A Web3D-Based Method for Visualization of LED Lighting Environment

Jiankang Du, Jianqing Zhang and Lilan Liu*
Shanghai Key Lab of Intelligent Manufacturing and Robotics
Shanghai University
Shanghai,China
854624303@qq.com, georgezhang@51jiecai.com, *lancy@shu.edu.cn

***Abstract*—This paper is on the inquiry of the Web3D visualization methods for LED illumination environment. In recent years, the three dimensional visualization technology has been widely applied in many fields, such as city planning, engineering, medicine, education and so on. However, technology of LED visualization was mostly applied in LED lamps and lanterns, seldom applied in visual effect of the LED environment. This paper presents a sort of Web3D technology--WebGL, which combines great advantages in the GPU acceleration, platform crossing, inheriting from the simulation tool of last generation, running smoothly in many major browsers without any plug-in. Three.js is a perfect3D graphics engine standing out of numerous WebGL framework, having a lot of built-in light source and material. Meanwhile, it provides a lot of convenient function for this study. In addition, this research can be applied to the customization of LED lamps and lanterns, the design of the indoor environment, etc.**

Keywords—Visualization; LED lighting; Web3D; Three.js

I. INTRODUCTION

LED is widely used as a type of energy-saving illumination source. Before an LED lamp is produced, the effects of the lighting are unknown. Moreover, before an LED light is decorated in the wall, no one knows if it is appropriate or not. As for the importance of visualization of LED lighting environment, it is direct advice provided as your reference when people were about to choose or on the collocation of LED lamps and lanterns [1]. Also empirically speaking, 3D visual interfaces has been proven to be perfect and amazing. Meanwhile, it has been regarded as an important element to increase value to some applications such as entertainment, e-Learning, e-business, etc. Moreover, it can benefit a great deal from the Web domains of 3D visual interfaces [2].

The Web3D technology can be traced back to the earliest VRML (Virtual Reality Modeling Language) [3]. The association of VRML was renamed the Web3D association in 1998, completed the conversion from VRML to Extensible 3D language, and the first to use the word "Web3D". Now, there are dozens of realization of the Web3D technology solutions, relevant software more than 30, new technologies constantly innovated, which greatly improves the Web3D rendering speed, image quality and modeling technology, interactive and data compression and optimization. Many world-famous manufacturers have launched their own Web3D technologies, including the Cycore's Cult3D, Sun Microsystems' Java3D, Adobe's Atmosphere, Microsoft's Direct3D, Metastream's Viewpoint, etc. However, there are still many problems in 3D Web, such as network bandwidth, CPU speed, hardware acceleration (independent GPU for thinning quality, and improving fluency of the images). All above factors will influence actual operation quality and efficiency of the Web3D. The disunity of the 3D model file formats and the 3D rendering engines has been one of the obstacles in the development on Web application for Web3D. Because each Web3D technologies has its own characteristics and advantages, the competition situation will exist for a long time before the unified standard is available.

With the development of network technology and the definition of new standard about network, the integration of HTML5 and WebGL, makes full play to the advantages of the web as a 3D visual interface, establishing a good foundation for function extension of the Web3D applications. The browsers supporting WebGL program (writen with HTML and JavaScript similar languages) can not only render the 3D graphics faster, but also be superior in memory management [4]. As for HTML5 and WebGL integration of development pattern, it is much faster than traditional Web3D solution based plug-ins with VRML, Flash, Java, X3D or others. As to the difficulty of the development, it is enough for the designers and developers to handle a single language (JavaScript) and a HTML document object model (DOM).

Section II of this paper introduces the WebGL technology and the Three.js framework. Section III presents the framework of the project. Section IV illustrates the key factors influencing the LED virtual environment. Section V demonstrates the system with research data, and the verification of the experiment.

II. SYSTEM ARCHITECTURE

A. WebGL Technology

WebGL is a 3D drawing standard combined JavaScript and OpenGL ES 2.0, provides hardware accelerated for HTML5 canvas rendering, it can display 3D scenes and models smoothly through system Graphics Processing Unit, and can also create complex navigation and data visualization. The relationship between the WebGL internal elements is showed in Fig. 1.

WebGL perfectly solves the existing two problems of the Web3D virtual simulation: first, it completes the Web3D scene through HTML itself, without any support of browser plug-ins; Second, it relies on the underlying graphics hardware

Supported by Shanghai Economic and Information Technolgy Commission-funded project, No.15521103502.

acceleration for graphics rendering, through a unified, standard, platform crossing OpenGL interface implementation [5]. The drawing progress of WebGL is showed in Fig. 2.

