Volume Graphics (2006)
T. Möller, R. Machiraju, T. Ertl, M. Chen (Editors)

GPU-Assisted Raycasting for Cosmological Adaptive Mesh Refinement Simulations

Ralf Kaehler[1], John Wise[2], Tom Abel[2] and Hans-Christian Hege[1]

[1]Zuse Institute Berlin, Germany
[2]Kavli Institute for Particle Astrophysics and Cosmology/Stanford University, USA

Abstract

In the recent years the advent of powerful graphics hardware with flexible, programmable fragment shaders enabled interactive raycasting implementations which perform the ray-integration on a per-pixel basis. Unlike slice-based volume rendering these approaches do not suffer from rendering artifacts caused by varying sample distances along different ray-directions or limited frame-buffer precision. They further allow a direct realization of sophisticated optical models.
In this paper we investigate the applicability of GPU-assisted raycasting to block-structured, locally refined grids. We present an interactive algorithm for artifact-free, high-quality rendering of data defined on this type of grid structure and apply it to render data of time-dependent, three-dimensional galaxy and star formation simulations. We use a physically motivated emission-absorption model to map the computed temperature and density fields to color and opacity.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation–Vieweing Algorithms I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism–Raytracing

1. Introduction

Nowadays astrophysics, in particular cosmology, probably like no other scientific area, is more and more depending on time-dependent, numerical 3D simulations, since for the most relevant research problems explicit analytical solutions do not exist and direct experiments are unfeasible. In order to interpret and verify such simulations, it is important to compare the results to observed image data. Interactive, photo-realistic visualization thus becomes an important tool. It allows the scientist to change parameters, e. g. the viewpoint as well as the shading model, on the fly to get an intuitive impression of the overall structure of the data. Furthermore it supports meaningful visual comparisons of different simulations.

Adaptive techniques are vital in this type of simulation since many length scales must be considered to accurately model the physical phenomena, which can range from several 10,000 light years (dynamics of the proto-galaxies) to several light hours for evolving stellar objects in certain regions. A specific adaptive approach for solving partial differential equations that is popular in astrophysics is called *AMR* (Adaptive Mesh Refinement). It was introduced by Berger et al. in the ’80s [BO84]. The basic idea is to combine the simplicity of structured grids and the advantages of local refinement by recursively overlaying regions of a coarse initial structured grid with grid patches of increasing resolution.

In this paper we present an interactive GPU-assisted raycasting algorithm for high-quality volume rendering of block-structured, locally refined grids like octrees or AMR hierarchies. In contrast to hardware accelerated, slice-based approaches our algorithm does not suffer from rendering artifacts due to limited frame-buffer precision or varying slice distances at boundaries of different resolution levels. We apply it to render three-dimensional galaxy and star formation simulations, which follow the hydrodynamics and gravity of gas and dark matter from the density fluctuation 400,000 years after the big bang. Based on these simulation data the influence of the first stellar objects within these galaxies on the surrounding gas is computed in a preprocessing step. The resulting temperature and density fields are processed by the GPU-assisted volume rendering algorithm that computes emission and absorption coefficients at each location within the data volume and allows to interactively adjust the relevant rendering parameters.

In the next Section we will review related work in the field. In Section 3 we sketch the grid structure on which the field variables are defined. Next will describe the GPU-assisted raycasting approach for locally refined structured grids (Section 4). In Section 5 we give some details about the cosmological simulations and specify the emission-absorption model we used to render the simulation data. We end with a comparison of the method to hardware-accelerated slice-based approaches in terms of image quality and rendering performance.

2. Related Work

Slice-based hardware-assisted volume rendering using 3D textures hardware was introduced by Cullip and Neumann in 1993 [CN93]. The underlying idea is to map the data volume to a 3D-texture, respectively a stack of 2D-textures and exploit graphics hardware to extract and blend a set of axes- or viewpoint-aligned slices to approximate the volume rendering integral.

The basic algorithm has been extended for multi-resolution data in [LHJ99, WWH*00, KH02]. Other acceleration techniques and also sophisticated optical models have been realized for slice-based approaches, see e. g. [KPHE02, EKE01, GWGS02].

Though slice-based approaches allow for interactive rendering of even larger data volumes, they suffer from several disadvantages, that might lead to visible artifacts: first the slice distance for perspective projection varies along different directions. Further currently hardware accelerated blending is only supported for 16-bit (or less) float render targets and thus the rendering performance is reduced drastically if 32-bit render targets are used, see e.g. [YNCP05]. This is especially problematic for adaptive grids, since in these cases usually wide viewing angles are required and a large number of highly transparent slices have to be blended.

