Visualization Tools for Adaptive Mesh Refinement Data

Gunther H. Weber[1] and Vincent E. Beckner[1] and Hank Childs[2] and Terry J. Ligocki[1] and Mark C. Miller[2] and Brian Van Straalen[1] and E. Wes Bethel[1]

[1]Computing Research Division, Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA 94720, USA

[2]Computing Applications and Research Department, Lawrence Livermore National Laboratory, Box 808, L-557, Livermore, CA 94551, USA

{GHWeber, VEBeckner}@lbl.gov, childs3@llnl.gov, TJLigocki@lbl.gov, miller86@llnl.gov, {BVStraalen, EWBethel}@lbl.gov

Abstract

Adaptive Mesh Refinement (AMR) is a highly effective method for simulations that span a large range of spatiotemporal scales, such as astrophysical simulations that must accomodate ranges from interstellar to sub-planetary. Most mainstream visualization tools still lack support for AMR as a first class data type and AMR code teams use custom built applications for AMR visualization. The Department of Energy's (DOE's) Science Discovery through Advanced Computing (SciDAC) Visualization and Analytics Center for Enabling Technologies (VACET) is currently working on extending VisIt, which is an open source visualization tool that accommodates AMR as a first-class data type. These efforts will bridge the gap between general-purpose visualization applications and highly specialized AMR visual analysis applications. Here, we give an overview of the state of the art in AMR visualization research and tools and describe how VisIt currently handles AMR data.

Appeared in: Proceedings of the 4th High-End Visualization Workshop, ISBN 978-3-86541-216-4, Lehmanns Media, http://www.lob.de/

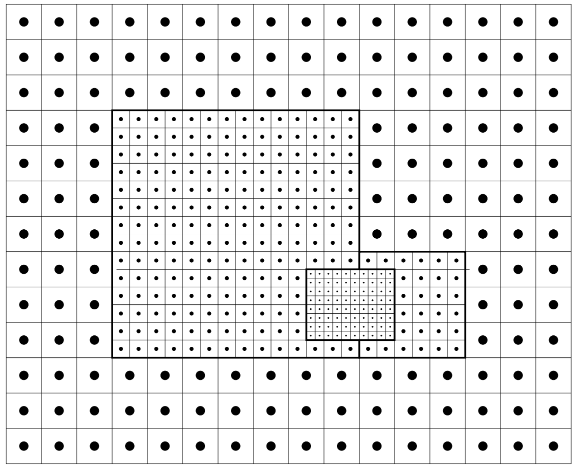

Figure 1: A simple Berger-Collela AMR hierarchy consisting of four patches organized in three hierarchy levels.

1 Introduction

Adaptive Mesh Refinement (AMR) techniques combine the compact, implicitly specified structure of regular, rectilinear with the adaptivity to changes in scale of unstructured grids. In this paper, we focus on block-structured, h-adaptive AMR techniques that represent the computational domain with a set of nested rectilinear grids or *patches* at increasing resolutions [Berger & Colella, 1989]. Figure 1 shows a simple example. Four regular patches are organized in three hierarchy levels. Grids belonging to a finer level are always completely enclosed by grids of the coarser levels.

Handling AMR data for visualization is challenging, since coarser information in regions covered by finer patches is superseded and replaced with information from these finer patches. During visualization it becomes necessary necessary to manage selection of which reslotuions are being used. Furthermore, it is difficult to avoid discontinuities at level boundaries, which, if not properly handled, lead to visible artifacts in visualizations.

While AMR data can be node centered, the majority of data sets we are currently visualizing use a cell centered format. This fact poses an additional challenge since many visualization algorithms expect data in a node centered format. Despite the growing popularity of AMR simulations, little research has been done in effective visualization of AMR data. Furthermore, there is a lack of tools that treat AMR as first-class data type. In this paper we give an overview of the current state of AMR research and describe ongoing work to extend VisIt to be one of the first mainstream visualization tools with "native" AMR support. Though other general visualization tools with AMR support exist, such as ParaView [Squillacote, 2006] and customized versions of Amira [Stalling et al., 2005], exist, we focus our discussion on VisIt due to its unique analytics capabilities.