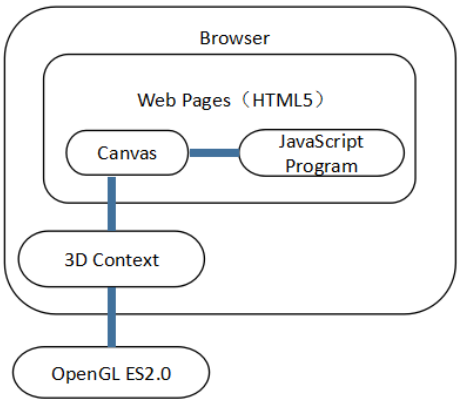

Fig. 1. The relationship between the WebGL internal elements

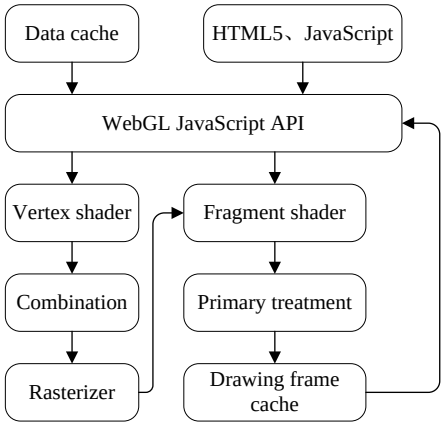

Fig. 2. Drawing progress of WebGL

B. *Three.js Framework*

Although WebGL allows us to show the complex and sophisticated 3D scenes, it is difficult to use the original WebGL API directly. Because we have to provide a vertex shader and fragment shader language (a kind of OpenGL shading language code, different from JavaScript). In particular, LED lighting environment must provide more details. Therefore, there are some utilities libraries which make WebGL more convenient, such as X3DOM, Scene.js, Babylon.js, etc. Each of them have advantages, but the most popular one is Three.js. We choose the Three.js to build our program.

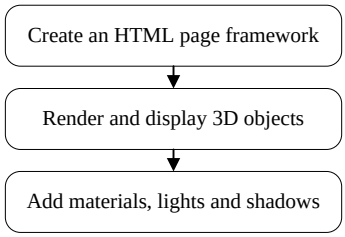

Fig. 3. The drawing progress of Three.js

Three.js in a simple, intuitive way to encapsulate 3D graphics programming in commonly used objects. It uses a lot of advanced techniques in the development of the graphics engine, greatly improving performance [6]. Further, since the common object and built a lot of easy-to-use tool, Three.js features are also very powerful. Finally, three.js is completely open source, thousands of developers on GitHub maintained in this framework together . The drawing progress of Three.js is showed in Fig.3.

III. PROGRAM FORMULATION

A. *HTML Framework*

To start this project, we need to create an HTML page framework, this is the first step in the LED lighting environment visualization. There are a lot of widely used HTML framework, you can choose the suitable project according to your need. We didn't introduce any one here, one simple HTML page framework is as follows:

```
<!DOCTYPE html>
<html>
<head>
<title>Example</title>
<script type=”text/javascript” src=”../libs/three.js”>
</script>
<style>
body{
margin: 0;
overflow:hidden;
}
</style>
</head>
<body>
<div id=”WebGL”></div>
<script type=”text/javascript”></script>
</body>
</html>
```

B. *Build a scene*

First, we should find the page elements for WebGL rendering, and stored in the variable of container. And initialize Three.js renderer object (which is responsible for all the drawing work of Three.js), then we can add it to the container as a DOM element. Then, we created a scene, which is the highest in the Three.js objects, used to hold all of the other graphic objects. When a scene is built, we may add other objects: a camera and a cube. The camera defines the position and the angle we are observing in the scene. Finally, by calling the renderer method “render()”, we render the whole scene.

```
var container = document.getElementById(“WebGL”);
var renderer = new THREE.WebGLRenderer();
```

```
container.appendChild(renderer.domElment);
var scene = new THREE.Scene();
var camera = new THREE.PerspectiveCamera(60, window.innerWidth/window.innerHeight, 1, 10000);
camera.position.set(10, 10, 0);
scene.add(camera);
var geometry = new THREE.BoxGeometry(1, 1, 1);
var material = new THREE.MeshMaterial({color:0xff0000});
var cube = new THREE.Mesh(geometry, material);
scene.add(cube);
renderer.render(scene, camera);
```

C. Add a model

Thee.js can read a variety of 3D file formats, such as JSON, Collada, STL, import the combination and the grid of them. OBJ is a simple 3D file format, created by the WaveFront technology company, it supports straight Line, Polygon, Surface and Free form Curve, has been the most widely used 3D file format.

