Voxel-Based Terrain for Real-Time Virtual Simulations

By

ERIC STEPHEN LENGYEL
B.S. Mathematics (Virginia Polytechnic Institute and State University) 1994
M.S. Mathematics (Virginia Polytechnic Institute and State University) 1996

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Computer Science

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA
DAVIS

Approved:

____
John D Owens, Chair

____
Kenneth I Joy

____
Nina Amenta

Committee in Charge

2010

Voxel-Based Terrain for Real-Time Virtual Simulations

Abstract

This dissertation provides the theoretical basis and implementation details for a complete and practical real-time voxel-based terrain rendering system. We first present a modified Marching Cubes algorithm designed to eliminate choices arising from ambiguities in the original algorithm and its successors in order to facilitate a faster implementation and to simplify the design of a level-of-detail algorithm. The modified Marching Cubes algorithm is extended to operate on voxel data at multiple resolutions in such a way that triangle meshes produced at all levels of detail correctly match geometrical features. We introduce a robust method for seamlessly joining voxel-based terrain meshes of different levels of detail and establish a transition structure that both simplifies the triangulation problem and eliminates the potential for shading artifacts. Finally, we discuss methods for applying texture maps and advanced shading techniques to voxel-based terrain meshes. These methods are designed to be fast and compatible with the widest possible range of graphics hardware across multiple platforms.

Contents

List of Figures

The shape and size of each paint brush is controlled by several settings in the tool panel. First, the general shape is selected by the spherical brush, cylindrical brush, and slope brush buttons, and four different slope functions can be selected for the slope brush. The radius of each brush shape is specified, in units of voxels, by the brush radius slider, and the height of a brush is specified, as a percentage of the brush diameter, by the brush height slider.

The drawing plane can be set to four different values:

- **Horizontal plane.** The brush is constrained to move only in the x and y directions.
- **Tangent plane.** The brush is constrained to the plane tangent to the terrain surface where the mouse is first clicked.
- **Camera plane.** The brush is constrained to move only in directions perpendicular to the current camera view direction.
- **Follow surface.** The brush always stays on the terrain surface that existed before the mouse was first clicked. Changes made by the brush do not affect the brush position until the user releases the mouse and begins drawing again.

The brush offset slider controls the position of each paint brush with respect to the current drawing plane. A value of -100% means the brush is completely submerged beneath the surface, and a value of $+100\%$ means the brush is just grazing the surface on the outside. Naturally, a value of 0% means the brush is split halfway between inside and outside. If the "Enable stylus pressure" box is checked, then the brush offset is dynamically adjusted based on the current pressure exerted by the user on a tablet device, if available. For an additive brush, the offset is increased with greater pressure, and for a subtractive brush, the offset is decreased with greater pressure.

The texture selectors at the bottom of the tool panel are used to select two sets of texture maps for the triplanar projection. In each set, one texture map can be chosen to be applied to the positive z direction, one can be chosen to be applied to the the negative z direction, and one can be chosen to be applied to all four directions in the x-y plane. Clicking on a texture image causes a palette to appear from which a texture map can be selected from the available palette entries. Textures from the two sets are mixed on the terrain using the blend brush and the texture blend slider.

A.3 Terrain Shaders

The C4 Engine includes a graphical fragment shader editor that provides an abstraction of the underlying textual shading language passed to the low-level rendering library. We added special types of nodes to this editor that encapsulate all of the shader code developed in Chapter 5 so that the user need not be concerned with the internals of texture fetches from texture arrays or texture palettes and the specifics of triplanar blending.

An example fragment shader graph showing the operations for bump-mapped diffuse and specular reflection is shown in Figure A.2. The "Terrain Texture" node fetches color samples from the texture maps assigned to the triangle being rendered and performs triplanar blending based on the interpolated normal vector. This node can be configured to blend between two different sets of textures using the blend factor α or selecting only one set or the other.