2 Current State of AMR Visualization

2.1 Visualization of AMR Scalar Fields

Scalar quantities describe a variety of important physical characteristics such as temperature or pressure. Most simulations, including AMR simulations, include several scalar variables. Commonly used scalar data visualization techniques include slicing planes, isosurface extraction and direct volume rendering. Spreadsheets, which are somewhat related to slicing planes, provide direct access to data value and are valuable for debugging and extracting data for further analysis with a wider range of tools such as Matlab or paper and pencil.

2.1.1 AMR Visualization by Conversion to Other Types

Initial work on AMR visualization focused on converting AMR data to suitable conventional representations, which are subsequently used for visualization. [Norman et al., 1999] described a method for visualizing AMR data using is then visualized using standard unstructured grid techniques. Their method converts an AMR hierarchy into an unstructured grid composed of hexahedral cells. This resulting grid is then visualized utilizing standard AVS, IDL, and VTK algorithms. By converting AMR data to an unstructured mesh, the implicit definition of grid connectivity is lost. Overhead resulting from the required separate storage of grid structure results in poor performance and does not scale well to large AMR data sets. Furthermore, this approach prohibits using of the hierarchical nature of AMR data for efficient visualization algorithms. Recognizing these fundamental problems, Norman et al. continue by extending VTK to handle AMR grids as first-class data structure. They have yet to publish detailed descriptions of their techniques.

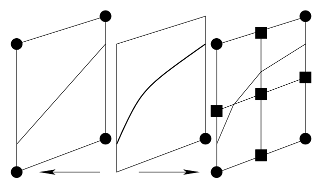

Figure 2: At the boundaries (center face in figure) between coarse (left face in figure) and fine levels (right face in figure), dangling nodes (black rectangles in the figure) occur in addition to vertices that are shared between grids (black circles in the figure). On cell faces, marching cubes approximates isosurfaces with line segments. This scheme can lead to cracks, as the actual contour (bold line on center face) is approximate by a line segment in the coarse level (left face) and a poly line in the fine level (right face).

2.1.2 Crack-free Isosurface Extraction

AMR data is particularly difficult to handle when visualizing data via isosurface extraction. This difficulty is due to the fact that AMR often uses a cell centered data format while the marching cubes algorithm [Lorensen & Cline, 1987], which is de-facto standard isosurface extraction algorithm in scientific visualization, expects values at the vertices of a grid. Furthermore, when extracting an isosurface using the marching cubes method, t-junctions will lead to visible cracks in an isosurface, even if dangling nodes have values that are consistent with the coarse level representation [Shekhar et al., 1996] (see Figure 2).

[Weber et al., 2001b, Weber et al., 2003a] developed a method that extracts crack-free isosurfaces from cell-centered AMR data by interpreting cell centers of each patch of the AMR hierarchy as the vertices of a new patch, which is the *dual grid* to the original patch. Within these dual grids, isosurfaces are extracted utilizing the standard marching cubes method. The use of dual grids leads to gaps between different levels of an AMR hierarchy. Weber et al. use a procedural scheme to fill these gaps with "stitch" cells (tetrahedra, pyramids, triangle prisms and deformed cubes) ensuring that this step produces no t-junctions. Subsequently, they extract isosurface portions within gaps between hierarchy levels utilizing the marching cubes methods by giving appropriate case tables for these new cell types.

[Fang et al., 2004] presented an alternate isosurface extraction approach for node-centered AMR data. Their main goal is preserving the original patch structure and "identity" of cells, enabling a user to determine to what particular patch cell a triangle of an isosurface belongs. Their method achieves this goal by extending refined patches until it is possible to assign values to dangling nodes that are consistent with interpolation results in the coarser level. Subsequently, they decompose coarse-level cells at the boundary to a finer level into a set of pyramids that connect the cell center with all boundary faces. For each "facet" of a subdivided face, i.e., a face at the boundary to a finer level, a separate pyramid is created, ensuring that marching cubes will not produce cracks in an extracted isosurface.

