Hello everyone, today we want to share our group work about real-time rendering in computer graphics. Since we have a lot to share, we assume you have enough basic knowledge.

So, what do we want to do? You know many fantastic video games worldwide, and the image quality and fluency significantly impact players. Furthermore, according to this project, we also want to understand computer graphics better. Anyway, today we will introduce some technology, including lighting and shadow.

The first thing is the Phong lighting model, and the fundamental idea is supposed the surface of an object is perfect and uses some basic lighting equation. But around 2010, physically based rendering became popular in video games. This idea emphasizes using real-world physics to simulate and render.

People used the microfacet surface model in lighting to simulate the object's surface roughness. Also, natural radiation was thought to be needed to describe the energy of light, called energy conserving. The metallic represented the proportion of metal in the material. As you know, metal materials have no diffuse reflection. You can see that those renderings have a great effect.

Here are some comparisons between Phong and PBR. Why do people prefer to use PBR nowadays? An essential reason is that TA, the technical artist, won't need to tune parameters! They can directly use real-world parameters and quickly get amazing pictures!

Here are some formula derivations, but we won't discuss the details. We only need to calculate this integration. In 2003, an epoch-making paper was published. They were using Spherical Harmonics to estimate this integration. In recent years, thanks to the rapid development of GPU, Unreal Engine 4 gives another approximation. This method is high-speed, and the photos shown before result from our reproduction.

The second thing we want to share is shadow. The traditional shadow technique was shadow map. It has many bugs, like shadow acne, peter panning, and it needs over-sampling to improve quality. Also, it doesn't create a soft shadow! Recall your life experiences. All the shadows you have seen are soft! People improved it by using Percentage-Closer Soft Shadow (Based on PCF) and Variance Soft Shadow Mapping (Faster PCSS).

Nowadays, a new technique has become popular. It is called Signed Distance Field Soft Shadow. We start from each shader point, emit a ray to the light point, and go through this step by step. The SDF can tell us there cannot be any occlusion inside these circles. So, an intuitive way is that the smallest theta describes the visibility of this shader point. But it is too difficult to reproduce! Many details were not shared with the public, and we can't read the source code of UE4. But we tried our best to do it.

This photo should be the correct one.

One is that we also need to draw a sphere at the light point. More than just on the path is required. Otherwise, you can see this.

Another thing is that the generation of the Signed Distance Field is tricky. Just a 64 times 64 times 64 pixels SDF is already very huge. But if we use some approximation, you can see this.

Also, why UE won't use vectors to describe distance? In this way, we can interpolate better approximations. Also, this whole method doesn't require much floating-point precision and easily achieves the same memory storage. We also apply this, and you can see a bit of improvement in the left one.

So, in our group work, we explored the real-time rendering technique and studied how to render more realistic lighting and shadows. Also, we raised some questions and made a little progress. The whole code is written by ourselves using OpenGL. This project is public on GitHub if you are interested.

Here are our references.

Also, we want to share a picture rendered by ourselves. Seems cool, right?