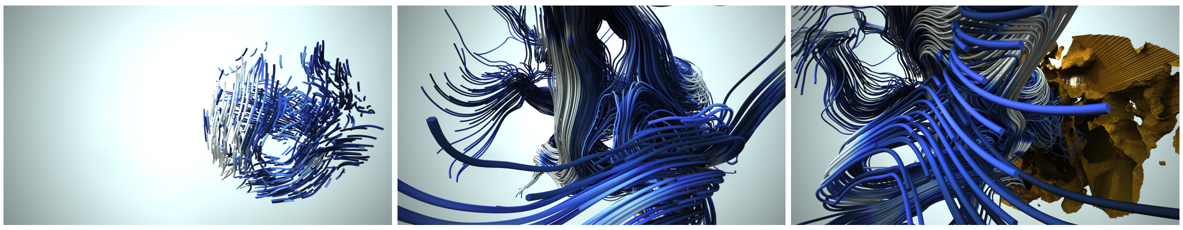
Eurographics Symposium on Parallel Graphics and Visualization (2015)
C. Dachsbacher, P. Navrátil (Editors)

Visualizing Groundwater Flow Through Karst Limestone

Carson Brownlee[1] Aaron Knoll[2] Paul Navrátil[1] Kevin J. Cunningham[3] Michael C. Sukop[4] and Sadé Garcia[5]

[1]Texas Advanced Computing Center [2]SCI Institute [3]USGS [4]Florida International University [5]Louisiana State University



Abstract

Water management is critical in Florida where freshwater is often rare or, in times of flooding, overabundant and seawater frequently contaminates available sources. Professor Michael Sukop from Florida International University, his student, Sadé Garcia, and Dr. Kevin Cunningham of the United States Geological Survey are developing techniques to better understand flow through aquifers in South Florida, which are vital sources of freshwater. 3D flow simulations of groundwater through Computed Tomography (CT) data from samples of karst limestone allowed them to more accurately predict the permeability values of the rock in their tests than existing laboratory measurement techniques. Researchers at TACC visualized these simulations by developing a rendering library which can render photo-realistic images using a path tracer built with Intel's Embree ray tracing kernels by intercepting calls to the OpenGL API. Using this software, they were able to generate significant improvements over native OpenGL rendering in existing tools and better illustrate the flow through thumb-sized holes in the limestone.

1. Introduction

Garcia et al. have been exploring ways to better determine the permeability of karst limestone found in the Biscayne Aquifer in South Florida [Gar13]. Rock structures were digitized using CT scans and flow simulations were conducted over the resulting data to measure permeability of the sample. Visualizations over the resulting 3D scalar field for the tomography and vector field from the flow simulations were then created in ParaView and animated using tessellated contours and streamlines. We then created improved renderings using in-house tools to create photo-realistic path-traced images without modification to existing visualization tools or the need to export geometry to external rendering programs. The primary novelty presented is the extension of previous work to support multiple ray tracing engines in the back end, providing a framework for a program-agnostic rendering library able to not only run on programs built with fixed function OpenGL, but to also render with different ray tracers to suite different requirements such as varying hardware or lighting algorithms.

2. Simulation

Aquifers are a vital resource in many parts of the world, especially in Florida where freshwater can often prove scarce and existing freshwater sources are often contaminated by seawater. Furthermore, flooding is a common occurrence which can cause significant challenges. Understanding flow through the complex rock structures present in aquifers that hold large quantities of fresh water is thus of great importance to managing these critical resources. Kevin Cunningham, Michael Sukop, and Sadé Garcia studied aquifers to determine the

DOI: 10.2312/pgv.20151160

permeability by combining tomographic scans with a computed flow simulation through the permeable structure of the Biscayne Aquifer in Florida, which about four million people rely on for fresh water [**?**]. The rock sample used contains a complex structure of megapores created by Calinassid shrimp burrowing into the rock, which resulted in high permeability of the aquifer.