2.1.3 Volume Rendering

[Max, 1993] described sorting schemes for cells during volume rendering including one method specifically geared toward AMR data. [Ma, 1999] described and compared two approaches for rendering of structured AMR data using the PARAMESH framework. A PARAMESH hierarchy organizes grids as blocks in a quadtree (in 2-d space) or an octree (in 3-d space) structure. Inner nodes of this tree correspond to regions that need further refinement while leaf nodes specify a grid whose resolution is given by the current hierarchy level. Ma described two approaches for volume rendering of AMR data. One method resamples a hierarchy on an uniform grid at the finest resolution. The resulting grid is evenly subdivided and each part rendered in parallel on a separate processor. Partial images are combined using binary-swap composition.

A second method preserves the AMR structure. Individual blocks (leaves of the octree) are distributed among the processors in a round-robin fashion to achieve static load balancing. Since a block structure can lead to many small ray-segments, Ma buffers these segments into larger messages to decrease communication overhead. Individual blocks are rendered using ray-casting. Two sampling schemes are used: A simple approach using a fixed, constant sample distance and an adaptive approach that decreases sample distance in finer resolution blocks.

[Weber et al., 2001a] described an interactive, hardware accelerated volume rendering approach to generate previews of AMR data and a higher-quality software approach based on cell projection. Both approaches use data duplicated in coarser hierarchy levels as a less accurate approximation for the data in finer levels. The hardware-based approach uses a k-d-tree-like structure to partition an AMR hierarchy into blocks of homogeneous resolution and renders these blocks in back-to-front order. Based on view-dependent criteria (e.g., the number of pixels covered by a voxel) and a measured rendering time for the current frame, the traversal depth into the individual patches of the AMR hierarchy is chosen to achieve interactive rendering rates.

[Weber et al., 2001a] also described a software cell-projection-based approach to render AMR data sets in higher quality. While rendering a level of an AMR hierarchy, additional information is stored for each pixel that makes it possible to "replace" the contribution of those parts of the domain that are refined by another hierarchy level with a more accurate representation, supporting progressive rendering of AMR data sets. In later work, [Weber et al., 2001a] used the dual mesh and stitch cells introduced for isosurface extraction [Weber et al., 2003a] to define a C^0 continuous interpolation scheme and utilized this interpolation method in their progressive cell-projection rendering approach.

[Kreylos et al., 2002] described a framework for remote, interactive rendering of AMR data. The framework consists of a "lightweight" viewer and a renderer running on one or several remote machines. The method of Kreylos et al. "homogenizes" an AMR hierarchy, i.e. partitions it in blocks of constant resolution using a k-d tree. Resulting blocks of constant resolution are distributed among processors and rendered using either a texture-based hardware-accelerated approach or a software-based cell-projection renderer. Two distribution strategies are implemented: One strategy attempts to distribute cost evenly among processors, the other variant tries to minimize data duplication. [Weber et al., 2003b] built on this work and compared various AMR partitioning strategies for parallel volume rendering of AMR data.

[Kähler & Hege, 2002] introduced a method that partitions Berger-Colella AMR data into homogeneous resolution regions and visualizes it using texture-based hardware-accelerated volume rendering. Their partitioning scheme uses a heuristic that is based on assumptions concerning the placement of refining grid to minimize the number of constant-resolution blocks. Generally, this approach generates fewer blocks than the approach described by [Weber et al., 2001a] and the approach developed by [Kreylos et al., 2002]. Subdivision into a smaller number of blocks is beneficial when data is rendered on a single machine.

In later work, [Kähler et al., 2002] used a set of existing tools to render results of a simulation of a forming star using the framework developed by [Bryan, 1999]. They define camera paths within a CAVE environment using the *Virtual Director* virtual reality interface. Subsequently, they render animations of the AMR simulation utilizing their previously developed hardware-accelerated volume rendering approach [Kähler & Hege, 2002]. To enhance depth perception, rendered images are augmented with a background that is obtained by rendering a particle simulation of the formation of the early universe. Recently, [Kähler et al., 2006] implemented a GPU-based ray-casting approach for AMR data, which improves rendering quality considerably compared to slicing-based approaches and supports a more complex light model with wavelength dependent absorption.

