Point Clouds can render up to 10x faster with compute shaders instead of glDraw

Rendering Point Clouds with Compute Shaders

Markus Schütz, Michael Wimmer, TU Wien

Abstract

Regular point rasterization with glDrawArrays(GL_POINT,...) can be slow due to the overhead of the rendering pipeline. Compute shaders with atomicMin and atomicAdd are often a faster alternative.

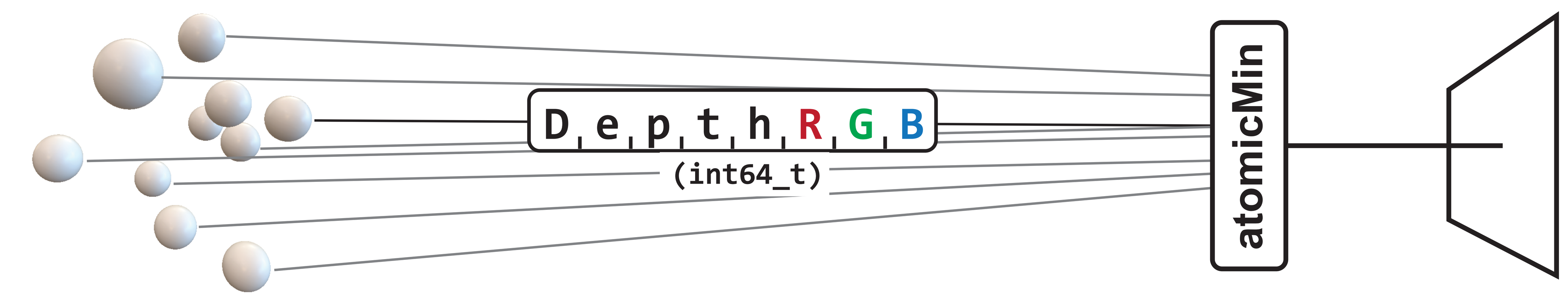
Method 1: Compute

A compute shader transforms points to pixel coordinates, and then encodes linear depth and color into a 64 bit integer. With atomicMin, we store the fragments with the lowest depth in a pixel buffer. A second compute shader transfers the pixel buffer into a texture.

Method 2: High-Quality

First, create a depth buffer with method 1. Then, use atomicAdd to sum up and count color values of points at most 1% behind depth buffer. Finally, divide sum of colors by number of fragments to get an average color value of overlapping points in a pixel. Compute shader implementation of Botsch et al. [1].

Custom Rasterization with atomicMin



Transform
to pixel coord

Encode
5 byte depth, 3 byte RGB

Write to Pixel Buffer
atomicMin keeps point with smallest depth, RGB in least significant bits → largely ignored

High-Quality Rendering with atomicAdd

Depth Buffer
using method 1

Attribute Buffer
sum of colors and fragment counts with atomicAdd

Normalize
Divide sum of colors by number of fragments

Results

GL_POINTS 60.26 ms | **Compute** 5.87 ms | **High-Quality** 15.48 ms

Regular | **High-Quality**

• Our compute and the classic GL_POINTS method produce the same result • The basic compute method is up to 2x to 10x faster than GL_POINTS • The high-quality method is up to 2x to 4x faster than GL_POINTS • Evaluated for point sizes of 1 pixel • GL_POINTS still faster for point sizes larger than 2x2 pixels

Dataset: San Simeon, 117M points, courtesy of PG&E
Code: github.com/m-schuetz/compute_rasterizer
Video: bit.ly/2nv48gl

SIGGRAPH ASIA 2019 BRISBANE

Acknowledgements

The authors wish to thank Riegl LMS for the point cloud of Retz, PG&E and Open Topography for funding and hosting the point cloud of San Simeon (Morro Bay), and the Ludwig Boltzmann Institute for Archaeological Prospection and Virtual Archaeology for the point cloud of the Heidentor.

References / Related Work

[1] M. Botsch, A. Hornung, M. Zwicker, and L. Kobbelt. 2005. High-quality surface splatting on today's GPUs. In Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics, 2005. 17–141. https://doi.org/10.1109/PBG.2005.194059

[2] Christian M Günther, Thomas Kanzok, Lars Linsen, and Paul Rosenthal. 2013. A GPGPU-based Pipeline for Accelerated Rendering of Point Clouds. Journal of WSCG 21 (2013), 153–161.

[3] Michael Kenzel, Bernhard Kerbl, Dieter Schmalstieg, and Markus Steinberger. 2018. A High-performance Software Graphics Pipeline Architecture for the GPU. ACM Trans. Graph. 37, 4, Article 140 (July 2018), 15 pages. https://doi.org/10.1145/3197517.3201374