REFERENCES

[Abram90] Gregory D. Abram and Turner Whitted, “Building Block Shaders”, Proceedings of SIGGRAPH 90 (Dallas, Texas, August 6–10, 1990). In *Computer Graphics*, v24n4. ACM SIGGRAPH, August 1990, pp. 283–288.

They present an implementation of Cook’s shade trees [Cook84]. Instead of constructing parse trees using an expression parser, the trees are directly constructed by linking building-block modules together in a graphical user interface.

[Akeley93] Kurt Akeley, “RealityEngine Graphics”, Proceedings of SIGGRAPH 93 (Las Vegas, Nevada, August 1–6, 1992). In *Computer Graphics* Proceedings, Annual Conference Series, ACM SIGGRAPH, August 1993, pp. 109–116.

This paper describes the implementation of the Silicon Graphics RealityEngine.

[Amburn86] Phil Amburn, Eric Grant and Turner Whitted, “Managing Geometric Complexity with Enhanced Procedural Models”, Proceedings of SIGGRAPH 86 (Dallas, Texas, August 18–22, 1986). In *Computer Graphics*, v20n4. ACM SIGGRAPH, August 1986, pp. 189–195.

They describe two ideas for procedural modeling. First is generalized subdivision, where a representation is subdivided until a transition test triggers a change of representation. The new representation may undergo further subdivision or be rendered as a primitive. Second is communication between models. Two interacting models send messages to adjust to each other.

[Bajura92] Michael Bajura, Henry Fuchs and Ryutarou Ohbuchi, “Merging Virtual Objects with the Real World: Seeing Ultrasound Imagery Within the Patient”, Proceedings of SIGGRAPH 92 (Chicago, Illinois, July 26–31, 1992). In *Computer Graphics*, v26n2, ACM SIGGRAPH, July 1992.

This paper describes an augmented reality system combining a head-mounted display and an ultrasound machine. The system gives a physician “x-ray” vision for seeing a fetus inside the mother. The ultrasound data was rendered using a special-purpose flat disk primitive on Pixel-Planes 5.

[Banks92] David Banks, “Interactive Manipulation and Display of Two-dimensional Surfaces in Four-dimensional Space”, Proceedings of the 1992 Symposium on Interactive 3D Graphics (Cambridge, Massachusetts, March 29–April 1, 1992). In *Computer Graphics* special issue. ACM SIGGRAPH, March 1992, pp. 197–207.

To help in understanding surfaces in 4D, Banks uses a custom primitive on Pixel-Planes 5 to display intersection and silhouette curves. He also uses some interesting shading techniques to light these surfaces.

[Banks93] David Banks, *Interacting with Surfaces in Four-dimensional Space Using Computer Graphics*, PhD Dissertation, Department of Computer Science, University of North Carolina at Chapel Hill, 1993.

This dissertation includes and expands on [Banks92].

[Barr84] Alan H. Barr, “Global and Local Deformations of Solid Primitives”, Proceedings of SIGGRAPH 84 (Minneapolis, Minnesota, July 23–27, 1984). In *Computer Graphics*, v18n3. ACM SIGGRAPH, July 1984, pp. 21–30.

Barr introduces deformations like twist, taper, bend, etc. These deformations are standard transformations parameterized by position.

[Bentley88] Jon Bentley, “Little Languages”, *More Programming Pearls*. Addison-Wesley, 1988, pp. 83–100.

This book reprints a number of articles from Jon Bentley’s Programming Pearls column in the Communications of the ACM. This article includes a definition of *little languages* and an argument in favor of their use. Shading languages are a prime example of the use of a little language

[Bishop94] Gary Bishop, Henry Fuchs, Leonard McMillan and Ellen J. Scher Zagier, “Frameless Rendering: Double Buffering Considered Harmful”, Proceedings of SIGGRAPH 94 (Orlando, Florida, July 24–29, 1994). In *Computer Graphics* Proceedings, Annual Conference Series. ACM SIGGRAPH, July 1994, pp. 175–176.

This paper introduces frameless rendering. The frameless rendering technique update pixels randomly to allow display updates during pixel computation.

[Blinn78] James F. Blinn, *Computer Display of Curved Surfaces*, PhD Dissertation, Department of Computer Science, University of Utah, 1978.

Blinn introduces an incremental scan-line algorithm for rendering spline curves and surfaces (also published in [Lane80]). The algorithm uses the points of intersection between the curve and scan-line as a starting point for a Newton’s iteration to find the intersection between the curve and the next scan line.

Blinn also (in a seemingly unrelated major contribution) introduces bump-mapping, where the gradient of a gray-scale texture map is used to perturb the surface normal.

For small-scale changes, this gives the appearance that the shaded surface is much more complicated than the original geometry.

[Blinn82] James F. Blinn, “A Generalization of Algebraic Surface Drawing”, *ACM Transactions on Graphics*, v1n3. ACM SIGGRAPH, July 1982, pp. 235–256.

This paper presents an algorithm for ray tracing blobby models. Blobs are implicit functions based on a mixture of Gaussian density function. Uses bounding spheres to limit the number of active blobs and solves for intersection using a hybrid Newton/regula falsi method.

[Blinn85] James F. Blinn, “The Ancient Chinese Art of Chi-Ting”, Proceedings of SIGGRAPH 85 (San Francisco, California, July 22–26, 1985). *Image Rendering Tricks seminar notes*, ACM SIGGRAPH, July 1985.