Though the precision problem should be solved for future generations of graphics hardware, another source of artifacts for slice-based rendering of adaptive grids remains. It is due to small regions at slice edges at level boundaries, where the sample distance changes from one slice to another. Weiler et al. addressed this problem in detail in [WWH*00]. They presented an algorithm to detect these problematic regions and render the corresponding slice parts with the correct opacity that corresponds to the actual sample distance in these regions. However, since this approach requires connectivity information between adjacent cells and the resolution of adjacent blocks might differ by an arbitrary number of refinement levels the proposed solution is cumbersome for AMR data.

With the advent of programmable graphics hardware that supports flexible fragment programs, it became feasible to perform a ray-integration on a per-pixel based at interactive frame rates, as described in [RGW*03, KW03, SSKE05]. In this approach the data volume is converted to a 3D texture and a fragment program is executed for each pixel that is covered by the projected bounding box of the data volume. The ray is parameterized in texture coordinates and the integral can be computed as for software implementations. This approach has been adopted for the rendering of large datasets, see for example [HQK05]. In this approach the data domain is decomposed into subvolumes that are sorted and processed front-to-back according the actual viewpoint. GPU-assisted raycasting is very attractive for adaptive grids, since it does not suffer from the rendering artifacts mentioned above, which limits the achievable image quality for this kind of data using slice-based methods.

A software-based raycasting approach for AMR data has been proposed in [WOK*03]. Photorealistic volume visualization of planetary and reflection nebulae has been presented in [MHLH05]. In this approach the aim was to model the 3D shape of nebulae based on 2D image data, rather than to render simulation data of the galactic nebula formations. In particular the authors present an approximation for multiple scattering events based on a multi-resolution method.

3. The Grid Structure

The basic idea of AMR is to combine the simplicity of structured grids with the advantages of local grid adaption. In this approach the computational domain is covered by a set of coarse, structured subgrids $\Gamma^0_{l=0\ldots n} \subset \mathbb{R}^3$. The union of these subgrids is called the *root level* $\Lambda^0 := \bigcup_{m=0}^{n} \Gamma^0_m$.

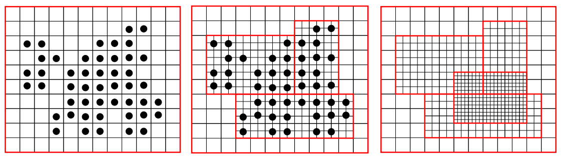

Figure 1: *Refinement process for AMR schemes: Cells that require refinement are determined using local error criteria (a) and clustered into separate subgrids (b), which cover the regions with higher resolution. This process is recursively continued until each cell fulfills the error criteria (c).*

During the computation, local error estimators are utilized to detect cells that require higher resolution. These cells are covered by a set of rectangular subgrids. Unlike in finite element approaches, these subgrids do not replace but rather overlay the refined regions of the coarse base grid. The equations are advanced on the finer subgrids and this refinement procedure recursively continues until all cells fulfill the considered error criterion, giving rise to a hierarchy of nested refinement levels, as shown in Figure 1.

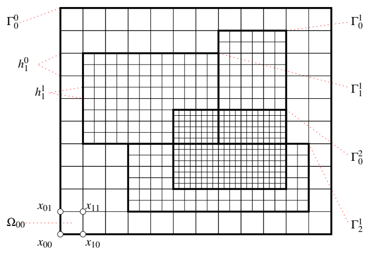

Figure 2: *Two-dimensional example of a structured AMR grid. The root level Γ_0^0 is refined by three subgrids $\Gamma_0^1, \Gamma_1^1, \Gamma_2^1$ that generate the refinement level Λ^1. Λ^1 itself is refined by one subgrid Γ_0^2.*

The mesh spacings of the finer grids are recursively defined by $\vec{h}^l := (h_0^{l-1}/r, h_1^{l-1}/r, h_2^{l-1}/r)$. The positive integer r denotes the so-called *refinement factor*, and $\vec{h}^0 = (h_0^0, h_1^0, h_2^0)$ is the mesh spacing of the root grid. In principle this factor r can differ for each direction and each level, but in order to simplify the notation we assume that it is constant. In the AMR approach, cells are either completely refined by cells of the next finer grid or remain completely unrefined. Each coarse cell can be decomposed into a set of r^3 cells of the next finer discretization. In the following the union of all level l subgrids $\Gamma_{m=0,1,2\ldots}^l$ is called *refinement level* Λ^l or just level l, compare Figure 2. By construction these levels are nested: $\Lambda^{l+1} \subseteq \Lambda^l \subseteq \Lambda^0$.

