Principles of Rendering: Real-time rendering

Callum Glover*
National Centre for Computer Animation

Abstract

The purpose of this report is to document how I utilised GLSL through C++ to create a photorealistic shaded model of a Sandisk USB memory stick, using my previous work with OSL and Renderman as a basis to improve and iterate upon.

1 Introduction

Figure 1: *The original reference image.*

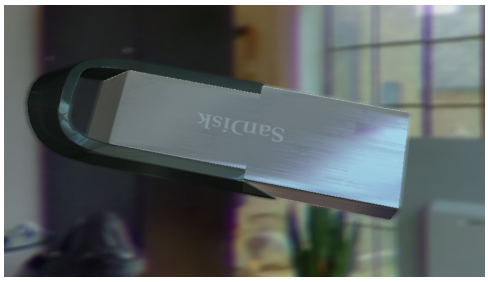

Figure 2: *The final rendered image.*

By analysing the object, the shader would break into the following properties:

1. The main metal surfaces have brushed marks, resulting in varying roughness over the surface and also small differences in the angle of reflected light.
2. The surfaces have defined highlights on the rounded edges, and wider highlights over the larger surfaces.
3. There are small scratches on the surfaces that seem less rough than their surrounding areas, leading to differently shaped highlights.
4. There is an etched logo in the centre, with a higher roughness than the rest of the surface.
5. Other than the logo, the memory stick seems to have a uniform material colour.
6. The innards of the memory stick are relatively uniform in appearance, consiting of blue plastic with a black plastic underlay, and gold contacts on top.
7. The gold contacts on top of the blue plastic are highly metallic, leading to large highlights when lit, however almost black when out of light.
8. There is a translucent plastic surround to the metal housing that has wide specular highlights that clearly reflect the environment.
9. The plastic is not overly transparent, and so from many angles it appears nearly entirely reflective.

2 Method

To start development of shaders, I initially created the base model in Maya, making sure to properly UV the object so later noise application worked as intended. After instancing the model into an OpenGL context I started development of the base PBR shader that I would modify for the individual components of the object. At first the shader used the Beckmann distribution for the specular component, though I switched to using GGX for aesthetic reasons, as it generally led to wider highlights on the metal surfaces. ***You should stick an image of that here***

I then worked on expanding my shader to work with environment lighting by passing through a set of pre-defined lights for lighting calculations, and an environment cubemap for the specular colour. I encountered issues with deciding how to calculate the view vector, as I was working with the mindset of using the camera's rotation to rotate the object rather than to rotate around the object, and as such I had thought a constant view vector down the Z axis would make sense. Unfortunately due to the flat nature of the main housing, the reflection vector would be the same for all fragments adjacent to each other on a flat surface, and so I came to the realisation the view vector would simply be the normalised, negative position of the fragment in unprojected eye space, as the cubemap is also handled via unprojected space.

The environment cube itself is simply a cube that the camera sits in, using the eye space fragment position of the cube as a lookup vector to the sampler.

As the memory stick is comprised of mutliple different elements with differing properties, I separated the mesh into groups of the same material, and created a USB structure to hold the meshes and their respective shaders and properties. I modified the base PBR shader to take in $roughness$, $metallic$, $diffuse$ and $specular$ colour and multipliers along with $alpha$ for transparency handling, all of which I defined in the USB structure.

3 Results

Results text [Gunawan 2013]

References

GUNAWAN, J., 2013. OSL / Shader Writing. `https://blendersushi.blogspot.co.uk/2013/08/osl-shader-writing-journey-to-unknown.html`. Accessed 12 December 2017.

*e-mail:s4907224@bournemouth.ac.uk