PHOTO-REALISTIC SCENE GENERATION FOR PC-BASED REAL-TIME OUTDOOR VIRTUAL REALITY APPLICATIONS

E. Yılmaz[a], H.H. Maraş [a], Y.Ç. Yardımcı [b] *

[a] GCM, General Command of Mapping, 06100 Cebeci, Ankara, Turkey - (eyilmaz, hmaras)@hgk.mil.tr
[b] Informatics Institute, Middle East Technical University, 06531 İnönü Bulvarı, Ankara, Turkey - yardimy@ii.metu.edu.tr

KEY WORDS: Outdoor Virtual Environment, Synthetic Scene, Crowd Animation

ABSTRACT:

In this study we developed a 3D Virtual Reality application to render real-time photo-realistic outdoor scenes by using present-day mid-class PCs. High quality textures are used for photo-realism. Crowds in the virtual environment are successfully animated. Ability of handling thousands of simultaneously moving objects is an interesting feature since that amount of dynamic objects is not common in similar applications. Realistic rendering of the Sun and the Moon, visualization of time of day and atmospheric effects are other features of the study. Text to speech is used to inform user while experiencing the visual sensation of moving in the virtual scene. Integration with GIS helped rapid and realistic creation of virtual environments. The overall rendering performance is deemed satisfactory when we consider the achieved interactive frame rates.

1. INTRODUCTION

Virtual Environments (VE) where we can pay a visit are no longer far away from us. Developments in rendering capabilities of graphics hardware and decrease in the prices can easily turn an average PC that we use in our daily life into a cyberspace. PC-based VR applications are used in many fields such as computer games, simulators, entertainment, education, medical applications or real-estate presentations. Higher realism level of the rendered scenes effects the users positively. Rough terrain models, thousands of trees, moving human beings, animals, vehicles, and buildings are some of the common elements of typical outdoor scenes. The scene needs to be refreshed as the user moves or any element of the virtual world changes position. These renderings must be performed timely so that the user is not annoyed. In a previous paper we addressed scene generation for low altitude flights (Yılmaz et al., 2004). Here we developed a cost-effective real-time VR application that is capable of rendering realistic crowded outdoor scenes from the eyes of first-person. Competition in the video game industry lead to development of algorithms that can be used to render faster and better VEs. Most of the algorithms or rendering techniques that we used in this study originated from entertainment industry.

1.1 Outdoor Virtual Reality

We focused on outdoor VR to be potentially used as a military application such as visualization of battlefield in which we plan to conduct further study. Outdoor VR has many challenging issues that take place between real-time and realism boundaries. A typical outdoor scene contains terrain, vegetation, culture, sky, clouds, people, animals etc. The total polygonal cost of these items is often beyond the rendering capabilities of many graphics cards.

1.2 Type of Virtual Reality Systems

We can categorize VR systems into two main groups: non-immersive and immersive. Non-immersive systems let the user observe virtual world through conventional display devices. Such systems are also called as desktop virtual reality. Immersive systems totally replace real world scenes with virtual ones. Head Mounted Display (HMD) is a typical immersive system tool. In this study we tested desktop virtual reality system but our study is ready to be use with immersive system tools.

1.3 Real-Time Image Generation

One of the primary requirements of VR application is the ability to update images at high speed. Ideally, this should be no slower than conventional video frame refresh rates, which are 25 Hz for PAL and 30 Hz for NTSC. Human eye is able to integrate a rapid succession of discrete images into a visual continuum that takes effect at the Critical Fusion Frequency (CFF), which can be as low as 20 Hz. Image size and brightness are two important factors that determine CFF (Vince, 1995). Commercial Image Generators used for flight simulators provide 60 Hz for daylight and 30 Hz for night scenes. In our application considering the capabilities of our target configuration and the complexity of the scenes we accepted 25 Hz and upper frame rates as real-time. Frame rates over 10 are considered as interactive. Frame rate is also strongly related with the type of VR system. While lower frame rates annoy the user in a non-immersive system it may cause motion sickness in an immersive system.

1.4 Object Database

Real-time VR applications mostly provide an object database that contains 3D models, 2D textures etc. to help creation of life like scenes. This database mostly covers different versions of the same model at different level of details (LOD). Low

* Corresponding author.

polygon models with high quality textures should be preferred. Another idea behind the use of preprocessed models is to save considerable amount of rebuilding time. In this study we used such models from Geometric and Dark Basic companies. Our library contains nearly 1000 objects consist of vehicles, buildings, people, animals, trees, bushes, fences etc.

