A Data Management Scheme for Effective Walkthrough in Complex Virtual Environments

Tsai-Yen Li and Wen-Hsiang Hsu
Computer Science Department, National Chengchi University
64, Sec.2, Chih-Nan Road, Taipei, Taiwan 11623, ROC
e-mail: {li, g8804}@cs.nccu.edu.tw

Abstract

Walkthrough technologies for virtual environments allow a user to experience realistic navigation in an unreachable scene. However, as the complexity and applications of virtual environments increase, it becomes a critical issue to maintain the quality of a walkthrough experience. In this paper, we propose an effective data management scheme to address this issue. First, we propose to use real-time scene management to manage the computing resources on the client side by reducing the amount of geometry data transmission. Second, we propose a prioritized MLM (Most Likelihood Movement) model to prefetch possible future objects, based on users' current motion intentions. This prefetch model has accounted for mouse control and the influence of obstacles in walkthrough to increase prefetch accuracy. Lastly, a hybrid coherence cache model is proposed to take advantages of both the temporal and spatial localities of walkthrough. We have implemented the front-end and back-end systems for the proposed scheme and done extensive experiments to demonstrate how these techniques can improve the effectiveness of walkthrough in a large virtual environment.

1. Introduction

Virtual environment (VE) is one form of multimedia presentation that aims to simulate realistic worlds with virtual reality (VR) technologies. As the computer and communication technologies are evolving, the applications of virtual environment are also emerging. For example, virtual museum, virtual shopping mall, traditional 3D games such as DOOM[22] and DIABLO[21], or even the popular online 3D games are all example of VE applications.

As the computer technologies are rapidly advancing, user demands for better and more delicate scenes also increase. Consequently, the complexity of a virtual scene in a virtual environment is rapidly expanding. Although the computer hardware and software developments on 3D technologies have great advances recently, we think the speed is difficult to catch up the user demands for better contents. In addition, most users are still equipped with 2D input devices on a regular personal computer only. Therefore, the quality of navigation experience remains difficult to control under such computational and input constraints. It is even more challenging for large virtual environments that usually contains a great number of object models located dispersedly.

In this paper, we propose a data management scheme to address the navigation issues in a large virtual environment. In such a scheme, we develop a real-time scene management method to reduce the amount of data retrieval and rendering times. We account for both the spatial and temporal localities when we access relevant object geometric models with a cache mechanism. In addition, we propose a Most Likelihood Movement (MLM) Prefetch Model that predicts a user's intended motions. We consider the mouse controllability and the existence of obstacles in the prefetch model. Finally, we have implemented the front-end and back-end systems for the virtual environment in which we have done extensive experiments to show the effectiveness of such a data management scheme.

2. Related Work

The related researches on handling large virtual environments started on the 90th[10] and have had made significant advances since then. According to the problems that have been addressed in the literature, we can classify the problems into two categories: *geometry replication* and *data management*. Geometry replication addresses the problem of transferring 3D geometry models from the server to the clients efficiently while data management addresses the problem of how to manage data on the client side in order to increase navigation smoothness.

2.1. Geometry duplication methods

In a navigation system with a client-server ar-

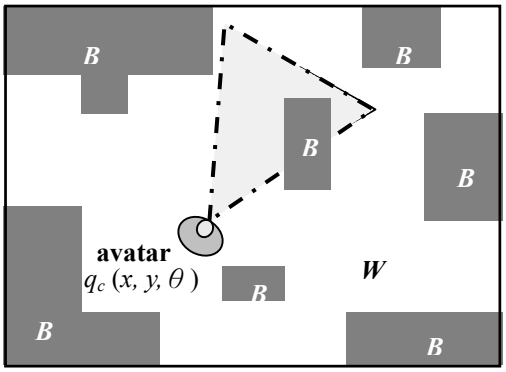


Figure 1. An example of visible objects from a viewpoint configuration in a sample world

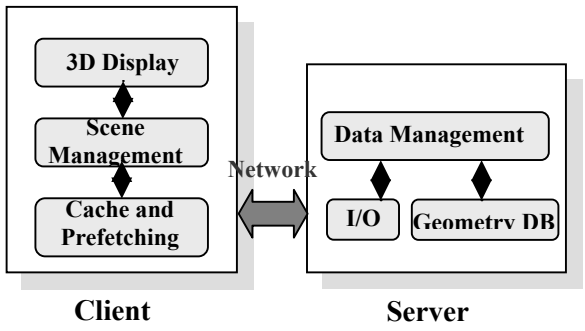


Figure 2. System architecture

chitecture, most geometric data are stored on the server side. Therefore, a common problem to face is about how to copy data to the clients effectively. The simplest method would be that the client preloads all models to the client side before starting the navigation[16][21][22]. NPSNET[16] is an example that preloads models before navigation. In such a method, the client side must be willing to wait and have enough memory space.

