«**Real-Time Rendering**

**with DirectX 12**»

By

XingKui Liu

Advisor

WenLai Ye

in the

Computer Engineering College

JiMei University

February 2019

**CONTENTS**

Abstract …………………………………………………… 2

Keywords ……………………………………………….… 2

1 Introductions ……………………...……………….…… 2

2 Convention ………………………………………........... 3

1.1 Mathematical notations …………………………... 3

1.2 Miscellaneous …………………………………….... 3

3 DirectX 12 ………………………………………………... 4

4 MODELS and VERTICES ………………………………. 5

5 LIGHTING …………………………………………...…... 6

5.1 Lighting quantities ………………………………… 6

5.2 Lighting equation …………………………………… 7

5.2.1 Ambient …………………………………………. 8

5.2.2 Diffuse …………………………………………… 9

5.2.3 Direct specular …………………………………. 10

5.2.3.1 Blinn-Phong ……………………………… 12

5.2.3.2 Cook-Torrance …………………………… 13

5.2.4 Indirect Specular ……………………………… 15

5.3 Reflection and Refraction …………………………….17

5.4 The result ………………………………………………17

6 SHADOW ……………………………………………………18

7 FUTURE WORK …………………………………………….19

8 References ……………………………………………………20

**ABSTRACT**

This paper is focused on Real-Time Rendering in the context of computer algorithm. Talking mainly about Rendering equations. We provide a review and explain lighting models which are prevalent. Implementing some technologies with DirectX 12.

**Keywords**

Computer Graphics, Real-Time Rendering, Lighting, Physical Based Rendering

**1.INTRODUCTION**

The Real-Time Rendering is a field of CG (Computer Graphics). CG is aimed at synthetic virtual images. Rendering in real time normally means producing three-dimensional images [1]. Nowadays, Rendering is divided into Real-Time, which renders images in seconds (e.g. 24 frames per second), and Off-Line which makes an image that costs a lot of time. Off-Line Rendering in film industry is widely used, because very complex calculations are present and have to run the synthetic process in some cluster of workstations. Even though costing a lot of time and money, images’ qualities usually very high. Real-Time relative to Off-Line trade-off between time and results. It’s a typical thought that is in rapidly making a series of wonderful images. There are many theories and methods which are adopted in Off-Line use in Real-Time. They may be reduced to a new one, simpler and more computable. We will only discuss Real-Time Rendering in this paper. **2.CONVENTION**

**2.1 Mathematical notations**

With homogeneous coordinates, we augment to 4-tuples and what we place in the fourth w-coordinate depends on whether we are describing a point or vector [2]. (x, y, z, 0) for vectors and (x, y, z, 1) for points (be all denoted with ).

(1)

represents a multiply of component-wise (e.g.).

**2.2 Miscellaneous**

All matrix has default 4x4 format. Vectors are indicated as

1x4 Matrix.

(2)

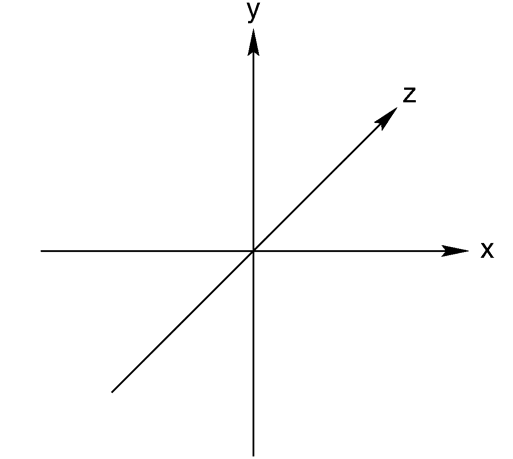
Row-major order of matrix is taken in Direct3D, that means in the multiply that vectors are followed by matrixes (i.e. ). This paper uses a so-called left-handed coordinate system; see Figure 1.

Figure 1. Left-handed coordinate

**3.DirectX 12**

DirectX plays a mainly role as a communicator between applications and hardware (e.g. Graphics card). It’s well known as Graphics APIs (application programming interfaces), and OpenGL, WebGL, Vulkan are similar to this one; Figure 2 has described that the timeline of graphics APIs and graphics hardware releases. The predominant consumer of Direct3D is the games industry, where higher level rendering engines are built on top of Direct3D. However, other industries need high performance interactive 3D graphics as well, such as medical and scientific visualization and architectural walkthrough. DirectX 12 has much improved in multi-thread and has more low-level controls which close to hardware.

