Visualization needs and techniques for astrophysical simulations

To cite this article: W Kapferer and T Riser 2008 New J. Phys. 10 125008

View the article online for updates and enhancements.

You may also like

- Analysis of algorithms magenta with 128 bit key length accompanied by shifting hash securing data
  Riandy Yap

- A computer simulation of the metallic artifacts in MRI caused by dental magnetic attachments
  Masatake Akutagawa, Takahiro Emoto, Tetsuro Tokitani et al.

- Comparison of Numerical Predictions and Experimental Results of Water Percolation in PEM Fuel Cell Porous Transport Layers
  Ezequiel Medici and Jeffrey Allen
Visualization needs and techniques for astrophysical simulations

W Kapferer\textsuperscript{1,3} and T Riser\textsuperscript{2}

\textsuperscript{1} Institute for Astro- and Particle Physics, University of Innsbruck, Technikerstrasse 25, 6020 Innsbruck, Austria
\textsuperscript{2} Institute for Mathematics, University of Innsbruck, Technikerstrasse 21a, 6020 Innsbruck, Austria
E-mail: wolfgang.e.kapferer@uibk.ac.at

New Journal of Physics 10 (2008) 125008 (15pp)
Received 31 March 2008
Published 1 December 2008
Online at http://www.njp.org/
doi:10.1088/1367-2630/10/12/125008

Abstract. Numerical simulations have evolved continuously towards being an important field in astrophysics, equivalent to theory and observation. Due to the enormous developments in computer sciences, both hardware- and software-architecture, state-of-the-art simulations produce huge amounts of raw data with increasing complexity. In this paper some aspects of problems in the field of visualization in numerical astrophysics in combination with possible solutions are given. Commonly used visualization packages along with a newly developed approach to real-time visualization, incorporating shader programming to uncover the computational power of modern graphics cards, are presented. With these techniques at hand, real-time visualizations help scientists to understand the coherences in the results of their numerical simulations. Furthermore a fundamental problem in data analysis, i.e. coverage of metadata on how a visualization was created, is highlighted.
1. Introduction

To understand the physical processes involved in all kinds of astrophysical objects, complex simulations are essential. The great success of numerical simulations in the field of astrophysics started in 1960s. All those early simulations had one thing in common: low resolution in terms of present numerical simulations. In the famous work by Toomres [1] on the origin of disrupted galaxies, only 120 particles, representing the matter distribution in a spiral galaxy, were incorporated. Only one decade later the first cosmological simulations revealed the importance of dark matter as a concept to understand structure formation in the universe [2]. These simulations were done with a thousand particles, one magnitude more than a decade before. In addition to the increasing resolution the physics embedded in the simulations became more and more complex. Hydrodynamics or magnetohydrodynamics coupled with gravity for the evolution of the gaseous matter with different source terms for cooling and heating, coupled with radiation are nowadays used to model the complex processes in astrophysical objects and their various interactions. A state-of-the-art simulation on the formation of large-scale structures in the universe, including gas physics, magnetic fields and element abundances, results in time series of up to 50 scalar and velocity fields. Keeping in mind that all those quantities are typically stored at high resolution, it is obvious that only visualization enables the scientist to interpret the results. Furthermore visualizations often reveal flaws in the models and the calculations, which make them very powerful in the debugging phase of a code.

In this paper, visualization techniques for complex data sets beside the current visualization methods in astrophysical simulations are presented. A special focus is given on real-time visualization.

2. The problem of multiscale simulations

Astrophysical objects of all kinds exhibit enormous ranges in physical quantities, like densities, temperatures or dimensions. For an example, simulations of large-scale structure formation...
Figure 1. Volumetric visualization of the dark matter distribution in a model galaxy cluster in a 10 Mpc h$^{-1}$ along the side cube. Whereas the left panel shows the dark matter density in linear colourmap, the right panel gives the square of the density in the same linear colourmap.

are done on Mpc (3.08 × 10$^{22}$ m) scales, with densities spanning over several magnitudes and temperatures in the range of 10$^4$ to several 10$^8$ K. One problem very often appearing in visualizations of such datasets is the limited range of dynamics in volumetric renderings. A way to overcome these problems is rendering from preprocessed data. As an example, the square of the density of a fluid reveals more substructure than directly rendering the density, as it pronounces denser regions more than less dense. An example is given in figure 1, the distribution of dark matter in a galaxy cluster simulation at the present epoch is shown. Whereas the left panel shows the dark matter density in linear colourmap, the right panel gives the square of the density in the same linear colourmap. By showing the square of the dark matter density an additional quantity is given, a direct image of dark-matter annihilation radiation, given that dark matter interacts with itself not only gravitationally. It is important to note that a visualization needs to be adjusted to the physical quantity that is examined.