Instead of transferring all geometry data to the client, some researches proposed to use on-demand transmission, which transfers only necessary data to the client as they are needed during navigation. Visibility culling [12][11][19][6], Level of Detail (LOD) [7][10][13][5], and image-based rendering[1] are all examples along this line of research. In addition, other researches use 3D compression techniques [18] to reduce the amount of geometric data for object models storage and transmission.

2.2. Data management methods

Navigation in a virtual environment means a sequence of interactions between a user and a virtual environment through user input and visual feedback. Data management for navigation in virtual environments aims to use data processing techniques to make the navigation process smooth and effective. These techniques include caching, prefetching[17], efficient memory management, etc.

In order to have effective prefetching, it is critical to be able to predict a user's intended motion. Some prediction researches use statistical methods [15] that usually require significant amount of processing time. On the other hand, Dead Reckoning (DR) [8][14] is one of the simple but common methods for predicting user locations and reducing data transmissions in real time for Distributed Virtual Environments (DVEs). It uses extrapolation from the current motion to predict future motions. It is a general prediction method that does not take any advantages of the characteristics of navigation in VE. [3] proposed to use the characteristics of mouse control to predict user motions. However, since the experiments are done on paths of artificial pattern instead of real navigation paths, the effects of such a prediction method on realistic navigation paths are difficult to evaluate.

Caching is one common method for reducing data access time. The effectiveness of such methods is usually determined by the selection of good replacement policies. A good caching model could retain useful old data to future use while bad caching model could waste cache space and even increase the processing time. Among the commonly used cache replacement policies, the Least-Recently-Used (LRU) method is a popular one that takes advantage of temporal locality. However, the policy might not be very effective for the data access pattern in a virtual environment where strong spatial locality can usually be assumed[4][9].

3. System Architecture

We first give a brief description of the problem we consider in this paper and then describe the architecture of the proposed system with data management scheme. As depicted in Figure 1, the Euclidean space where an avatar exists is called *workspace*, denoted by W. The set of parameters for describing an avatar are called *Configuration*. The set of all possible configurations for an avatar is called *Configuration Space*, or *C-space* in short. The environmental objects that an avatar cannot penetrate are called obstacles, denoted by B.

In this system, we assume that the virtual environment is static, which means that the obstacles do not move during user navigation. In addition, we assume that a user can only use a mouse to control the navigation on a 2D plane in the virtual environment. We assume that the current configuration of an avatar, denoted by q_c, contains three parameters: (x, y, θ). x, y represent the current horizontal location of the avatar

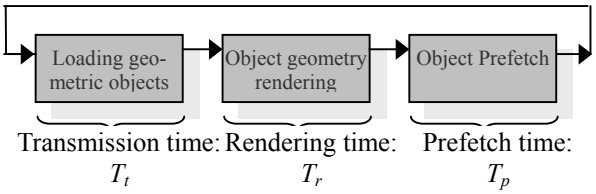


Figure 3. Frame update procedure

while θ represents the orientation. Since an avatar usually contains by a viewpoint in its local coordinate system to percept the environment, we will assume that the terms of avatar configuration and viewpoint configuration can be used interchangeably in this paper.

The architecture of the proposed system is a client-server architecture depicted in Figure 2. The client side contains three components: *3D display component*, *scene management component*, and *cache and prefetch component*. The 3D display component is in charge of graphics rendering while the scene management component decides what objects need to be fetched for rendering. The cache and prefetch component prepares or acquires necessary objects from the network. On the other hand, the server side includes a geometry and scene database, a data management component, and an I/O component. The data management component is responsible for retrieving and assembling geometry data for a given scene, and the I/O component transfers the data to the client via network.