4. The GPU-assisted Raycasting Approach

The outline of the GPU-assisted raycaster for structured adaptive mesh refinement data is as follows:

- First the hierarchy of nested refinement levels is decomposed into blocks of disjoint, axis-aligned blocks that cover only cells from the same level of resolution.
- The blocks are traversed front-to-back in a view-consistent order and rendered separately into an offscreen render buffer. The resulting contribution from each block to the final image is stored in a offscreen render target.
- After all blocks are processed the offscreen target is rendered into the frame-buffer.

In the next subsections the single steps are discussed in more detail.

4.1. Space Partitioning

In order to take advantage of the GPU-assisted raycasting approaches the data volume should be processed block-wise, with blocks consisting of cells from the same resolution level. Rendering the separate subgrids directly would result in severe rendering artifacts. This is because of facts: In the AMR approach the patches of finer subgrids do not replace but rather overlap regions of coarser levels, so refined regions of the data volume would be rendered multiple times. It is further not possible in general to traverse the subgrids separately in a view-consistent order, due to the nesting of the refinement levels and the fact that subsets of the grids might form visibility cycles.

We therefore decompose the data domain into axis-aligned blocks $B_m^l \subset \Lambda^l$ with

$$(B_i^l \cap B_j^l) = \emptyset \vee (B_i^l \cap B_j^l) \subset (\partial B_i^l \cup \partial B_j^l) \quad \text{for } i \neq j$$

that consist either of cells that are refined by subgrids, or of cells which are not further refined. Each block is processed separately during the rendering phase, so it has to be ensured that no subsets of the blocks build visibility cycles for any viewpoint. In [KH02] a decomposition scheme is proposed that fulfills these constraints. In particular the resulting blocks are arranged in a kD-tree structure, allowing an efficient determination of the view-consistent order for each viewpoint. The resulting decomposition consists of three types of nodes:

- nodes representing areas of the computational domain which cover only cells that are further refined,
- nodes that contain only cells that are unrefined and represent leaves of the decomposition tree
- and nodes that contain refined and unrefined cells; these nodes are used to traverse the tree in a view-consistent order.

Notice that no data from the original AMR hierarchy is copied, but only bounding box information and pointers to the original data are stored in the kD-tree. Once a node is rendered for the first time, see next subsection, a 3D-texture is allocated on-the-fly for the subgrid the node refers to and the texture name is stored for later rendering passes. This ensures that only those textures needed for the actual viewpoint are generated. Furthermore it allows to keep the data of the original hierarchy out-of-core and just load it once it is required for the rendering.

We employ nearest-neighbor interpolation for cell-centered AMR data and trilinear interpolation for vertex-centered data. In the first case the texels are aligned with the centers of the cells, while in the second one they are aligned with the vertices of the grid. To avoid artifacts originating from discontinuities between adjacent subgrids for trilinear interpolation, adjacent texture-blocks share a row of data samples at their common boundary faces and the data at dangling nodes has to be replaced to the interpolated texel values of the abutting, coarse texture.

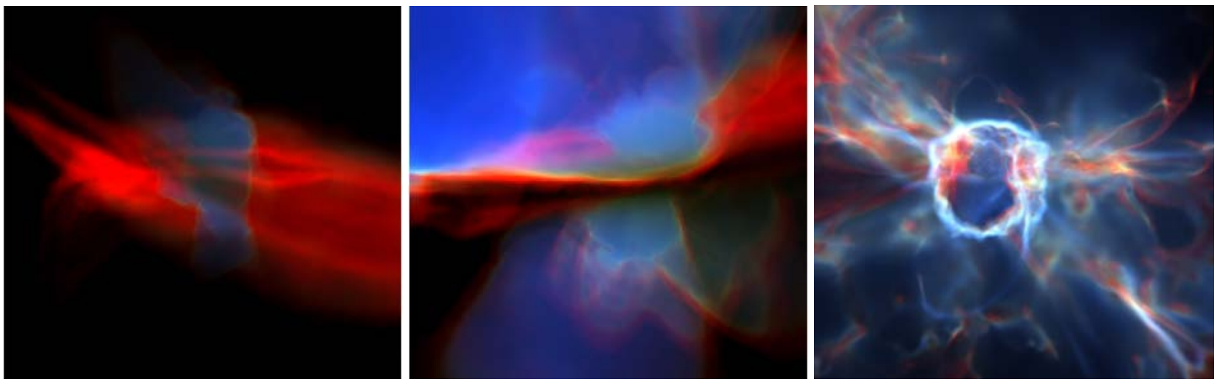

Figure 3: *Three time-steps from a star formation simulation rendered with the optical model presented in Sect. 5.1, that maps density and temperature fields to emission and absorption coefficients. The dark regions in the middle image result from non-illuminated gas that absorbs large amounts of light from behind it.*