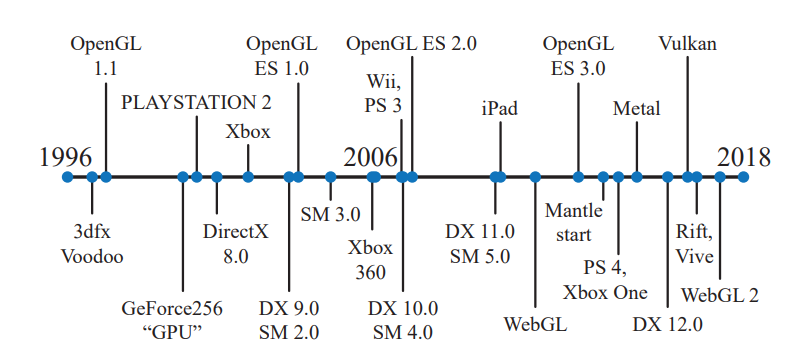


Figure 2. A timeline of APIs and graphics hardware releases. [2]

**4.MODELS and VERTICES**

No matter what 3D DCC (Digital Content Creation) applications are, 3D models are must primary. Assimp, a prevalent model-loader, is used. It is based on C++ and can read much model format, such as: .obj, .fbx, .ply etc. The model file mainly has vertices which have useful properties.

Vertices properties used in this paper:

1 Position in three- dimension space.

2 Normal attach each vertex.

3 UV—texture coordinates

4 Tangent attach each vertex.

Some properties maybe don’t include in model file, and so we would generate them via codes.

**5.LIGHTING**

A most important, but also much difficult, factor is lighting. In the real-world, light is a big clue to perceive other things.

You can image that entering a cave, and you are in the pitch dark. In contrast, dim light can illuminate shapes of things.

In computer graphics, the research of lighting has a very long history. There have two lighting models in real-time rendering: local illumination models, global illumination models. With a local model, we can ignore light that bounced off other objects (i.e. not emitted from the light source directly, so-called indirect-light); a global model considers all of them, direct-light and indirect-light. We only talk about local illumination models.

**5.1 Lighting quantities**

Many basic quantities in lighting model had been raised in previous decades. Light-Surface intersection position**,** Normal, LightVector, ViewVector , ReflectVector, HalfVector, Normal- LightVector intersection angle **,** Normal- ReflectVector intersection angle ; Figure 3 describes these quantities.

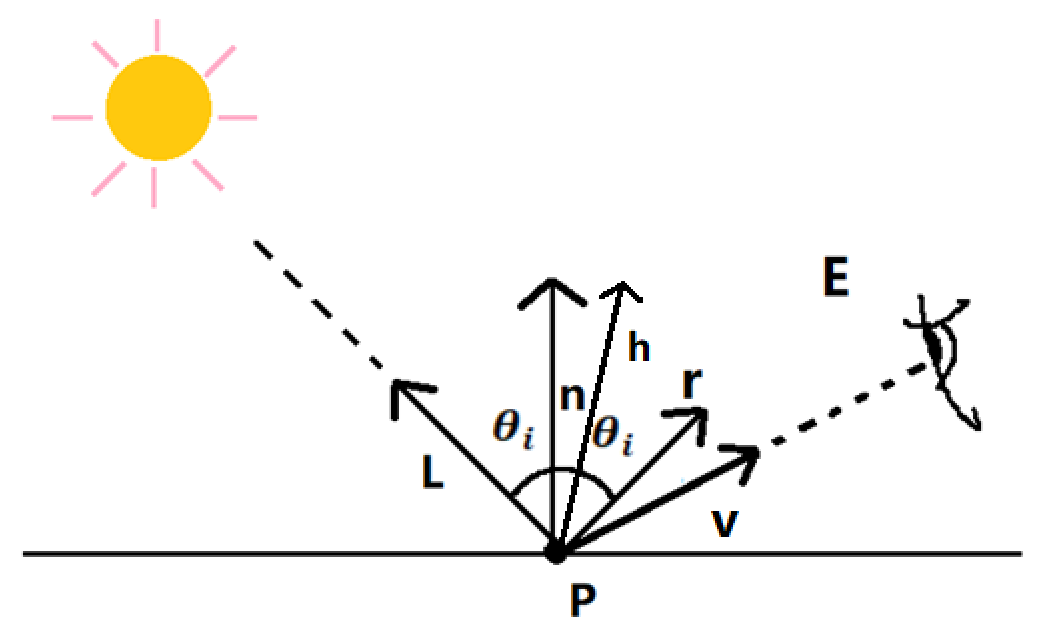


Figure 3. Basic lighting quantities

**5.2 Lighting equation**

As a matter of fact, lighting is the sum of all light’s components. To calculate final colors, we must construct an equation of lighting (a. k. a “lighting models”). The intensity of reflected light at a point on a surface is calculated for red, green, and blue wavelengths simultaneously [4]. We define to stand for colors. The total light reflected off a surface is a sum of ambient light reflectance, diffuse, indirect specular and direct specular.