To better understand flow through the aquifer, Garcia et al. used the sample data to conduct groundwater flow simulations using the Lattice Boltzmann Method (LBM) lb3d-prime program created by Daniel Thorne and Michael Sukop (https://code.google.com/p/lb3d-prime-dev) to show the applicability of LBM for simulating flow in megaporous structures. They set out to determine if simulation models were better suited for determining permeability of sampled rock structures of this type than existing air permeameter measurements. The initial sample was digitized using a CT scan and LBM was then used to compute the permeability of the sample and to simulate groundwater flow. This technique represented the fluid as discrete particles which were advected through the scanned data. Through a series of tests with laboratory apparatus, they were able to validate that their computational models proved to be a more correct method for analysis of permeability than traditional laboratory methods that often underestimated the permeability of the highly porous rocks found in the Biscayne Aquifer.

3. Visualization

The visualization was conducted on the Stampde supercomputer at the Texas Advanced Computing Center (TACC). Paraview 4.1.0 was used for rendering, utilizing the python interface for creating the rendering pipeline and animation. Initial renderings were done in 4K resolution and 1024 samples per pixel, but were reduced to 1080p to match submission criteria on file sizes. Parallelization of the rendering process was done through parallel batch processing, with each node given a different frame.

Two different geometric representations were used to display the rock sample data and the flow through it. For the rock, an isosurface was used to create a surface around homogeneous data values in the rock structure. It should be noted that in this representation not all data of the sample is shown, but rather a surface was extracted from the underlying density values along a single isovalue. "Emtpy" regions of the data are thus not necessarily empty, but rather don't contain the isovalue. This technique shows us the contours of that data value, however, while still leaving much of the data area empty so that users can better see into the data. To further facilitate this, an animated clipping plane was used to initially clip away the entire isosurface and slowly reveal the rest of the structure. A useful further addition to this work would be a sweep of the isovalues or a volumetric representation, however this was not used in this video.

Streamlines were used for representing flow through a single timestep of the simulated flow. 8000 points were created in a spherical region and advected through the vector field of flow velocities resulting in 3D streamlines with around 86 million triangles in total. The streamlines are colored according to the magnitude of flow velocity from blue to white. In order to animate the streamlines, the maximum advected length of the streamlines was increased each frame, resulting in expanding curves. The speed of expansion is constant and not dependent on the velocity field, however an interesting addition would be to vary the velocity of streamline growth by the velocity of flow. The camera was dollied out with the streamline expansion, resulting in streamlines snaking out towards the camera and weaving in and out of the rock structure. The resulting animation gives a revealing visualization of the streamlines, many of which would otherwise be occluded by rock or other streamlines.