The three "Terrain Normal" nodes function a little differently. Each one fetches normal vector samples from one or two of the texture maps assigned to just one plane of the triplanar projection. As before, these samples can be blended, or just one texture map can be sampled and have its value passed through by itself. The results of the "Terrain Normal" nodes are passed into nodes called "Terrain Diffuse Reflection" and "Terrain Specular Reflection". These two nodes calculate the quantities $\max\{\mathbf{N}\cdot\mathbf{L},0\}$ and $(\max\{\mathbf{N}\cdot\mathbf{H},0\})^e$ for each input normal vector in the appropriate tangent space and then perform triplanar blending on the results in the manner shown in Listing 5.10.

The presence of the "Terrain Texture" and "Terrain Normal" nodes in the graph implicitly cause the three sets of texture coordinates and the triplanar blending weights to be generated and shared among the nodes that need them.

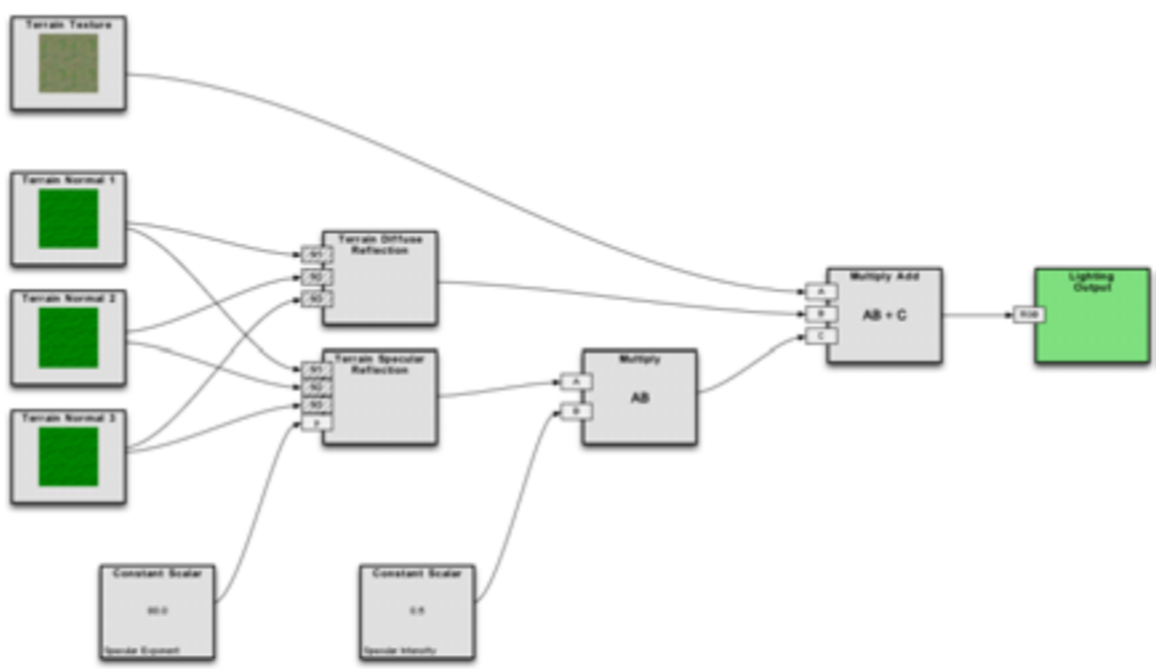

Figure A.2. A terrain fragment shader graph that includes bump-mapped diffuse and specular reflection.

Bibliography

ASIRVATHAM, ARUL, AND HOPPE, HUGUES. 2005. “Terrain Rendering Using GPU-Based Geometry Clipmaps”. *GPU Gems 2*, Addison-Wesley, Chapter 2, pp. 27–45.

BLINN, JAMES F. 1977. “Models of Light Reflection for Computer Synthesized Pictures”. *Proceedings of the 4th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 192–198.

CHERNYAEV, EVGENI V. 1995. “Marching Cubes 33: Construction of Topologically Correct Isosurfaces”. Technical Report CERN CN 95-17, CERN.