By specifying a transfer function, and a range of isovalues, [Park et al., 2002] produced volume-rendered images of AMR data based on hierarchical splatting, see [Laur & Hanrahan, 1991]. Their method converts an AMR hierarchy to a k-d-tree structure consisting of blocks of constant resolution. Each node of this k-d tree is augmented with an octree. Octree and k-d-tree nodes contain a 32-bit field, where each bit represents a continuous range of isovalues. Using the k-d tree and the octree, regions containing values within the specified range are identified and rendered back-to-front using hierarchical splatting.

2.2 Visualization of Time-varying AMR Scalar Field Data

[Chen et al., 2003] introduced the *feature tree* that describes, how connected components of an isosurface change as one successively adds AMR refinement levels. Nodes in a tree represent connected components of an isosurface, with each level in the tree corresponding to a level in the AMR hierarchy. The resulting tree describes whether a connected component splits or merges when considering a finer hierarchy level. Using the resulting feature tree, it becomes possible to describe how individual isosurface components change over the course of time in a simulation.

[Kaehler et al., 2005] described a framework for visualization of time-varying AMR data. Their method addresses the problem that most AMR simulations update finer AMR patches more frequently than coarse patches. Considering two subsequent time steps, their interpolation scheme first ensures that both time-steps have the same refinement configuration, i.e., that each cell that is refined in one time step is also refined in the other time step. Values for cells that are not refined in the current time step but the other are obtained by interpolation. Subsequently, they define an interpolation scheme to compute intermediate values in regions that are covered by coarser level and thus, updated less frequently. In addition to this interpolation scheme, their framework automatically handles remote data access and computes interpolated values on the machine, which also runs the simulation.

3 Custom AMR Visualization Tools

3.1 ChomboVis

ChomboVis [Ligocki et al., 2003] is an AMR visualization tool built to visualize AMR data produced by Chombo [Chombo, 2007]—an AMR library distributed by the Applied Numerical Algorithms Group (ANAG). Data is stored in files using HDF5. ChomboVis reads these files and uses VTK along with custom Python and C++ code for visualization. A variety of methods are provided for visualizing the data sets including color mapping, grid display, slicing, isosurfaces/contours and streamlines. Users can browse the data directly via spreadsheets by selecting grids graphically or via indices. ChomboVis has both a graphical user interface and a Python application programming interface (API) to facilitate interactive use and scripting.

ChomboVis has rudimentary support for vector field visualization. A GUI interface allows a user to specify which scalar data fields should be combined to form a d-dimensional vector in $\mathbb{R}^d$, as well as specify a *rake* (a line segment in $\mathbb{R}^d$ with seed points space equidistantly). These seed points are integrated either forward or backward (using negative vectors) using backward Euler integration with a user-specified step length for a user-specified number of steps. After each step a check is made to determine if the current position is still inside the current AMR box. If it is, integration is continues, otherwise, the AMR hierarchy is searched, from finest to coarsest, to determine a box in which integration can be continued. These are *streamlines*, not to be confused with streaklines or pathlines.

3.2 Amrvis

Amrvis is a visualization and data analysis tool for examining data files generated by the Center for Computational Sciences and Engineering (CCSE) using their AMR codes. A user can view color planar images of the data, grids, numerical values in a chosen format, subregions, animations, volumetric renderings, contour plots, line graphs and 2D vector fields using arrow glyphs. Amrvis works with 2D and 3D data, requires no specialized graphics hardware, and can run in parallel on both SMP and distributed memory parallel machines. The user can interactively choose the displayed variable (density, pressure, etc.) and AMR level, scale the images, set arbitrary data ranges, and set viewing planes. Amrvis also has batch capabilities for extracting subregions, planes, lines, and points from data files and for preprocessing volume renderings. A separate tool, Amrmovie, can be used to view, animate, and output volume renderings. This tool can animate volume data, which has been preprocessed with Amrvis, at arbitrary orientations and through time. Another separate tool, Amrderive, can be used to access and manipulate AMR data. Examples include deriving a new variable such as vorticity or log of density, calculating integrals, and finding average values across multiple files.