4.2. Raycasting the Subgrids

The GPU-accelerated raycasting of the separate blocks is based on the single-pass approach described in [SSKE05]. In this approach first the 3D data texture and colortable texture are generated, respectively bound to different texture units. Next the front faces of the block's bounding box are rendered into an offscreen 2D RGB-floating point texture. The texture coordinates of the vertices are set identical to the vertex coordinates in order to access object space coordinates in the fragment shader. Within the fragment shader the ray direction in object space coordinates is computed for each pixel. Next the ray-entry point and the direction vector are transformed to texture coordinate space and the raycasting is performed. The data values are obtained via texture lookups and mapped to color and opacity via the colortable. Finally the resulting color and opacity of the each ray-segment are blended according to the front-to-back blending equation

$$\begin{aligned} C_{dst} &= C_{dst} + (1-\alpha_{dst})\alpha_{src}C_{src}\,, \\ \alpha_{dst} &= \alpha_{dst} + (1-\alpha_{dst})\alpha_{src}\,. \end{aligned}$$

In order to meed the requirements for rendering the multi-resolution block-structured data, we modified this approach. To prevent multiple computation of the ray direction for pixels that are contained in the projected screen space of multiple subgrids, we initially render the front-faces of the enclosing bounding box of the rendered subgrids, compute the ray direction for these pixels and store them in a separate texture that is used to lookup the ray directions when the separate subgrids are processed. When processing a block, for each ray the blended color and opacity from the last ray-segment are required as initial values. According to the specification of the *OpenGL framebuffer_object* extension [FBO] reading from the texture that is bound as a texture target is currently undefined. However, it worked in our scenario, so we additionally bound the texture render target onto a texture unit, in order to access it within the fragment shader.

Alternatively one could copy the area required from the render buffer in a separate rendering pass by binding the previous texture target, rendering the front-faces of the bounding box and loading a fragment shader that just copies the color and opacity information into a separate buffer. This buffer would be additionally bound as a texture in the raycasting pass to provide the initial values for each ray-segment.

In order to increase the rendering performance the sample distance along the ray is adapted to the actual resolution of the precessed block. Therefore corrected opacity values are precomputed according for the step size that is used for each refinement level according to

$$\alpha_i(l) = 1 - (1-\alpha_i(0))^{\frac{1}{r^l}}\,. \qquad (1)$$

Here $\alpha_i(0)$ are the opacity values used for the root level and r is refinement factor of the hierarchy. These values are stored as l separate 1D-RGBA-textures, one for each level of refinement. In general the last interval for each ray-segment will only be a fraction of the actual interval extension used for the integration. Artifacts due to wrong opacity values for these intervals are avoided by computing the correct value for the last integration interval on-the-fly.

More details about the implementation of the renderer will be given at the end of subsection 5.1.

5. The Application Scenario

Recently several studies (see [ABN02, BCL02]) established that the first stars in the universe were massive, about 100 solar masses, and were formed when the universe was only 200 million years old. These stars form isolated in their host gas cloud, in contrast to stars in clusters that we see in the local universe. Due to their large mass, these first stars are very luminous, which lead to intense heating and ionization of the adjacent regions. The first star calculations are *ab initio*,

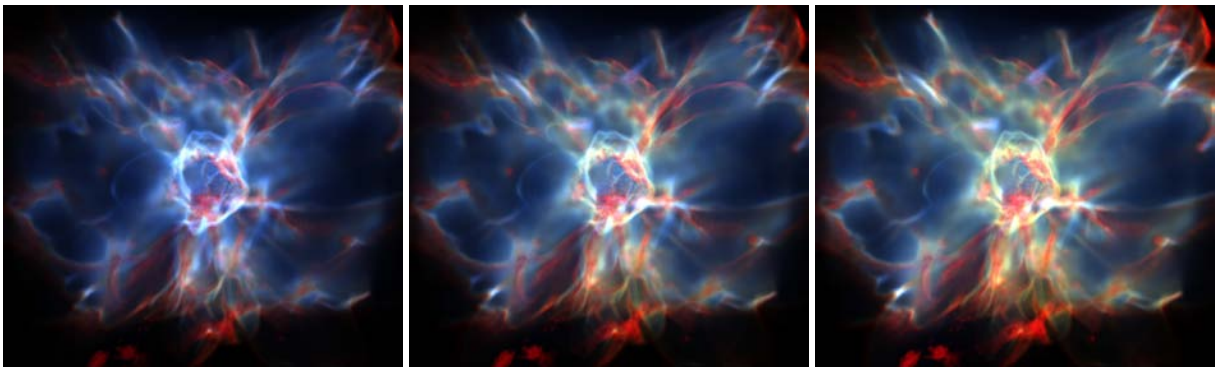
Figure 4: *The image sequence shows the effect of wavelength dependent absorption used in the optical model. From left to right the absolute amount of absorption was increased, resulting in a reddening of the resulting color.*

i.e. without any assumptions, and follow the hydrodynamics, gravity, and chemistry in a cosmological environment. The simulations end at the birth of the first star and do not follow its impact on the host gas cloud.

