Noname manuscript No.
(will be inserted by the editor)

Real-Time Astrophysical Visualisation Using OpenGL Shaders

me · you · him · ....

Received: date / Accepted: date

Abstract Splotch is a rendering algorithm for exploration and visual discovery in particle-based datasets coming from astronomical observations or numerical simulations. The strengths of the approach are production of high quality imagery and support for very large-scale datasets through an effective mix of the OpenMP and MPI parallel programming paradigms. This article reports our experiences in developing an interactive OpenGL application mimicking Splotch imagery. Our application is light. i.e. avoids dependencies on external libraries and allows exploitation of a wide range of hardware configurations ranging from low spec PCs to supercomputers. We describe a number of custom built renderers and present performance results and comparisons with standard Splotch imagery in the context of a number of use cases. Our application is useful in providing fast Splotch-like previews of astrophysical simulations thus allowing to quickly inspect large-scale datasets. We conclude by outlining future work developments including possibilities for further optimisations.

Keywords Scientific Visualisation · Astrophysics · OpenGL · GPUs · Shaders · High-Performance Computing ·

me
place
tel
E-mail: me@blah.blah.blah

you
place
tel
E-mail: you@blah.blah.blah

him
place
tel
E-mail: him@blah.blah.blah
....

1 Introduction

Splotch [5] is an open-source ray-casting algorithm for effectively visualising large-scale, particle-based numerical simulations. Very high-quality visualisations can be generated for large-scale cosmological simulations, e.g. the Millennium trilogy [9], the Horizon and MareNostrum runs [11] or the DEUS simulation [4]. The underlying models in these simulations typically reproduce the evolution of a representative fraction of the universe by means of hundreds of billions of fluid elements (represented as particles), interacting through gravitational forces. The typical size of a time output (or *snapshot*) can range from several hundreds of GBs to tens of TBs, recording ID, position and velocity of particles together with additional properties, e.g. local smoothing length, density and velocity dispersion. Although developed for numerical simulations, Splotch has being successfully employed also in other contexts, e.g. for visualizations of observed galaxy systems [28] or mesh-based datasets such as [**?**] .

The original Splotch algorithm has been optimised in terms of memory usage and exploitation of standard HPC architectures, e.g. multi-core processors and multi-node supercomputing systems by adopting the MPI paradigm [25] and OpenMP [**?**]. Recently a new version of Splotch was released [**?**] using the CUDA paradigm [12] to fully exploit modern HPC infrastructures. A number of optimised solutions for rendering particles have been implemented, based on a novel data classification and organization strategy. Significantly improved performances were achieved without affecting the linear scalability on the number of particles of the original Splotch. Nevertheless the focus of Splotch always remained on being light and fast code providing scientists with a handy tool for extracting scientific content from large-scale datasets, e.g. for identifying new features or patterns or even isolating small regions of interest within which to apply time-consuming algorithms. The code runs as an executable in a variety of modes, to handle a wide array of hardware setups, which are configured using a series of text files. These allow control not only over how the program runs but also over a number visualisation parameters.

A number of needs for real-time visualisation tools are not met by the current implementations of Splotch as the usability of the approach is quite limited. Whilst the configuration files are self-explanatory, in most cases a broad understanding of the datasets under scrutiny is required before algorithm values can be suitably defined. If these values are to be changed the program must be restarted. Another issue is the speed of rendering - depending on the dataset sizes it can take very long times to produce a render. Our objective in this paper was thus to develop an alternative approach to mimic the feature set offered by Splotch while at the same time meeting the objective of producing interactive real-time visualisations. The method would be able to sacrifice some visual quality in order to increase performance, but the visualisation should still provide meaningful representations. This way we could quickly generate Splotch-like imagery from astrophysical datasets - effectively previewing the datasets - to assist in configuring visualisation parameters before generating a full high-resolution image using the full Splotch. Our previewer was to exploit the capability of OpenGL which has already been shown to deliver good performances for visualisation (see Section 2).

