'Photogrammetric Week 05' Dieter Fritsch, Ed. Wichmann Verlag, Heidelberg 2005.

Photo-Realistic Outdoors Scene Visualizations with PCs[1]

ERDAL YILMAZ, GCM
YASEMIN YARDIMCI CETIN, METU

ABSTRACT

Even though we are not aware of the some effects of computer games and similar applications that offer high reality visualization, they made virtual reality a part of our technological life. Current technology demos already shown that it is possible to visualize the growth of a grass and micro details on its surface by using upcoming game consoles. Such cutting-edge applications designed by talented graphics artists and realized by industries most smart software developers set new standards for virtual reality applications. No matter what type of real environments visualization you are working on, you should meet many standards defined by other virtual reality applications. In this study, we tried to develop software that can be used to visualize outdoors scene of the landscapes that really exist on the earth. Instead of texture-mapped model scenes we tried to generate virtual worlds that simulate the dynamism of the life as it is. In order to increase the usage areas we used on-the-shelf PC hardware. Our software can be evaluated as satisfactory considering the quality of the scenes and rendering performance.

1. INTRODUCTION

In our previous work we tried to render photo-realistic outdoors scenes by using our in-house-built software (Yilmaz et al, 2004). In this study we improved this work and also focus on new issues like flight simulator scenes and panoramic views.
Nowadays computer graphics applications develop at an incredible speed. Just a decade ago simple 3D scenes were good enough to impress users. Virtual reality was mostly depicted by the picture of a user that wears some future-looking stuff like head-mounted-display and data gloves and tries to touch some 3D primitives like a spinning cube. Today such simple 3D scenes do not meet the user needs who really desire demanding graphics. As a result of this trend, once simple wire-frame terrain models of real earth, evolved into texture mapped complicated models. These textures range from satellite images to airborne photographs. Although satellite image-draped terrain models are quite new to PC users, they have been commonplace in flight simulators of million-dollar graphics hardware for many years. The new challenge in the image generators of flight simulators is to produce exact replica of real world on the monitors mounted in the cockpit. Having been inspired by these high-end applications, we tried to port realistic outdoor scenes to PCs. Our software is designed to:

- run on low-cost PCs,
- render scenes at reasonable speed,
- provide visual sensation of realism,
- Let the user to create or populate scenes easily.

These requirements and our solution are illustrated in Figure 1.

[1] This work is supported by 3D TV European Network of Exellence funded by the EC 6th Framework IST Programme.

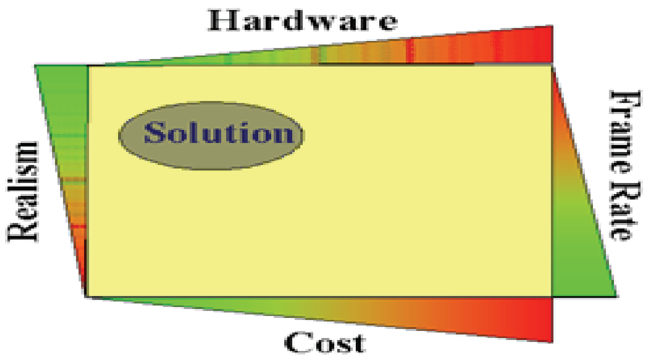

Figure 1: Rendering Requirements

2. OUTDOORS SCENE COMPONENTS

There is no single way of determining components of outdoors scene. For simplicity we used the below listed components. Terrain model is the base of this structure. Excluding sky model every element takes their places over terrain model. They can also be considered as layers.

- Terrain Model
- Sky Model
- Vegetation and Culture
- Vehicles, Animals and Human Beings

Everything in this list changes at different periods. For example the color or the quantity of the leaves of the trees changes in every season, the moon moves, new buildings appear, children walk etc. Even terrain shape changes after a natural phenomenon or when we construct highways, dams etc. We tried to reflect this dynamic nature as much as possible in our study. For example when the user changes season the status of the vegetation layer also changes. Let us see these components in detail.

3. TERRAIN

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.