Another big issue in three-dimensional simulations of astrophysical objects is the dynamic in the spatial scales. With mesh refinement techniques or Lagrangian-based simulations, often the situation occurs that small-scale features are blurred out in visualizations. In principle two ways to solve this problem are possible, high-resolution displays and renderings, or the natural way to zoom into the computational domain. Magnifying regions of the computational domain includes several problems, one easily gets lost in large domains and very often the information in front of the zoomed region is skipped, which leads to misinterpretations in the visualization. Therefore high-resolution displays are increasingly important in the investigation of high-resolution simulations. A possible construction of a high-resolution visualization display with custom high-definition systems and standard PC hardware is described in section 5.

3. The power of real-time visualization

Enabling the user to alter a visualization in real-time is the goal of all real-time visualization software packages. Strictly speaking no exact definition of what is real-time is given. In principle one could consider a responsive visualization being executed in real-time at a rendering speed in the range of 30 frames per second. To achieve this behaviour two principle approaches
Table 1. A comparison of the computational power and memory bandwidth of a commonly used CPU and GPU.

|                         | CPU (Intel Core 2) | GPU (Nvidia G80) |
|-------------------------|--------------------|------------------|
| Total floating point operations per second (FLOPS) | 24 GFLOPs per core | 576 GFLOPs per card |
| Peak memory bandwidth   | 10.67 GB s$^{-1}$  | 103.6 GB s$^{-1}$ |

are suitable, the first is via software rendering on the central processing unit (CPU) and the second way is hardware accelerated rendering on the graphics processing unit (GPU). The software rendering can be done on a single machine, in a cluster environment using all different paradigms of parallelization, like Open Multi-Processing (OpenMP) for shared memory architectures, Message Passing Interface (MPI) for a Beowulf [3] computing cluster or in a GRID environment [4]. Often such techniques enable the user to investigate very large datasets, which would otherwise not fit in the memory of a single visualization workstation, but the visualization process is often quite slow and Beowulf clusters or shared memory machines are restricted in access. The path many visualization tools follow nowadays is hardware accelerated rendering. The hardware accelerator most often used is the graphics card. Whereas software real-time raytracing solutions, such as OpenRT [5] and MANTA [6], require expensive computational environment, modern graphics hardware is a cheap alternative with some limitations, e.g. memory size.

3.1. The power of state-of-the-art graphics cards

In the past, graphics cards were either very expensive special purpose hardware devices for specific rendering applications or low-end hardware for the entertainment industry. They were basically solely capable of accelerating the rendering process of textured triangles in a three-dimensional space. Given their limited flexibility, scientific utilization was restricted. In particular, the results from numerical simulations of astrophysical systems can benefit greatly from utilizing techniques beyond simple textured polygons. In addition, large volumetric data sets could not be processed in real-time on those early graphics cards.

In the past 10 years graphics cards evolved enormously, thanks to an extraordinary growing market in the gaming industry. However, the most important development for graphics cards in the field of scientific computing was the very recent introduction of programmable, hence more flexible, graphics pipelines. Now the path was cleared for graphics cards to overcome their limited use in polygon rendering and become general-purpose highly parallelized vector processing units. Combining the number-crunching power of modern graphic-accelerators with programmable graphics interfaces improves the creation process of scientific visualization.

A comparison of the computational power and the memory bandwidth of a commonly used CPU and GPU is given in table 1. The direct comparison reveals the powers of current graphics accelerators.