1.5 Rendering Performance Issues

It is a known fact that developments in the graphics hardware dramatically reduced the time needed to render VEs. But parallel to this development the demand for more realism increases the complexity of the VEs. The most well known techniques for handling this complexity are:

- LOD Management,
- Visibility Culling,
- Object Organization.

In LOD management, main idea is to render same object with different number of polygons considering the distance with the virtual position of the observer. Throughout this paper the virtual location where the user observes VE is called as Virtual eye (V-eye). Objects that are far from V-eye rendered in a low-polygon form. The problem of removing redundant data that are not to be displayed is named as visibility culling. Back face culling, frustum culling and occlusion culling achieve typical reduction. Back face culling is simply described as not drawing the inner faces of the closed shape objects since they are invisible to the V-eye. The viewing volume of perspective projection is called as frustum. In frustum culling objects that are outside of the viewing volume are not sent to the graphics pipeline. This method is useful when polygonal cost of the object is high. As the name implies occlusion culling is discarding the objects that are occluded such as objects behind a wall. Depth buffer mechanism of 3D graphics libraries prevents drawing occluded objects. The problem in here is to do this job prior to rendering phase and minimize the workload of graphics hardware. Object organization defines the hierarchical organization of objects according to binary, quad or octal trees or according to some other criteria such as objects in the room.

2. IMPLEMENTATION

We developed an application that can render dynamic crowded outdoor scenes in real-time. The excessive number of objects in the virtual environment is probably the most significant capability of our system. Common features of this software and the methodology that we used are explained in this part.

2.1 Terrain Modeling

Terrain modeling is one of the most popular research topics of 3D computer graphics. This issue can be divided into two sub problems: generating geometric terrain model and painting or texturing terrain surface.

2.1.1 Geometric Terrain Models: Various types of Digital Elevation Models (DEM) are used to construct geo-specific terrain models whereas pre-rendered grey-scale bitmap images or 2D array of height values that are known as heightmap are used to construct generic models. Our implementation supports popular DEM formats. The outdoor scenes sometimes contain more than a million polygons. An average hardware accelerated graphics card cannot display that many polygons in real-time. A 7.5 minute DEM that covers 1:25K scale map contains 203401 height points which corresponds to 405000 triangles. When we consider the frame rates that should be met it is clear that it is necessary to reduce the number of polygons that are going to be rendered. Many research papers have dealt with different LOD algorithms and aggressive frustum culling. Famous methods for terrain rendering are the LOD algorithm (Lindstrom et al., 1996) and Real-time Optimally Adapting Meshes (ROAM) method (Duchaineau et al., 1997). These methods and derivatives eliminate some of the triangles by combining them with other triangles. In this study we implemented these basic algorithms to improve performance.

We also designed a terrain editor in which user can generate generic terrain models by using the tools provided. User can define the width and length of the terrain model and the interval of height points. At this point we let user to create a random base map if he wants. The random functions used by compilers are not appropriate to produce generic terrain models due to discontinuities of the functions they produce. It is possible that very high and very low two points might be side by side. A random function should change smoothly in order to be used in terrain model generation. Ken Perlin proposed a method known as P*erlin noise*, which became very popular in many fields including motion picture industry (Perlin, 1984). This method has been widely used in computer graphics. To create Perlin noise one first generates a random sequence with maximum allowable dynamic amplitude range. This sequence is smoothed using interpolation techniques. As a next step another random sequence with twice the frequency and half the dynamic range is created and interpolated. This procedure is repeated until the desired spatial resolution is obtained. Finally all intermediate random sequences are combined by addition. As it can be easily seen in Figure 1, sum of noises looks like a silhouette of a mountainous area. We used 2D Perlin noise functions to generate generic terrain model.