(3)

**5.2.1** **Ambient**

Ambient light present low-intensity light which reflects off from other surfaces. In real-world, even though there not are punctual lighting sources, our eyes can percept things around us. That is because sun or moon light would emit light to surfaces which will reflect light. For simplicity, a low-intensity constant color will become ambient light.

(4)

Figure 4 shows a scene that only has ambient light.



Figure 4. An ambient only scene

**5.2.2 Diffuse**

Light emits from source and strike many surfaces. Some of them transmission into surface. The transmitted light undergoes scattering and absorption events until finally some of it is re-emitted back from the surface [5]. See Figure 9. The simplest formula of diffuse is Lambert’ law (is incident light’s color, is the surface’s diffuse color):

(5)

See Figure 5 that how this formula can affect the scene. (This formula was extended, in order to use image texturing.)

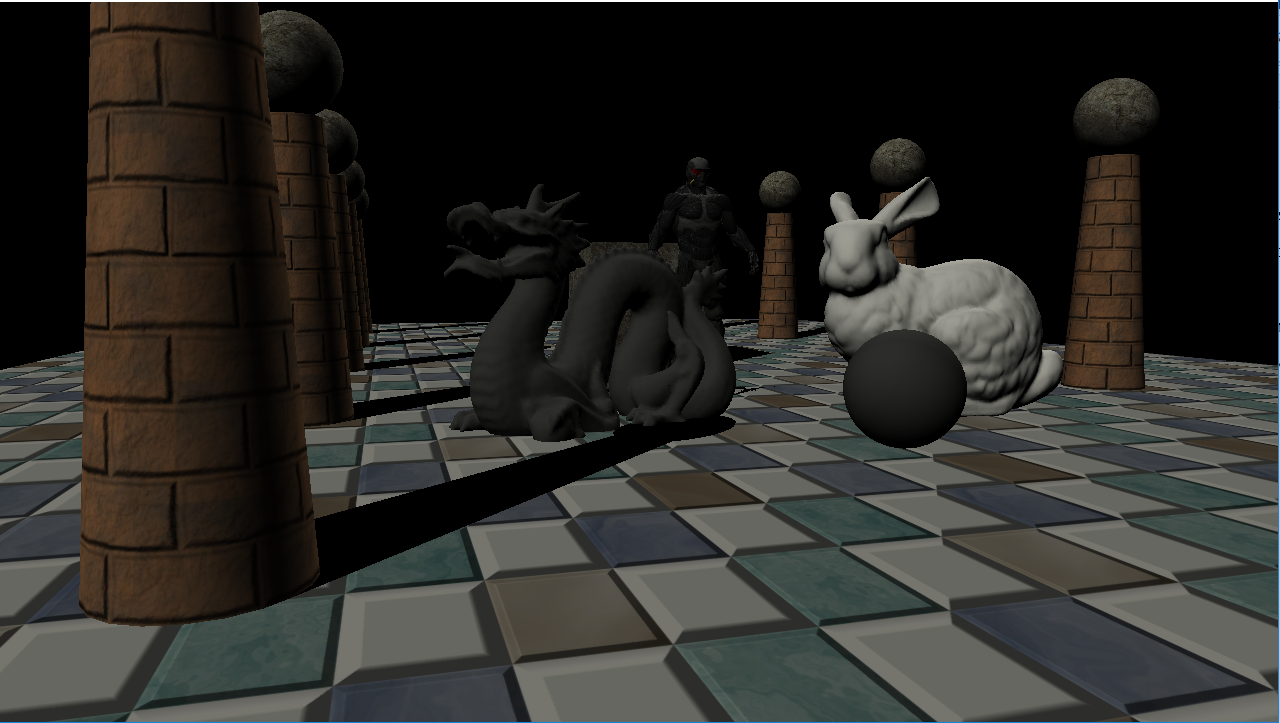


Figure 5. A diffuse only scene

**5.2.3 Direct specular**

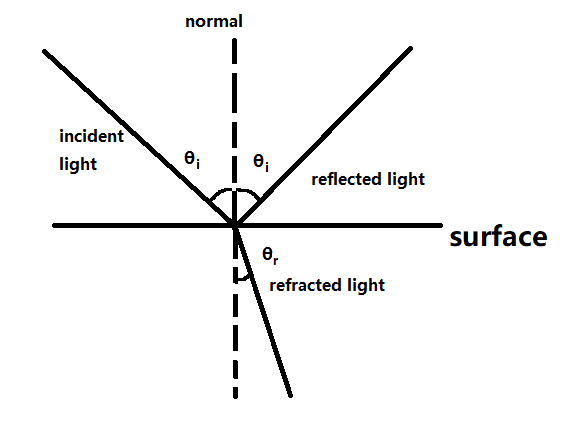