CRASSIN, CYRIL; NEYRET, FABRICE; LEFEBVRE, SYLVAIN; AND EISEMANN, ELMAR. 2009. “GigaVoxels: Ray-guided Streaming for Efficient and Detailed Voxel Rendering”. *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games*, pp. 15–22.

CRASSIN, CYRIL; NEYRET, FABRICE; LEFEBVRE, SYLVAIN; SAINZ, MIGUEL; AND EISEMANN, ELMAR. 2009. “Beyond Triangles : Gigavoxels Effects In Video Games”. SIGGRAPH 2009 Technical Talk.

DICK, CHRISTIAN; KRÜGER, JENS; AND WESTERMANN, RÜDIGER. 2009. “GPU Ray-Casting for Scalable Terrain Rendering”. *Proceedings of Eurographics 2009*, pp. 43–50.

DUCHAINEAU, MARK; WOLINSKY, MURRAY; SIGETI, DAVID E.; MILLER, MARK C.; ALRICH, CHARLES; AND MINEEV-WEINSTEIN, MARK B. 1997. “ROAMing Terrain: Real-time Optimally Adapting Meshes”. *Proceedings of the 8th Conference on Visualization ’97*, pp. 81–88.

DÜRST, M. J. 1988. “Additional reference to marching cubes”. *Computer Graphics*, Volume 22, Number 2, pp. 72–73.

GEISS, RYAN. 2008. “Generating Complex Procedural Terrains Using the GPU”. *GPU Gems 3*, Addison-Wesley, Chapter 1, pp. 7–37.

GEISS, RYAN, AND THOMPSON, MICHAEL. 2007. "NVIDIA Demo Team Secrets—Cascades". Game Developers Conference. http://developer.download.nvidia.com/presentations/2007/gdc/CascadesDemoSecrets.zip

HAZEWINKEL, MICHIEL (editor). 2001. "Poincaré–Hopf Theorem". Encyclopaedia of Mathematics, Kluwer Academic Publishers.

HOPPE, HUGUES. 1996. "Progressive Meshes". *Proceedings of SIGGRAPH 1996*, pp. 99–108.

HOPPE, HUGUES. 1998. "Smooth View-Dependent Level-of-Detail Control and its Application to Terrain Rendering". *Proceedings of the Conference on Visualization '98*, pp. 35–42.

JU, TAO; LOSASSO, FRANK; SCHAFER, SCOTT; AND WARREN, JOE. 2002. "Dual Contouring of Hermite Data". *Proceedings of SIGGRAPH 2002*, pp. 339–346.

KAZHDAN, MICHAEL; KLEIN, ALLISON; DALAL, KETAN; AND HOPPE, HUGUES. 2007. "Unconstrained Isosurface Extraction on Arbitrary Octrees". *Proceedings of the 5th Eurographics Symposium on Geometry Processing*, pp. 125–133.

KOBBELT, LEIF P.; BOTSCH, MARIO; SCHWANECKE, ULRICH; AND SEIDEL, HANS-PETER. 2001. "Feature Sensitive Surface Extraction from Volume Data". *Proceedings of SIGGRAPH 2001*, pp. 57–66.

LEECH, JON AND KILGARD, MARK. 2008. GL_EXT_texture_array OpenGL extension specification. http://www.opengl.org/registry/specs/EXT/texture_array.txt

LENGYEL, ERIC. 2002. *Mathematics for 3D Game Programming and Computer Graphics*. Charles River Media, Chapter 6.

LENGYEL, ERIC. 2005. *C4 Engine*. Terathon Software LLC. http://www.terathon.com/c4engine/

LINDSTROM, PETER; KOLLER, DAVID; RIBARSKY, WILLIAM; HODGES, LARRY F.; FAUST, NICK; AND TURNER, GREGORY A. 1996. "Real-Time, Continuous Level of Detail Rendering of Height Fields". *Proceedings of SIGGRAPH 1996*, pp. 109–118.

