Shader Development at OLM

Shinji Ogaki*
OLM Digital, Inc.

Abstract

Recently our studio started using Arnold renderer to create cinematics for games. It was not an easy task for artists to optimize shader parameters for a modern renderer because they were not familiar with it. In this paper we describe the challenges we had and how our shaders were developed to tackle this situation.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Raytracing;

Keywords: shader, ray tracing

1 Introduction

Nowadays, physically based shaders are common tools for realistic image synthesis. Sometimes we have to break the physical correctness to satisfy the needs of artists. However, even in such situations it is still very important to be physically plausible to generate convincing images. Shaders have to be not only efficient but also comfortable for artists. We designed our shaders with these points in mind.

2 Challenges

Simplicity *Easiness to learn* could be directly interpreted as simplicity. Especially when artists are working on a tight schedule, it is very important to provide a better usability. To that end, the number of parameters was reduced as much as possible. The range of a parameter is limited from 0 to 1 whenever possible, and parameters required by skilled artists are hidden in a tab which is closed by default because they should not be exposed for everyone. **Rapid Development** Development speed is one of the most important factors if the deadline is approaching. Prototypes were built on a daily basis to meet the requirements from our artists. **Reliability** Needless to say, all features should work perfectly. For example, our artists do not use a shader if it does not support per light AOVs (Arbitrary Output Variables). Therefore, this type of feature should be implemented in all shaders without any bugs. Not making mistakes is impossible, however, the quality of our shaders was improved in collaboration with our artists. **Lack in Literature** Designing physically based shaders is not an extremely hard task unless a completely new shading model is required. However, it is not the case

*e-mail:shinji.ogaki@gmail.com

SA'15 Workshop on RDVG, November 02 – 06, 2015, Kobe, Japan.
ACM 978-1-4503-3934-6/15/11.
http://dx.doi.org/10.1145/2818419.2818423

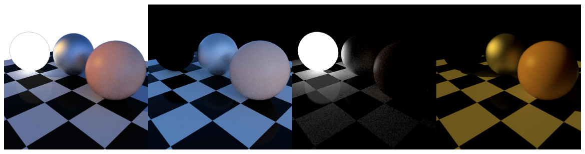

Figure 1: *Per light AOVs. From left to right: beauty, contribution from sky, contribution from light-emitting object, and contribution from key light. Each AOV channel contains not only direct illumination but also indirect illumination.*

with non-physically based shaders because they are usually subjectively evaluated by artists. Sometimes their feature requests are described in abstract terms and solutions cannot be found in academic literature. A good example is our eye shader (Section3.2).

3 OLM Shaders

By working through the above described challenges, OLM shaders were created. In this section we describe the details of our main shaders.

The most important common feature of our shaders is per light AOVs so that artists can adjust the intensity of key lights without re-rendering images. Our implementation is based on alShaders[Hill et al. 2014]. Unfortunately, alShaders had some limitations so we implemented custom ray traced subsurface scattering and a hair shader that work together with this feature.

3.1 Surface/Hair Shader

We developed an über shader (surface shader) which has a base lambertian/subsurface scattering layer and two specular/glossy layers on top of it (Fig2 right). This shader is used for almost any kinds of materials including glass, metal, skin, etc. Our hair shader uses Eugene d'Eon's model[d'Eon et al. 2013]. As reported in [Pekelis et al. 2015], generating samples and evaluating pdf are very slow, however, in practice they can be done very quick with pre-computed lookup tables. We carefully chose the size of the lookup tables so that users do not feel stressed when they are updated due to the change of a parameter. And also we are not aiming at creating realistic human hair fibers. Therefore, Eugene's model fits perfectly for our production, although it lacks the support of eccentricity. One interesting feature of our hair shader is that it shares the parameters with the surface shader (Fig2 left and Table1). For example, the specular/glossy layer of the surface shader has color, roughness, and refractive index. These are also used in the hair shader to compute the R component. Thus artists do not have to learn specific controls for the hair shader and can imagine how it looks like once they become familiar with the surface shader.

Surface	diffuse	S/G layer1	S/G layer2	Backlight
Hair	diffuse	R	TRT	TT

Table 1: *How surface shader components are mapped to the ones of hair shaders. Each component has common parameters such as color, roughness, and refractive index.*

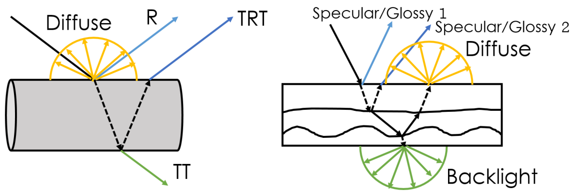

Figure 2: *Surface shader and hair shader share most of their parameters.*

We also created a small helper shader to control the parameters of the hair shader along each hair curve as Obq_Shaders[Oblique FX 2011-2015]. This shader has two inputs so that user can change the arbitrary parameter of a hair strand from root to tip. With this tool, users have more controls while keeping the user interface simple (Fig3).