The paper is organised as follows. Section 2 is a short description of Splotch and OpenGL shader technology. Our solutions for a number of interactive renderers mimicking Splotch are described in Sect. 3. Section 4 elaborates on the issues faced in extending our application to HPC infrastructures. Section 5 presents our reference datasets for benchmarking and discusses achieved performances and image quality in the context of a number of use cases. Finally Sect. 6 presents conclusions from our experiences and offers pointers to future developments.

2 Background

2.1 Splotch

Splotch renders particle datasets by accumulating contributions for individual particles along lines of sight originating from image pixels using a radiative transfer equation [32]:

$$\frac{d\mathbf{I}(\mathbf{x})}{dr} = (\mathbf{E}_p - \mathbf{A}_p\mathbf{I}(\mathbf{x}))\rho_p(\mathbf{x}), \tag{1}$$

where $\mathbf{I}(\mathbf{x})$ represents radiation intensity at position $\mathbf{x}$, r is a coordinate along the line of sight, $\mathbf{E}_p$ and $\mathbf{A}_p$ are the coefficients of radiation emission and absorption for particle p respectively and $\rho_{0,p}$ is a physical quantity (e.g. mass density or temperature) transported by p according to the Gaussian distribution:

$$\rho_p(\mathbf{x}) = \rho_{0,p}\exp(-||\mathbf{x} - \mathbf{x}_p||^2/\sigma_p^2), \tag{2}$$

where $\mathbf{x}_p$ denotes particle coordinates. This distribution is clipped to zero at a given distance $\chi \cdot \sigma_p$, where χ is a suitably-defined multiplicative factor and σ_p is the particle's smoothing length. Any rays passing at distances larger than $\chi \cdot \sigma_p$ are unaffected by ρ_p. Typically $\mathbf{E}_p$ is chosen identical to $\mathbf{A}_p$ as this situation not only produces visually appealing renderings but it also allows a simplified parallel implementation of the algorithm (calculations are independent of the integration order of particles). Nevertheless for special applications independent emission and absorption coefficients can be used. These coefficients can vary between particles and can be defined as functions of their properties.

2.2 Shaders

Survey of articles on exploitation of GPU shaders for visualisation (especially as applied to particle-based datasets) to set the scene for the shader implementations described in Sect. 3. Start from papers by Bailey as below (and references in them), relate to this paper's work (ref. applications and compute shaders).

[Bailey2009], Using GPU shaders for Visualisation, IEEE Comp. Graphics and Applications, 29(5), 2009, pp. 96-100.

[Bailey2011], Using GPU shaders for Visualisation Part 2, IEEE Comp. Graphics and Applications, 31(2), 2011, pp. 67-73.

[Bailey2013], Using GPU shaders for Visualisation Part 3, IEEE Comp. Graphics and Applications, 33(3), 2013, pp. 5-11.

3 Fast Previews

Nowadays GPU shaders can generate spectacular special effects and are used extensively in the computer games and film industries. Nevertheless as discussed in [Bailey2009] and [Bailey2011] they can be exploited effectively for scientific visualisation as their main strengths of high-quality imagery and interactivity can potentially aid significantly in gaining meaningful insights into modern large-scale datasets. GPU shaders can be deployed in a variety of visualisation contexts for rendering, e.g. point clouds, jitter clouds, cutting planes, contour planes, data probes, line integral convolutions and terrain bump-mappings. [Bailey2013] recently discussed compute shaders and Shader Storage Buffer Objects (SSBOs). A compute shader is a single-stage GLSL program (OpenGL Shading Language) playing no direct role in the graphics rendering pipeline as it simply resides outside the pipeline and manipulates data it finds in the OpenGL buffers. SSBOs make getting data into and out of compute shaders much easier. This means that using GPUs for data-parallel visualisation is intrinsic to OpenGL to greatly assist real-time visualisation techniques.