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. Typical Outdoors scene contains 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 and decreasing resolution of the regions that are far away. In order to get better performance we implemented these basic algorithms together with frustum culling techniques. Another trick that we used is the quad-tree structure of the terrain data which significantly decreased the number of controls that must be done frequently. Now we can explain the works mentioned above. The LOD algorithm for terrains works best for height fields consisting of (2^n +1) (2^m +1) points, where m and n are any integer values. The height field is initially divided into subparts called as tile that contains 2^k+1 points on each side. The value of k is half of the lower of m and n. Each tile shares the vertices at the borders with neighboring tiles in order to avoid gaps. As the name implies, there are different levels in the LOD algorithm. The tiles are at Level 0 initially. The next level omits the even number grid points thus decrease the number of triangles down to the one fourth of the preceding level. Level 2 has 16 times as fewer triangles as the Level 0. In Figure 2b, a 9X9 points heightfield is divided into 4 tiles of 5X5 points, and Level 0, Level 1 and Level 2. It can be seen that the edge rows or columns of the low level tile is modified if two tiles at different level are neighbors. This prevents cracks. As we can see the number of triangles is reduced significantly. At the initial rendering there were 128 triangles and at the final rendering there were 44 triangles.

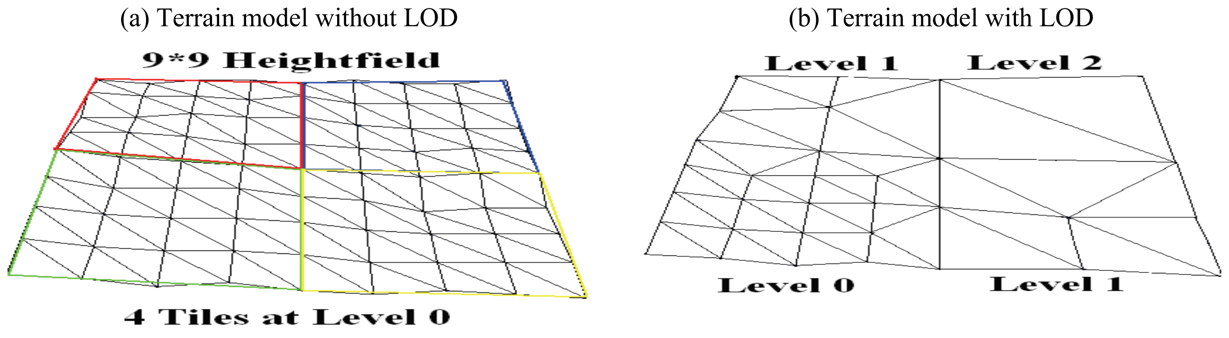

Figure 2: Illustration of LOD implementation for Terrain Models

Now let us examine the contribution of frustum culling and the tile organization implemented by using quad-trees. First step is the division of the terrain into tiles as shown in Figure 3. In order to determine the level of the tiles to be rendered we use the neighboring relation of the tiles. The center tile where the camera or its projection is in and the neighbor tiles are rendered as Level 0 that are gray boxes in Figure 3. The blue boxes are rendered in Level 1 and the red ones are in Level 2. The tiles are kept in a quad tree structure in order to speed up the frustum culling operation and spatial querying. As we can easily see in this example it is enough to render only %7 of the total triangles rendered in Brute Force approach.

We also designed a terrain editor in which user can generate generic terrain models by using the Perlin noise method proposed by Ken Perlin which became very popular in many fields including motion picture industry (Perlin, 1984). This method has been widely used in computer graphics.