The picture drawn by these numbers may suggest that modern graphics cards makes any CPU driven software obsolete. However, there are severe limitations to the programmability of these devices. Firstly, the software has to be specifically designed for each single graphics-card generation; secondly, GPU programs lack the flexibility of CPU driven code; thirdly, the power of the graphic processing unit is only available if the problem consists of tasks which can be highly parallelized.
While most of the tasks within the field of visualization fit perfectly in the schemes of GPU driven rendering, some still cannot be performed well in parallel, e.g. depth sorting transparent objects. On the other hand, more and more vectorized numerical simulations can be swapped to the GPU to greatly reduce execution times. In conclusion, modern general-purpose graphics hardware augments the existing CPU driven environment with enormous number-crunching potential.

However, since the introduction of these graphic processing units only happened very recently, applications that completely benefit from that type of hardware are still in their infancies.

3.2. Commonly used freely available real-time visualization packages

Visualization packages provide the user common visualization techniques to investigate data. With slices through the computational domain, height-fields of 2D scalar-fields, isosurfaces of 3D data sets, vector glyphs and volume renderings, the user can explore the computational domain in detail. Not all real-time visualization packages offer all techniques, some are more appropriate for scalar-fields, some are better suited for tensor-fields and so on. In this section, we want to give a short overview over commonly used real-time visualization tools and their advantages and disadvantages. This list is not meant to be complete.

1. IFrIT—Ionization FRont Interactive Tool. The IFrIT was developed by Nick Gnedin and is distributed under the GNU GPL license. The tool is capable of visualizing scalar, vector- and tensor-field data (a symmetric $3 \times 3$ tensor) in 3D space. In addition particle data with several optional attributes per particle can be displayed. The package offers slices, isosurfaces and volumetric rendering for scalar field data. Vector-field data is represented by vector glyphs or streamlines along the vector field. Besides IFrIT’s own file format, VTK [18] files can be processed, and an extension to read Gadget [7] files, a commonly used tree/smoothed particle hydrodynamics (SPH) code in astrophysics, exists. The software package can be operated with two specialized script languages: an animator script and a control script. Basic tools for data analysis and measurements within the computational domain are integrated. IFrIT and its documentation can be found at http://home.fnal.gov/gnedin/IFRIT/.

2. The MayaVi data visualizer. MayaVi is a scientific data visualizer that was developed by Prabhu Ramachandran. It is distributed under the conditions of the BSD license. It runs on any platform where both Python and VTK are available (including Mac OSX or Windows). MayaVi supports VTK data sets including rectilinear, structured and unstructured grid data and also polygonal data. Besides the capability of visualizing cuts, isosurfaces, volume renderings and vector glyphs of data sets, MayaVi offers quite complex filter modules to investigate the computational domain, like delaunay triangulation of points and many other VTK-filters. According to the project’s website a new version will be released in the near future, called MayaVi2 with an improved user- and scripting-interface in Python.

3. ParaView. ParaView supports a distributed computation model to process large data sets. The tool is developed by Kitware, Sandia National Laboratories and CSimSoft and freely available in the stable version 3.0. When compiled together with the MPICH [8] library, the tool can be integrated in a Beowulf computing cluster [3] architecture. While the software is freely available the full documentation has to be paid for. ParaView supports
the volume rendering of unstructured meshes and is capable of visualizing adaptive-refined mesh data sets. The packages can be obtained from http://www.paraview.org/New/index.html.

4. VisIt. VisIt is an interactive parallel visualization and graphical analysis tool. It was developed by the Department of Energy (DOE) Advanced Simulation and Computing Initiative (ASCI). The tool produces contour slices (cone, plane, sphere), vector visualizations with glyphs, isosurfaces and volumetric renderings. It supports two- and three-dimensional point, rectilinear, curvilinear, and unstructured meshes, as well as adaptive-refined structured meshes. VisIt is powerful in data exploration with many different tools to extract information from the computational domain. It offers native support for many different file formats, like FITS [9], FLASH [10], ENZO [11] or ZEUSMP [12] files.

In figure 2, isosurfaces of the distribution of a distinct iron value in a model galaxy cluster within a box (1.5 Mpc h$^{-1}$ on a side) are shown. The isosurfaces were generated from the same VTK data file in the software packages described above. The resolution of the visualized cube is 128$^3$ cells—all tools reproduce the same isosurface. The extraction and the rendering of the isosurface take less than a second for all visualization packages. For computational domains with resolutions above 256$^3$ cells the extraction of the isosurface can become quite slow (tens of seconds), nevertheless the rendering of the extracted isosurface is still in the regime of real-time.

