A Synthetic Shader Generator for Building Large Shader Databases

Min-Hee Jang[1], Du-Yeol Kim[1], Sang-Wook Kim[1], Jae-Ho Lee[2], Min-Ho Kim[3], and Sung-Woo Nam[2]

[1] Dept. Of Electronics and Computer Engineering, Hanyang University, Korea
[2] Content Research Division, ETRI, Korea
[3] Studio Floationg Island, Korea
[1] {zzmini, gorila, wook}@hanyang.ac.kr
[2] {jhlee3, swnam}@hanyang.ac.kr
[3] romare@hanmail.net

Abstract. There are significant difficulties during the shading process when using commercial rendering systems: (1) a lot of technical details on shaders required, (2) long rendering time, and (3) repeated cumbersome trial-and-errors. To overcome such difficulties, we had developed *Shader Space Navigator*, a system that efficiently searches for shaders similar to a given query shader. *Shader Space Navigator* requires a shader database that contains a large number of quality shaders. It is not easy to secure quality shaders in reality. To overcome this situation, we propose a shader generator that produces shaders in a synthetic way. First, a small number of seed shaders are collected, and then their variations are generated by a synthetic shader generator by randomly changing of attributes values on each seed shader. Next, CG artists give feedback by picking good shaders among the generated ones. Then, a meaningful value range for each attribute is estimated. This process is repeated until the stable value range is reached for each attribute. Finally, the required number of quality shaders are generated with determined attribute value ranges.

Keywords: Shader, Rendering, Shader Databases, Shader Generator.

1 Introduction

Rendering is one of most significant processes in computer graphics(CG) [1]. Rendering indicates two dimensional image synthesizing using environmental information such as lighting, positions, and 3D geometry [2]. As CG techniques have been improved and popularized, rendering is also widely utilized in movies, animation, games, science visualization, and industrial design.

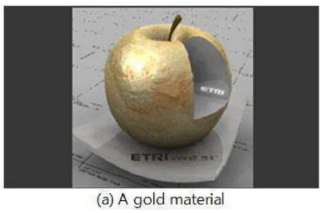
A *shader* is an essential element that defines the final look of a target object, and represents the object features such as color, texture, and material[3]. Usually, a shader is presented as a network of shading node, called a *shading network*[8], which is found in many commercial rendering systems. A shading node has its specific features to show the object, and is very diverse in practical usage[7]. The shading

node has a number of attributes for an object, counting up to hundreds. 3D CG artists compose these shading nodes into a shading network to prepare the rendering process[9]. Subsequently, the rendering system executes synthesizing of the graphic object using the values of all the attributes in the shading network thus composed[10]. Figure 1 shows an example of a shading network. Normally, the complicated shading network, in Figure 1(b), was composed to represent a practical gold material in Figure 1(a) by a 3D content production.

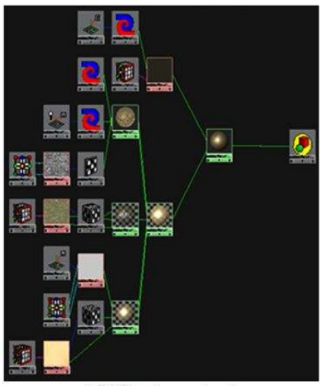
There are some serious difficulties during the shading process when using commercial rendering systems. First, lots of knowledge and learning are required for CG artists to create a reasonable shader for a practical object. A CG artist has to fully understand the features of shader nodes to compose a shading network as in Figure 1.

Second, rendering requires a lot of processing time[4]. Rendering is performed to verify the artist's composed shading network even though it is in intermediate state. However, the test rendering to verify a current shader also demands a long processing time, which is dependent upon the complexity of a shading network[11].

Third, the above processes, composing a shading network, tuning attributes, and test rendering have to be executed repeatedly until the shader is satisfied by the CG artist. The final look of a target object is difficult to imagine accurately before rendering since there are many shading components and attributes in a shading network[12]. Therefore, CG artists should repeat composing, tuning, and rendering multiple times when they create the final quality shader for the scene of contents.



(a) A gold material



(b) A Shading network

Fig. 1. An example of a shading network.

Due to these difficulties, a fairly long processing time is needed to create a shader[13]. However, we claim that these difficulties can be avoided if we maintain a

shader database that contains a large amount of useful shaders produced by good CG artists. If CG artists find their shaders on demand directly from the shader database, the shading process would be much simpler and more creative than before. To the end, Electronics and Telecommunications Research Institute and Data and Knowledge Engineering Lab at Hanyang University have been working together to develop *Shader Space Navigator*[5], a system that efficiently searches for similar shaders from a shader databases.