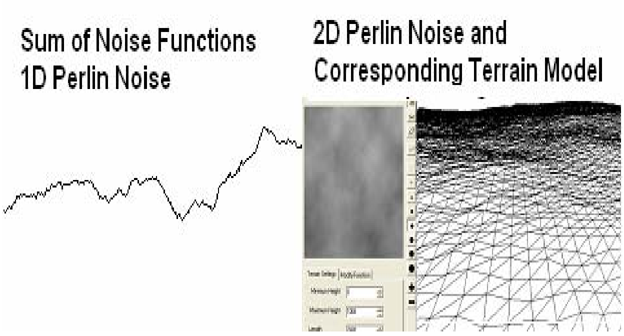

Figure 1. Illustration of Perlin Noise

The user can modify this model with mouse or use it directly. In the Terrain Editor module, Gaussian equation is used to edit the terrain model. The point that the user clicks with the mouse is considered as the centre point. The region around this point can be raised or lowered to create hills or pits. It is possible to change the parameters of the Gaussian function in order to generate different shapes. Repetitive operations enable the user to create the terrain that he wants. The user can also use a flat terrain model as a starting template. There are two additional functions that can be used for fine-tuning. First one is convolution function that smoothes the model and the second one is addition of random noise that makes the model rough. All of the operations can be performed either on a 2D heightmap image or in a 3D model. Terrain Editor tool is also very useful to construct real terrain models if the height data is not available. The user can edit the terrain by using paper maps.

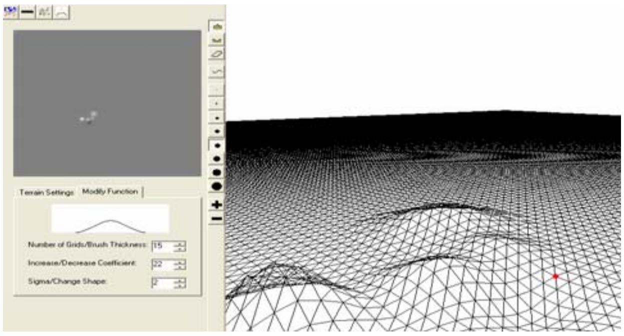

Figure 2. Terrain Editing

2.1.2 Terrain Surface: Aerial or space-borne images are the main tools that are used to texture map terrain surface in many VEs. This approach works when user looks at terrain from higher altitudes. Typical resolutions of these images are not sufficient when user walks through. As explained before human eye resolution is 1 arc minute in normal day light conditions. We can use this as rule of thumb to find the average resolution that the V-eye sees when walking through the scene. This value corresponds to 1.5 mm at 5 meters and 2 cm at 70 meters. This calculation is dependent on contrast, shape of object, visibility conditions, lighting, concentration, speed etc. To achieve desired image resolution, game industry uses pre-rendered high quality artificial textures that reflect the surface property such as grass, arid, rocky, cement etc. We also used such textures prepared by artists.

2.2 Crowd Animation

Crowds are part of real life. It is possible to render very realistic virtual 3D model of a wonder such as Hagia Sophia by using augmented reality techniques but it may not be enough to feel the viewers in the VE as they are in the real world unless the environment includes walking people, pigeons, vehicles etc. Historical battlefield scenes are typical examples where it is necessary to render thousands of animated characters. Our software managed to handle this issue. Thousands of marching soldiers over rough terrain are rendered successfully in real time. We conducted rendering performance test in virtual battlefield.

2.2.1 Real-Time Animation: Animation of rigid body objects such as vehicles are relatively easier when compared to animation of bipeds or quadrupeds. For example human movement is a very complex task. Many researchers deal with the inclusion of animated human actors in virtual environments. The synthesis of human motion is one of the most challenging areas in computer graphics since human being possesses more than 200 degrees of freedom (Chung, 2000). Computer game industry, which is the leading power in the development of real-time rendering techniques, simplifies this complicated issue. Below are some popular techniques that are used for real-time character animation (Anderson, 2001).

- 3D hierarchic articulated object animation.
- Key-frame animation.
- Skeletal animation.
- Real-time inverse kinematics.

3D hierarchic articulated object animation uses local and general transformation matrices to perform animation of each body part separately and character as a whole. This method consumes less memory and computation cost is low but the quality of visual output is very poor due to gaps between separate body parts. In key-frame animation a character model is taken and using 3D modeling tools animates a loop of action such as walk. This animation contains different but limited number of 3D character poses while moving. These poses are known as key frames. In order to smooth this animation new frames are interpolated which are called in-betweens. Realistic animations can be done with this technique. The advantage is low computational cost. Main disadvantages are the size of required memory space and limitation of using only predefined actions. Skeletal animation technique is used to make more realistic animation of articulated characters in virtual environments. Many popular 3D games use skeletal animation. Inverse kinematics can be considered as an alternative animation method. When initial and final positions of objects at specified times are given motion parameters are computed by the system (Hearn & Baker 1997). This method requires more computation than preceding methods.