We further assume that the system uses a one-processor machine. In such a machine, the update of each frame is divided into three phases, as shown in Figure 3. First, the object geometry must be loaded from the server side by the scene management component. The time for transferring this data is called *transmission time* or T_t for short. Once the geometric data is ready, the system renders the scene from the current viewpoint. The time for rendering is called *rendering time* or T_r for short. Finally, the cache and prefetch component use the *prefetch time* (T_p for short) to do prefetching. In other words, a frame update consists of T_t、T_r 與 T_p. The available time for T_p depends on the acceptable frame rate, $FR_{acceptable}$, specified by the user. That is,

$$T_p \geqq \left(\frac{1}{FR_{acceptable}} \right) - T_t - T_r.$$

3.1. Real-time Scene Management

The scene management component uses on-demand transmission to load geometry data as object models are needed. We use a real-time visibility-culling algorithm to determine the visible objects from the current viewpoint configuration. The algorithm is a plane-sweeping algorithm[2] that takes a viewpoint configuration and a 2D geometric description of the objects in the world as input. We assume that the visible region from a given configuration q is denoted by $R(q)$. An object is visible if any portion of its boundary intersects with the visible region. We denote the set of visible objects for q by $V(q)$.

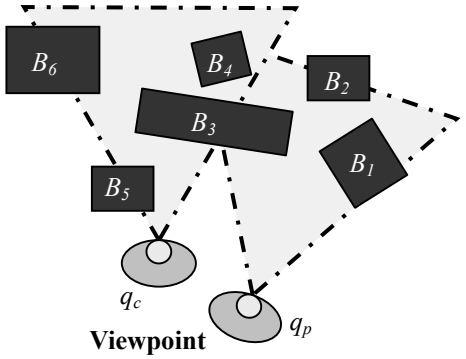


Figure 4. Example of real-time scene management

Since usually only a small number of objects are visible by a viewpoint, the amount of geometry data that need to be retrieved and rendered can be greatly reduced. For the example in Figure 4, $V(q_p)=\{B_1, B_2, B_3\}$ and $V(q_c)=\{B_3, B_5, B_6\}$. Objects that are visible in the previous viewpoint configuration q_p but not visible in the current configuration q_c, (for example, $\{B_1, B_2\}$ in the example in Figure 4) are managed by the cache and prefetch component. The component will determine when to release these objects according to the rules described in the next section. Since objects are handled in an on-demand fashion, a user will be able to perform navigation in a large virtual environment for a long time without exhausting the system resources. As the number of visible objects is limited, T_t and T_r can be reduced, which leaves more time for the system to prefetch objects for the future use.

3.2. Hybrid Coherence Cache Model

LRU is a common replacement policy that has been shown to be effective in many applications. When the cache is full, the LRU policy replaces the objects that are not accessed for the longest time. The policy is mainly based on the principle of temporal locality. Objects that are visible during navigation in a virtual environment also have such temporal cohesion. However, spatial locality could be as important for such applications. For example, objects that are closer to the current viewpoint would be more likely to be view again in the near future. Consequently, we propose to use a hybrid cache model to perform caching.

The hybrid cache model determines the order of replacement by both temporal and spatial coherences. Objects that are retrieved at difference frames will be ordered according to the LRU policy. That is, objects that are used the longest time ago will be ordered highest and replaced first. However, for objects that are retrieved at the same time frame, we order them by the distance and orientation relative to the viewpoint. In other words, the distance is computed according to a metric on the local polar coordinate system of the viewpoint. We have adopted such a hybrid cache model because in our prefetch method described in the next subsection, objects that are more likely to be seen are loaded earlier at the same frame. Therefore, it is important to consider the factor of spatial coherence in addition to the temporal based policy such as LRU.

3.3. MLM Prefetch Model

Good prefetch mechanisms make smooth navigation in a retrieve-on-demand VE system. In such a system, a user often encounters sluggishness when the viewpoint performs abrupt changes since the number of object models that need to be retrieved increases suddenly. A prefetch mechanism retrieves useful models that are likely to be used in the next few frames. Consequently, a good prefetch mechanism should be able to distribute the load of retrieving object models into earlier frames, which results in smoother navigation experiences. In order to retrieve the right object models in advance, it is critical to have a good prediction method based on user navigation behavior. We propose a prediction model, called *Most Likelihood Movement* (MLM) model, that aims to obtain a set of possible future viewpoint locations based on the current user navigation behavior.