There are a lot of 3 software tools can be used to create OBJD format files, such as Blender, Maya, and 3 Studio Max, etc. Three.js provides the OBJ format file loader called OBJLoader. we load a model from a URL through OBJLoader class in the following code:

```
var loader = new THREE.OBJLoader();
loader.load('../models/model1.obj', function(geometry){
var material = new THREE.MeshLambertMaterial({
color: 0x235A47
});
geometry.childern.forEach(function(child){
child.children.material = material;
});
scene.add(geometry);
});
```

IV. LED Lighting Environment

LED lighting environment visualization needs consideration of the characteristics of the LED light source, as well as the environmental performance of the object.

A. Light

There are two main types of light in the LED application environment: Directional light, which is similar to the nature of the sun; Point light, which is similar to the LED light. Moreover, ambient light is used to simulate non-direct light in the real world.

B. Materials

Materials determines the geometry appearance of objects. Three.js has many built-in materials. For example, MeshBasicMaterial does not consider the effect of light, which is suitable for flat object; MeshDepthMaterial enables an effect of gradual disappearance, which is determined by the distance between the camera and the object; MeshNormalMaterial can make colorful effect, because the color of this kind of material refers to the outward normal vector calculated from the surface of the object, giving each subtle surface a slightly different color.

C. Textures

The surface of an object is always not smooth in the lighting environment. A material called THREE.ShaderMaterial is used to present a realistic scene. This material allows developers to customize a shader code for commonly visual effects directly. Under normal circumstances, in order to realize the vivid picture, including bump and specular highlights, we will use three kinds of texture map: a color map (diffuse map) the map provides the closest to the real color of the pixel; A normal map (bump map), it is essentially an additional grid attribute coding into RGB values stored in a bitmap file, normal determines the brightness of the light to the grid surface, brighter or darker areas depends on the normal value. Relative to defining additional vertex attributes, the normal can show rough grid surface using less resource consumption at a relatively high performance; Finally, it is a specular highlights or specular reflection map which refers to the reflection of the surface of the grid and reflective. Similar to normal mapping, specular mapping is an effective way of data storage. A bright RGB value indicates the areas of high luminosity, darker RGB value indicates the luminosity to low areas.

V. Evaluation

The system constructed by the 3D models and the optical models of the LED lighting environment, and was expressed in the browser. In order to verify the rationality of the simulation, we used the DIALux, an excellent optical software, which can calculate optical data according to optical scene [7]. Models built in the DIALux are as shown in the fig.s below. Fig. 4 is an aerial view on the left side of the whole 3D scene.

Fig. 4. 3D Rendering

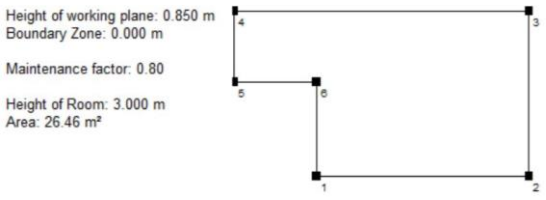

Fig. 5. Input protocol

Fig. 6. Luminaire parts list

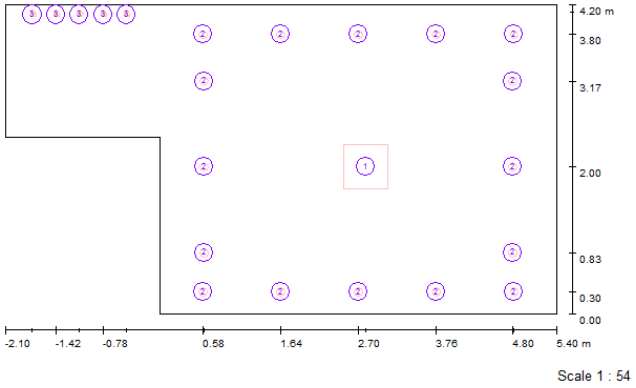

Fig. 7. Luminaries (layout plan)