It was argued that radiative transfer in a cosmological sense would best split the highly anisotropic point sources, such as stars, from diffuse sources, such as recombinations of electrons and ions in dense regions. For the point sources, ray tracing is an effective way of accurately calculating the evolution of ionized regions. Abel et al. [AWB06] have successfully coupled an adaptive ray tracing scheme [AW02] into the AMR code *enzo* [BL97]. By doing so, they were able to precisely follow the radiation of the first star, whose radiation drives a shock through the surrounding gas that effectively expels the majority of it. At the end of the star's life, this gas is traveling outward at 30 km/s, and the gas up to 5,000 light years away have been ionized by this one star. The adaptive ray tracing scheme splits rays in the framework of HEALPix as they venture farther from its source to guarantee each grid cell in the hydrodynamic simulation contains at least 5 rays. This ensures that all relevant cells are sampled well. Furthermore it results in efficient ray tracing close to the source when not many rays are needed for a consistent calculation.

We use the GPU-assisted raycaster presented above to interactively render the output of this raytracing step. It is basically the resulting temperature and density distribution of the gaseous matter within the proto-galaxy. In the next subsection we describe the physically motivated optical model that was used to map the data to emission and absorption coefficients.

5.1. The Optical Model

In order to render the datasets, we must map the temperature to physical colors in the same manner as observational data. In telescopic observations, light is passed through various filters with a transmission function $\mathcal{T}_i(\nu)$ and post-processed to create color images. The most common filters, Johnson U, B, V, R, and I, are shown in Figure 5; however any other choice of filters is possible. We can utilize the temperature field to create a realistic colormap. A parcel of gas with a temperature T emits a blackbody spectrum,

$$B_\nu(\nu,T) = \frac{2h\nu^3}{c^2}\frac{1}{\exp(h\nu/kT)-1}, \tag{2}$$

where ν is frequency, h = 6.673 × 10^{-27} erg·s is the Planck constant, c is the speed of light, and k = 1.38 × 10^{-16} erg K^{-1} is Boltzmann's constant. Now we can convolve B_ν with a choice of three filters that correspond to RGB values. The flux F_i in the i^{th} filter, where i ∈ [1,2,3], is

$$F_i = \int B_\nu \mathcal{T}_i \, d\nu.$$

We normalize F_i by evaluating $\tilde{F}_i = \frac{F_i}{\max(F_1,F_2,F_3)}$ so that the filter with the maximum flux has a value of 1. For example, to represent realistic colors to the human eye, we choose the R, V, and B filters to represent RGB so that $RGB = [\tilde{F}_0(R), \tilde{F}_1(V), \tilde{F}_2(B)]$. To get the final emission coefficient we scale this term by the normalized gas density $\frac{\rho}{\rho_{max}}$ present at the voxel location.

In addition to emission, the scattering of light by particles larger than its wavelength λ can affect the incoming light. For the correct absorption properties detailed models are needed that must be adapted exactly to the type of simulations considered. E.g. the ionized regions need to be treated differently than non-ionized regions. For the latter one wavelength independent Compton scattering is appropriate. Such level of detail is beyond the scope of the current work. We rather use Rayleigh scattering for the whole computational domain as an attempt towards physical based rendering. This is motivated by the fact at the time when the first stars formed there was no dust present in the intergalactic medium. It was not produced until the first generation stars ended in supernova explosions. As a result, in our ap-

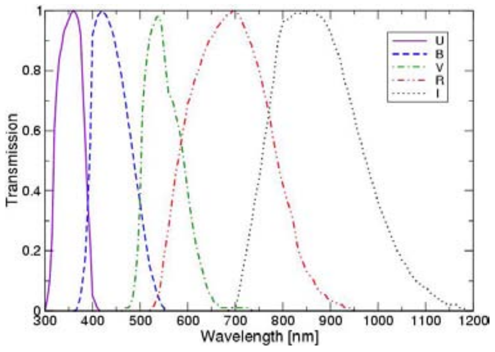

Figure 5: *Transmission functions $\mathcal{T}$ for Johnson U, B, V, R, I filters. These filters are most common when conducting astronomical observations.*

plication area the particle diameter of the medium is very small compared to the wavelength of the light that is scattered, respectively absorbed. So Raleigh scattering applies in our case (instead of Mie scattering which would become relevant if the particles would be bigger than $\frac{1}{10}$ of the light wavelength).

