2011 International Conference on Computer Science and Information Technology (ICCSIT 2011)
IPCSIT vol. 51 (2012) © (2012) IACSIT Press, Singapore
DOI: 10.7763/IPCSIT.2012.V51.16

Designing of Interactive Stage Lighting Simulation

Yuhua Fang [a,*], Yinghua Shen [a], Chaohui L ü[a]

[a]School of Information Engineering Communication University of China, Beijing, China

Abstract. A method of using OGRE engine to simulate the realistic virtual stage lighting is presented. Through virtual stage modeling and rendering engine, a variety of stage effects and lighting effects are simulated realistically. The system provide a lot of stage operations, so that stage designers can design the stage lighting environment, adjust various lighting parameters, and add a variety of stage effects easily. The experimental results show that the proposed method can achieve various stage effects and lighting effects.

Keywords: OGRE, Stage rendering, Lighting effects

1. Introduction

Stage lighting Designers need to design and select the type of lighting, locate lighting arrangements, adjust the parameters of the lights and so on. A good simulation environment can help them a lot. The main contribution of this paper is to provide a solution to the problem of stage lighting simulation.

In this work two innovations have been made. One is to simulate the realistic virtual stage lighting effects, the other is to provide real-time interactivity. Using the OGRE engine [1], realistic three-dimensional stage lighting environment can be simulated more easily, convenient UI operations can make the system easy to use [2]. Extensive using of shader technology and material systems could help simulate a variety of lighting effects [3].

In the next section we will discuss the logical structure. In section 3 the organization and components of scene are presented, view space transformation and entity rotation are discussed, and we will also discuss implementation details of lighting effects, with results including scene, smog effect and lighting effect are presented in section 4. Conclusions are given in section 5.

2. System structure

Virtual scenes are often complex in order to guarantee realistic game experience and virtual reality. Virtual scene won't be constructed instantly under usual circumstance. The system is required to deal with high complexity, huge amount of calculation and good scalability. An overall implementation framework which can do calculation with high efficiency, good scalability and functional modules which have loose coupling and strong cohesion is needed, as is shown in Fig.1:

* Yuhua Fang. Tel.: 15101146202
E-mail address: fangyuhua16@gmail.com

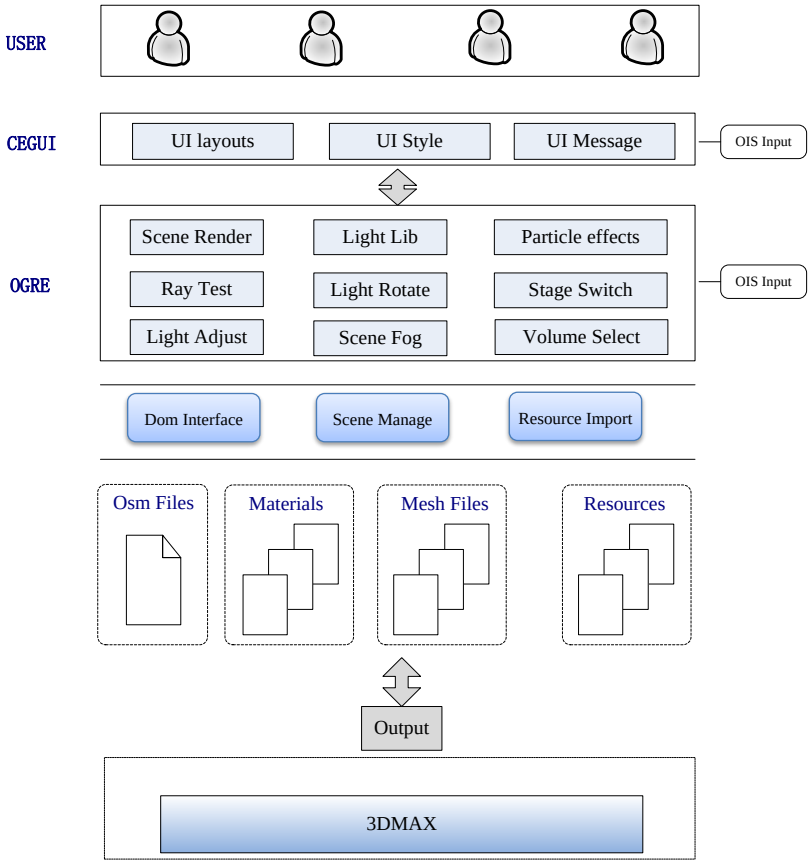

Fig. 1. System structure

The system includes resource layer, interface layer and rendering layer. Resource layer: imported by 3DMAX+OFUSION plug-in, it generates resource files like framework, material, substantiality and texture, etc. Interface layer: imports resource files into the scene. Rendering layer: simulates scene and lighting, and it also deals with operation of resource files and interactive real-time response.

3. Key technique of system

3.1. The organization and components of scene

Resource files are imported by 3DMAX and OFUSION plug-in. The organization files are in XML format, they record some basic parameters of the stage, the location and orientation of each stage entity.