Blinn presents a collection of rendering tricks from JPL space movies and the Mechanical Universe. The first section has text, the rest is in outline form. He has a section on specialized primitives that describes his 2 polygon cylinders (also published in [Blinn89])

[Blinn89] James F. Blinn, “Jim Blinn’s Corner: Optimal tubes”, *IEEE Computer Graphics and Applications*, v9n5. ACM SIGGRAPH, September 1989, pp. 8–13.

This paper derives two or three polygon cylinders used in JPL space movies and the Mechanical Universe. This also appeared in the harder-to-find [Blinn85].

[Briggs92] Preston Briggs, *Register Allocation via Graph Coloring*, PhD Dissertation, Department of Computer Science, Rice University, Houston, Texas, 1992.

This dissertation is a thorough work on register allocation and SSA (single static assignment) analysis.

[Cabral93] Brian Cabral and Leith (Casey) Leedom, “Imaging Vector Fields Using Line Integral Convolution”, Proceedings of SIGGRAPH 93 (Anaheim, California, August 1–6, 1993). In *Computer Graphics*, Annual Conference Series. ACM SIGGRAPH, August, 1993, pp. 263–272.

This paper introduces line integral convolution as a method for understanding 2D vector fields. Line integral convolution uses a local directional blurring of an underlying texture. The blur direction lies either along or across the flow direction.

[Cook84] Robert L. Cook, “Shade trees”, Proceedings of SIGGRAPH 84 (Minneapolis, Minnesota, July 23–27, 1984). In *Computer Graphics*, v18n3. ACM SIGGRAPH, July 1984, pp. 223–231.

Cook parses arbitrary simple expressions into a parse tree form that is interpreted for shading. Reasonable speed is obtained though a powerful set of precompiled library functions.

[Cook87] Robert L. Cook, “The Reyes Image Rendering Architecture”, Proceedings of SIGGRAPH 87 (Anaheim, California, July 27–31, 1987). In *Computer Graphics*, v21n4. ACM SIGGRAPH, July 1987, pp. 95–102.

This paper presents the Reyes (Renders Everything You Ever Saw) rendering algorithm. The algorithm works by recursively splitting all objects into sub-pixel micropolygons.

[Coquillart90] Sabine Coquillart, “Extended Free-Form Deformation: A Sculpturing Tool for 3D Geometric Modeling”, Proceedings of SIGGRAPH 90 (Dallas, Texas, August 6–10, 1990). In *Computer Graphics*, v24n4. ACM SIGGRAPH, August 1990, pp. 187–196.

This paper extends free-form deformations [Sederberg86] by using non-rectangular lattices.

[Coquillart91] Sabine Coquillart and Pierre Jancène, “Animated free-form deformation: An interactive animation technique”, Proceedings of SIGGRAPH 91 (Las Vegas, Nevada, July 28–August 2, 1991). In *Computer Graphics*, v25n4. ACM SIGGRAPH, July 1991, pp. 23–26.

They animate objects either by animating an EFFD (extended free-form deformation) containing the object or by moving the object through the space of an existing EFFD.

[Crow82] F. C. Crow, “A More Flexible Image Generation Environment”, Proceedings of SIGGRAPH 82 (Boston, Massachusetts, July 26–30, 1982). In *Computer Graphics*, v16n3, ACM SIGGRAPH, July 1982, pp. 9–18.

Crow presents a testbed renderer, in which a control process does some sorting and forks off separate processes for each primitive. The image results from each primitive are later composited together.

[Deering88] Michael Deering, Stephanie Winner, Bic Schediwy, Chris Duffy and Neil Hunt, “The Triangle Processor and Normal Vector Shader: A VLSI System for High Performance Graphics”, Proceedings of SIGGRAPH 88 (Atlanta, Georgia, August 1–5, 1988). In *Computer Graphics*, v22n4, ACM SIGGRAPH, August 1988, pp. 21–30.

This is the first hardware system I know of that uses deferred shading.

[Dietz92] Henry G. Dietz, “Common Subexpression Induction”, Proceedings of the 1992 International Conference on Parallel Processing (Saint Charles, Illinois, August 1992). pp. 174–182.

Presents common subexpression induction (CSI), a technique for SIMD compilers similar to common subexpression elimination, used in traditional compilers. Instead of combining subexpressions that compute the same value, CSI combines equivalent sections of code that execute on different processors of the SIMD array. For example,

code in the two branches of an `if` are normally executed sequentially on a SIMD array, but with CSI, portions of the two branches can be combined.

[Dunlavey79] M. R. Dunlavey, "The Procedural Approach to Interactive Design Graphics", *Computer Graphics*, v13n2. ACM SIGGRAPH, March 1979, pp. 110–147.

This presents a language based CAD system. The user communicates using DL, a simple stack based language with registers. Objects are created by controlling a *turtle* with a milling tool attached.

[Ebert94] David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin and Steven Worley, *Texturing and Modeling: A Procedural Approach*, Academic Press, Boston, 1994.

This is the book version of their long-running SIGGRAPH course. It includes a great deal of practical information about creating procedural shaders and some information on procedural models. The forms of procedural modeling that are covered include hypertexture, volume-based smoke, and fractals (primarily mountains).

[Ellsworth91] David Ellsworth, "Parallel Architectures and Algorithms for Real-time Synthesis of High-quality Images using Deferred Shading". Workshop on Algorithms and Parallel VLSI Architectures (Pont-á-Mousson, France, June 12, 1990).

Ellsworth describes the deferred shading technique, where the surface visible at each pixel is determined before any shading. This avoids spending time to shade pixels that will never show up in the final image.