4 Visualization of AMR Data with VisIt

VisIt [Childs & Miller, 2006] is a richly featured visualization and analysis tool for large data sets. It employs a client-server model where the server is parallelized. The nature of parallelization is data parallel; the input data set is partitioned among VisIt's processors. In the case of AMR data, each patch is treated as an atomic unit and assigned to one of the processors on VisIt's parallelized server. For example, patches "level zero, patch zero" and "level one, patch two" may be assigned to the server's processor zero, while patches "level zero, patch one," "level one, patch zero," and "level one, patch one" may be assigned to processor one.

Visualization and analysis of massive scale data sets is an important use case for VisIt. As such, it employs many optimizations to enable the processing of this data. For example, VisIt is able to use spatial extents meta-data to reduce the amount of data that is processed. When a slice of a three-dimensional data set is being rendered, VisIt is able to to limit the patches processed to those that actually intersect the slice. Although this functionality may sound straight forward, it is difficult to implement in a richly featured, module based framework. More information about the contract methodology that enables these optimizations can be found in [Childs et al., 2005].

VisIt's handling of AMR data is made possible by marking coarse cells that are refined at a lower level as "ghost." This marking is done by adding an array to each patch that designates the status of each cell (ghost or non-ghost). Most algorithms can ignore the ghost markings; they operate identically on ghost and non-ghost cells. One advantage of using the ghost cells is that it allows structured grids to retain their native form. That is, removing the ghost cells before applying visualization algorithms would create a grid that was no longer structured. The resulting grid would often be unstructured and that unstructured grid could have a memory footprint that is an order of magnitude larger. Another advantage of using ghost cells is that they allow for proper interpolations to take place, which would not be possible if refined cells were removed before applying visualization algorithms. After all algorithms have been applied, a module walks the data set and removes all cells or geometry resulting from a ghost cell.

VisIt employs the standard Marching Cubes algorithm to contour data. Most AMR data is cell-centered, requiring interpolation to the nodes. For hanging nodes at the boundary of patches at different refinement levels, this interpolation is done incorrectly in VisIt and cracked isosurfaces can result. However, VisIt does *not* produce cracked isosurfaces when abutting patches are at the same refinement level. In this case, VisIt can create a layer of ghost cells around each patch that contains the values of neighboring cells from the other patches. These ghost cells allow for correct interpolation to take place, meaning that a consistent contouring takes place from patch to patch and no cracks are created in this case.

VisIt's employs a data parallel volume rendering scheme that is able to resolve the types of complex sorting issues that arise in unstructured meshes [Childs et al., 2006]. AMR meshes present a special type of load balancing challenge for VisIt's volume rendering algorithm, however. The running time of the algorithm is dependent on the amount of data and the amount of samples. For AMR meshes, the patches at the coarser refinement levels occupy a larger spatial footprint, and, as such, often cover a much larger portion of the picture and contribute more samples. Hence, sampling the patches at coarser refinement levels typically takes much longer than the sampling for patches at finer refinement levels. VisIt attempts to counteract this problem by minimizing the amount of patches at the coarse refinement levels on any given processor. In terms of additional AMR handling, the volume rendering algorithm ignores all samples from cells that are marked ghost. So if a sample point is contained by many patches at different refinement levels, only the value at the finest level will be accepted, since all other levels will have the corresponding cell marked as ghost.

A trend of increasing importance is where visualization and analysis capabilities are coupled in one production quality application. Here, analysis means computing statistical moments of subsets of AMR hierarchies, distribution functions, computed/derived quantities, temporal analysis, etc. VisIt provides a rich set of analysis capabilities (such as integrating density over volume to obtain mass, calculating volumes, surface areas, and moments of inertia), all of which execute in parallel.

