the cache mechanism is introduced, both t_{ave} and \mathbf{s} are improved. However, \mathbf{a}

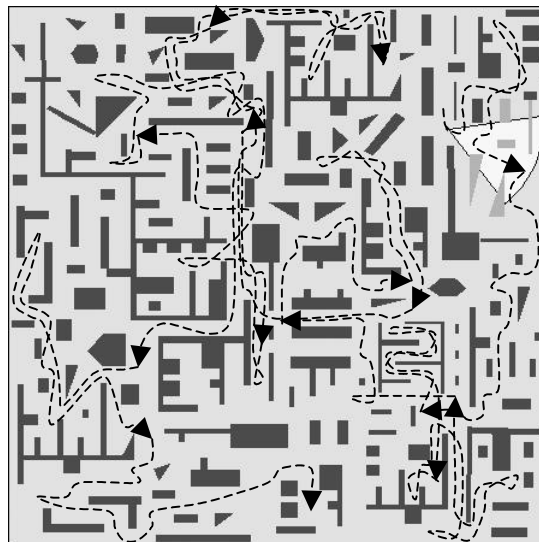


Figure 5: An example of large virtual environment and navigation path for experiments

is only 44%. When the traditional one-step prefetching is introduced, t_{ave} is slightly higher but \mathbf{s} and \mathbf{a} are improved. When the proposed relevance mask is used for prefetching, \mathbf{s} and \mathbf{a} are further improved. However, t_{ave} is also increased.

4.2. Discussions

For objective comparisons, we use the same navigation path for each experiment. The system performance is greatly improved by adopting the caching mechanism as one can expect. However, the hit ratio is low when the LRU cache replacement policy is used. This is due to the facts that the viewpoint seldom returns to the same configuration in the test path and the LRU policy does not accounts for spatial locality. In addition, the traditional one-step prefetching may not be effective if no new objects need to be retrieved for the predicted configuration or the predicted configuration is not correct. By adopting a prioritized prefetching strategy for a range of configurations, we can trade some frame rate (average update time) with navigation smoothness (standard deviation).

5. Conclusions and Future Work

In this paper we have described a data management scheme that utilizes discrete visibility precomputation, hybrid cache replacement policy, and prioritized object prefetching to achieve interactive visualization of large virtual worlds. The preliminary experiments show encouraging results on the effectiveness of these techniques. We need to further test this system on more realistic examples and conduct more experiments to find out the effects of available cache size

Table 1: comparison of experimental system with difference parameters

	case 1	case 2	case 3	case 4
AVRG (ms)	143	127	128	144
STD (ms)	93	61	57	53
Hit Ratio (%)	N/A	44	54	89

case 1: no caching or prefetching

case 2: LRU caching only

case 3: w/ caching and one-step prefetching

case 4: w/ caching and relevance mask prefetching

and allowable prefetching time.

References

- [1] J. Airey, J. Rohlf, and F. Brooks, "Towards Image Realism with Interactive Update Rates in Complex Virtual Building Environments," in *ACM 1990 Symposium on Interactive 3D Graphics*, pp. 41-50, 1990.
- [2] D. Aliaga, et. al., "MMR: An Interactive Massive Model Rendering System Using Geometric and Image-Base Acceleration," in *Proceedings of 1999 Symposium on Interactive 3D Graphics*, pp. 199-206, 1999.
- [3] Blaxxun Interactive, Blaxxun Community-Cypertown, <http://www.blaxxun.com/community/index.html>.
- [4] C. Carlsson and I. Hagsand, "DIVE – a Multi-User Virtual Reality System," in *Proceedings of IEEE Virtual Reality Annual International Symposium*, pp. 394-400, 1993.
- [5] J. Chim, R. Lau, H. Leong, and A. Si, "Multi-Resolution Cache Management in Digital Virtual Library," in *Proceedings of IEEE Advances in Digital Library*, pp. 66-75, 1998.
- [6] J. Chim, R. Lau, A. Si, H. Leong, D. To, M. Green, and M. Lam, "Multi-Resolution Model Transmission in Distributed Virtual Environment," in *Proceedings of ACM Virtual Reality Software Technology (VRST' 98)*, pp. 25-34, Taipei, Taiwan, Nov. 1998.
- [7] S. Coorg, and S. Teller, "Temporally Coherent Conservative Visibility," in *Proceedings of ACM Conference on Computation Geometry*, pp. 78-87, 1996.
- [8] F. Durand, G. Drettakis, and C. Puech, "The Visibility Skeleton: A Powerful And Efficient Multi-Purpose Global Visibility Tool," in *Proceedings of ACM Computer Graphics Conference (SIGGRAPH97)*, pp. 89-100, 1997.
- [9] M. Franklin, M. Carey, and M. Livny, "Global Memory Management in Client-Server DBMS Architectures," in *Proceedings of the International Conference on Very Large Database*, pp.596-609, 1992.
- [10] T. A. Funkhouser, C. H. Sequin, and S. Teller, "Management of Large Amounts of Data in Interactive Building Walkthroughs," in *Proceedings of ACM Symposium on Interactive 3D Graphics*, pp. 11-20, 1992.
- [11] T.Y. Li, J.M. Lien, S.Y. Chiu, and T.H. Yu, "Automatically Generating Virtual Guided Tours," in *Proceedings of the Computer Animation '99 Conference*, Geneva, Switzerland, pp.99-106, May 1999.
- [12] M. Macedoniam M. Zyda, D. Pratt, P. Brutzman, and P Barham, "Exploiting Reality with Multicast Groups: A Network Architecture for Large-Scale Virtual Environments," in *Proceedings of IEEE Virtual Reality Annual International Symposium*, pp 2-10, 1995.
- [13] M. Pocchiola, and G. Vegter, "The Visibility Complex," in *Proceedings of 9th ACM Annual Computational Geometry*, pp. 328-337, 1993.
- [14] S. Teller and C. Sequin, "Visibility Preprocessing for Interactive Walkthroughs," in *Proceedings of ACM Computer Graphics Conference (SIGGRAPH91)*, pp.61-69, 1991.
- [15] VRML97 International Standard, URL: <http://www.web3d.org/technicalinfo/specifications/vrml97/index.htm>