[Eyles97] John Eyles, Steven Molnar, John Poulton, Trey Greer, Anselmo Lastra, Nick England and Lee Westover, "PixelFlow: The Realization", Proceedings of the 1997 SIGGRAPH/Eurographics Workshop on Graphics Hardware (Los Angeles, California, August 3–4, 1992). ACM SIGGRAPH, August 1997, pp. 57–68.

This is a description of the PixelFlow hardware, as it was finally built. It includes a little on the PixelFlow software, but mostly covers the hardware architecture.

[Fleischer87] Kurt Fleischer, "Implementation of a modeling testbed", Proceedings of SIGGRAPH 87 (Anaheim, California, July 27–31, 1986). *Object-Oriented Geometric Modeling and Rendering seminar notes*. ACM SIGGRAPH, July 1987.

This paper has much of the text of [Fleischer88] with a little more detail

[Fleischer88] K. Fleischer and A. Witkin, "A modeling testbed", *Proceedings of Graphics Interface '88* (Edmonton, Alberta, June 6–10, 1988). Canadian Inf. Process. Society, 1988, pp. 137–137.

They present a LISP based testbed rendering system. Transformations are generalized to functions for position, normal, and an inverse position transform. Objects are generalized to parametric functions for surface position and normal and implicit functions

for surface position. Shaders use a function of position, normal, and parametric position to get the appearance parameters for a reasonably standard shader definition. Also uses a graphical interface similar to building block shaders for connecting the objects, transformations, and shaders together.

[Foley90] James Foley, Andries van Dam, Steven Feiner, John Hughes, *Computer Graphics: Principles and Practice*, Second Edition, Addison-Wesley, 1990.

This is the de-facto default reference for graphics. It is hard to imagine that a single book can have both this scope and detail.

[Fuchs82] Henry Fuchs, John Poulton, "Pixel-Planes: A VLSI-Oriented Design for a Raster Graphics Engine," *VLSI Design*, 2(3), 1982, pp. 20-28.

This is an early paper on the Pixel-Planes style of graphics hardware, with hardware support for evaluating a linear function at every pixel.

[Fuchs85] Henry Fuchs, Jack Goldfeather, Jeff P. Hultquist, Susan Spach, John D. Austin, Frederick P. Brooks, Jr., John G. Eyles and John Poulton, "Fast Spheres, Shadows, Textures, Transparencies, and Image Enhancements in Pixel-Planes", Proceedings of SIGGRAPH 85 (San Francisco, California, July 22–26, 1985). In *Computer Graphics*, v19n3, ACM SIGGRAPH, July 1985, pp. 111–120.

This paper presents the Pixel-Planes 4 architecture. Pixel-Planes 4 was the machine two generations before PixelFlow, and the first UNC graphics engine to receive day-to-day use by other projects in the Computer Science Department.

[Fuchs89] Henry Fuchs, John Poulton, John Eyles, Trey Greer, Jack Goldfeather, David Ellsworth, Steve Molnar, Greg Turk, Brice Tebbs and Laura Israel, "Pixel-Planes 5: A Heterogeneous Multiprocessor Graphics System Using Processor-Enhanced Memories", Proceedings of SIGGRAPH 89 (Boston, Massachusetts, July 31–August 4, 1989). In *Computer Graphics*, v23n3. ACM SIGGRAPH, July 1989, pp. 79–88.

This paper outlines Pixel-Planes 5 hardware and software.

[Glassner91] Andrew S. Glassner, "*Spectrum*: A Proposed Image Synthesis Architecture", Proceedings of SIGGRAPH 91 (Las Vegas, Nevada, July 28–August 2, 1991). *Developing Large-scale Graphics Software Toolkits seminar notes*. ACM SIGGRAPH, July 1991.

This is an initial proposal for the *spectrum* system described more completely in [Glassner93]. It includes code from an initial implementation, written in the MESA language.

[Glassner93] Andrew S. Glassner, "*Spectrum*: An Architecture for Image Synthesis Research, Education, and Practice", Proceedings of SIGGRAPH 93 (Anaheim, California, August 1–6, 1993). *Developing Large-scale Graphics Software Toolkits seminar notes*, ACM SIGGRAPH, August 1993.

Glassner defines a generic architecture for the ultimate testbed rendering system. It is being implemented in C++ to be freely distributed. Spectrum is designed for radiosity and ray tracing, but the modules scheduling is also programmable so it could probably wash your shirts for you. Normal module types are cameras, samplers, reconstructors, seeders, shaders, shapes.

[Glassner95] Andrew S. Glassner, *Principles of Digital Image Synthesis*, Morgan Kaufman, 1995.

This two volume set contains a wealth of knowledge about graphics. It assumes a fairly sophisticated reader, but is perfect you want to understand graphics from a mathematical point of view.

[Grant86] Eric Grant, Phil Amburn and Turner Whitted, "Exploiting classes in modeling and display software", *IEEE Computer Graphics and Applications*, v6n11. IEEE, November 1986, pp. 13–20.

They discusses class hierarchy for their C++ system. Not directly useful to this work, but it is the basis for [Grant87].

[Grant87] Eric Grant, "Class Design for a Modeling Testbed", Proceedings of SIGGRAPH 87 (Anaheim, California, July 27–31, 1987). *Object-Oriented Geometric Modeling and Rendering seminar notes*. ACM SIGGRAPH, July 1987.

This is a discussion of object-oriented testbed approach in [Grant86]. Display classes divisions exist for Z-buffer, A-buffer, etc. It appears that primitives must either reduce to polygons or use custom code for each.