As we known, light which hits the surface might reflect along surface normal, that’s described in Figure 6, also with refraction. In real life, there is a physical phenomenon that decides how light and surface react is so-called Fresnel effect. For example, you stand by a calm lake and look down, your sight can under the lake. On the other hand, taking your sight far from you, then you can’t see anything under the lake. Check for this with Figure 7. 

Figure 6. Light reacts with surface



Figure 7. Fresnel effect

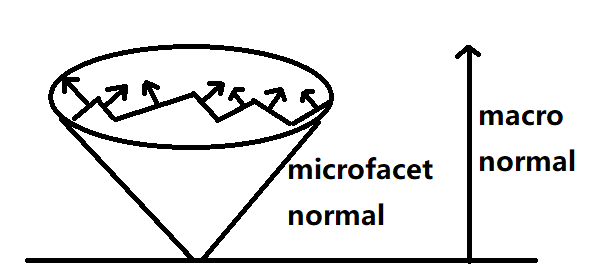
The Fresnel effect calculates how many lights reflect and other refract. There has another quantity called roughness which describe the fitness of the surface’s macroscopic normal vector and the microfacet’s normal vector. For actually, even though a surface in the life looks so flatten, it combines many micro surface (In Cook-Torrance theory, called that microfacets [3]); See Figure 8. So, when we calculate lighting equation, there is a matter that taking the macro or micro normal vector. 

Figure 8. Macroscopic normal vector and microfacet normal vector

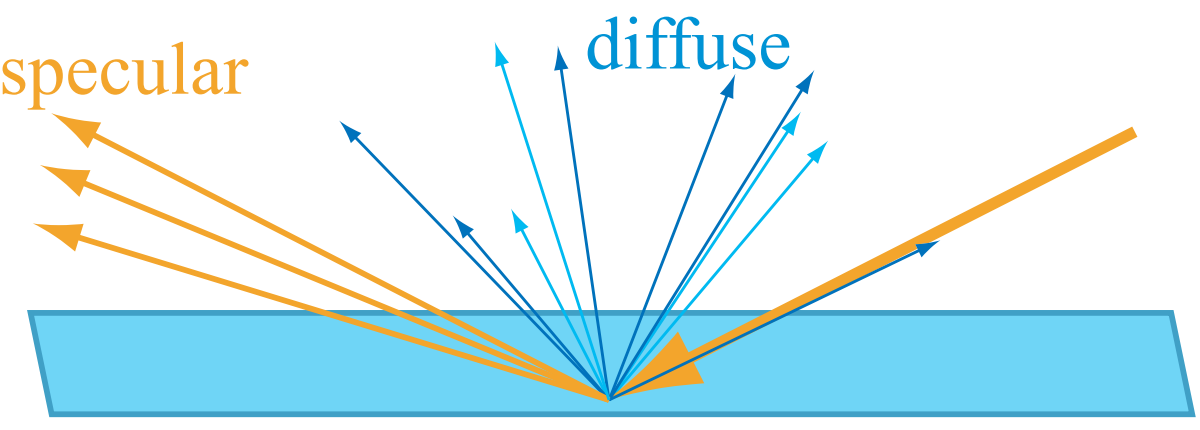


Figure 9. Light hits surface, generate diffuse and specular

**5.2.3.1 Blinn-Phong**

In previous decades, the most popular specular BRDF is Blinn-Phong. It’s just an empirical model, that is non-physically based.