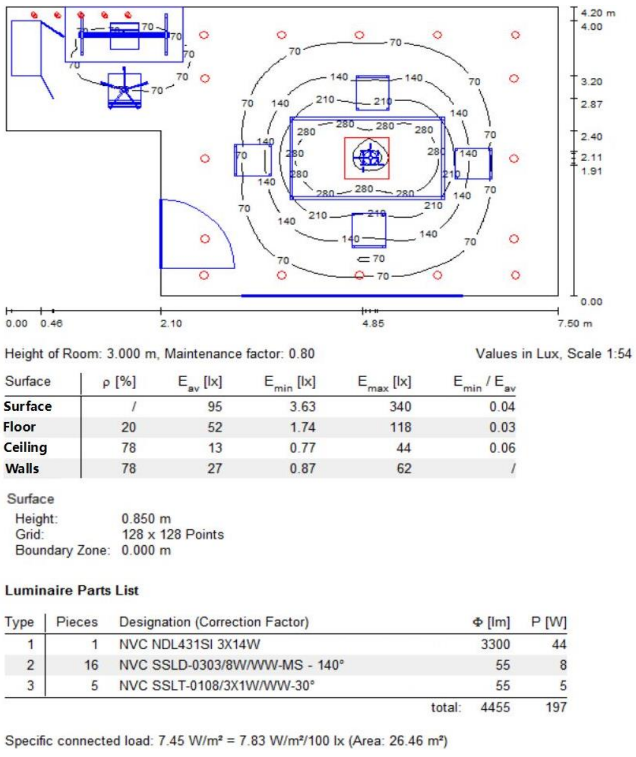

Fig. 8. Analysis report

DIALux can help designers to determine the lighting energy consumption according to DIN V18599 criteria. Characteristics and all other parameters, also the framework of the standard specified value have been deposited in the DIALux, ready for the user at any time [8]. In lighting design inputing information, such as space shape, shape of glass Windows, skylights, composite ceiling, lamps and lanterns, etc., will be directly delivered into energy evaluation by DIALux, and then used for automatical calculation.

For details, fig. 5 is about the size of the space, fig. 6 is the information graph of the LED lamps and lanterns used in the scene, fig. 7 is the location map of the LED lamps and lanterns. fig. 8 is the illumination analysis of the working intensity of general layout. According to the analysis report, the intensity of illumination is in line with the total standard, and the design of LED lighting environment is reasonable.

VI. Conclusion

This paper introduces a method of LED light environment visualization, which can not only display the scene directly through the browser without any plug-ins, but also accelerate rendering engine using the GPU. WebGL is a new kind of technology in recent years, received much attention and support, and has great potential development. Combing WebGL technology and visualization, has realized the cross-platform and real-time performance, which is a real meaning breakthrough in this field. In the development of the virtual light environment, there are many methods in handling light sources, materials and textures, it will take great effort to find the best ones. However, the foundation of our research is still at the primary stage, unable to deal with complex visual requirements and high quality visual effect. In the deeper study, the LED light environment visualization should be subdivided, details inside will be modularized, which may greatly improve the usability and stability of the system.

Acknowledgment

The authors would like to thank Shanghai Key Lab of Intelligent Manufacturing and Robotics and Shanghai University for their valuable support and computing resources.

References

[1] Zheng, Jun, and J. Jian. "LED Lighting Simulation Control System Design of Highway Tunnel." Transportation Science & Technology (2015).

[2] Bochicchio, M. A., A. Longo, and L. Vaira. "Extending Web applications with 3D features." (2011):93-96.

[3] Ziwar, F., and R. Elias. "VRML to WebGL Web-based converter application." International Conference on Engineering and Technology IEEE, 2014:1-6.

[4] Khronos Group. WebGL - OpenGL ES 2.0 for the Web. http://www.khronos.org/webgl/ Restrieved 20 May 2016.

[5] Weigang, et al. "A WebGL-based method for visualization of intelligent grid." Electric Utility Deregulation and Restructuring and Power Technologies (DRPT), 2011 4th International Conference on IEEE, 2011:1546 - 1548.

[6] Dirksen J. Learning Three.js: The Javascript 3D Library for WebGL[J]. 2013.

[7] Aman, M. M., et al. "Analysis of the performance of domestic lighting lamps." Energy Policy 52.3(2013):482-500.

[8] Acosta, Ignacio, J. Navarro, and J. J. Sendra. "Towards an Analysis of Daylighting Simulation Software." Energies 4.12(2011):1010-1024.