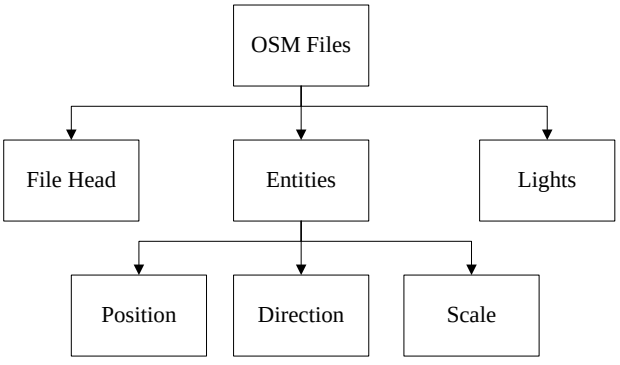

Fig. 2. The organizational chart of stage scene

These files are parsed through DOM interface, so OGRE can render each entity and assign corresponding parameters. The object to be rendered is usually managed by scene graph in OGRE. Scene graph can be used to retrieve and inquire objects, do collision detection and manage object bonding polygons.

3.2. Related operations on entity

3.2.1. Space transformation of entity view:

In world coordinate system, the view needs to be transformed, that is, converse the point defined in world coordinate system into view coordinate system.

To obtain the transformation matrix, four vectors including position vector, right vector, upper vector and fixation vector are defined as $p=\left(p_x, p_y, p_z\right)$, $r=\left(r_x, r_y, r_z\right)$, $u=\left(u_x, u_y, u_z\right)$ and $k=\left(k_x, k_y, k_z\right)$. The transformation matrix is $T*A$, in which T is translation matrix and A is rotation matrix.

Translation matrix: move the camera back to the origin. It completes the transformation of position vector, that is:

$$T=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -p_x & -p_y & -p_z & 1 \end{bmatrix} \tag{1}$$

Rotation matrix: make the lens toward Z axis direction. Here B is defined:

$$B=\begin{bmatrix} r_x & r_y & r_z \\ u_x & u_y & u_z \\ k_x & k_y & k_z \end{bmatrix} \tag{2}$$

The rotation matrix A needs to be obtained. Take $B*A$ as identity matrix, thus $A=B^{-1}$. Since Matrix B is a unit orthogonal matrix, so that $A=B^{-1}=B^T$.Matrix A are 3*3, and the entity view transformation matrix V can be obtained by doing an extension about matrix A:

$$V=T*A=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -p_x & -p_y & -p_z & 1 \end{bmatrix}*\begin{bmatrix} r_x & u_x & k_x & 0 \\ r_y & u_y & k_y & 0 \\ r_z & u_z & k_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

3.2.2. Space transformation of entity view

When an object moves in 3D space, the movement often includes rotation. A rotation in four-dimensional space can be expressed by a pair of unit quaternion, which is more compact and instant. In OGRE quaternion needs to be constructed according to axis and angle of rotation in OGRE. Here is an equation expressing a quaternion: $q=w+xi+yj+zk$, in which w stands for rotation, x, y, z rotation axis. When $w^2+x^2+y^2+z^2=1$, q is a unit quaternion.

In order to construct equivalent matrix of unit quaternion, the matrix as follow needs to be obtained:

$$R_q=\begin{bmatrix} 1-2\left(y^2+z^2\right) & 2\left(xy-wz\right) & 2\left(xz+wy\right) \\ 2\left(xy+wz\right) & 1-2\left(x^2+z^2\right) & 2\left(yz-wx\right) \\ 2\left(xz-wy\right) & 2\left(yz+wx\right) & 1-2\left(x^2+y^2\right) \end{bmatrix} \tag{4}$$

It's easy to prove that R_q is a unit matrix, and it can be used to rotate entities around any axis.

3.2.3. Simulation of lighting effect

Lighting is a key element of stage effect as well as the core technology of this design system [4]. Some special effects such as spotlight, should be realized. Spotlight needs more calculation than point source and parallel light. On one hand it attenuates with distance increment, on the other hand, the part on luminescent vertebral body needs to be told which bonds the vector from spotlight to the cone point. It means SportFactor inside and outside the vertebral need to be obtained. Given variable：

$$Rho=-p*S_{direction} \tag{5}$$

p is unit direction vector from the cone point to lighting source [5].

Then SportFactor can be obtained: in which phi and theta are angles inside and outside the vertebral.

$$SpotFactor = \left(\frac{\max\left(\left(Rho - \cos\left(phi/2 \right) \right), 0 \right)}{\cos\left(theta/2 \right) - \cos\left(phi/2 \right)} \right)^{falloff} \quad . (6)$$

Now the spotlight lighting equation can be obtained through attenuation factor transformation:

$$I = M_{emmissive} + M_{ambient} \otimes A_{glob} + \left(S_{ambient} + \max\left(N*P, 0 \right) * S_{diffuse} \otimes M_{diffuse} \right) * \frac{1}{Attention0 + Attention1*d + Attention2*d^2} * SpotFactor \quad . (7)$$

Here A_{glob} is global scene ambient, M and S are components of scene ambient of material and source, N is unit normal vector on the cone point, P is unit direction vector from lighting cone point to source, d is distance from the cone point to source which is used to calculate attenuation factor.