[Green88] Mark Green and Hanqiu Sun, "MML: A language and system for procedural modeling and motion", *Proceedings of Graphics Interface '88* (Edmonton, Alberta, June 6–10, 1988). Canadian Inf. Process. Society, June 1988, pp. 16–25.

Based on C, MML is just a preprocessor. It can generate rule or grammar based models (particle systems, L-systems, graftals, fractals). Objects have state, code for procedural generation, code for motion verbs, and code for rendering. Their examples eventually render using a existing low-level primitives.

[Gritz96] Larry Gritz and James K. Hahn, "BMRT: A Global Illumination Implementation of the RenderMan Standard", *Journal of Graphics Tools*, v1n3, 1996, pp. 29–47.

They describe BMRT (Blue Moon Rendering Toolkit), a ray tracer implementing the RenderMan standard [Upstill90]. BMRT implements both the RenderMan API and shading language.

[Guenter95] Brian Guenter, Todd B. Knoblock and Erik Ruf, “Specializing Shaders”, Proceedings of SIGGRAPH 95 (Los Angeles, California, August 6–11, 1995). In *Computer Graphics* Proceedings, Annual Conference Series, ACM SIGGRAPH, 1995, pp. 343–348.

They use a dependency analysis on their shading language to allow interactive modification to just one or a couple of parameters. The dependency analysis allows them to only recompute the intermediate results that change based on the parameters that change.

[Hall83] R. A. Hall and D. P. Greenberg, “A Testbed for Realistic Image Synthesis”, *IEEE Computer Graphics And Applications*, v3n11. IEEE, November 1983, pp. 10–20.

This is a ray tracing system concerned primarily with the ability to use new illumination models. It includes spectral distributions instead of just RGB. It also has brief mention of the typical ray tracing primitive extensibility using functions to find the intersection of the primitive with an arbitrary ray. They claim their testbed could use other rendering techniques, but it appears that would only be possible for the illumination calculations; the rendering code would need to be replaced.

[Hanrahan90] Pat Hanrahan and Jim Lawson, “A Language for Shading and Lighting Calculations”, Proceedings of SIGGRAPH 90 (Dallas, Texas, August 6–10, 1990). In *Computer Graphics*, v24n4. ACM SIGGRAPH, August 1990, pp. 289–298.

This paper introduces and explains the RenderMan shading language. It includes some details on compilation issues.

[Hart89] John C. Hart, Daniel J. Sandin and Louis H. Kauffman, “Ray Tracing Deterministic 3-D Fractals”, Proceedings of SIGGRAPH 89 (Boston, Massachusetts, July 31–August 4, 1989). In *Computer Graphics*, v23n3, ACM SIGGRAPH, July 1989, pp. 289–296.

They present an algorithm for ray-tracing surfaces defined by the quaternion extension to the Julia set fractal equation [Mandelbrot77]. The method is based on the ability to compute bounding spheres guaranteed **not** to contain the object. A progression of these *unbounding-spheres* allow as accurate an approximation of the fractal surface as necessary.

[Hedelman84] H. Hedelman, “A Data Flow Approach to Procedural Modeling”, *IEEE Computer Graphics and Applications*, v3n1. IEEE, January 1984, pp. 16–26.

With this rendering system, a user can hook together procedural models, transformations, and shaders into a tree. The paper discusses parallel execution of nodes of the tree. It also discusses high level smart culling by procedural models and multiple level-of-detail representations of a single object provided by one model procedure.

[Heidrich96] Wolfgang Heidrich, Philipp Slusallek, Hans-Peter Seidel, “Sampling of Procedural Shaders Using Affine Arithmetic”, Technical Report 11/1996, Universität Erlangen-Nürnburg, 1996.

Introduces the use of affine arithmetic as a method to automatically antialias a procedural shader. Affine arithmetic represents each value as a base value and a number of *noise symbols*, which encode the potential error in the value. Using affine arithmetic for all computations, they are able to compute an average value for a shader over an area of surface and an error bound on the result.

[Hill97] B. Hill, Th. Roger and F. W. Vorhagen, “Comparative Analysis of the Quantization of Color Spaces on the Basis of the CIELAB Color-Difference Formula”, *ACM Transactions on Graphics*, v16n2. ACM, April 1997, pp. 109–154.

This paper presents an analysis of the number of bits required for accurate color representation in different color spaces based on the just-noticeable-difference between colors. They indicate that 12-13 bits per component are necessary for linear encoding in either RGB space or CIE XYZ space.

[Kajiya83] J. T. Kajiya, “New techniques for ray tracing procedurally defined objects”, *ACM Transactions on Graphics*, v2n3. ACM SIGGRAPH, July 1983, pp. 161–181.

This paper presents three related ray tracing techniques. Fractals are contained within hierarchical bounding volumes. The sub-volumes (and final primitives) are only computed when a ray intersects the parent volume. For prisms, the ray is projected into the plane of the prism base and intersection proceeds in 2D. The 2D base is contained by a strip tree. For surfaces of revolution, the ray is transformed to a parabola in a squared space and intersection proceeds in 2D as for prisms.

[Kanus96] Urs Kanus, Michael Meißner, Wolfgang Straßer, Hanspeter Pfister, Arie Kaufman, Rick Amerson, Richard J. Carter, Bruce Culbertson, Phil Kuekes and Greg Snider, “Cube-4 Implementations on the Teramac Custom Computing Machine”, Proceedings of the 11th Eurographics Workshop on Graphics Hardware (Poitier, France, August 26–27, 1996), pp. 133–143.