In the proposed MLM model, since more than one viewpoint configuration are predicted, the concept of object priority becomes important. Most other prediction methods used in other related researches only predict the next single configuration and retrieve all object models that can be seen in the configuration. However, whenever the result of prediction is not accurate, the retrieved models might be not only meaningless but also wasting system resources. In order to increase the prediction accuracy, we propose to predict a set of possible configurations (called *Possible Configuration Set*, or *PCS* for short) in a region instead of a single configuration. We order the configurations by giving them priorities that are computed according to the rules described later. Within the allowed prediction time, T_p, of a frame update, the system will retrieve as many models as possible according to the priorities of PCS.

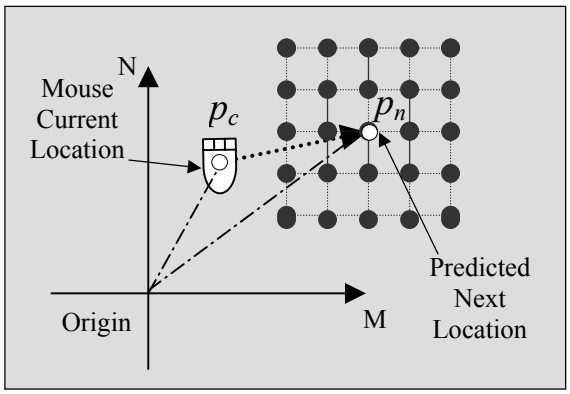


Figure 5. Possible set of mouse locations in the mouse space

3.3.1. Effects of mouse control on navigation behavior

The effectiveness of a prediction method greatly depends on the test cases. In many other related researches, we found that the test cases are either computer-generated paths of certain patterns or smooth paths resembling human motions. A prediction method that works well for these test cases might not yield good results for paths that are sampled from input devices in real navigation. We think the missing factor is on accounting for human ergonomics, the possible constraints of the input devices, and their effects on the generated paths. Therefore, in this paper, in order to increase the prediction accuracy of the MLM model, we attempt to define the prediction problem in the Cartesian space of the most commonly used input device: mouse.

As shown in Figure 5, a vector dragged by a mouse in a VE system is usually decomposed into vertical and horizontal components, which represent linear and angular velocities of the viewpoint, respectively. Transforming the vectors into velocities is usually done via some mapping functions. Since a mouse motion must obey physical rules, the continuity of the mouse trace can therefore be guaranteed. Therefore, instead of performing prediction in the viewpoint configuration space, we perform prediction in the so-called *Mouse Space*, which is in the plane where a mouse is operated and dragged. The origin of the mouse space is defined at the point where the user presses a button and starts a drag motion. The vertical and horizontal axes in the space are called the *M* and *N* axes, respectively. During a prefetch period, we predict the next possible location of the mouse based on its current motion. The predicted location is then mapped to the predict viewpoint configuration to retrieve necessary object models.

The predicted mouse location, denoted by p_n, is

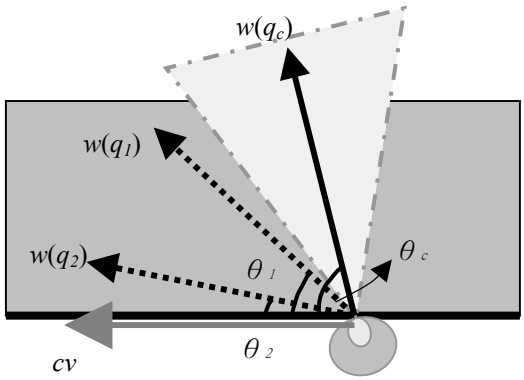


Figure 6. Effects of obstacles on the possible set of configurations

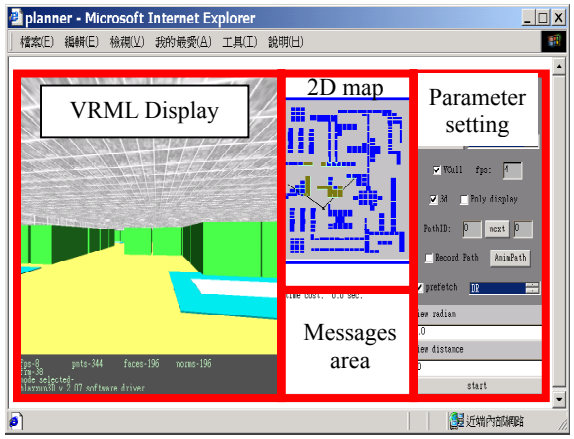


Figure 7. The frame rate variation for case 4

computed according to the current location, p_i, and the commonly used second-order Dead Reckoning (DR) method. However, instead of predicting a single location, we define a rectangle grid region centered at the predicted location. Each point in the grid represents a possible mouse location for the next frame. This grid of locations is called the *Possible Set of Configurations for mouse* (or Mouse PCS for short), as shown in Figure 5. Each mouse location is associated with a priority in the range of (0,1), which is computed according to the following formula:

$$S_d(p_i) = (1 - \frac{d(p_n, p_i)}{Q_d}), \qquad (1)$$

where d is a distance function and Q_d is a constant for quantization. The farther from the center of the region, the lower the priority.