According to the Rayleigh law the intensity of the scattered light is proportional to density and λ^{-4}. Thus blue light is scattered more strongly than red light. A prime example of this process is the atmosphere scattering the Sun's radiation and causing the sky to be blue. To consider scattering, we correct the incoming blackbody emission by the scattering light intensity in the line of sight of the observer for the three different dominant wavelengths. Multiple scattering would change the appearance of our rendered objects as soon as it starts to scatter significant amounts of photons into the line of the sight. In the low density, dust free environment of the first proto-galaxies multiple scattering events have a low probability and thus we do not take them into account.

We precompute the color emission (without the density weighting) and the absorption coefficients for each level and stored them in 1D floating point textures. The density-weighting is performed on-the-fly. Since 6 floats have to be stored after the integration of each segment in order to accumulate the 3 color and 3 absorption values, we use two 32-bit floating point 2D RGB-textures as render targets to store the intermediate results of the raycasting passes. The two scalarfields for density and gas temperature are stored as 2-channel (*GL_LUMINANCE_ALPHA*) 3D-textures with one byte per channel.

6. Results

The algorithms have been implemented as extensions to Amira [SWH05], an object-oriented, expandable 3D data visualization system developed at ZIB.

	#sub-grids	**GPU-Raycasting**	**Slice-based**
Data I	65	19 fps	59 fps
Data II	670	7 fps	23 fps
Data III	4100	1 fps	4 fps

Table 1: *The first column denotes number of subgrids, the second one frame-rates for the GPU-assisted raycasting and the last one the frame-rate for the slice-based approach.*

We tested the presented approaches on a standard PC system (Pentium 4, 3.0 GHz, 2 GByte Main Memory) that was equipped with a *NVIDIA GeForce 6800 GT* graphics card with 256 MByte of graphics memory. *OpenGL* was used as the graphics API and the fragments shaders have been implemented with the OpenGL *Shading Language*. For generating the rendering targets we used the *OpenGL framebuffer_object* extension [FBO], since it is available on *Windows* and *Linux* systems.

We compared the performance and image quality of the GPU-assisted raycasting approach with a slice-based volume renderer. In this case we used a standard emission-absorption model with three color and one alpha channel that is realized via a user defined colortable. For the GPU-assisted raycasting approach we performed early ray termination, if the alpha component exceeded 0.99. We used three datasets with an increasing number of subgrids for the comparison. The number of cells at the root level was 32^3, 64^3, respectively 128^3 for the examples. The sampling distance was increased for each level of resolution and was about half the actual cell size. Information about the number of subgrids as well as the performance numbers are given in Table 1. For all examples the size of the viewport was $760x700$. The performance of the raycasting method was about 30% of the one obtained with the slice-based renderer. Renderings of the datasets are shown in Figure 7.

Figure 6 shows a comparison of the image quality of the two approaches. Though opacity corrections according to Equation 1 have been applied, the middle image shows severe rendering artifacts for the refined regions. These stem from insufficient framebuffer precision during blending the highly-transparent slices and from small regions at slice edges at level boundaries, where the sample distance changes from one slice to another, see Section 2. Since 32-bit floating-point precision is used in the raycasting approach and the rays are computed on a per-pixel basis, no artifacts are visible in the right image.

Examples of the emission-absorption model presented in Section 5.1 are shown in Figure 3 and 4. The first one shows three timesteps of the time-dependent star-formation simulation described in Section 5, while the latter one points out the effect of the wavelength dependent absorption model.