(6)

Let be the angle between the light vector and half vector , then tells us the amount of light reflected about into due to the Fresnel effect, represent for roughness. The original Fresnel formula is very complicated. Instead of that, Schlick [6] propose an approximate one which consumes less computing power and basically closes to the original effect.

(7)

The reason why use is that according with energy conservation [7]. The result of Blinn-Phong with energy conservation and no energy conservation can see Figure 10.

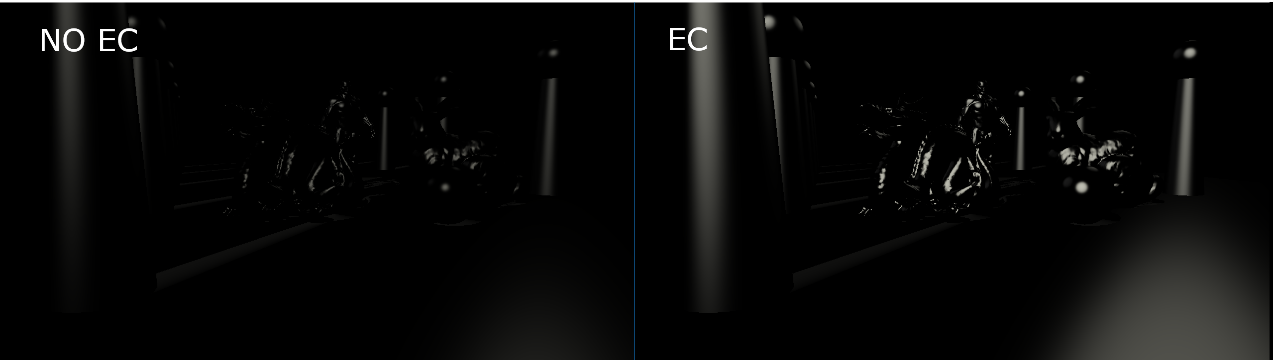


Figure 10. No energy conservation(left) and energy conservation(right) with Blinn-Phong

**5.2.3.2 Cook-Torrance**

There is a popular specular BRDF physically-based model called Cook-Torrance [3].

(8)

(9)

(10)

(11)

(12)

In here, we use Schlick approximate for (Fresnel). and have a derivative relation. The lights reflect from surfaces that may be occluded by other surfaces. There two functions are responsible for estimating how many lights are occluded. Figure 11 see this situation.

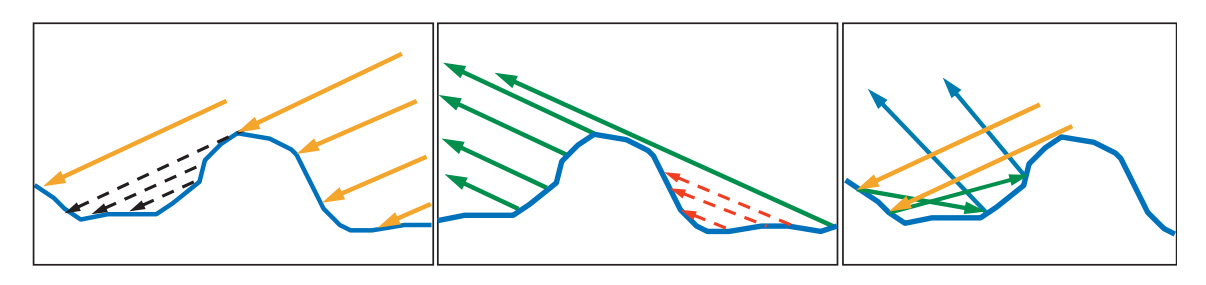


Figure 11. Microgeometry

We use GGX for (i.e. Distribution Term), Schlick geometry for (i.e. Geometry Term). It is worth mentioning, there are so many BRDF except Cook-Torrance and other implements for [8]. The effect of this implement that we can see Figure 12.