3.3.2. Effects of obstacles on navigation behavior

We consider the effects of environmental obstacles to the possible viewpoint configuration. A common feature of a virtual environment system is detecting collisions of the viewpoint with the obstacles in the environment. A viewpoint usually will be forced to move along an obstacle if the system detects collisions. Therefore, in the possible set of viewpoint configurations, we would like to account for the influence of the obstacles by giving configurations that is making a smaller angle with obstacle boundaries higher priorities.

Assume that the facing direction of a viewpoint configuration q is represented by $w(q)$. Suppose the current configuration is denoted by q_c and q_1 and q_2 are two other neighboring configurations. The tangential component of $w(q_c)$ along the obstacle boundary is called *collision edge vector*, or *cv* for short. The priority of a viewpoint configuration q after being mapped from the mouse space is defined as

$$S_{obs}(q) = (1 - \frac{|\theta(w(q), cv)|}{\pi}). \qquad (2)$$

Therefore, as shown in the example of Figure 6, since $\theta_2 < \theta_1$, q_2 will be given a higher priority than q_1.

3.3.3. Combining the effects of mouse control and obstacles

The object models in our MLM approach are retrieved according to the order of configuration priority, which are in turn determined by the combined influences of mouse control and environment obstacles. We first compute the possible set of mouse locations p_n and their associated distance priorities by eq. (1). Then we map p_n from the mouse space into the viewpoint configuration to determine if the viewpoint will collide with the obstacles. If there is no collision, the priority will be the distance priority S_d only. Otherwise, we will take a linear combination of the distance priority $S_d(p)$ and obstacle priorities $S_{obs}(B_{m,c}(p))$ for the final priority of a mouse location p.

$$\begin{cases} S(p) = S_d(p), & \text{if no collision} \\ S(p) = w * S_d(p) + (1-w) * S_{obs}(B_{m,c}(p)) & \text{if collision} \end{cases}$$

At run time, the priorities of each possible mouse locations and their viewpoint configurations are computed according to the above equations. Object models with higher priorities will be retrieved earlier as long as the prefetch time does not exceed T_p, the prefetch time.

4. Experiments

The system proposed in this paper has been fully implemented in Java. The architecture of the system follows the common three-tier architecture for web applications. All the client side needs is a web browser, such as IE or Netscape, supporting Java applets. The server side is a regular web server (an

Apache server in our case) and a database server (the MS SQL Server in our case). The 3D models of the objects in the virtual environments are stored in individual files in the VRML (Virtual Reality Modeling Language) [23] format.

The computer that has been used in our experiments is a personal computer with an AMD TB1.2GHz CPU and 256MB RAM and the operating system is MS Windows 2000. An example of the graphical user interface at the client side is shown in Figure 7. A blaxxun3D[20] VRML browser has been integrated into the system for 3D display. The paths that have been used in our experiments are recorded from navigations controlled by a real user. The same path for a scene is tested on different experimental settings such as different cache and prefetch methods. Figure 8 shows an example of large virtual environment of size 64 by 64 meters containing 343 objects. The largest view distance is set to be 20 meters while the view angle is 2 radians. The example path shown in the figure consists of 760 configurations. We test the scene with five different cases. In case 1, no cache and prefetch methods are used, while case 2 uses the LRU cache method. Case 3 use the hybrid cache method, and case 4 uses the hybrid cache method and the 2nd order dead reckoning method for prefetch. Case 5 uses the MLM prefetch method and the hybrid cache method.

4.1. Results

We first try to observe if the real-time scene management component can effectively manage the resources on the client side. Figure 9 shows the records of frame rates for all 760 steps. We notice that the frame rate does not decrease as we navigate through the whole path. In an on-demand system, this means that the system correctly release resources for later uses. In addition, we observe that the memory consumptions before and after the navigation are 30,588KB and 32,400KB, respectively. The memory usage is only slightly increased during the navigation. This also explains the effectiveness of the real-time scene management scheme.

