COMP220-Research Journal

Graphical Shaders and Optimization

1506530

November 27, 2016

1 Journal

For this research journal, shaders and compilers will be looked at. Specifically shader languages that allow for an easier time when creating custom shaders, a look into optimization will also be included.

Title: A System for Rapid Exploration of Shader Optimization Choices[1]
This is an open source project that provides a new shader language and complier to allow rapid exploration of shader optimization options. Being open source means that anyone can contribute as well as the software being free. This kind of software can be incredibly useful when creating shaders as it allows the user to quickly optimize their game for multiple platforms. It also is designed in a way that allows the program to keep up to date with the most current underlying rendering pipeline's, so if a new GPU has been releases the program will continue to work as per usual.

Title: Interactive Shader Development[2]
As programming becomes more complex and GPU's are in a constant state of evolution, it is very desirable to create software that can aid in keeping up to date. With the newest technology we need to do this without overcompensating the situation. This paper proposes a visual method to creating shaders, like blueprint it is a set of nodes that you connect to create viable algorithms. Unlike blueprint this method is not as optimized, so this paper also goes into a two step optimization process. Placing the generated code in the most optimal positions and a local optimization on the inserted transformations. This could be very useful for those who do not have the technical skill of a programmer but want to create custom shaders of their own.

Title: Shader Development at OLM[3]
This paper describes the challenges that a development team faced and how they overcome them. This type of paper can be really useful when developing shaders yourself as it contains information on how you could go about doing something while explaining whats happing along the process. This paper specifically covers hair and eye shaders while also warning about the issues that creating such shaders can cause. By doing this it can help the reader to not repeat their mistakes as well as justification on why something should be done in the way presented to them.

Title:Semiautomatic Shader Code Generation for Rendering Voxelized Polygonal Models[4]
Here is an example of semiautomatic code generation, it looks into the rendering of voxelized polygonal models. Voxelized rendering means that the scene is reprisented with spatial data opposed to conventional rasterization view. This paper proposes generating code based on a minimal extension of the shading language with a small number of voxelization specific keywords. They call rendering shaders accessing voxel data using their language 'voxel shaders'. The keywords chosen are of an XML-like look, and these words are replaced with full shader code once compiled. This can be very useful for medical purposes as CT scans require voxel information and writing the shader for it with this could make the task easier.

Title:A System for Rapid, Automatic Shader Level-of-detail[5]
Similar to the other papers this paper proposes a method of creating shader LOD policies for shaders typical of those in modern games using an LOD system. They state that most shader generation software is too slow and can lead to messy code with hours of waiting time for the shaders to be written. So one of their main focuses is on optimization of their system so the user only has to wait moments for the software to work. To do this they used a greedy search algorithm, adding parameter binding time processing capabilities. Their current system uses GLSL, though they want to use a higher shader authoring system.

Title:3D graphics hardware and role of shaders[6]
Pathak describes in this paper, the different methods, languages and terms used in graphics programming. A good place to start for someone looking to engage in the more technical side of 3D or 2D rendering. Many things are covered from a language like GLSL to the hardwares graphics pipeline. This paper is a nice summary that covers the basics that one could refer back to when working on shaders.

References

[1] Y. He, T. Foley, and K. Fatahalian, "A system for rapid exploration of shader optimization choices," *ACM Trans. Graph.*, pp. 112:1–112:12, 2016.

[2] P. D. E. Jensen, N. Francis, B. D. Larsen, and N. J. Christensen, "Interactive shader development," in *Proceedings of the 2007 ACM SIGGRAPH Symposium on Video Games*, (New York, NY, USA), pp. 89–95, ACM, 2007.

[3] S. Ogaki, "Shader development at olm," in *SIGGRAPH Asia 2015 R and D in the Video Game Industry*, (New York, NY, USA), pp. 5:1–5:2, ACM, 2015.

[4] J. Fischer, D. Whittaker, A. Lefohn, and B. Gooch, "Semiautomatic shader code generation for rendering voxelized polygonal models," in *Proceedings of the 2008 Symposium on Interactive 3D Graphics and Games*, (New York, NY, USA), pp. 14:1–14:1, ACM, 2008.

[5] Y. He, T. Foley, N. Tatarchuk, and K. Fatahalian, "A system for rapid, automatic shader level-of-detail," *ACM Trans. Graph.*, pp. 187:1–187:12, 2015.

[6] S. K. Pathak, "3d graphics hardware and role of shaders," in *Confluence 2013: The Next Generation Information Technology Summit (4th International Conference)*, pp. 351–356, IEEE, 2013.