LIVNY, YOTAM; KOGAN, ZVI; AND EL-SANA, JIHAD. 2009. "Seamless Patches for GPU-Based Terrain Rendering". *The Visual Computer: International Journal of Computer Graphics*, Volume 25, Number 3, pp. 197–208.

LORENSEN, WILLIAM E., AND CLINE, HARVEY E. 1987. "Marching Cubes: A High Resolution 3D Surface Construction Algorithm". *Computer Graphics (Proceedings of SIGGRAPH 87)*, Volume 21, Issue 4, pp. 163–169.

LOSASSO, FRANK AND HOPPE, HUGUES. 2004. "Geometry Clipmaps: Terrain Rendering Using Nested Regular Grids". *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2004)*, Volume 23, Issue 3, pp. 769–776.

NEWMAN, TIMOTHY S., AND YI, HONG. 2006. "A Survey of the Marching Cubes Algorithm". *Computers & Graphics*, Volume 30, Number 5, pp. 854–879.

NIELSON, GREGORY M. 2004. "Dual Marching Cubes". *Proceedings of the Conference on Visualization '04*, pp. 489–496.

NIELSON, GREGORY M., AND HAMANN, BERND. 1991. "The Asymptotic Decider: Resolving the Ambiguity in Marching Cubes". *Proceedings of the 2nd Conference on Visualization '91*, IEEE Computer Society Press, pp. 83–91.

NING, PAUL, AND BLOOMENTHAL, JULES. 1993. "An Evaluation of Implicit Surface Tilers". *IEEE Computer Graphics and Applications*, Volume 13, Number 6, pp. 33–34.

PAJAROLA, RENATO, AND GOBBETTI, ENRICO. 2007. "Survey on Semi-Regular Multiresolution Models for Interactive Terrain Rendering". *The Visual Computer: International Journal of Computer Graphics*, Volume 23, Number 8, pp. 583–605.

RÖTTGER, STEFAN; HEIDRICH, WOLFGANG; SLUSALLEK, PHILIPP; AND SEIDEL, HANS-PETER. 1998. "Real-Time Generation of Continuous Levels of Detail for Height Fields". *Proceedings of WSCG '98*, pp. 315–322.

SEGAL, MARK, AND AKELEY, KURT. 2009. *The OpenGL Graphics System: A Specification, Version 3.2*. The Khronos Group.

SCHNEIDER, JENS, AND WESTERMANN, RÜDIGER. 2006. "GPU-Friendly High-Quality Terrain Rendering". *Journal of WSCG*, Volume 14, pp. 49–56.

SHEKHAR, RAJ; FAYYAD, ELIAS; YAGEL, RONI; AND CORNHILL, J. FREDRICK. 1996. "Octree-Based Decimation of Marching Cubes Surfaces". *Proceedings of the 7th Conference on Visualization '96*, pp. 335–342.

SHU, RENBEN; ZHOU, CHEN; AND KANKANHALLI, MOHAN S. 1995. "Adaptive Marching Cubes". *The Visual Computer*, Volume 11, pp. 202–217.

STANLEY, RICHARD P. 2001. *Enumerative Combinatorics, Volume 2*. Cambridge University Press, Chapter 6.

TANNER, CHRISTOPHER C.; MIGDAL, CHRISTOPHER J.; AND JONES, MICHAEL T. 1998. "The Clipmap: A Virtual Mipmap". *Proceedings of SIGGRAPH 1998*, pp. 151–158.

ULRICH, THATCHER. 2002. "Super-size it! Scaling up to Massive Virtual Worlds". SIGGRAPH 2002 Courses.

UNITED STATES GEOLOGICAL SURVEY. 2006. National Elevation Dataset. http://ned.usgs.gov/

VAN GELDER, ALLEN, AND WILHELMS, JANE. 1994. "Topological Considerations in Isosurface Generation". *ACM Transactions on Graphics*, Volume 13, Issue 4, pp. 337–375.