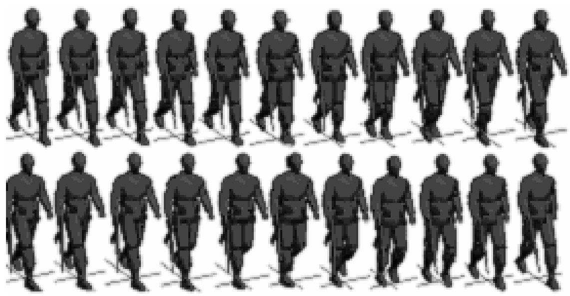

Figure 3. Walk Animation

In this study considering memory, computation costs, visual quality and other requirements of real-time rendering we decided to use key-framed animations prepared in 3D Studio Max format. Since we deal with real-time rendering of high number of virtual characters, we had to minimize polygons to be rendered. LOD management and visibility culling are the main methods used in this study for crowd animation.

Three different sets of virtual characters with high, medium and low polygon counts are used in this study. Distance is the criterion to choose the appropriate LOD. The user can perceive distinct transition between two models at different levels of detail. This disadvantage can be eliminated by using progressive meshes that are redefined at run-time to provide smooth transition (Sullivan et al., 2002). Progressive meshes are not implemented in this study and considered as a future work.

We applied standard back-face culling. We also implemented frustum culling. To minimize number of comparisons we organized virtual characters into groups. Inspired by roman army structure we called these groups as legions. Each legion has its own bounding sphere that is used for frustum culling. We also used special indices to keep track of terrain block/blocks which legion and every single virtual character is on. Since we control visibility of terrain blocks according to quad-tree structure prior to every visibility and LOD management, this mechanism decreases frustum-culling check significantly. If a legion passes frustum-culling test we then check every virtual character's bounding sphere. The distance between the observer and the midpoint of the legion, which can be obtained easily during frustum-culling test, is used by LOD management mechanism to prevent duplication. Occlusion culling could also be useful. Some topographic features like high hills or deep valleys may block objects behind or inside them. We have not implemented this control yet. It is also another future work for this study.

2.2.2 Moving Over Terrain: Rendering of bipeds or quadrupeds that are moving on non-flat surfaces is a challenging issue. On rough terrain segments where slope is high, feet of virtual characters may seem to sink or rise unless exact actions of the alive are modeled. This problem is inevitable when key-framed animations prepared for flat surfaces are used as in our case. Shifting few centimeters above from ground level considering the degree of slope may decrease the effect of this problem since V-eye cannot easily perceive little rise from ground. Sink/rise problem is tolerable when the degree of slope is small even if no precautions are taken.

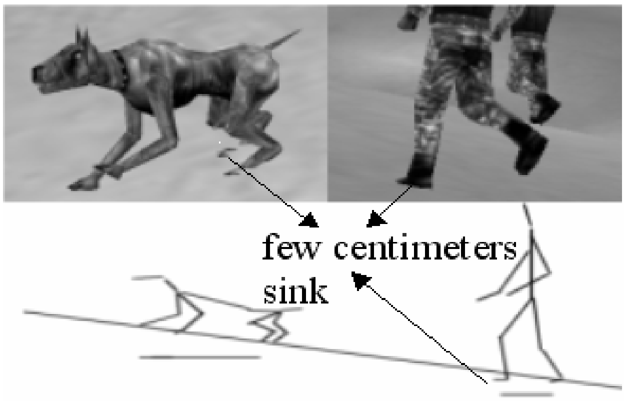

Figure 4. Sink/rise problem

In order to model actions such as walking, it is necessary to calculate the actual three-dimensional position of each foot. The exact height of a point on terrain model can be calculated by using plane geometry. Choosing triangles as a primitive to construct terrain model simplifies the calculation of height of the point on terrain. Below is the method that can be used to get height of a point on terrain model when horizontal coordinate pair is provided.