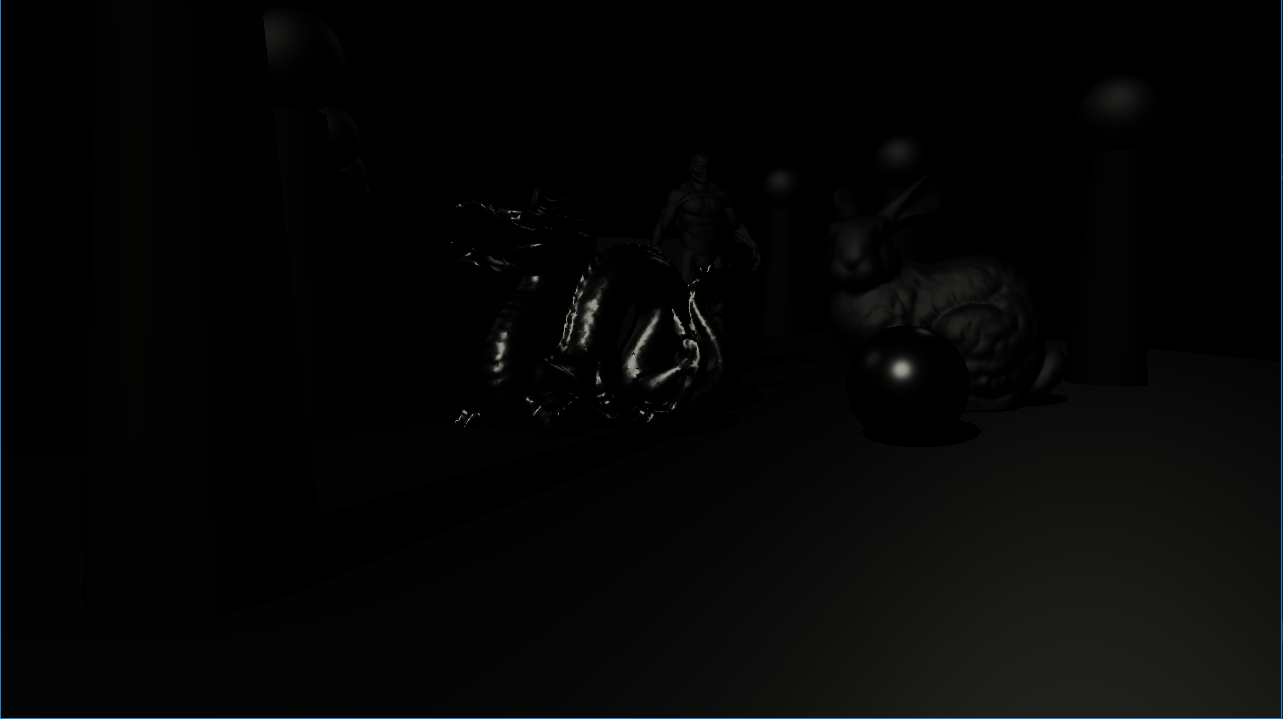


Figure 12. Cook-Torrance Specular BRDF

**5.2.4 Indirect Specular**

We use environment mapping to simulate the indirect specular. The environment mapping is a kind of cube mapping which is a type of three-dimension texture. For these distant things (e.g. sky, mountains), we can see them but can’t go there. So, this trick can improve the realistic of results, as we known “Sky-Box”. Environment mapping use textures that shows in Figure 13.

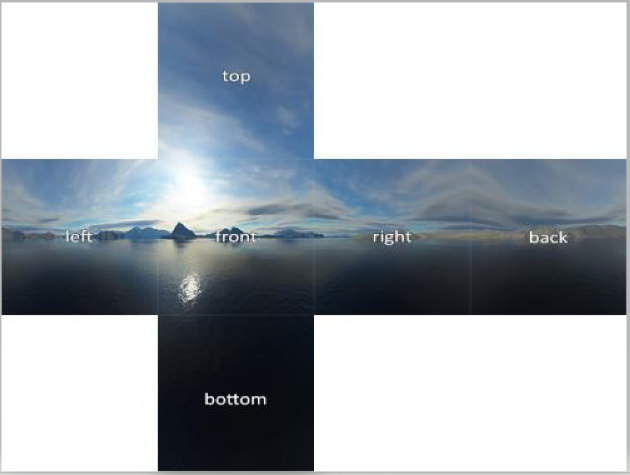


Figure 13. Environment mapping for Sky-Box

We use the reflect vector to query pixel of color in the specified cube map, and assigning them to our scene as indirect specular color. See Figure 14.

