Shader Pipeline and Effect Encapsulation using CLOS

Nicolas Hafner
Shirakumo.org
Zürich, Switzerland
shinmera@tymoon.eu

ABSTRACT

KEYWORDS

Common Lisp, GLSL, OpenGL, GPU, CLOS, Object Orientation

ACM Reference Format:
Nicolas Hafner. 2019. Shader Pipeline and Effect Encapsulation using CLOS. In *Proceedings of the 12th European Lisp Symposium (ELS'19).* ACM, New York, NY, USA, 2 pages.

ELS'19, April 1–2 2019, Genova, Italy

1 INTRODUCTION

2 RELATED WORK

Courreges[1] presents an in-depth analysis of the rendering procedure employed by the modern, high-production game GTA V. It illustrates the many stages to produce a final image, as well as their data dependencies.

Harada et al.'s work on Forward+[2][3] also clearly illustrates the need for systems that support multi-staged rendering pipelines with complex data interaction schemes.

Gyrling[4] presents an overview of the techniques used to perform parallel rendering in Naughty Dog's commercial engine. Individual steps within a stage, render stages of a frame, and multiple frame renderings are divided up into many small jobs that can run in parallel and are synchronised using counters on a shared structure.

The case study of the Unity game engine by Messaoudi et al[**?**] shows the availability of a set of fixed rendering pipelines that can be customised in a very limited extent with custom shaders. However, these shaders must fit into Unity's existing lighting and overall rendering model. While Unity does allow building a custom pipeline via their Scriptable Rendering Pipeline[**?**], they do not seem to offer any encapsulation or modularity features.

3 OVERVIEW

4 PASSES

5 PIPELINES

6 ALLOCATION

7 CONCLUSION

8 FURTHER WORK

9 ACKNOWLEDGEMENTS

10 IMPLEMENTATION

An implementation of the proposed system can be found at
https://github.com/Shirakumo/trial/blob/f34a79f0a6df21d1ed9259e85fbb3c7eed39352b/shader-pass.lisp
https://github.com/Shirakumo/trial/blob/f34a79f0a6df21d1ed9259e85fbb3c7eed39352b/pipeline.lisp
https://github.com/Shinmera/flow

A more in-depth discussion of the system can be found at
https://reader.tymoon.eu/article/363
https://reader.tymoon.eu/article/364

REFERENCES

[1] Adrian Courreges. Gta v-graphics study. http://www.adriancourreges.com/blog/2015/11/02/gta-v-graphics-study/, 2015. [Online; accessed 2019.01.24].
[2] Takahiro Harada, Jay McKee, and Jason C Yang. Forward+: Bringing deferred lighting to the next level. 2012.
[3] McKee, Jay Harada, Takahiro. Forward rendering pipeline for modern gpus. https://www.gdcvault.com/play/1016435/Forward-Rendering-Pipeline-for-Modern, 2012. [Online; accessed 2019.01.24].
[4] Christian Gyrling. Parallelizing the naughty dog engine using fibers. https://www.gdcvault.com/play/1022186/Parallelizing-the-Naughty-Dog-Engine, 2015. [Online; accessed 2019.01.24].