- Find the triangle on which the point lies.
- Get vertex coordinates of the triangle.
- Calculate the normal vector of the triangle by using the Equation 1.
- Calculate the plane-shift constant value of the triangle by using the Equation 2.
- Calculate the height value by using the Equation 3.

$$\underline{N} = (\underline{V}_2 - \underline{V}_1) X (\underline{V}_3 - \underline{V}_1) \quad (1)$$

$$D = N_x V_{1x} + N_y V_{1y} + N_z V_{1z} \quad (2)$$

$$P_y = (N_x P_x + N_z P_z - D) / N_y \quad (3)$$

where $\underline{N}$=normal vector of the plane
$\underline{V}_1$, $\underline{V}_2$, $\underline{V}_3$=vertices of the triangle in vector form
D=plane-shift constant
P_x, P_y, P_z=3D coordinates of the point

In order to decrease the computational cost we calculated the normal vectors and plane-shift constants of every triangle of the terrain model and kept these pre-calculated values in memory.

During an animation it is also necessary to calculate the new position of the virtual character in every frame. We used Equation 4 to calculate step distance for every animation frame. For faster operation pre-calculated slope values of the triangles should be better kept in computer memory. Equation 5 is used for getting new horizontal position of the virtual character. Finally by using Equation 3 vertical coordinate can be extracted. This operation is conducted only for objects inside viewing frustum.

$$SD = \frac{1000}{3600} . \frac{AS}{FR} . \cos \alpha \quad (4)$$

$$\begin{aligned} P_x^n &= P_x^o + SD \ \cos \beta \\ P_z^n &= P_z^o + SD \ \sin \beta \end{aligned} \quad (5)$$

where α=slope angle
SD= step distance
AS=average speed in km/hour
FR=frame rate in 1 second
ß=heading angle
P^n, P^o=new and old position.

2.2.3 Behavior Simulation: Simulation of crowd behaviors is a popular research area since more and more crowded scenes are being included in real-time animation applications. Autonomous agents in VE increase the level of realism. When we examine the conceptual and technical requirements of multi-agent systems we can see that we are facing a task that is not straightforward. First of all variety of individual agents' visualizations and behaviors such as variety of individual trajectories for the group traveling along the same path, variety of the animations for agents having same behavior or different reactions of individuals facing the same situations is needed. This prevents monotony of the scenes that include same individuals with the same behaviors. Multi agent models require more computational sources, which linearly (agent-environment interaction) or quadratically (agent-agent interaction) increase with the number of simulated agents (Ulincy & Thallman, 2001). In our study we did not focus on the AI of virtual crowds since our primary aim is achieving high frame rates at crowded scenes. Collision has avoided by assigning same average speed and direction to each group. Virtual characters are arranged to have the same motion type: walk, idle, wait etc. The user can also externally guide the total crowd or sub groups by changing motion type, direction and average speed. Our implementation is crowd animation rather than crowd simulation.

2.2.4 Sample Scene: We generated a VE to measure the rendering performance of the study. In this VE 10.000 animated soldiers walk in groups of 100 legions over non-flat terrain model which consists of 524288 triangles. Each soldier model is made up of 700-1000 triangles. Test PC configurations and rendering results are given in Table 1 and Table 2 respectively. Number of soldiers in the viewing frustum is rounded.

Figure 5. 10.000 animated characters in test VE

Name	Processor	Memory	Graphics Hardware
Test PC#1	Intel Pentium 4, 1.8 Mhz	256 mb	NVidia GeForce2 go 400, 32 mb
Test PC#2	Intel Pentium 4, 2.4 Mhz	512 mb	NVidia GeForce4 go, 460 64 mb
Test PC#3	Intel Pentium 4, 3.2 Mhz	2 gb	NVidia GeForceFX 5900, 128 mb

Table 1. Configuration of Test PCs

Test PC	# of Soldiers in Viewing Frustum	Frame Rate
Test PC#1	3000	4 fps
Test PC#1	380	20 fps
Test PC#1	110	30 fps
Test PC#1	1100	10 fps
Test PC#2	300	30 fps
Test PC#2	800	15 fps
Test PC#2	1970	6 fps
Test PC#3	1700	20 fps