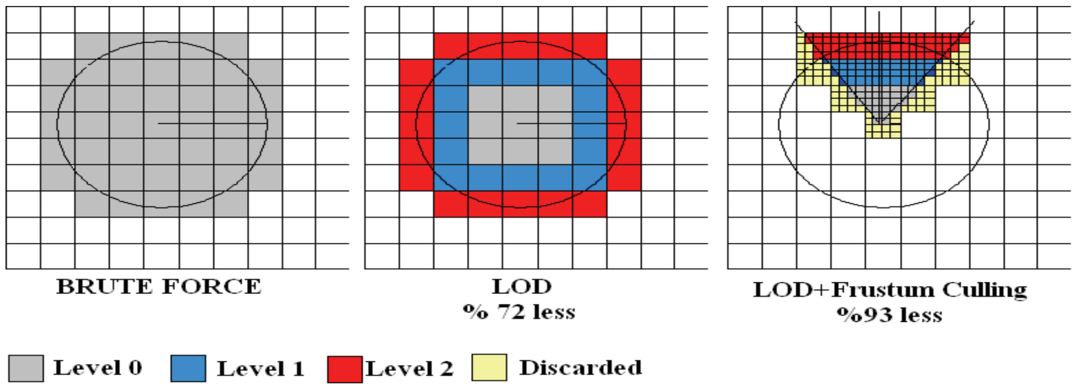

Figure 3: Illustration of LOD, Frustum Culling and Quad-Tree

Aerial or space-borne images draped over geometric models to make realistic terrain models. Typical resolutions of these images are not sufficient when user very close to the ground level. Human eye resolution is 1 arc minute in normal day light conditions and resolve details less then a millimeter while walking. This calculation is dependent on contrast, shape of object, visibility conditions, lighting, concentration, speed etc. It is very clear that even we use highest aerial image resolution that is commercially available; it is far away from simulating the real human vision while walking through. 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. These special textures construct a seamless structure when combined. There are no artifacts at the borders. In Figure 4a we can see the real pavement stones in front of the New Mosque in Istanbul. In Figure 4b same area is visualized in a Computer game from a different camera location. As we can see it is very difficult to distinguish the differences of the real pavement in Figure 4a and artist made ones in Figure 4b.

(a) Front view of New Mosque, Istanbul (b) New Mosque Area in a computer game

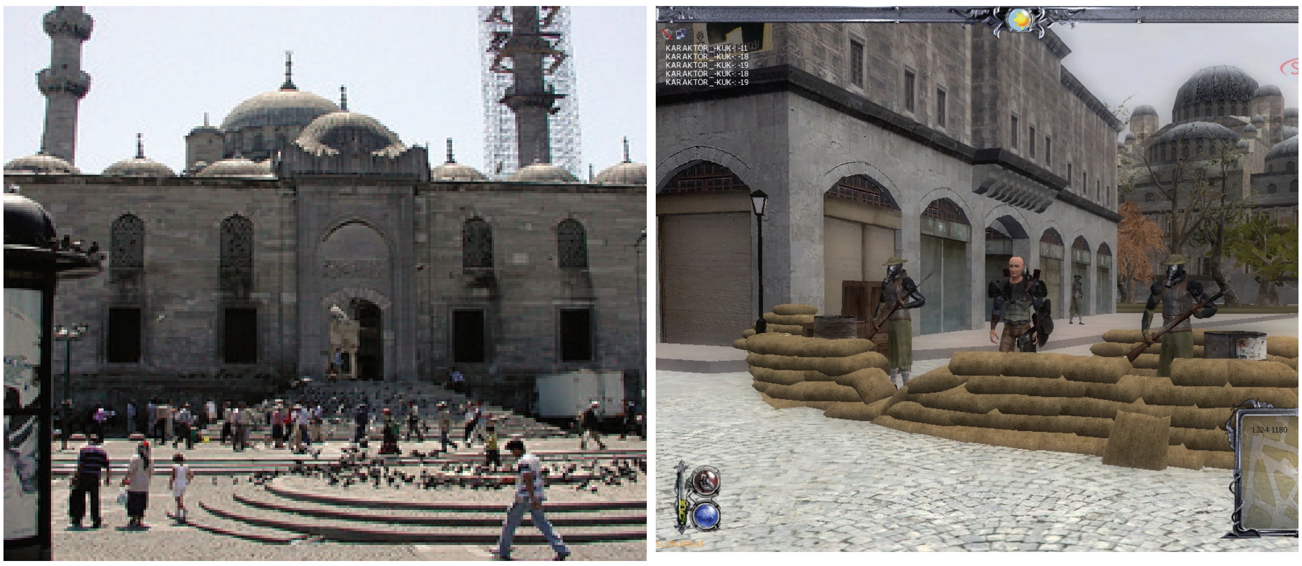

Figure 4: Comparison of pavements in real image and computer game scene.