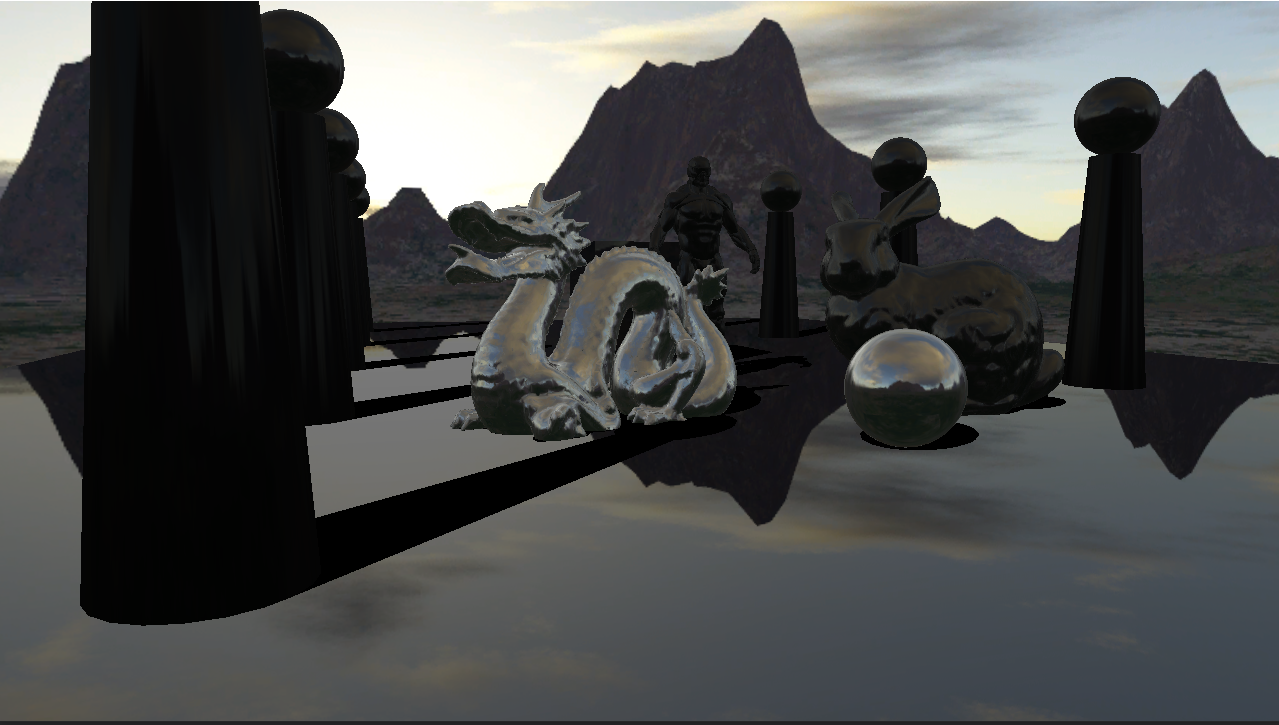


Figure 14. Sky-Box with indirect specular light

**5.3 Reflection and Refraction**

In the field of physical optics, because of different IOR (i.e. Index of Refraction) astride a surface, the light reflection and refraction. We can use cube map and the vector of reflection and refraction to simulation them. See Figure 15.

Figure 15. Reflection and Refraction

**5.4 The result**

As a matter of fact, the lighting simulation is analyzing the components of practical lighting. Allow for everything is this scene has different roughness, I modify the lighting result formula.

(13)

The rougher surface will blur the specular component and augment the diffuse component, vice versa. Figure 16 shows the final rendering results.