Table 2. Rendering performance results

2.3 Sound

Let us imagine a scenario that happens in immersive VR system: The user walks in the geo-specific VE. When he wants to learn the name of the hill he faces, one way is to display text that shows the name. Although the user is informed, the displayed text destroys realism, as in the real world we do not see geographic names on the hills. Also this solution is not different than using traditional 2D GIS or paper map. Another solution, which we prefer, is text to speech mechanism. By using GIS import tool we get the coordinates of geographic locations in Gazetteer and defined a buffer around them. For example when the user touches a hill with data glove he hears the name of the hill. We simulated this scenario on desktop VR with mouse and evaluated it useful. Regarding the use of sound in VE we used library of wav files that contains sound effects such as wind, marching group, rain, various engine sounds etc. Although these simple sounds contributed the realism of the environment more realistic use of sound is essential. Real world effects such as Doppler effect may better impress the user.

2.4 Conceptual Elements

2.4.1 Sky and Clouds: Atmospheric rendering is an important step to generate impressive virtual environments. There are a lot of methods for sky and atmosphere rendering, ranging from the use of single color to very realistic models. The sky color is time and location dependent. It is a known fact that the sky color around the horizon is not same with the sky color around zenith at the same time. It is also known that the sky color around horizon becomes red at sunrise and sunset. The altitude of the sun, the viewing direction, the height of the observer, conditions of the atmosphere, and the reflected light from the ground are the parameters that affect the color of the sky (Nishita et al., 1996). It is a very complex task to try to render sky according to criteria listed above. In order to simplify this complicated task we built pre-rendered skybox library. Skybox is a cube in which inner faces are texture mapped with five or six pre-rendered images. When this cube is folded, inner faces create a seamless scene. Aesthetic skyboxes need artist work. The easy way to create them is to use special landscape rendering packages. It is also possible to obtain skybox image sets on Internet. Our skybox library contains many consequent scenes that complete a day loop.

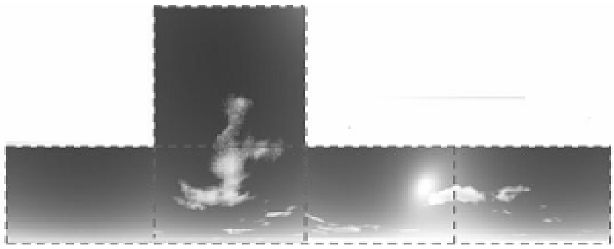

Figure 6. Skybox

2.4.2 The Sun and the Moon: VR applications that render real world conditions use the Sun and the Moon as light sources and complementary objects of the VE. In both usages, it is important to place them into their correct positions in the three-dimensional scene. To calculate the positions of the Sun and the Moon at a given time and location, some methods use astronomic almanacs and complex equations, which give precise results and some others use simple formulas to get rough results. Jean Meeus, a Belgian astronomer published a book *Astronomical Algorithms* for computer calculations, which became popular among amateur astronomers and computer programmers (Meeus, 1991). Geocentric positions are accurate to within a few arc-seconds, which is many times higher than typical desktop display resolution precision. In order to correctly visualize the Sun and the Moon it is necessary to calculate angular sizes and locate them on the outer border of the limited VE.

$$\Theta = 2\arctan(r/d) \quad (5)$$

where Θ= angular size of celestial body
r= radius
d= distance to the Earth.

Rendering of the Moon is quite different since it necessary to determine the visible portion and the bright limb angle, which corresponds to inclination with respect to rotation axis.

2.5 Editing Features

By using our software realistic geo-specific or generic VE can be generated easily. The user can populate the scene with thousands of trees, moving people, animals, buildings, bridges, vehicles etc. With few mouse clicks walls, fences or roads can be created. The user can browse any object from object library and place them in 3D environment. It is also possible to select any object/s in 3D environment and modify it.

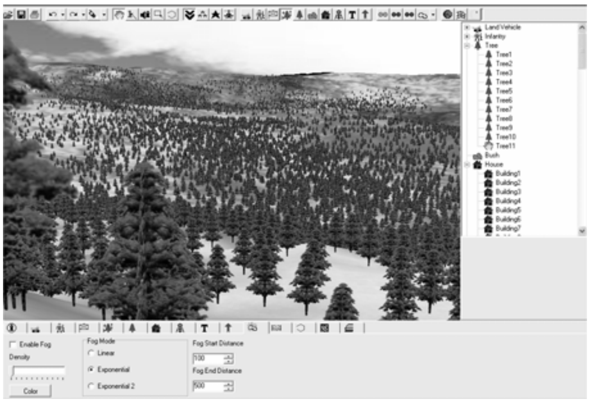

Figure 7. Virtual Environment Editor

3. CONCLUSION AND FUTURE WORK

Our study can be evaluated from two different viewpoints: VE editor and the rendering performance.