4. SKY MODEL

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. Dynamism is the main lacking phenomenon of the sky dome and skybox methods. Dynamic clouds are the solution of this problem. A gray level cloud image is tiled over the sky. Values below threshold are set as transparent. This cloud layer can be rotated or translated to give user the impression that clouds are moving. An extra layer with different clouds above the first layer adds more realism to the scene. This dynamism brings extra calculation cost since there are a lot of extra triangles to move, rotate and finally render.

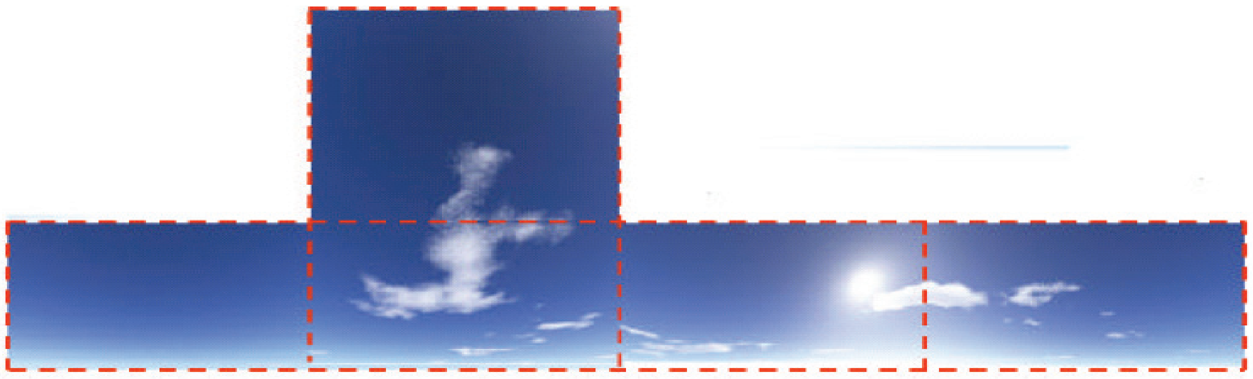
Figure 5: Unfolded skybox

Rendering of night scenes quite different than day light rendering since it is necessary to position sun and moon correctly and calculate the value of illumination. The ability to render night scenes accurately would be useful for many applications including film, flight and driving simulation, games and planetarium shows (Jensen et al., 2001).
Virtual reality applications that render real world conditions use the Sun and the Moon as light sources and complementary objects of the scene. 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). We used the algorithms given in this book to handle this issue.

5. VEGETATION AND CULTURE

"The visualization of landscape models that only feature terrain and building data tends to look dull and lifeless. Strikingly is particularly the absence of trees and other vegetation" (Fritsch and Kada, 2004).Vegetation rendering in real-time and interactive applications is one of the challenging problems in computer graphics due to very complex nature of the plants. A realistic tree model may consist of thousands of polygons. Polygonal models offer realistic true three-dimensional solutions, but they are not preferred in real time applications due to high number of polygons. It is possible to use image based solutions known as Billboarding. Although it offers poor solutions from the window of human perception, it only requires rendering 4 triangles at all for a single tree. Hybrid methods can be used in order to fill the gap between expensive/high-quality and cheap/low-quality solutions. A tree model whose trunk and main branches are made of polygons and leaves and small branches are represented with image slices can be used as an intermediate level solution (Jakulin, 2000). In this study Billboarding technique is used due to the faster solution it offers, so it is possible to generate scenes with forests that consist of thousands of trees. In Figure 6 we can see the contribution of thousands of trees to the outdoors scene visualization when the user is near the ground level. Without trees it is not possible to feel that you are flying over forest which is illustrated in 6c and 6d.