This section describes an application for generating fast renderings mimicking Splotch. The basic requirement was that the application should be light, i.e. avoiding unnecessary dependencies upon external libraries, and that it should be able to run on a variety of hardware configurations ranging from low performance PCs (i.e. without support for programmable graphics pipelines) to High Performance Computing (HPC) devices, e.g. multi-core and multi-node systems which nowadays are increasingly populated with GPUs. We implemented a number of renderers suitable for handling this wide range of hardware configurations.

As described below a number of profiles have been devised to detail the level of OpenGL feature support that can be expected. Minimal hardware support assumes no programmable graphics pipelines and only limited memory capabilities on the underlying GPUs. The choice to support non-programmable pipelines means that the corresponding renderer can achieve very limited visual effects. The renderer consists purely of spheres drawn using the position of particles with some support for scaling, e.g. based on particle properties. A more sophisticated image could be obtained by using other geometric primitives (glyphs) e.g. pyramids, but that would result in increased rendering times thus decreasing (potentially significantly) the overall performance of the application.

The following sections describe the developed GPU rendering techniques in detail focusing on the achieved performances on a number of reference datasets. We conclude by describing the use of a blurring algorithm to deliver improved visual quality. A Gaussian blur was used with linear sampling which improves rendering quality, i.e. closer mimicking of Splotch imagery at the expense of slowing down overall rendering performances.

3.1 Fixed Function Graphics Pipelines

– OpenGL specification: Vertex Buffer Objects. Support: minimal HW configurations.

3.2 Programmable Graphics Pipelines

– OpenGL specification: Vertex Buffer Objects / Geometry Shaders. Support: middle range HW configurations.

– OpenGL specification: Vertex Buffer Objects / Geometry Shaders / Frame Buffer Objects. Support: high-end HW configurations.

– OpenGL specification: Vertex Buffer Objects / Geometry Shaders / Frame Buffer Objects / Visual Quality Effects - Post Processing (Blurring). Support: high-end HW configurations.

4 HPC Infrastractures

– Adapting previewer to run on HPC systems;

– Frame synchronization;

– The previewer client application.

5 Results

An N-body-SPH simulation performed using Gadget [8] was used for the tests we discuss in this section. The simulation is based on about 400 million particles consisting of 200 million *dark matter* particles, 200 million baryonic matter particles (hereafter referred to as *gas*) and around 10 million *star* particles. Particles possess a number of physical quantities, e.g. vectors capturing spatial coordinates and velocities, or scalars representing smoothing length. Additionally gas particles are associated with *temperature* and *mass density*. Also, *spectral type* and *age* are properties of star particles. Such quantities can be exploited in highly-detailed renderings, e.g. by modulating colours and intensities appropriately.

– PC performance results (low spec laptop/PC);

– HPC performance results;

– Use cases - focus on comparison with Splotch imagery, meaningfulness of renderings / interactivity (questionnaire of users?)

6 Conclusions and Future Work

– OpenGL capability to be used in real-time scientific visualisation;

– Reflection on the previewer in relation to the problems in Splotch and against generic problems in scientific visualisation;

– Indication of future work to be done.