The average frame update rates and standard deviations for five different cases are shown in Figure 10. The data for case 1, case 2, and case 3 show that the cache mechanism can help the average frame rate by as much as 24%. Both case 2 and case 3 are all effective in saving model retrieval time. However, the frame rate for the hybrid method is not necessarily much higher than the LRU-based method. We think the main reason is that the cache size we used in our experiment is large enough to accommodate the objects seen in most recent frames. Therefore, the spatial cohesion for objects that are seen long time ago is not as important.

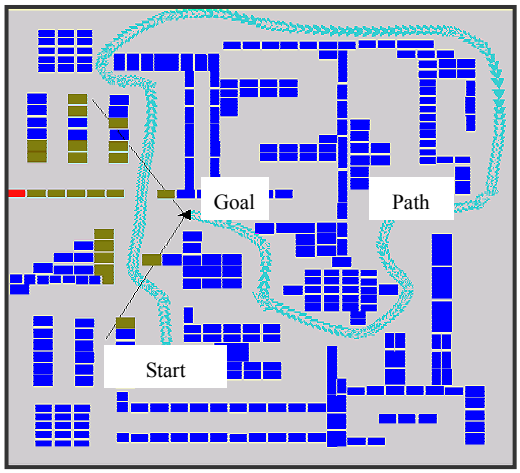


Figure 8. An example of a large virtual environment and navigation path for experiments

We then compare the effectiveness of the prefetch model. In Figure 10(b), we observe that the standard deviations (STDVs) for case 4 and case 5 are less than case 2 and case 3. The STDV for case 4 that use the DR method is about 12% less while case 5 (the MLM method) is even 83% less. This means that the MLM method can greatly improve the smoothness of the navigation experience. However, we also observe that the average frame rates for case 4 and case 5 decrease since the objects that are prefetched in each frame update may not be always used in the later frames due to prediction errors. In other words, the average and standard deviation of the frame rates are tradeoffs affecting each other. A user can tune the system behavior by adjust the acceptable frame rate ($FR_{acceptable}$) to affect the prefetch time in each frame update. For example, a user can trade degree of smoothness for higher frame rate by increasing $FR_{acceptable}$. Nevertheless, we think the frame rate does not always reflect the usability of a navigation system since fast but sluggish navigation could be even troublesome than slow but consistent navigation.

5. Conclusions and Future Work

In this paper, we have proposed a data management scheme to solve the problem of smooth navigation in a large virtual environment. We have use the visibility culling method for real-time scene management in order to reduce the amount of data that needs to be transferred from the server. We also have designed a hybrid cache mechanism to replace objects with the principles of spatial as well as temporal coherences. In addition, we have adopted an MLM prefetch model to predict the possible viewpoint con-

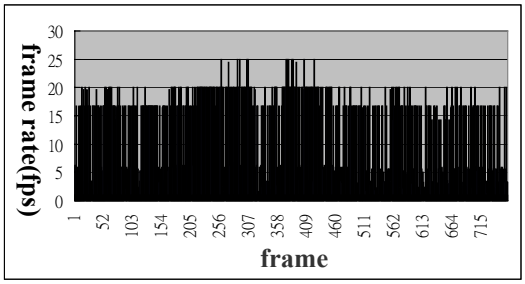
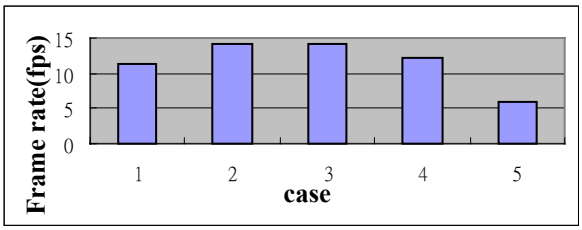


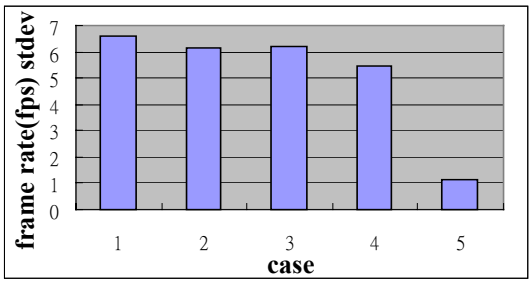
Figure 9. The frame rate variation for case 4

figurations. In order to increase prediction accuracy, we have taken the mouse control characteristics and environment obstacles into consideration. The experiments that we have done on our virtual environment navigation system show that the proposed data management scheme can increase the average frame rate and the navigation smoothness.