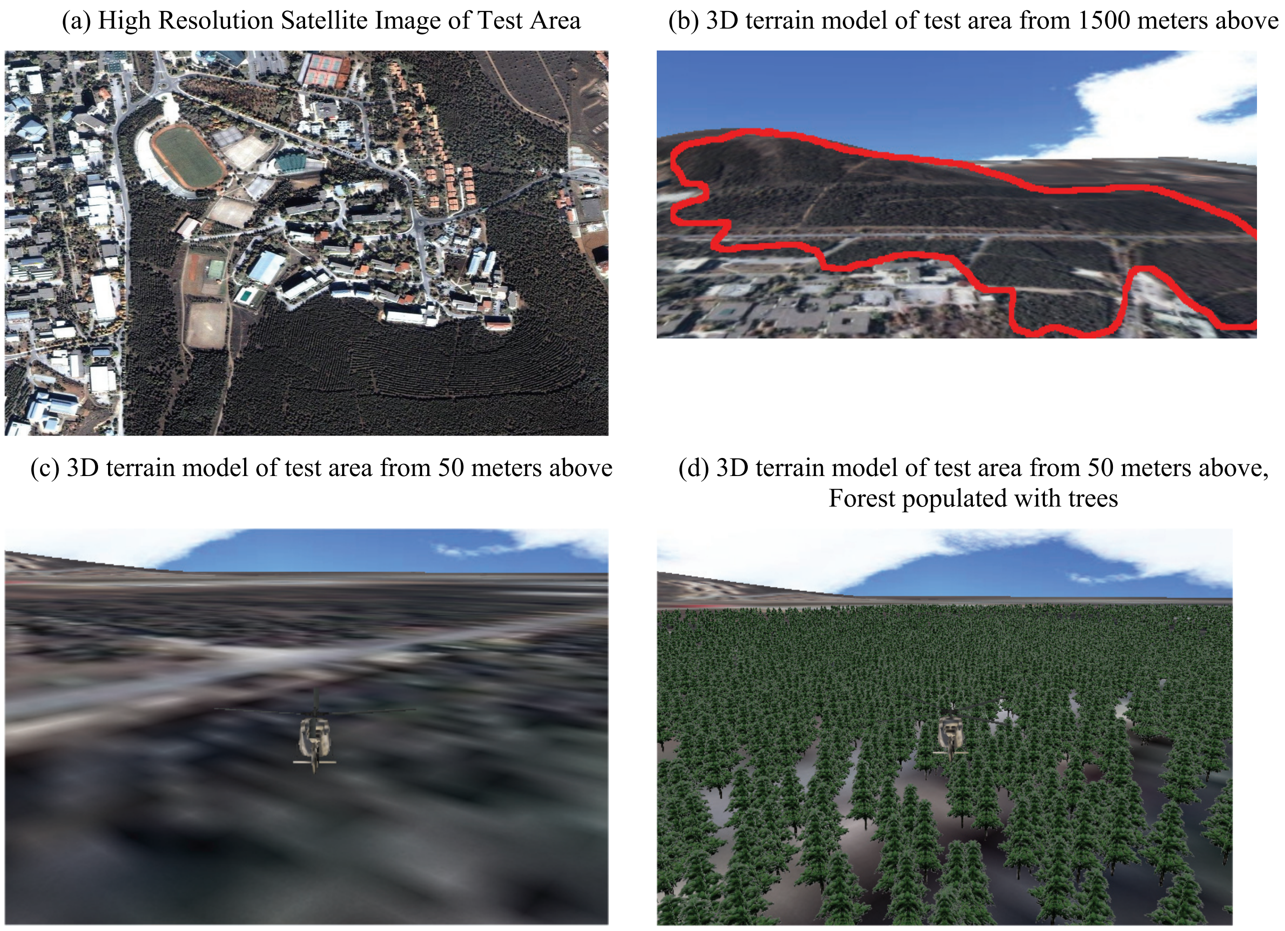
(a) High Resolution Satellite Image of Test Area
(b) 3D terrain model of test area from 1500 meters above
(c) 3D terrain model of test area from 50 meters above
(d) 3D terrain model of test area from 50 meters above, Forest populated with trees

Figure 6: The need for 3D Trees at low altitude camera views

Rendering of culture elements like buildings, roads and bridges is also necessary for achieving higher level of realism. At this point generic building models can be used to visualize thousands of buildings in a city but users want to see real replicas of well known buildings like temples, skyscrapers, malls, stadiums etc. In Figure 4 we can see some historical buildings in Istanbul. It is very difficult to port such structures to virtual worlds and there is no easy method introduced yet. In this study we only dealt with generic buildings and used model structure library that contains hundreds of generic structures. By using these models we succeeded to visualize small populated places like villages.