**4. Facilitating the power of modern graphics cards**

The tools mentioned in section 3.2 already offer a wide range of visualization techniques, and are sufficient to represent a variety of astrophysical data. However, they do not facilitate the capabilities of modern freely programmable graphics pipelines. Therefore, any custom visualization has to be individually prepared and precalculated. It is obvious that the real-time feature can be lost if for every change in an arbitrary property (time range, projection of the velocity field along the line of sight, ...) the precalculation has to be done for the
whole data set. As mentioned before, state-of-the-art graphics cards provide means to overcome these limitations and preserve real-time requirements. The so-called shaders are arrays of freely programmable vector processors on the GPU that can execute an arbitrary shader program for every single displayed primitive. This allows the user to relocate the necessary custom calculations from the CPU to the graphics card, benefiting from its calculative power. Even recent standard rendering techniques like the GL_POINTS_SPRITES mentioned in the section below are implemented by the driver using shaders.

Independently, utilizing those shader programs can also improve the image quality of scientific visualizations. Below one specific kind of astrophysical data is used to illustrate possible enhancements to the well-established visualization software.

4.1. Improving visual appearance

Particle data set visualization can easily be improved by using programmable parallelized computational units on the graphics card. In order to render particle data sets in a three-dimensional scene, traditional techniques use point representations of fixed size (e.g. GL_POINTS with 1 pixel in size in openGL [13]). They fit well in the well-established fixed pipeline architecture, and already deliver an appealing visual representation. Once a user wants to zoom in, however, the main drawbacks of point primitives become obvious. Since their size is constant in screen space, any particle is always represented by a fixed amount of pixels (usually one), independent of its distance from the viewpoint. This results in a complete loss of any three-dimensional experience. An example is given in figures 3(a) and (b). In this figure, different visualizations showing the result of an SPH simulation is given. A model galaxy suffers from ram-pressure as it moves through an ambient medium (not shown in the visualizations). The gas is coloured red, whereas the stellar component of the model galaxy is coloured green and yellow. In figure 3(b), the particles in front seem to vanish while zooming in comparison to figure 3(a). To cope with that using traditional techniques one can either add additional geometry like using little spheres instead of points, or calculate the screen size of every particle separately on the CPU. Given the amount of data visible in each frame, both would inevitably drop the frame-rate drastically. Additional problems would arise like adding dynamic level of detail to glyph representations. Rendering techniques utilizing recent GPUs like GL_POINTS_SPRITES are not only capable of individually adjusting the size of the point representation with respect to the distance of the viewer without any noticeable frame rate drop, figures 3(c) and (d), combined with shader programs they can even add visual ‘eye candy’ to further enhance the three-dimensional experience. As seen in figure 3(e), simulated and fully interactive lighting was added to the point primitives giving them the visual appearance of a lit sphere instead of a monochrome circle. Invoking the power of modern graphics cards the described quality enhancement can easily be performed in real-time, without adding additional geometry. This results in no further consumption of graphics memory.

However some commonly used techniques are still not enhanced by using modern GPUs. Transparency effects for instance require visualization data to be displayed from back to front, therefore a sorting is needed for every change in camera angle. This task performs badly on graphics hardware and must still be done on the CPU. Considering that the given examples are easily capable of displaying over 10 million particles simultaneously in real-time (i.e. more than 30 frames per second) any particlewise operation that requires the CPU will significantly drop the frame rate.
Figure 3. Different levels of visualization complexity of particle data using the programmable hardware units on graphic cards are shown. The images give the distribution of gas and stellar components of a model galaxy, feeling the ram-pressure as it moves with 1000 km s\(^{-1}\) through a surrounding rarefied gas (not shown in the visualizations). By adding more features via the programmable units of the graphics card, like adjusting the size of a circle as a function of the viewpoint or introducing a lit sphere (e), the user gets an extraordinary three-dimensional impression of the simulation. Note that all operations are completely interactive in real-time, without any noticeable delay.