Shader Space Navigator can reduce carrying out trial-and-errors in a shading process. *Shader Space Navigator* is based on the assumption that there is an abundant shader database that contains a lot of quality shaders created from CG artists. For attaining this, *Shader Space Navigator* has to provide a large pre-rendered shader database, well organized for efficient navigation. However, since there are an insufficient number of shaders available until now, we decided to generate shaders in a synthetic way.

First, a small number of seed shaders are collected, and then their variations are generated by a synthetic shader generator by randomly changing of attributes values on each seed shader. Next, CG artists give feedback by picking good shaders among the generated ones. Then, a meaningful value range for each attribute is estimated. This process is repeated until the stable value range is reached for each attribute. Finally, the required number of quality shaders are generated with determined attribute value ranges. Namely, shader generator makes the shading retrieval efficient and effective by providing large number of quality shaders.

This paper is organized as follows. Section 2 proposes shader database construction. Section 3 shows the examples of generated shaders from our shader database. Finally, Section 4 summarizes and concludes this paper.

2 Shader Database Construction

The As mentioned in the previous section, a large number of shaders should be prepared for retrieving satisfactory results. Actually, some world-famous major studios such as PIXAR, WETA, and ILM have their abundant shader database. Unfortunately, they do not open their shaders because they consider those as their intellectual property and competitive edge. Therefore, gathering of shaders was not easy to accomplish during building of a shader database. So, we decided to generate a large number of quality shaders in a synthetic way at this time. The steps of building a shader database are described as below:

1. Define shader categories.

2. Create some seed shaders in every category (by artists).

3. Generate more shaders by changing attribute values in each seed shader.

4. Analyze the feedback from artists to find out major attributes and reasonable value ranges for attributes in generated shaders.

5. Re-generate more shaders with the attribute value ranges obtained in step 4.

6. Repeat steps 4 and 5 until the attribute value ranges converge.

7. Generate the required number of shaders with finally determined attribute value ranges.

Fig. 2. Seed shaders in 17 categories.

For constructing a shader database, some artists have created seed shaders belonging to each of 17 categories: architecture, coating, emitter, fabric, glass, grass,

ground, liquid, metal, paper, pattern, plastic, skin, stone, wall, wood, and extra. Each category also has its sub-categories: for example, the metal category has gold, silver, and iron as its subcategories. In addition each sub-category has a number of seed shaders, which have different types of shader components even though the rendering results look similar. Figure 2 shows examples of seed shaders belonging to 17 shader categories.

Shader generation has been performed by randomly changing of attributes values on each seed shader at first. This naive approach is to estimate major shader attributes and their acceptable value ranges. Artists provides a lot of information on shading components and values based on their experiences. They gave feedback by picking shader images, which look nice as variations from their seed shaders. After the feedback step, a meaningful value range for each attribute is estimated. Repeat this process until we reach the stable value range for each attribute. This procedure is described in Figure 3.

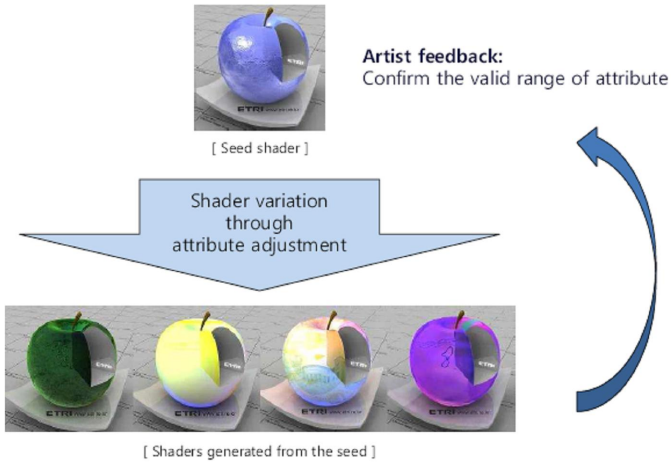


Fig. 3. Generating synthetic shaders.

The number of initially re-generated shaders was approximately 100,000 because we produced about 1,000 shaders from one seed shader by changing its attribute values. However, initially generated shaders include a number of garbage that are not useful in their categories. The number of these garbage shaders decreases as the stpes of feedback and re-production proceed repeatedly while the total number of shaders was fixed as 100,000. For convenient expansion, *Shader Space Navigator* provides functionality that artists can easily add new shaders into the shader database.

3 Examples of Generated Shaders

In this section, we show the examples of generated shaders in some categories: fabric, metal, and pattern. Figure 4 shows the examples of generated shaders in the fabric category.

Fig. 4. Examples of generated shaders in fabric category.

Figure 5 shows the examples of generated shaders in the emitter category. Finally, Figure 6 shows the examples of generated shaders in the pattern category.