6. VEHICLES, ANIMALS AND HUMAN BEINGS

These elements are partially different from the above mentioned components. Except the vehicles in parking position or the guards in Buckingham Palace gates, it is necessary to reflect the movement in every animation frame. If we mention the higher level of realism we must be able to render the worst case scenes which contain massive crowds. Crowds are part of real life as seen on Figure 4a. It is possible to reconstruct very realistic virtual 3D model of the New Mosque in this image but it may not be enough to feel the users as they are in the real world unless the environment includes walking people, pigeons, vehicles etc. We successfully managed to render 10.000 moving characters in real time. 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; Baille-De Byl, 2004).

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

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. A typical walk sequence that consists of more than 20 animations is illustrated in Figure 7. In order to handle crowd animation LOD management and visibility culling techniques mentioned in terrain models are used.

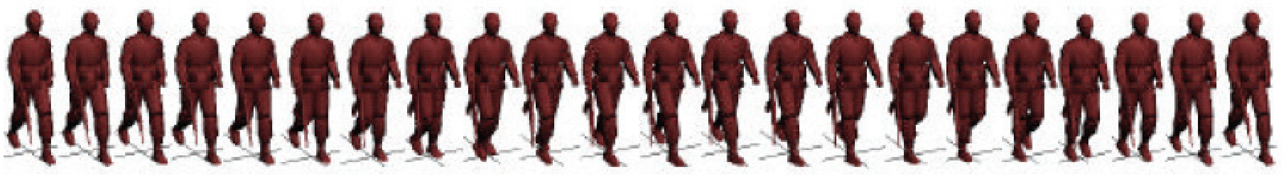
Figure 7: Walk Animation

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 (O'Sullivan et al., 2002). In order 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.
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.
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.

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

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

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

$\underline{N}$=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.

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

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

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.

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 virtual worlds 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 and 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. Our implementation is crowd animation rather than crowd simulation.

7. CONCLUSIONS

It seems that parallel to the huge advances in hardware the demand for better realism will continue to increase until the exact replica of the real life bring into our monitors. We can use our standard PCs till that day comes. In order to visualize very complicated outdoors scenes by using PCs we have to use software solutions mostly offered by computer games industry. Hierarchic organization of data such as cliché tree structures is a must when scenes get complicated. As a result we can also offer that we should prefer to render massive crowds or populated scenes by using simple techniques rather than trying to render very realistic single model if the aim is to visualize outdoors scene.

8. 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 3 June 2005).

Baille-De Byl, P., (2004), Meeus, J., (1991). "Programming Believable Characters for Computer Games". Charles River Media, Massachusetts, USA.

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", Proceed. IEEE Visualization '97, pp. 81-88.

Fritsch, D., Kada, M., (2004): "Visualization Using Game Engines", Proceed. XXth ISPRS Congress, Vol. XXXV, part B5, pp. 621-625.

Jakulin, A., (2000) "Interactive Vegetation Rendering with Slicing and Blending", EUROGRAPHICS, 2000. Available: URL: http://zeus.fri.uni-lj.si/~aleks/slicing-and-blending/trees-electronic.pdf, as accessed on 12 February 2003.

Jensen, H. W., Durand, F., Dorsey, J., Stark, M. M., Shirley, P. and Premoze, S., (2001): "A Physically-Based Night Sky Model", Proceed. 28th Annual Conference on Computer Graphics and Interactive Techniques, pp. 399-408

Lindstrom, P., Koller, D., Ribarsky, W., Hodges, F.L. and Faust, N., (1996). "Real-Time Continuous Level of Detail Rendering of Height Fields", Proceed. 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", Proceed. 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". Proceed. Third Irish Workshop on Computer Graphics, pp 15-20.

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

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

Yilmaz, E., Maras, H. H. and Cetin, Y. Y., (2004): "Photo-Realistic Scene Generation for PC-Based Real-Time Outdoor Virtual Reality Applications", Proceed. XXth ISPRS Congress, Vol. XXXV, part B5, pp. 615-620.