They present the latest of the *Cube* architectures for real-time volume rendering. Cube-4 still uses the cubic frame buffer of the earlier Cube architectures, but uses a slice-parallel decomposition for parallel rendering. They implemented two versions of the Cube-4 machine on Teramac, a software-configurable hardware machine.

[Kaufman88] Arie Kaufman and Reuven Bakalash, “Voxel Based Processing”, *IEEE Computer Graphics and Applications*, v8n6, November 1988, pp. 10–23.

This paper describes the first *Cube* architectures for volume rendering. The core of the Cube architecture is a cubic frame buffer, storing data for every voxel in the volume. Supporting this frame buffer are three processing units: one for volume acquisition, one for geometric scan conversion, and one for final rendering.

[Kiesewetter80] H. Kiesewetter, “ALGRA, an algebraic-graphic programming language for modeling”, *Eurographics ’80.* North-Holland, September 1980, pp. 249–254.

ALGRA is a follow on to DIGRA 73 – algebraic description of models with some control structures.

[Knoll90a] Thomas Knoll, *Filter Module Interface for Adobe Photoshop,* Adobe Photoshop Developers Kit 1990.

This is the technical specification for writing plug-in filters for Adobe Photoshop.

[Knoll90b] Thomas Knoll, *Writing Plug-in Modules for Adobe Photoshop,* Adobe Photoshop Developers Kit, 1990.

This presents an overview of the workings of Photoshop plug-ins.

[Koenderink90] Jan J. Koenderink, *Solid Shape*, MIT Press, Cambridge, Massachusetts, 1990.

This book gives an introduction to differential geometry as applied to surfaces in 3D. It avoids proofs and generality in favor of giving a visual understanding of the concepts.

[Kolb92] Craig E. Kolb, *Rayshade User’s Guide and Reference Manual,* January 1992.

Rayshade is an extensible ray tracer, evolved from [Kuchkuda88]. New primitives or shading functions require understanding and changing the code. Unfortunately, very little is said in the documentation on how to go about doing either.

[Kuchkuda88] Roman Kuchkuda, “An introduction to ray tracing”, *Theoretical Foundations of Computer Graphics and CAD,* volume F40. Springer-Verlag, 1988, pp. 1039–1060.

This paper includes full C code for a simple ray tracer. The code is designed to make it simple to add new primitives. Each primitive requires instancing, intersection, and normal calculation functions and a small amount of extra lex/yacc code and support code.

[Kumar95] Subodh Kumar, Dinesh Manocha and Anselmo Lastra, “Interactive Display of Large-scale NURBS Models”, *Proceedings of the 1995 Symposium on Interactive 3D Graphics* (Monterey, CA, April 9–12, 1995). ACM SIGGRAPH, April 1995, pp. 51–58.

They present a method of rendering NURBS (non-uniform rational B-splines) using dynamic tessellation. Polygons from the tessellated splines are cached and used in later frames. One implementation used the procedural primitive interface on Pixel-Planes 5, while another used the API on a Silicon Graphics machine.

[Kumar96] Subodh Kumar, “Interactive Rendering of Parametric Spline Surfaces”, *PhD Dissertation*, Department of Computer Science, University of North Carolina at Chapel Hill, 1996.

This dissertation includes the work in [Kumar95].

[Kylander97] Karin Kylander and Olof S. Kylander, *Gimp User Manual v0.901*, ftp://ftp.gimp.org/pub/gimp/manual/Gimpmanual_v0.9.01.pdf, 1997.

This is a draft of a manual for the GIMP (Gnu Image Manipulation Program). As the acronym implies, it is a free image manipulation program similar to Adobe’s Photoshop. The GIMP was initially written by Spencer Kimball and Peter Mattis, but numerous others have contributed to its public software development. It includes a flexible interface for writing plug-ins and extensions in C. This interface has been used to add several interpreted scripting languages (scheme, tcl and perl). The most popular, *Script-fu*, is based on scheme.

[Lane80] J. Lane, L. Carpenter, T. Whitted and J. Blinn, “Scan line methods for displaying parametrically defined surfaces”, *Communications of the ACM*, v23n1. ACM, January 1980, pp. 23–34.

They present three algorithms for scan converting spline surfaces. Blinn’s tracks edges and silhouettes with Newton’s method. Whitted’s approximates edges and silhouettes with cubic curves and subdivide when they are not monotonic or not accurate enough. The Lane-Carpenter algorithm subdivides to a flatness or size limit.

[Lastra95] Anselmo Lastra, Steven Molnar, Marc Olano and Yulan Wang, “Real-time Programmable Shading”, Proceedings of the 1995 Symposium on Interactive 3D Graphics (Monterey, California, April 9–12, 1995). ACM SIGGRAPH, 1995, pp. 59–66.

This paper describes the PixelFlow shading architecture. It includes details about timing, scheduling, and simulation to show that an animation of the classic Pixar bowling image could be possible on a reasonably-sized PixelFlow.

[Leech98] John Leech, “OpenGL Extensions and Restrictions for PixelFlow”, Technical Report TR98-019, Department of Computer Science, University of North Carolina at Chapel Hill, 1998.

This document presents the PixelFlow extensions to the OpenGL graphics API.

[Lewis81] Harry R. Lewis and Christos H. Papadimitriou, *Elements of the Theory of Computation*, Prentice-Hall, 1981.