4. Rendering Techniques

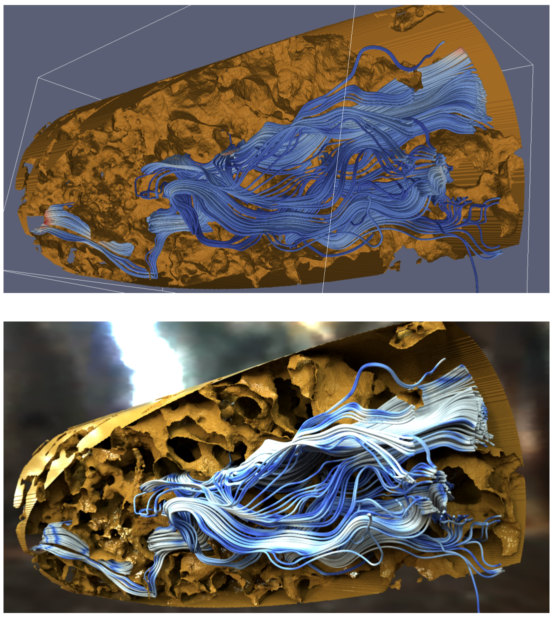


Figure 1: *Rendering with OpenGL (left) and GLuRay (right).*

Fixed function OpenGL utilized in many visualization programs such as ParaView results in local lighting only. Using global illumination techniques results in enhanced visual perception by giving additional depth cues from light occluded by other geometry, resulting in darker shading in areas with close proximity to other regions of the data. This is especially apparent in visualizations with large numbers of streamlines, giving clearer indications of where a streamline lies in regard to other streamlines. High curvature present in contours can also lead to indiscernible structures in the data due to lack of depth cues present in natural lighting. The Manta interactive ray tracer was integrated into ParaView by Brownlee et al. [BPL*12], however this implementation only supports a Whitted style of ray tracing with shadows, reflections and ambient occlusion. Instead, we extended the GLuRay framework developed by Brownlee et al. [BFH12] with a path tracing implementation built on the Embree Ray Tracing Kernels as a back end [WBW*14]. Figure 1(d) shows a rendering of the simulation data with standard OpenGL shading, while Figure 1(e) demonstrates the same scene rendered using GLuRay and an implementation of the Embree ray tracing kernels.

GLuRay presented a program-agnostic implementation of multiple ray tracers through a common interface— OpenGL. Fixed function Opengl state changes are mapped to ray tracing representations. Adapting additional rendering back ends is done through a selectable module system allowing for trivial extension of the GLuRay system by mapping a minimal set of primitives, material parameters, lights, and camera specifications represented through an intermediate GLuRay representation over the far more complex OpenGL state. Standard calls to set glLight, for example, are mapped to an intermediate representation in a set of C++ classes that ray tracers can easily translate into their own internal representations. GLuRay enabled high fidelity rendering within the existing ParaView program without having to export geometry and material properties to an external rendering program. This system also allows us to run GLuRay with different back ends for different computational environments to take advantage of specific hardware. The Embree Ray Tracing Kernels were chosen for Stampede as they were designed to be run with the Xeon Phi Coprocessor, and a path tracing implementation was easily adapted from examples given from Intel. Rendering was conducted on host Xeon CPUs for convenience and expanded memory space, however the system has been tested on the Phi and benchmarked for other performance critical projects. While not yet implemented, Optix or other GPU ray tracers could also be implemented for running a ray tracer on GPU systems.

The scene was lit from an emissive high dynamic range image map to give a realistic look to the scene, as if one were to place the sample in direct sunlight through image-based lighting. Depth-of-field effects were achieved using a thin lens camera simulation with a wide aperture at the start of the animation and a small aperture towards the end for a uniformly focused image. Two white quads were inserted to block out the HDR background image and, thanks to the path tracer, were used as bounce cards as one would use in photography. These options were specified in an external configuration file for the interception library as they could not be exposed through the existing OpenGL interface.

5. Conclusion

We have demonstrated the applicability of GLuRay for creating presentation quality renderings within existing visualization tools using selectable back ends suitable for different system architectures. While the motivation of this presentation video is primarily artistic, the performance benefits of such a system for CPU rendering and enhanced visual cues present in global illumination are compelling reasons for researchers to use such techniques in their exploratory and everyday use cases in addition to creating presentation quality videos.

6. Acknowledgements

This work was funded in part by National Science Foundation grants OCI-1134872 and ACI-1339863. Further funding was generously provided by the Intel Corporation.

References

[BFH12] BROWNLEE C., FOGAL T., HANSEN C. D.: Gluray: Enhanced ray tracing in existing scientific visualization applications using opengl interception. In *Eurographics Symposium on Parallel Graphics and Visualization* (2012), The Eurographics Association, pp. 41–50. 3

[BPL*12] BROWNLEE C., PATCHETT J., LO L.-T., DEMARLE D., MITCHELL C., AHRENS J., HANSEN C. D.: A study of ray tracing large-scale scientific data in two widely used parallel visualization applications. In *Eurographics Symposium on Parallel Graphics and Visualization* (2012), The Eurographics Association, pp. 51–60. 3

[Gar13] GARCIA S.: *Lattice Boltzmann Modeling and Specialized Laboratory Techniques to Determine the Permeability of Megaporous Karst Rock.* Master's thesis, FIU, 2013. 1

[WBW*14] WOOP S., BENTHIN C., WALD I., JOHNSON G. S., TABELLION E.: Embree - a kernel framework for efficient cpu ray tracing. In *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH, to appear* (2014). 3