In our current real-time scene management scheme, we have used a plane sweeping method to cull invisible object. However, the computation of such an operation could be rather time consuming for a large environment. Consequently, it could become a bottleneck for real-time scene management. One possible solution for this scalability problem is dividing the environment into several subdivisions so that we can focus on only a small portion of the world at a time. In addition, an important parameter that needs to be adjusted at run time in our system is $FR_{acceptable}$. Since there are no objective criteria for this parameter, a user has to set its value by experience. In the future, we would like to develop a mechanism to feedback the degree of navigation smoothness to provide automatic adjustment of the overall frame rate.



(a) Average frame rates for different cases



(b) Standard deviations on frame rates for difference cases

Figure 10. Comparison of frame rates for different cases

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] M. Berg, M. Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry-Algorithms and Applications*, pp. 20-29, 1997.

[3] A. Chan, R. W. H. Lau, and A. Si, "A Motion Prediction Method for Mouse-Based Navigation," in *Proceedings Of CGI'2001, IEEE Computer Society Press*, pp. 139-146, July 2001

[4] B. Y. L. Chan, A. Si, and H. V. Leong, "Cache Management for Mobile Databases: Design and Evaluation," in *Proceedings of the IEEE International Conference on Data Engineering*, pp. 54-63, Feb. 1998.

[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] S. Coorg and S. Teller, "Temporally coherent conservative visibility," in *Proceedings of the twelfth annual symposium on Computational geometry*, pp. 78-87, 1996.

[7] M. DeHaemer and M. Zyda, "Simplification of Objects Rendered by Polygonal Approximations," in *Computers and Graphics*, 15(2):175-184,1991.

[8] DIS Steering Committee, "IEEE Standard For Distributed Interactive Simulation-Application Protocols," *IEEE Standard 1278*, 1998.

[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] A. Funkhouser, H.Seqin and J. Teller, "Management of Large Amounts of Data in Interactive Building Walkthroughs," in *Proceedings of the 1992 symposium on Interactive 3D graphics*, pp. 11-20, 1992.

[11] Z. Gigus, J. Canny, and R. Seidel, "Efficiently Computing and Representing Aspect Graphs of Polyhedral Objects," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6), 542–551, 1991.

[12] Z. Gigus and J. Malik, "Computing the Aspect Graph for Line Drawings of Polyhedral Objects," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(2), pp. 113– 122, 1990.

[13] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Mesh Optimization," in *ACM Computer Graphics (SIGGRAPH'93)*, volume 27, pp.

19-26, August 1993.

[14] A. Katz and K. Graham, "Dead Reckoning for Airplanes in Coordinated Flight," in *Proceedings of Workshop on Standards for Interoperability of Defense Simulations*, pp. 5-13, 1994.

[15] G. Liu and G. Maguire, "A Class of Mobile Motion Prediction Algorithms for Wireless Mobilecomputing and Communications," in *Mobile Networks and Applications*, 1(2), pp. 113-121, 1996.

[16] M. R. Macedonia, M. J. Zyda, D. R. Pratt, P. T. Barham, and S. Zeswitz, "NPSNET: A Network Software Architecture for Large-Scale VEs," *Presence* 3(4), pp. 265-287, 1994.

[17] S. Park, D. Lee, M. Lim, and C.Yu, "Scalable Data Management Using User-Based Caching and Prefetching in Distributed Virtual Environments," in *Symposium on Virtual Reality Software and Technology*, 2001.

[18] G. Taubin and J. Rossignac, "Geometric Compression through Topological Surgery," in *ACM Transactions On Graphics*, 1998.

[19] S. Teller and C. Sequin, "Visibility Preprocessing for Interactive Walkthroughs," in *Proceedings of ACM Computer Graphics Conference (SIGGRAPH91)*, pp. 61-69, 1991.

[20] blaxxun3D, URL: http://www.blaxxun.com/products/blaxxun3d/.

[21] Blizzard: DIABLO, Computer game, URL: http://www.blizzard.com/worlds-diablo.shtml.

[22] ID Software: DOOM, Computer game, 1994. URL: http://www.idsoftware.com/games/doom/doom-final.

[23] VRML97 International Standard, URL: http://www.web3d.org/technicalinfo/specifications/vrml97/index.html.