This is a text on automata and Turing machines. I include it because it has a simple and concise definition and explanation of the halting problem: that certain problems simply cannot be solved by a computer in a finite amount of time.

[Lewis89] John-Peter Lewis, “Algorithms for Solid Noise Synthesis”, Proceedings of SIGGRAPH 89, (Boston, Massachusetts, July 31–August 4, 1989). In *Computer Graphics*, v23n3. ACM SIGGRAPH, July 1989, pp. 263–270.

Lewis gives an excellent overview of noise functions.

[Luebke95] David Luebke and Chris Georges, “Portals and Mirrors: Simple, Fast Evaluation of Potentially Visible Sets”, *Proceedings of the 1995 Symposium on Interactive 3D Graphics*, (Monterey, California, April 9–12, 1995). ACM SIGGRAPH, April 1995, pp. 105–106.

They describe a simple method for reducing the size of the rendered database. The model is partitioned into cells, and each cell is only rendered if it is visible from the current position. Visibility is determined by a conservative view-frustum check. The original implementation was done on Pixel-Planes 5 using the procedural primitive interface.

[Lyon93] Richard Lyon, “Phong Shading Reformulation for Hardware Renderer Simplification”, *Apple Technical Report #43*, Apple Computer, Inc. 1993.

Lyon gives an approximation to Phong shading with much improved error behavior. Ordinary Phong shading has an error $O(\varepsilon^2)$ in the output color from an error of ε in the components of the surface normal. His approximation only has an error of $O(\varepsilon)$.

[Mandelbrot77] Benoit Mandelbrot, *The Fractal Geometry of Nature*. Freeman, San Francisco, 1977.

This is the father of all books on fractals by the grandfather of the field.

[MasPar90] MasPar Computer Corporation, *MasPar Parallel Application Language (MPL) User Guide*, 1990.

This is the user’s guide for the variant of C used to program the SIMD MasPar MP-1 computer. The language uses `singular` and `plural` variable attributes to indicate that operations should happen once for the entire PE array or separately on each PE.

[Max81] Nelson L. Max, “Vectorized Procedural Models for Natural Terrain: Waves and Islands in the Sunset”, Proceedings of SIGGRAPH 81 (Dallas, Texas, July 1981). In *Computer Graphics*, v15n3. ACM SIGGRAPH, August 1981, pp. 317–324.

He describes details for the creation of a “Waves and Islands in the Sunset” animation. It uses procedural models for both the waves and islands.

[Max89] Nelson L. Max, “Smooth appearance for polygonal surfaces”, *The Visual Computer*, v5n3, June 1989, pp. 160–173.

Nelson Max uses polygon mesh and vertex normals to define a quadratic Beziér triangle mesh. He can then uses the derived Beziér mesh to render C^1 smooth silhouettes, normals, shadows, and texture coordinates.

[Max90] Nelson L. Max, “Cone-Spheres”, Proceedings of SIGGRAPH 90 (Dallas, Texas, August 6–10, 1990). In *Computer Graphics*, v24n4. ACM SIGGRAPH, August 1990, pp. 59–62.

This paper presents a generalized cylinder primitive: two spheres and a section of cone tangent to both connecting them.

[Middleton78] T. Middleton, “A Language for Regular Operations in Graphics”, *Computer Graphics*, v11. ACM SIGGRAPH, March 1978, pp. 39–57.

Middleton argues that a base language without support for graphics makes writing graphics code unnatural and inconvenient. He presents a 2d system built on ALGOL 68 with a handful of special data types and operations.

[Molnar91] Steven Molnar, “Image Composition Architectures for Real-time Image Generation”, PhD Dissertation, Department of Computer Science, University of North Carolina at Chapel Hill, 1991.

This dissertation describes the graphics architecture that became PixelFlow.

[Molnar92] Steven Molnar, John Eyles and John Poulton, “PixelFlow: High-speed rendering using image composition”, Proceedings of SIGGRAPH 92 (Chicago, Illinois, July 26–31, 1992). In *Computer Graphics*, v26n2. ACM SIGGRAPH, July 1992, pp. 231–240.

This paper describes the PixelFlow hardware. It lacks the detail of [Molnar91], but the design that is presented is closer to the final PixelFlow architecture.

[Montrym97] John S. Montrym, Daniel R. Baum, David L. Dignam and Christopher J. Migdal, “InfiniteReality: A Real-time Graphics System”, Proceedings of SIGGRAPH 97 (Los Angeles, California, August 3–8, 1997). In *Computer Graphics*, Annual Conference Series, ACM SIGGRAPH, August, 1997. pp. 293–303.

They describe the architecture of the Silicon Graphics InfiniteReality graphics machine. This is a large, parallel graphics machine, completed at about the same time as PixelFlow.

[Muchnick97] Steven Muchnick, *Compiler Design and Implementation*. Morgan Kaufmann, San Francisco, CA, 1997.

This is a text on some of the more advanced techniques of compiler design. It consists of about one chapter on traditional parsing and symbol table concerns, a few chapters on control flow and data flow analysis, and about a dozen chapters on different forms of optimization.

[Nadas87] Tom Nadas and Alain Fournier, "GRAPE: An Environment to Build Display Processes", Proceedings of SIGGRAPH 87 (Anaheim, California, July 27–31, 1987). In *Computer Graphics*, v21n4. ACM SIGGRAPH, July 1987, pp. 75–84.

They have a data flow based C testbed system. The C functions are hooked together in a directed acyclic graph. The paper also includes excellent overview of other systems and a good set of references.