### 4.2. Gaining deeper insight into the computational domain

#### 4.2.1. Interactive 3D movies.  Investigating the dynamical behaviour of astrophysical objects is a major goal in numerical simulations. These usually produce data sets discretized in time-space, i.e. snapshots of the simulated objects. Traditionally these snapshots are either visualized separately or a preprocessed movie extracted from a set of snapshots with fixed visualization properties and camera-angles or a predefined camera-path is created. Introducing real-time adjustment for all these components simultaneously, yields a fully interactive time-driven animation. To obtain a fluent motion for every object in a time-dependent simulation at any given time, the trajectory has to be interpolated between time adjacent snapshots. Since astrophysical simulations produce huge amounts of data it is not possible to interpolate the data on the CPU without losing real-time capabilities.
This calculation can be independently performed for every single particle, perfectly fitting the paradigm for GPU programming. Depending on the type of interpolation (in the example shown, only linear interpolation was used), the approximate trajectory of every particle is calculated completely on the GPU. This enables the user to utilize the highly parallelized architecture of modern graphics hardware and provides perfectly fluent motions in real-time while maintaining complete interactivity. In the movie of figure 4, the real-time approach in space and time can be seen.

4.2.2. Real-time vector field visualization. By investigating the velocity-field of model galaxies often one has to project the velocity vectors of the galaxies’ components on the current line of sight, in order to obtain an image which is comparable with observations. This requires a recalculation of the appropriate colour for every change in the viewing angle, and is therefore again perfectly suited for execution on the GPU. In figure 5, the described projection of the velocity field of two interacting model galaxies is shown. The applied colourmap represents blue- and red-shifted regions.

The movie of figure 6 shows the real-time behaviour of this approach.

5. High-resolution display techniques

A commonly used technique for high-resolution displays is tiled walls. They can be assembled with active screens of different technologies like plasma or LCD or with projectors. Using the advantage of real-time visualizations together with high-resolution tiled walls provides the viewer deep insight into the numerical simulation. In this section, a short description on how to build a tiled wall with standard components is given.

Modern PCs are capable of managing more than one graphics adapter. For full glx (OpenGL Extension to the X Window System) support under Linux the graphics driver options are combined with Xinerama [14], which is an extension of the X-server enabling the window
**Figure 5.** Visualizing a simulated collision of two galaxies with respect to red- and blue-shift of moving particles. (a)–(d) Precollision frame (e)–(f) mid-collision frame with varying camera angles.

**Figure 6.** This movie (available from stacks.iop.org/NJP/10/125008/mmedia) shows a demo of the real-time visualization approach in projecting the velocity vector on the line of sight using the shader programmes.
managers to use several physical displays as one virtual display. In figure 7, a possible mounting of a tiled wall for back projection is illustrated. By integrating two identical graphics cards (e.g. 2x NVIDIA GeForce 8800GTX) into a standard PC, four full high-definition projectors can be used simultaneously. This results in an overall resolution of 3840 × 2160 pixels. Using back projection in combination with mirrors, as indicated in figure 7, the space requirement is reduced. Standard PC components reduce the costs below €15 000, including the workstation, the projectors, the back projection screen and mounting materials.

In figure 8, the tiled wall, built at the University of Innsbruck, is shown. Note that on such a system all visualization packages described in section 3.2 maintain their real-time behaviour.

The advantage of a high-resolution tiled-wall projection in comparison with a single projection in standard high-definition resolution lies in the combination of the large projection area together with higher resolution. The visual investigation of all different kinds of data, whether observed or simulated, becomes much more natural and easier. Small-scale structures are detectable faster and one gets less easily lost in the data than with the typical zoom in/out technique. Another important feature of tiled walls, especially with back-projection techniques, is the presentation and discussion of data in a group. All group members have a direct view of the data.
6. Different simulation techniques—different visualization needs