Rendering performance can be evaluated as satisfactory when we consider the frame rates achieved at scenes where thousands of static and dynamic entities included. When we consider the VR scene editor functionality easy interaction with 3D environment and objects may be the reason behind the rapid scene development capability. Ability to construct groups from single objects such as forest from a tree or crowd from a human helps to populate scenes in a short time. This feature additionally decreases frustum-culling computations. Rich content of the object library, skybox library or terrain surface texture library also meets most of the requirements for realistic scene generation. Data import ability from existing GIS systems is also another feature that contributes the editor functionality.

Crowd simulation rather than animation is important future work. Autonomous agents that act like a real human should replace the virtual characters in our implementation. Common algorithms such as *boids* (Reynolds, 1987) that represent animal flock behaviors could also be used to model flocks in this study. Occlusion culling which is believed to contribute achieving better frame rates is another future work. Audio is complementary element of any VE. Although we implemented sound, it should be renewed to meet needs of VEs. 3D sound feature that works according to position of V-eye is considered to improve realism and the feeling of immersion.

ACKNOWLEDGEMENT

We would like to express our sincere thanks to Mevlüt Dinç who conducted rendering test on various PCs.

REFERENCES

Anderson, E.F., 2001. Report on computer animation, "Real-Time Character Animation for Computer Games", National Centre for Computer Animation, Bournemouth University. http://ncca.bournemouth.ac.uk/newhome/alumni/docs/CharacterAnimation.pdf (accessed 11 March 2004)

Chung, S., 2000. Interactively Responsive Animation of Human Walking in Virtual Environments. Ph.D. Thesis, George Washington University, USA.

Duchaineau, M., Wolinsky, M., Sigeti, D.E, Miller, M.C., Aldrich, C. and Mineev-Weinstein, M.B., 1997. ROAMing Terrain: Real-time Optimally Adapting Meshes, *Proc. IEEE Visualization '97*, pp. 81-88.

Hearn, D. and Baker, M.P., 1997. *Computer Graphics*. Prentice Hall, New Jersey, USA, 2nd. Ed., pp. 595-596.

Lindstrom, P., Koller, D., Ribarsky, W., Hodges, F.L. and Faust, N., 1996. Real-Time Continuous Level of Detail Rendering of Height Fields, *Proc. ACM Siggraph96*, pp. 109-118.

Meeus, J., 1991. *Astronomical Algorithms*. Willmann-Bell, Richmond Va., USA.

Nishita, T., Dobashi, Y., Kaneda, K., Yamashita, H., 1996. Display Method of the Sky Color Taking into Account Multiple Scattering. *Proc. Pacific Graphics*, pp. 117-132.

O'Sullivan, C., Cassell, J., Vilhjalmsson, H., Dobbyn, S., Peters, C., Leeson, W., Giang, T. and Dingliana, J., 2002. Crowd and Group Simulation with Levels of Detail for Geometry, Motion and Behaviour. *Proc. Third Irish Workshop on Computer Graphics*, pp 15-20.

Perlin, K., 1984. ACM Siggraph 84 conference, course in Advanced Image Synthesis.

Reynolds, C. W., 1987. Flocks, Herds, and Schools: A Distributed Behavioral Model. *Proc. SIGGRAPH '87*, volume 21, pp. 25-34.

Ulincy, B. and Thalmann D., 2001. Crowd simulation for interactive virtual environments and VR training systems, *Proc. Eurographic workshop on Computer animation and simulation '01*, pp. 163 – 170.

Vince, J., 1995. *Virtual Reality Systems*. Addison-Wesley, Singapore, pp. 9-263.

Yılmaz, E., Maraş, H.H. and Yardımcı, Y.Ç., 2004. PC-Based Generation of Real-Time Realistic Synthetic Scenes for Low Altitude Flights, Appear in the *Proceedings of SPIE Vol. # 5424*, Orlando, USA.