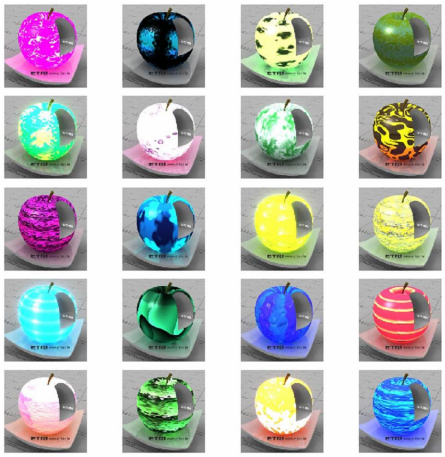


Fig. 5. Examples of generated shaders in emitter category.

4 Conclusions

There are significant difficulties during the shading process when using commercial rendering systems: (1) a lot of technical details on shaders required, (2) long rendering time, and (3) repeated cumbersome trial-and-errors. To overcome such difficulties, we had developed *Shader Space Navigator*, a system that efficiently

searches for shaders similar to a given query shader. *Shader Space Navigator* requires a shader database that contains a large number of quality shaders. It is not easy to secure quality shaders in reality.

To overcome this situation, we proposed a shader generator that produces shaders in a synthetic way. First, a small number of seed shaders are collected, and then their variations are generated by a synthetic shader generator by randomly changing of attributes values on each seed shader. Next, CG artists give feedback by picking good shaders among the generated ones. Then, a meaningful value range for each attribute is estimated. This process is repeated until the stable value range is reached for each attribute. Finally, the required number of quality shaders are generated with determined attribute value ranges. The proposed shader generator makes the shading retrieval efficient and effective using a large number of quality shaders produced by our shader generator.

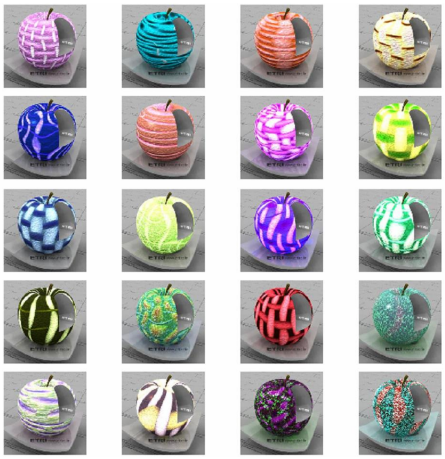


Fig. 6. Examples of generated shaders in pattern category.

Acknowledgments. This work was supported by the IT R&D program of MIC/IITA[2006-S-045-01, Development of Function Extensible Real-Time Renderer] and was partly supported by the MKE(Ministry of Knowledge Economy), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute of Information Technology Advancement) (IITA-2008-C1090-0801-0040).

References

1. J. Birn, *Digital Lighting and Rendering,* New Riders Press (2006)
2. M. Pharr and G. Humphreys, *Physically Based Rendering: from Theory to Implementation,* Elsevier (2004)

3. S. Upstill, *The RenderMan Companion: A Programmer's Guide to Realistic Computer Graphics,* Addison-Wesley Professional (1990)
4. T. Kajiya, "The Rendering Equation," In *Proc. Annual Conf. on Computer Graphics*, pp. 143--150 (1986)
5. J. H. Lee, M. H. Jang, D. Y. Kim, S. W. Kim, M. H. Kim, and J. S. Choi, "Shader Space Navigator: A Turbo for an Intuitive and Effective Shading Process," *ACM Int'l. Symp. on Applied Computing* (2009) (accepted to appear)
6. S. Beyer and J. Goldstein, "When is Nearest Neighbor Meaningful?," In *Proc. Int'l. Conf. on Database Theory*, pp. 217--235 (1999)
7. P. Pellacini and F. Ferwerda, "Toward a Psychophysically-Based Light Reflection Model for Image Synthesis," In *Proc. Int'l. Conf. on Computer Graphics and Interactive Techniques*, pp. 55--64 (2000)
8. AutodeskMayaPress, *Learning Autodesk Maya 2008: The Special Effects Handbook*, Autodesk Maya Press (2008)
9. K. Proudfoot and W. Mark , "A Real-time Procedural Shading System for Programmable Graphics Hardware," In *Proc. Int'l. Conf. on Computer Graphics and Interactive Techniques*, pp. 159--170 (2001)
10. X. Sun and K. Zhou, "Interactive Relighting with Dynamic BRDFs," In *Proc. Annual Conf. on Computer Graphics*, pp. 26--27 (2007)
11. A. Ngan and F. Durand, "Image-Driven Navigation of Analytical BRDF models," In *Proc. Eurographics Symp. on Rendering* (2006)
12. J. Nimeroff and J. Dorsey, "Implementation and Analysis of an Image-Based Global Illumination Framework for Animated Environments," *IEEE Transaction on Visualization and Computer Graphics*, pp. 283—298 (1999)
13. C. Goral and K. Torrance, "Modeling the Interaction of Light Between Diffuse Surfaces," *Computer Graphics*, pp. 213—222. (1984)