Not all visualization techniques are applicable to all kinds of data sets derived by different numerical techniques. Along general lines, simulations represent their computational domain either by particles or by grids, structured or unstructured. Particles can have many different quantities, like temperatures, velocities or densities. In SPH, the fluid is represented by particles folded by smoothing kernels with distinct smoothing lengths, therefore visualization techniques for grid-based schemes are not directly applicable. On the other side, many visualization techniques like isosurfaces are implemented in graphics packages only for grid-based data sets. To overcome this problem one can sample the particles on structured, unstructured or adaptively refined grids. By applying such sampling algorithms often resolution is intensively reduced or artificial structures are introduced. This could lead to misinterpretations. Figure 9 gives three isosurfaces from the same dark matter particle distribution. The difference in the three isosurfaces is the sampling of the dark matter distributions on different uniform grids with $64^3$, $128^3$ and $256^3$ cells, respectively. The sampling was done with a cloud in cell algorithm. The threshold of the three isosurfaces is held to be constant for all. Not only are different structures visible in the isosurface drawn from the higher resolved grids, but also the volume of the isosurfaces changes from 17.8 to 21.4 $(\text{Mpc h}^{-1})^{-3}$, depending on the resolution of the grid. How important visualizations are for the discovery of flaws and systematic effects in models is also visible in figure 9. The isosurfaces reveal a cross-like structure in the dark matter distribution. This structure is caused by the box size of the computational domain, which is too small to contain a representative piece of the universe. Without the help of visualizations, such flaws in the data may remain undetected. From this basic example it is obvious that visualization techniques for grid-based algorithms are not directly adaptable to particle-based algorithms.

A solution to this specific problem of isosurfaces from particle data is presented in [15]. Rosenthal et al developed a method for direct surface extraction from SPH simulation data. The method takes into account the position and the smoothing length of the gas particles in SPH simulations and constructs isosurfaces in a very precise way.
Figure 9. Three isosurfaces with a fixed threshold from the same dark matter distribution obtained from three different resolved grids. The difference in substructures as well as the different volumes of the isosurfaces can easily be seen.

A common problem in visualizations of astrophysical simulations is the size of the computational domain together with the finite speed of light. Many scientists show movies or images of simulated objects with diameters of Mpcs. Visualizations of such simulations often do not take into account the fact that the light from the simulated domain does not arrive at the virtual observer from all areas of the computational domain at the same time. In order to compare observations with simulations, e.g. galaxy distributions on large scales, often mock data from simulations are extracted, which take the finite speed of light into account.

7. The need for information on visualizations

A typical problem in comparing results of numerical simulations is provenance—recording metadata about how results were extracted and which algorithms on the raw data were necessary. Showing an elaborate visualization does not necessarily enable the viewer to understand all the processes involved. When comparing different numerical algorithms the problem becomes urgent. One possibility to address this problem is metadata in data files and visualizations like images or movies. Unfortunately not all file formats are suitable for carrying metadata in an appropriate way. One approach is the hierarchical data file format (HDF [16]) with its capability to store appropriate metadata on the simulation or data-reduction, nevertheless no automated process exists. Another way to keep track of how visualizations or results in general were created from raw data of simulations is realized in VisTrails [17]. This is a scientific workflow management system that provides support for data exploration and visualization. Workflows can be setup in an interactive way. In figure 10, a workflow and the corresponding result in VisTrails are shown. The isosurfaces show different element abundances within the intra-cluster medium in a hydrodynamical galaxy-cluster simulation. Different workflows can be created in parallel, which enables the user to start easily from existing data explorations. In addition, other users are able to follow exactly the different steps involved in creating visualizations. The tool keeps track of all changes, comparable with systems like subversion, in the field of software development.
Figure 10. VisTrails workflow and visualization output from an intra-cluster medium galaxy cluster simulation. The six isosurfaces are drawn from the same data file. With the help of the workflows the user is able to keep a record of different simulation techniques and data manipulations. In addition, the workflow enables other persons to understand in detail how the visualization was created.

Providing the data, the list of manipulations and the corresponding workflow enables the viewer of a visualization to understand what is shown in detail. In addition, the user can alter the visualization for her/his own requirements in an easy way, by starting from a previous workflow. Similar activities are also incorporated in workflow driven applications like the virtual observatory (VO) or the Planck Process Coordinator Workflow Engine (ProC).