[Nakamae90] Eihachiro Nakamae, Kazufumi Kaneda, Takashi Okamoto and Tomoyuki Nishita, "A Lighting Model Aiming at Drive Simulators", Proceedings of SIGGRAPH 90 (Dallas, Texas, August 6–10, 1990). In *Computer Graphics*, v24n4. ACM SIGGRAPH, August 1990, pp. 395–404.

They present a reflection model for wet roads.

[Neider93] Jackie Neider, Tom Davis and Mason Woo, *OpenGL Programming Guide: the official guide to learning OpenGL release 1*. Addison-Wesley, 1993.

This guide explains the OpenGL application-level graphics library.

[Newell75] Martin Newell, *The Utilization of Procedure Models in Digital Image Synthesis*, PhD dissertation, Department of Computer Science, University of Utah, 1975.

In this dissertation, Newell introduces *procedure models*. The procedure model is an object-oriented form of model representation. it takes arbitrary data, which it interprets in whatever way it wants to build the object. It also is able to respond to a couple of messages: "draw yourself" and "return bounds". He also includes inheritance, so some specialized models are derived from more general ones.

[Niimi84] Haruo Niimi, Yoshirou Imai, Masayoshi Murakami, Shinji Tomita and Hiroshi Hagiwara, "A Parallel Processor System for Three-dimensional Color Graphics", Proceedings of SIGGRAPH 84 (Minneapolis, Minnesota, July 23–27, 1984). In *Computer Graphics*, v18n3. ACM SIGGRAPH, July 1984, pp. 67–76.

This paper describes a graphics architecture based on scan-line rendering. The architecture consists of a set of parallel scan-line units. Each scan-line unit has a set of pixel processors that handle a sub-span on the scan line.

[Olano97] Marc Olano and Trey Greer, “Triangle Scan Conversion Using 2D Homogeneous Coordinates”, Proceedings of the 1997 SIGGRAPH/Eurographics Workshop on Graphics Hardware (Los Angeles, California, August 3–4, 1997). ACM SIGGRAPH, August 1997, pp. 89–96.

This paper describes a triangle scan conversion algorithm that completely avoids traditional clipping tests. Clipping is necessary to avoid potential division problems for vertices that are even with or behind the viewer. The algorithm uses 2D homogeneous coordinates to avoid the problem divisions.

[ONeill66] Barrett O’Neill, *Elementary Differential Geometry*, Academic Press, San Diego, CA, 1966.

This is a great resource for differential geometry.

[OpenGL92] OpenGL Architecture Review Board, *OpenGL Reference Manual: The Official Reference Document for OpenGL, Release 1*. Addison-Wesley, 1992.

This reference manual has *man-page*-like descriptions of all of the OpenGL graphics library functions. It includes a fold-out graphics system diagram.

[Ostby93] Eben F. Ostby, “Implementation of MENV”, Proceedings of SIGGRAPH 93 (Anaheim, California, August 1–6, 1993). *Developing Large-scale Graphics Software Toolkits seminar notes*. ACM SIGGRAPH, August 1993.

Ostby gives background and implementation details on MENV. He includes some information not found in [Reeves90] on partial updates of models from *avar* changes.

[Perlin85] Ken Perlin, “An Image Synthesizer”, Proceedings of SIGGRAPH 85 (San Francisco, California, July 22–26, 1985). In *Computer Graphics*, v19n3. ACM SIGGRAPH, July 1985, pp. 287–296.

Perlin presents his ground-breaking *pixel stream editor*. Essentially it expands Cook’s shade trees work to a full language [Cook84]. This is the earliest example I have of a full language for shading. Includes the Perlin noise function, one of the most useful tools for procedural shading.

[Perlin89] Ken Perlin and Eric M. Hoffert, “Hypertexture”, Proceedings of SIGGRAPH 89, (Boston, Massachusetts, July 31–August 4, 1989). In *Computer Graphics,* v23n3. ACM SIGGRAPH, July 1989, pp. 253–262.

They propose a form of procedurally defined solid objects (volume rendered).

[Pineda88] Juan Pineda, “A Parallel Algorithm for Polygon Rasterization”, Proceedings of SIGGRAPH 88 (Atlanta, Georgia, August 1–5, 1988). In *Computer Graphics*, v22n4, ACM SIGGRAPH, August 1988, pp. 17–20.

He presents an incremental triangle scan conversion algorithm using the same edge-equation math as the Pixel-Planes style implicit scan conversion algorithm.

[Pixar89] Pixar Animation Studios, *The RenderMan Interface*, September 1989.

This provides a technical description of RenderMan, RIB, and the shading language.

[Pixar89b] Pixar Animation Studios, *KnickKnack*, film short, 1989.

This is last of Pixar’s funny short animations before they started work on the Toy Story movie. As with all of their animations, it makes extensive use of procedural shading.

[Pixar97] Pixar Animation Studios, *PhotoRealistic RenderMan 3.7 Shading Language Extensions*, March 1997.

This addendum to [Pixar89] defines the recent additions to the RenderMan shading language standard. These additions include vector, normal and matrix types, matrix operations, arrays, new noise and step functions, time derivatives, and message passing between shading stages.

[Potmesil87] Michael Potmesil and Eric M. Hoffert, “FRAMES: Software Tools for Modeling, Rendering and Animation of 3D Scenes”, Proceedings of SIGGRAPH 87, (Anaheim, California, July 27–31, 1987). In *Computer Graphics*, v21n4, ACM SIGGRAPH, July 1987, pp. 85–93.