Lighting effect obtained by the formula makes the luminance on luminescent vertebral axis stronger, so lighting effect can be more vivid.

$$I_{tot} = \sum_{i=1}^{n} I_i \quad (8)$$

After several overlaps and radial blurs, more special lighting effects would be available [6].

4. Key technique of system

Fig.3 (a) is rendering of fountain effect. Fig.3 (b)is rendering of smog effect. Fig.4 is rendering of lighting effect. The system can do operations like switching stages, simulating different scene effects and other interactive real-time operations.

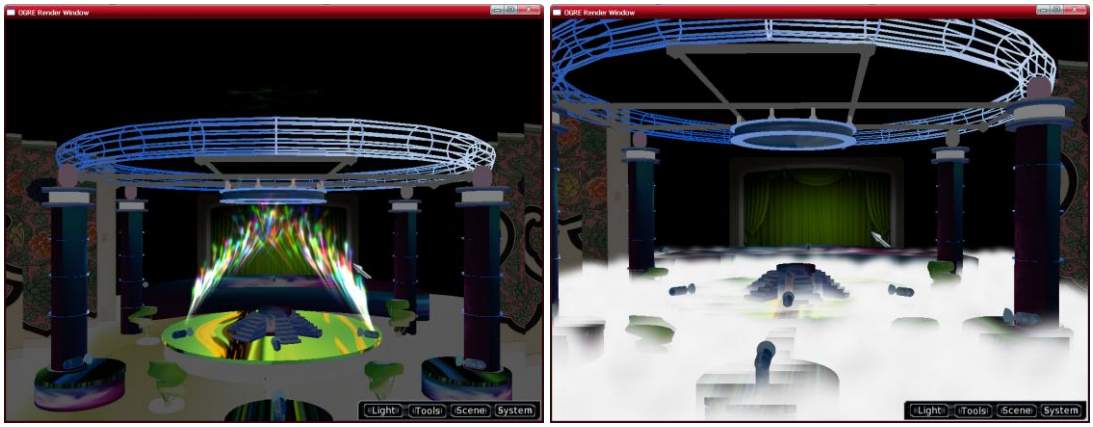

Fig. 3. (a) Fountain Effect; (b) Smog Effect

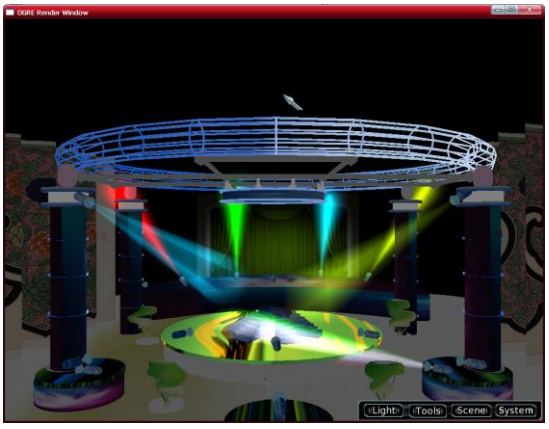

Fig. 4. Lighting Effect

5. Conclusions

Stage lighting design has been a major problem for lighting designers. They often face obstacles like high cost, high energy consumption and time-consuming quality. With booming information industry, professional stage lighting steps into full digital era. So our system is implemented to present real time design and adaption of stage lighting, and also a lot of interactive functions are provided. Next step, more lighting effects should be realized, such as back light, soft light etc, and the UI system should also be more and more friendly.

6. Acknowledgements

This work is supported by the National University Student Innovation Test Plan of Communication University of China (No.G2010033), the Science and Technology Innovation project of Ministry of Culture (No.2010-10), and the University Level Research project of Communication University of China (No.XNG1008).

7. References

[1] S Yan, D Xu, B Zhang, H. Zhang, "Graph embedding: A general framework for dimensionality reduction," IEEE Conference on Computer Vision and Pattern Recognition, 2005, pp.830-837.

[2] Wendy Jones, "Beginning DirectX9," MA. Boston: Prem ier Press, 2004.

[3] ZL Zheng, J Yang, "Supervised Locality Pursuit Embedding for Pattern Classification," Image and Vision Computing, 2006, 24,pp: 819- 826.

[4] Randima Fernando, "GPU Essence-Skills And Art of Real Time Image Programming," Beijing: Posts & Telecom Press 2006,pp:100-106.

[5] Tenghui Zhu, Xuehui Liu, Enhua Wu, "Lighting Calculating Skills Based on Pixel," Journal of Computer-Aided Design & Computer Graphics, 2002, 14(9),pp: 861-865.

[6] Tomas Akenine-Moller, Eric Haines, "Real Time Rendering," Beijing: Beijing University Press, 2004,pp: 20-21.