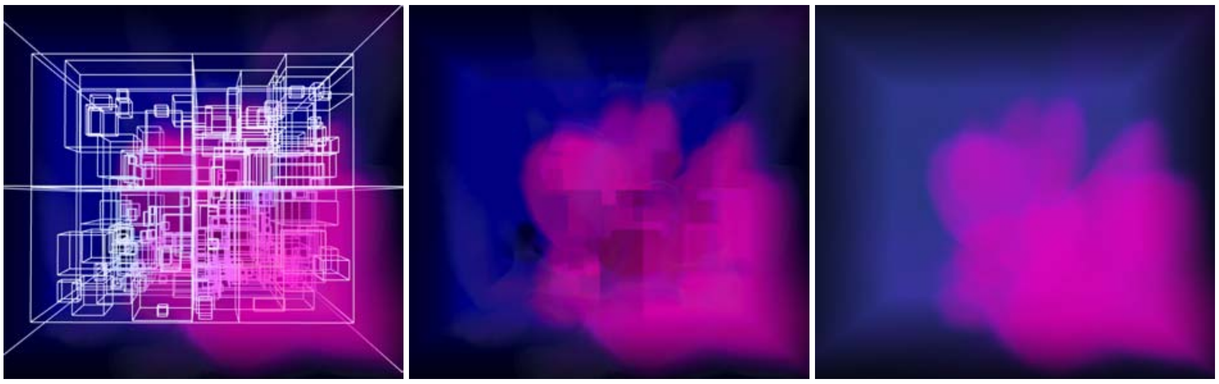

Figure 6: *A comparison of the image quality of slice-based (middle) and GPU-assisted raycasting (right) approaches. The left images shows the bounding boxes of the refined region of an adaptive simulation. The middle image show severe rendering artifacts for these regions, due to insufficient framebuffer precision to correctly blend the highly-transparent slices. Since floating-point precision is used in the raycasting approach, no artifacts are visible in that case.*

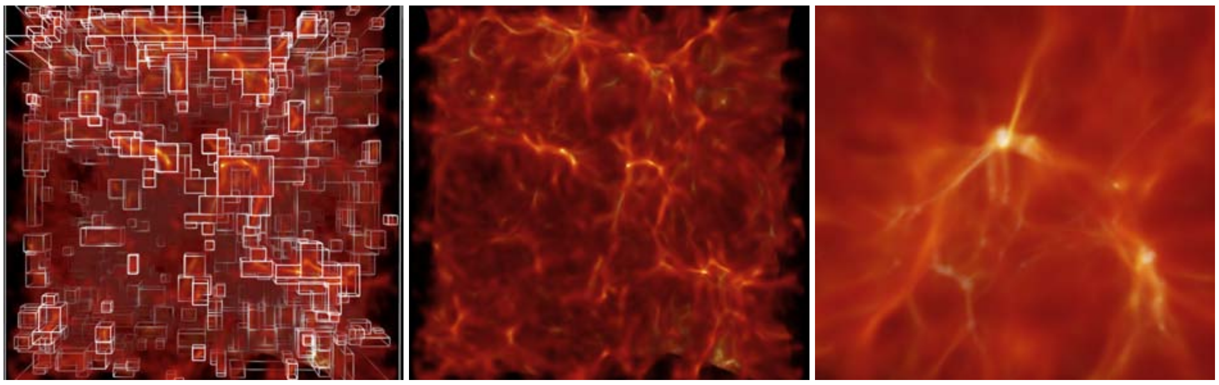

Figure 7: *Galaxy formation simulations that were used to compare the rendering performance of the slice-based and GPU-assisted raycasting approach.*

7. Conclusions and Future Work

In this paper we presented a frame-work for high-quality, interactive volume rendering of data defined on locally refined, block-structured grids. We chose a GPU-based raycasting approach that does not suffer from the drawbacks affecting the image quality of hardware-accelerated slice-based methods. It has superior image quality compared to slice-based methods and directly supports the implementation of advanced optical models, like the wavelength-dependent emission-absorption model used for rendering the cosmology simulation data. Though the performance of the GPU-based approach is only about 30% of the slice-based method for the tested datasets, this performance penalty is expected to decrease for future generations of graphics hardware.

There are several ways the extend the work presented in this paper. Pruning of tree-traversal based on local error criteria as well as empty space leaping should speed up the rendering performance considerably.

Internally the simulation computes the densities for the different components of the gas separately, so it would be interesting to use this data directly and apply measured extinction curves for the different compoments. Further for other kind of simulations, e.g. scenarios with dust present in the intergalactic medium, the effects of multiple scattering should be investigated, e.g. based on the approach described in [MHLH05].

8. Acknowledgements

This work was partially supported by the *Max-Planck-Institute for Gravitational Physics (Albert-Einstein Institute)*, Potsdam/Germany and the *NSF CAREER* award AST-0239709 from the *National Science Foundation*.

We thank Malte Clasen (*Zuse Institute Berlin*) for fruitful discussions.

References

[ABN02] ABEL T., BRYAN G. L., NORMAN M. L.: The Formation of the First Star in the Universe. *Science 295* (Jan. 2002), 93–98.

[AW02] ABEL T., WANDELT B. D.: Adaptive ray tracing for radiative transfer around point sources. *Monthly Notices of the Royal Astronomical Society 330* (Mar. 2002), L53–L56.

[AWB06] ABEL T., WISE J. H., BRYAN G. L.: The HII Region of a Primordial Star. *Astrophysical Letters* (2006).