5 Future Work

We are currently working on extending VisIt's visualization capabilities for AMR data. To this end, we had extensive meetings with members of the LBNL Applied Numerical Algorithms Group (ANAG) and the LBNL Center for Computational Sciences and Engineering (CCSE). Some feature requests, such as requests for support of picking abilities and spreadsheet support (duplicating functionality present in Amrvis and ChomboVis) indicate that visualization is still used frequently for debugging. On the other hand there is a growing need for for data analysis and visualization capabilities to interpret the results of production AMR simulations. For example, we are currently working on adding line integral convolution-based vector field visualization capabilities to VisIt. Some extensions, such as the spreadsheet interface, are already part of the recent VisIt 1.6 release, even though this functionality is still under development.

6 Acknowledgments

This work was supported by the Director, Office of Advanced Scientific Computing Research, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231 through the Scientific Discovery through Advanced Computing (SciDAC) program's Visualization and Analytics Center for Enabling Technologies (VACET). We thank the members of the LBNL Visualization Group, the LBNL ANAG, the LBNL CCSE, and the VisIt development team.

References

[Berger & Colella, 1989] Berger, M. & Colella, P. (1989). Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics*, 82, 64–84.

[Bryan, 1999] Bryan, G. L. (1999). Fluids in the universe: Adaptive mesh refinement in cosmology. *Computing in Science and Engineering*, 1(2).

[Chen et al., 2003] Chen, J., Silver, D., & Jiang, L. (2003). The feature tree: Visualizing feature tracking in distributed amr datasets. In *Proceedings of the IEEE Symposium on Parallel and Large-Data Visualization and Graphics 2003* (pp. 103–110).: IEEE Computer Society.

[Childs et al., 2005] Childs, H., Brugger, E. S., Bonnell, K. S., Meredith, J. S., Miller, M., Whitlock, B. J., & Max, N. (2005). A contract-based system for large data visualization. In *IEEE Visualization 2005* (pp. 190–198).

[Childs et al., 2006] Childs, H., Duchaineau, M. A., & Ma, K.-L. (2006). A scalable, hybrid scheme for volume rendering massive data sets. In *Eurographics Symposium on Parallel Graphics and Visualization* (pp. 153–162).

[Childs & Miller, 2006] Childs, H. & Miller, M. (2006). Beyond meat grinders: An analysis framework addressing the scale and complexity of large data sets. In *SpringSim High Performance Computing Symposium (HPC 2006)* (pp. 181–186).

[Chombo, 2007] Chombo (2000–2007). `http://seesar.lbl.gov/ANAG/chombo/`.

[Fang et al., 2004] Fang, D. C., Weber, H., G., Childs, H., Brugger, E., Hamann, B., & Joy, K. (2004). Extracting geometrically continuous isosurfaces from adaptive mesh refinement data. In *Proceedings of 2004 Hawaii International Conference on Computer Sciences (DVD-ROM conference proceedings)* (pp. 216–224). ISSN 1545-6722.

[Kaehler et al., 2005] Kaehler, R., Prohaska, S., Hutanu, A., & Hege, H.-C. (2005). Visualization of time-dependent remote adaptive mesh refinement data. In *IEEE Visualization 2005* (pp. 175–182).: IEEE Computer Society.

[Kähler et al., 2002] Kähler, R., Cox, D., Patterson, R., Levy, S., Hege, H.-C., & Abel, T. (2002). Rendering the first star in the universe – a case study. In *IEEE Visualization 2002* (pp. 537–540).: IEEE Computer Society.

[Kähler & Hege, 2002] Kähler, R. & Hege, H.-C. (2002). Texture-based volume rendering of adaptive mesh refinement data. *The Visual Computer*, 18(8), 481–492. Zuse Institut Technical Report ZR-01-30.

[Kähler et al., 2006] Kähler, R., Wise, J., Abel, T., & Hege, H.-C. (2006). GPU-assisted raycasting for cosmological adaptive mesh refinement simulations. In *Proceedings of Volume Graphics*: Eurographics Association.

[Kreylos et al., 2002] Kreylos, O., Weber, G. H., Bethel, E. W., Shalf, J. M., Hamann, B., & Joy, K. I. (2002). *Remote Interactive Direct Volume Rendering of AMR Data*. Technical Report LBNL 49954, Lawrence Berkeley National Laboratory.