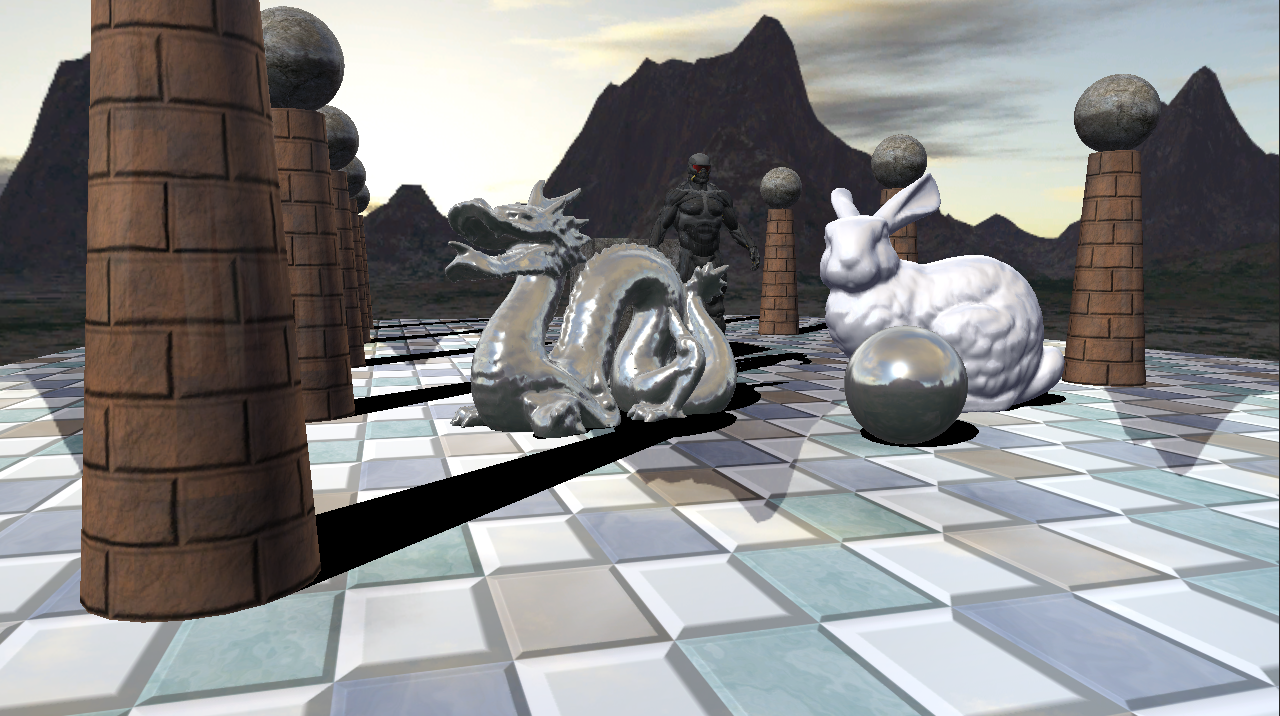


Figure 16. The final rendering results

**6 SHADOW**

Shadows are important for creating realistic images and in providing the user with visual cues about object placement References [1]. There are many prevalent methods to produce shadow. To demonstrate, here we take the simplest version of the shadowing, planar shadow. Along the light’s direction, projecting models’ vertices into the specified plane and shade them with shadowing color. This technology is illustrated in Figure 17.

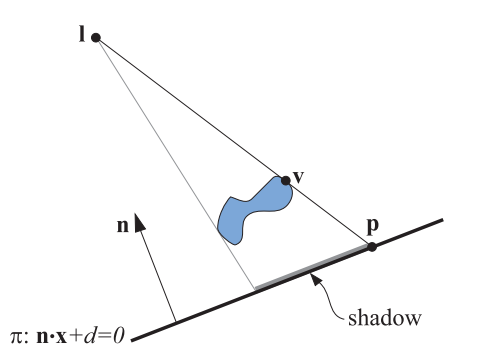


Figure 17. The shadow is being cast onto a plane, *π*: ***n · x*** + d = 0

In order for calculating vertex ***p*** in plane *π*, this yield:

(14)

**7 FUTURE WORK**

To display a realistic synthetic image, we must analyze the component of real-world. In practice, these methods are used in real-time rendering don’t follow the laws of physics, and many of them are just tricks to achieve the expected results. Nvidia just had launched new graphic cards which carried ray tracing core. It allows developers to write ray tracing programs and run them in real-time. To known much more about these, See Nvidia’s the new book [9].

**8 References**

[1] Tomas Akenine-Möller, Eric Haines, and Natty Hoffman. Real-Time Rendering 4rd Edition. A K Peters/CRC Press, USA, 2018.

[2] Frank D. Luna. Introduction To 3D Game Programming with DirectX 12. Mercury learning and Information, 2016.

[3] Robert L. Cook, Kenneth E. Torrance. A Reflectance Graphics Model for Computer. ACM Transactions on Graphics, Vol. 1, No. 1, January 1982

[4] Eric Lengyel. Mathematics for 3D Game Programming and Computer Graphics Third Edition. Course Technology, USA,2012.

[5] Naty Hoffman. Background: Physics and Math of

Shading. ACM Siggraph 2013.

[6] Christophe Schlick. An Inexpensive BRDF Model for Physically-based Rendering. France,1994.

[7] Eric P. Lafortune, Yves D. Willems. Using the modified Phong reflectance model for PBR. 1994

[8] Rosana Montes, Carlos Ureña. An Overview of BRDF Models. University of Granada, Granada, Spain. 2012

[9] Tomas Akenine-Möller, Eric Haines. Ray Tracing Gems. NVIDIA, 2019.