[BCL02] BROMM V., COPPI P. S., LARSON R. B.: The Formation of the First Stars. I. The Primordial Star-forming Cloud. *Astrophysical Journal 564* (Jan. 2002), 23–51.

[BL97] BRYAN G. L., L. N. M.: A Hybrid AMR Application for Cosmology and Astrophysics. In *Workshop on Structured Adaptive Mesh Refinement Grid Methods, IMA Volumes in Mathematics No. 117* (1997), Chrisochoides N., (Ed.), pp. 165–+.

[BO84] BERGER M. J., OLIGER J.: Adaptive mesh refinement for hyperbolic partial equations. *Journal of Computational Physics 53* (1984), 484–512.

[CN93] CULLIP T., NEUMANN U.: *Accelerating volume reconstruction with 3D texture mapping hardware*. Tech. Rep. TR93-027, Department of Computer Science at the University of North Carolina, Chapel Hill, 1993.

[EKE01] ENGEL K., KRAUS M., ERTL T.: High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware* (2001), pp. 9–16.

[FBO] OpenGL Framebuffer-Objects Specification. http://oss.sgi.com/projects/oglsample/registry/EXT/framebuffer_object.txt.

[GWGS02] GUTHE S., WAND M., GONSER J., STRASSER W.: Interactive rendering of large volume data sets. In *VIS '02: Proceedings of the conference on Visualization '02* (Washington, DC, USA, 2002), IEEE Computer Society, pp. 53–60.

[HQK05] HONG W., QIU F., KAUFMAN A.: Gpu-based object-order ray-casting for large datasets. In *Proceedings of Fourth International Workshop on Volume Graphics* (2005), pp. 177–240.

[KH02] KÄHLER R., HEGE H.-C.: Texure-based Volume Rendering of Adaptive Mesh Refinement Data. *The Visual Computer 18*, 8 (2002), 481–492.

[KPHE02] KNISS J., PREMOZE S., HANSEN C., EBERT D.: Interactive translucent volume rendering and procedural modeling. In *Proceedings of IEEE Visualization 2002* (2002), IEEE Computer Society Press, pp. 109–116.

[KW03] KRUGER J., WESTERMANN R.: Acceleration techniques for gpu-based volume rendering. In *VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03)* (Washington, DC, USA, 2003), IEEE Computer Society, p. 38.

[LHJ99] LAMAR E. C., HAMANN B., JOY K. I.: Multiresolution techniques for interactive texture-based volume visualization. In *IEEE Visualization '99* (San Francisco, 1999), Ebert D., Gross M., Hamann B., (Eds.), IEEE, pp. 355–362.

[MHLH05] MAGNOR M., HILDEBRAND K., LINTU A., HANSON A.: Reflection nebula visualization. In *Proceedings of Visualization 2005* (2005), IEEE, pp. 255–262.

[RGW*03] ROETTGER S., GUTHE S., WEISKOPF D., ERTL T., STRASSER W.: Smart hardware-accelerated volume rendering. In *VISSYM '03: Proceedings of the symposium on Data visualisation 2003* (Aire-la-Ville, Switzerland, Switzerland, 2003), Eurographics Association, pp. 231–238.

[SSKE05] STEGMAIER S., STRENGERT M., KLEIN T., ERTL T.: A simple and flexible volume rendering framework for graphics-hardware-based raycasting. In *Fourth International Workshop on Volume Graphics* (Washington, DC, USA, 2005), pp. 187– 241.

[SWH05] STALLING D., WESTERHOFF M., HEGE H.-C.: Amira: A highly interactive system for visual data analysis. In *The Visualization Handbook* (2005), Hansen C. D., Johnson C. R., (Eds.), Elsevier, pp. 749–767.

[WOK*03] WEBER G. H., OEHLER M., KREYLOS O., SHALF J. M., BETHEL W., HAMANN B., SCHEUERMANN G.: Parallel cell projection rendering of adaptive mesh refinement data. In *Proceeding of the IEEE Symposium on Parallel and Large-Data Visualization and Graphics* (Los Alamitos, California, 2003), Koning A., Machiraju R., Silva C. T., (Eds.), IEEE, IEEE Computer Society Press, pp. 51–60.

[WWH*00] WEILER M., WESTERMANN R., HANSEN C., ZIMMERMAN K., ERTL. T.: Level-of-detail volume rendering via 3D textures. In *IEEE Volume Visualization and Graphics Symposium 2000* (2000), pp. 7–13.

[YNCP05] YUAN X., NGUYEN M. X., CHEN B., PORTER D. H.: High dynamic range volume visualization. *Proceedings of IEEE Visualization 2005* (2005), 327–334.