[Laur & Hanrahan, 1991] Laur, D. & Hanrahan, P. (1991). Hierachical splatting: A progressive refinement algorithm for volume rendering. *Computer Grahpics (Proceedings of ACM SIGGRAPH 91)*, 25(4), 285–288.

[Ligocki et al., 2003] Ligocki, T. J., Straalen, B. V., Shalf, J. M., Weber, G. H., & Hamann, B. (2003). A framework for visualizing hierarchical computations. In *Hierarchical and Geometrical Methods in Scientific Visualization* (pp. 197–204). Springer Verlag.

[Lorensen & Cline, 1987] Lorensen, W. E. & Cline, H. E. (1987). Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics (Proceedings of ACM SIGGRAPH 87)*, 21(4), 163–169.

[Ma, 1999] Ma, K.-L. (1999). Parallel rendering of 3D AMR data on the SGI/Cray T3E. In *Proceedings of Frontiers '99 the Seventh Symposium on the Frontiers of Massively Parallel Computation* (pp. 138–145).: IEEE Computer Society.

[Max, 1993] Max, N. L. (1993). Sorting for polyhedron compositing. In *Focus on Scientific Visualization* (pp. 259–268). Springer-Verlag.

[Norman et al., 1999] Norman, M. L., Shalf, J. M., Levy, S., & Daues, G. (1999). Diving deep: Data management and visualization strategies for adaptive mesh refinement simulations. *Computing in Science and Engineering*, 1(4), 36–47.

[Park et al., 2002] Park, S., Bajaj, C., & Siddavanahalli, V. (2002). Case study: Interactive rendering of adaptive mesh refinement data. In *IEEE Visualization 2002* (pp. 521–524).: IEEE Computer Society.

[Shekhar et al., 1996] Shekhar, R., Fayyad, E., Yagel, R., & Cornhill, J. F. (1996). Octree-based decimation of marching cubes surface. In *IEEE Visualization '96* (pp. 335–342, 499).: IEEE Computer Society.

[Squillacote, 2006] Squillacote, A. H. (2006). *The ParaView Guide*. Kitware.

[Stalling et al., 2005] Stalling, D., Westerhoff, M., & Hege, H.-C. (2005). Amira: A highly interactive system for visual data analysis. In *The Visualization Handbook* (pp. 749–767). Elsevier.

[Weber et al., 2001a] Weber, G. H., Hagen, H., Hamann, B., Joy, K. I., Ligocki, T. J., Ma, K.-L., & Shalf, J. M. (2001a). Visualization of adaptive mesh refinement data. In *Proceedings of the SPIE (Visual Data Exploration and Analysis VIII)*, volume 4302 (pp. 121–132).

[Weber et al., 2001b] Weber, G. H., Kreylos, O., Ligocki, T. J., Shalf, J. M., Hagen, H., Hamann, B., & Joy, K. I. (2001b). Extraction of crack-free isosurfaces from adaptive mesh refinement data. In *Proceedings of the Joint EUROGRAPHICS and IEEE TCVG Symposium on Visualization, Ascona, Switzerland, May 28–31, 2001* (pp. 25–34, 335).: Springer Verlag.

[Weber et al., 2003a] Weber, G. H., Kreylos, O., Ligocki, T. J., Shalf, J. M., Hagen, H., Hamann, B., & Joy, K. I. (2003a). Extraction of crack-free isosurfaces from adaptive mesh refinement data. In *Hierarchical and Geometrical Methods in Scientific Visualization* (pp. 19–40). Springer Verlag.

[Weber et al., 2003b] Weber, G. H., Öhler, M., Kreylos, O., Shalf, J. M., Bethel, E. W., Hamann, B., & Scheuermann, G. (2003b). Parallel cell projection rendering of adaptive mesh refinement data. In *Proceedings of the IEEE Symposium on Parallel and Large-Data Visualization and Graphics 2003* (pp. 51–60).: IEEE Computer Society.

Index