References

1. Ahrens, J., Rogers, D., Springmeyer, B.: Visualization and data analysis at the exascale. NNSA White Paper, Accelerated Strategic Computing (ASC) Exascale Environment Planning Process (2011)
2. http://www.intel.com/content/www/us/en/architecture-and-technology/many-integrated-core/intel-many-integrated-core architecture.html: (accessed on 25/03/2013)
3. Berriman, G., Groom, S.: How will astronomy archives survive the data tsunami? Communications of the ACM **54**(12), 1150–1164 (2011)
4. http://www.deus consortium.org/: (accessed on 25/03/2013)
5. Dolag, K., Reinecke, M., Gheller, C., Imboden, S.: Splotch: Visualizing cosmological simulations. New Journal of Physics **10** (2008)
6. Dykes, T., Rivi, M., Gheller, C., Krokos, M., Dolag, K., Reinecke, M.: Cuda splotch: Gpu accelerated astrophysical visualization. NVIDIA GPU Technology Conference (2013)
7. Fraedrich, R., Schneider, J., Westermann, R.: Exploring the millennium run - scalable rendering of large-scale cosmological datasets. IEEE Transactions on Visualization and Computer Graphics **15**(6), 1251–1258 (2009)
8. http://www.mpa garching.mpg.de/gadget/: (accessed on 25/03/2013)
9. http://www.mpa garching.mpg.de/galform/virgo/millennium/: (accessed on 25/03/2013)
10. Hassan, A., Fluke, C., Barnes, D.: Unleashing the power of distributed cpu/gpu architectures: Massive astronomical data analysis and visualization case study. Astronomical Data Analysis Software and Systems XXI, P. Ballester (Ed.), ASP Conference Series **461**, 45–48 (2012)
11. http://www.projet horizon.fr/: (accessed on 25/03/2013)
12. http://developer.nvidia.com/cuda/: (accessed on 25/03/2013)
13. https://wci.llnl.gov/codes/visit/: (accessed on 25/03/2013)
14. http://thrust.github.com/: (accessed on 25/03/2013)
15. http://users.monash.edu.au/dprice/splash/: (accessed on 25/03/2013)
16. http://virdir.ncsa.illinois.edu/partiview/: (accessed on 25/03/2013)

17. http://visivo.oact.inaf.it:8080/: (accessed on 25/03/2013)
18. http://www.atnf.csiro.au/projects/askap/: (accessed on 25/03/2013)
19. http://www.hdfgroup.org/HDF5/: (accessed on 25/03/2013)
20. http://www.hpcc.astro.washington.edu/tools/tipsy/tipsy.html/: (accessed on 25/03/2013)
21. http://www.oamp.fr/dynamique/jcl/glnemo/: (accessed on 25/03/2013)
22. http://www.paraview.org/: (accessed on 25/03/2013)
23. http://www.skatelescope.org/: (accessed on 25/03/2013)
24. http://www.vtk.org/: (accessed on 25/03/2013)
25. Jin, Z., Krokos, M., Rivi, M., Gheller, C., Dolag, K., Reinecke, M.: High-performance astrophysical visualization using splotch. Procedia Computer Science **1**(1), 1775–1784 (2010)
26. Kaehler, R., Hahn, O., Abel, T.: A novel approach to visualizing dark matter simulations. IEEE Transactions on Visualization and Computer Graphics **18**(12), 2078–2087 (2012)
27. http://www.nvidia.co.uk/content/PDF/kepler/NVIDIA Kepler-GK110-Architecture-Whitepaper.pdf: (accessed on 25/03/2013)
28. Koribalsky, B., Gheller, C., Dolag, K.: http://www.atnf.csiro.au/people/bkoribal/3d-visualisation/ (accessed on 25/03/2013)
29. Merrill, D., Grimshaw, A.: Revising sorting for gpgpu stream architectures. Technical Report CS2010-03, Department of Computer Science, University of Virginia (2010)
30. Rivi, M., Calori, L., Muscianisi, G., Slavnic, V.: In-situ visualization: State-of-the-art and some use cases. PRACE White Paper (2012), http://www.prace-ri.eu/Visualisation/ (accessed on 25/03/2013)
31. Sciacca, E., Vitello, F., Becciani, U., Costa, A., Massimino, P., Bandieramonte, M., Krokos, M., Petta, C., Pistagna, C., Riggi, S., Vitello, F.: Visivo workflow-oriented science gateway for astrophysical visualization. Proc. of Parallel, Distributed, and Network-Based Processing, (to appear) (2011)
32. Shu, F.: Physics of Astrophysics: Volume I Radiation. University Science Books, New York, NY 10012 (1991)
33. Song, G., Zheng, Y., Shen, H.: Paraview-based collaborative visualization for the grid. Advanced Web and Network Technologies, and Applications, Lecture Notes in Computer Science **3842**, 819–826 (2006)
34. Woodring, J., Heitmann, K., Ahrens, J., Fasel, P., Hsu, C., Habib, S., Pope, A.: Analyzing and visualizing cosmological simulations with paraview. The Astrophysical Journal Supplement Series **195**(11) (2011)