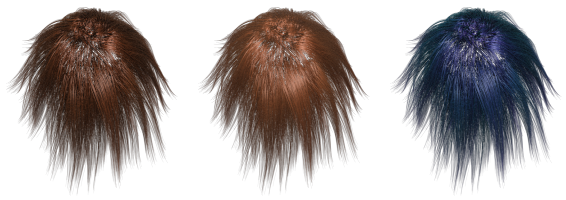

Figure 3: *Examples of hair rendering. Normal hair (left), varying shadow transparency from root to tip (middle), and varying color from root to tip (right). Note that the same color and lighting are used to render the left and middle images.*

3.2 Eye Shader

Our eye shader is one of the most used non-physically based shaders. The shader can be simply assigned to an ellipsoidal shape and the eye pattern is created in the projected uv space. In this way, artists can avoid modeling eyes part by part and assigning different materials to them. However, there were some technical issues that we had to address: *iris shape*, *catch light*, *caustics*, and *refraction*.

Initially we tried to find a biological model to generate the shape of iris. However, we could not find a suitable one for our characters. We thus ended up using the shape of the Go stone [Fiedler 2013].

Catch light is an important feature to make characters lively. The position of a catch light should be determined according to a specific key light but should not be a physically correct specular highlight because our artists want to change the shape and intensity. Therefore, a user specified catch light pattern is texture mapped at the position obtained from the key light information.

A similar approach is taken for caustics. Its intensity and size are controlled manually by artists but color and position are determined by the direction of a specific key light. The shape of caustics is a simple ellipsoid created in polar coordinates (Fig4 right).

Refraction is the most tricky part of this shader. Fake refraction effect is achieved by shifting the center of the eye $\vec{c}$ depending on the viewing direction and generating a deformed iris pattern (Fig4 middle). Generating the pattern in the deformed space is somewhat cumbersome. We thus create an iris pattern in the canonical space (Fig4 left). We calculate a new polar coordinates (r, θ) for the shading point p by solving the following equation:

$$|\vec{p} - (1-r)\vec{c}|^2 = r^2, \qquad (1)$$

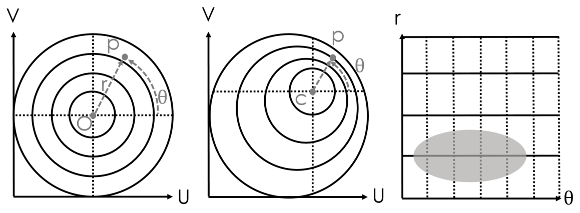

Figure 4: *Coordinates used for iris pattern creation. Canonical coordinates (left), deformed coordinates (middle), and fake caustics pattern in polar coordinates (right).*

where $\vec{p}$ is a 2d position in the texture coordinate, and r $(0 \leq r \leq 1)$ the radius we want to obtain. Note that θ is unchanged. As shown in Fig5, creating a deformed iris pattern gives an impression that the iris is located deep inside.

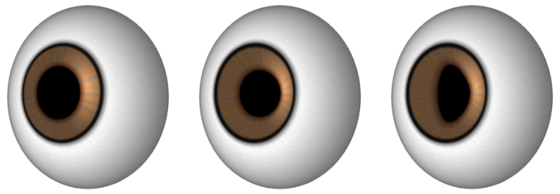

Figure 5: *Fake refraction effect. Disabled (left), enabled (middle), and enabled for a catty eye (right). Note that the pupil shape of the catty eye is bilaterally symmetric without fake refraction.*

4 Conclusion

We designed shaders to reduce artists's work amount and they are successfully used in production. Besides addressing technical issues, having good communication was crucial to lead this project to success.

Acknowledgements

The author would like to thank Marcos Fajardo for offering us Arnold research licenses. This work would not have been done without the support of our artists Tatsuya Horii, Yuma Sadakata, Tomosato Taniguchi, Godai Takanishi, Satoshi Ohkubo, Yuki Sugiyama, and CG Supervisor Yoshinori Moriizumi.

References

D'EON, E., MARSCHNER, S., AND HANIKA, J. 2013. Importance sampling for physically-based hair fiber models. In *SIGGRAPH Asia 2013 Technical Briefs*, ACM, New York, NY, USA, SA '13, 25:1–25:4.

FIEDLER, 2013. The shape of the go stone.

HILL, S., MCAULEY, S., DUPUY, J., GOTANDA, Y., HEITZ, E., HOFFMAN, N., LAGARDE, S., LANGLANDS, A., MEGIBBEN, I., RAYANI, F., AND DE ROUSIERS, C. 2014. Physically based shading in theory and practice. In *ACM SIGGRAPH 2014 Courses*, ACM, New York, NY, USA, SIGGRAPH '14, 23:1–23:8.

OBLIQUE FX, 2011-2015. Obq_shaders.

PEKELIS, L., HERY, C., VILLEMIN, R., AND LING, J. 2015. A data-driven light scattering model for hair.