8. Summary and outlook

With the enormous developments in computer science, both hardware and software, astrophysical simulations result in huge amounts of data with high complexity. Therefore the needs of visualization techniques, or data reduction in general, have to adapt. With modern standard PC hardware many scientists are able to investigate their simulations in real-time by using software packages that provide appropriate tools. In this paper some of these commonly used freely available packages are presented, like IFrIT, MayaVi, ParaView or Visit. All these packages provide stable modules to visualize all different kinds of data sets. Many simulations are time-dependent and therefore the capability to treat time series is necessary. One way to address this problem is real-time visualization using graphics hardware, i.e. programming the shader units on a state-of-the-art graphics adapter. With this technique it is possible to move virtually through both space and time, in the same real-time quality. Making use of the enormous computational power of recent graphics adapters, many complex algorithms can be performed in real-time parallelized directly on the GPU, like projection of velocities along the line of sight, a powerful approach to study velocity fields in 3D. Besides the typical visualization techniques for grid- and particle-based algorithms, new techniques have to be developed. One example is direct extraction of isosurfaces from particle/SPH data. Another important issue, especially in astrophysical simulations, is the real-time inclusion of relativity effects in visualizations.

That an image says more than a thousand words is not always true, especially in the case of visualizations of astrophysical simulations. It is important to provide an appropriate visualization combined with metadata on how it was created. A possible and elegant way to this
is workflow management systems. VisTrails is an excellent attempt to combine visualizations with the necessary information about their creation.

Of course there are many open issues, like the treatment of simulations with varying grid-geometries applied on different regions in the same computational domain, or the handling of huge data sets on a single workstation.

Acknowledgments

We acknowledge the Austrian Science Foundation (FWF) through grants P18523-N16, P19300-N16, Y-123INF and the UniInfrastrukturprogramm des BMWF Forschungsprojekt Konsortium Hochleistungsrechnen. In addition, we also acknowledge the ESO Mobilitätsstipendien des BMWF (Austria), and the Tiroler Wissenschaftsfonds (Gefördert aus Mitteln des vom Land Tirol eingerichteten Wissenschaftsfonds).

References

[1] Toomre A and Toomre J 1972 *Astrophys. J.* **178** 623
[2] Press W H and Schechter P 1974 *Astrophys. J.* **187** 425
[3] Sterling T, Salmon J, Becker D J and Savarese D F 1999 *How to Build a Beowulf: A Guide to the Implementation and Application of PC Clusters* (Cambridge MA: MIT Press) ISSN 0-262-69218-X
[4] Kranzlmueller D, Heinzlreiter P, Rosmanith H and Volker J 2004 *Grid-Enabled Visualisation with GVK* (Berlin: Springer) ISSN 0302-9743
[5] The OpenRT Real-Time Ray-Tracing Project, http://www.openrt.de/index.php
[6] Manta Open Source Interactive Ray Tracer, http://code.sci.utah.edu/Manta/index.php/Main_Page
[7] Springel V 2005 *Mon. Not. R. Astron. Soc.* **364** 1105
[8] The MPICH Home Page, http://www-unix.mcs.anl.gov/mpi/mpich1/
[9] FITS, The Astronomical Image and Table Format, http://fits.gsfc.nasa.gov/
[10] Rosner R et al 2000 *Comput. Sci. Eng.* **2** 33
[11] O’Shea B W, Bryan G, Bordner J, Norman M L, Abel T, Harkness R and Kritsuk A 2004 arXiv:astro-ph/0403044
[12] Norman M L 2000 *Rev. Mex. Astron. Astrofis. Ser. Conf.* **9** 66
[13] The Industry’s Foundation for High Performance Graphics, http://www.opengl.org/
[14] Xinerama, http://sourceforge.net/projects/xinerama/
[15] Rosenthal P, Rosswog S and Linsen L 2007 *Proc. of the 4th High-End Visualisation Workshop* ISBN 978-3-86541-216-4
[16] The HDF Group, http://hdf.ncsa.uiuc.edu/
[17] VisTrails, http://www.vistrails.org/index.php/Main_Page
[18] *The Visualisation Toolkit, An Object-Oriented Approach To 3D Graphics* 3rd edn ISBN 1-930934-12-2

*New Journal of Physics* **10** (2008) 125008 (http://www.njp.org/)