FRAMES is a testbed rendering system based on UNIX processes. Each stage of the graphics pipeline is a single process, and the pipeline stages communicate using standard UNIX command-line pipes. The model enters one end of the pipeline as the *standard input* to the first pipeline stage program, and the final frame is produced as the *standard output* of the last pipeline stage program.

[Potmesil89] Michael Potmesil and Eric M. Hoffert, “The Pixel Machine: A Parallel Image Computer”, Proceedings of SIGGRAPH 89, (Boston, Massachusetts, July 31–August 4, 1989). In *Computer Graphics*, v23n3, ACM SIGGRAPH, July 1989, pp. 69–78.

The AT&T Pixel Machine has a pipeline of DSP processors, corresponding to the stages of the graphics pipeline, and an array of DSP processors for final rendering. It had reasonable polygon rendering performance but was really designed for ray-tracing. It did not quite achieve general real-time ray tracing, but could ray trace a simple one-plane, one-sphere scene in a small window at interactive speeds.

[Prusinkiewicz88] Przemyslaw Prusinkiewicz, Aristid Lindenmayer and James Hanan, "Developmental Models of Herbaceous Plants for Computer Imagery Purposes", Proceedings of SIGGRAPH 88, (Atlanta, Georgia, August 1–5, 1988). In *Computer Graphics*, v22n4, ACM SIGGRAPH, August 1988, pp. 141–150.

This is hardly the first work on *L-systems* (named for Lindenmayer), but a good overview. L-systems provide a description of plant and tree branching structures using a grammar where each symbol (terminal or non-terminal) has some meaning for the length or branching structure of the plant.

[Reeves83] William T. Reeves, "Particle Systems – A Technique for Modeling a Class of Fuzzy Objects", Proceedings of SIGGRAPH 83 (Detroit, Michigan, July 25–29, 1983). In Computer Graphics, v17n3, ACM SIGGRAPH, July 1983, pp. 359–376.

This paper introduces particle systems. Particle systems are form of animated model in which a large number of particles obey simple rules for generation, movement, and extinction.

[Reeves90] William T. Reeves, Eben F. Ostby and Samuel J. Leffler, "The MENV Modeling and Animation Environment", *Journal of Visualization and Computer Animation*, v1n1, August 1990, pp. 33–40.

They Describe the *MENV* system in use at PIXAR. MENV provides a set of cooperating tools that communicate through shared memory, semaphores, and message passing. All modeling is done using a special purpose modeling language ML. ML consists of C-like statements, calls to geometric primitives, and calls to geometric operations. Includes distinct concepts of variable scoping hierarchy, object hierarchy, and transformation hierarchy. Animation is accomplished through the use of articulated variables (*avars*).

[Rhoades92] John Rhoades, Greg Turk, Andrew Bell, Andrei State, Ulrich Neumann and Amitabh Varshney, "Real-time procedural textures", Proceedings of the 1992 Symposium on Interactive 3D Graphics (Cambridge, Massachusetts, March 29–April 1, 1992). In *Computer Graphics* special issue. ACM SIGGRAPH, March 1992, pp. 95–100.

This work on our previous machine, Pixel-Planes 5, provided some of the inspiration for this dissertation. Pixel-Planes 5 could do real-time procedural textures, written in an assembler-like interpreted texture language.

[Rubin80] S. M. Rubin and T. Whitted, "A 3-Dimensional Representation for Fast Rendering of Complex Scenes", Proceedings of SIGGRAPH 80. In *Computer Graphics*, v14n3. ACM SIGGRAPH, July 1980, pp. 110–116.

They use hierarchical bounding boxes to accelerate ray tracing. Ray intersection only done with bounding boxes, surfaces are rendered by recursive subdivision. They mention the application of recursive subdivision for procedurally defined surfaces.

[Sabin79] M. A. Sabin, “Software Interfaces for Graphics”, *Methodology in Computer Graphics,* R. A. Guedj and H. Tucker, editors. North-Holland, 1979, pp. 49–78.

Part of the proceedings of the IFIP Workshop on Methodology in Computer Graphics, Seillac, France, 1976. Sabin gives an overview and comparison of picture description languages, stream protocols, and procedure libraries. He covers the DRAW, ARTIST, GPDL, G439, PDL2, MONSTER, and DIGRA languages. The article is followed by a critique by L. Kjelldahl, and notes taken during Sabin and Kjelldahl’s presentations and the following panel.

[Sederberg86] Thomas W. Sederberg and Scott R. Parry, “Free-Form Deformation of Solid Geometric Models”, Proceedings of SIGGRAPH 86 (Dallas, Texas, August 18–22, 1986). In *Computer Graphics Proceedings,* v20n4. ACM SIGGRAPH, August 1986, pp. 151–160.

This paper introduces Free Form Deformation (FFD). Objects are embedded in a 3D tensor product Beziér volume. As the volume is deformed, the objects inside deform as well. This is a simple and powerful form of non-linear transformation.

[Segal92] Mark Segal, Carl Korobkin, Rolf Van Widenfelt, Jim Foran and Paul Haeberli, “Fast Shadows and Lighting Effects Using Texture Mapping”, Proceedings of SIGGRAPH 92 (Chicago, Illinois, July 26–31, 1992). In *Computer Graphics Proceedings,* v26n2. ACM SIGGRAPH, July 1992, pp. 249–252.

They Present methods of projecting textures onto surfaces by using perspective transformations of the texture coordinates.

[Slusallek94] Philipp Slusallek, Thomas Pflaum and Hans-Peter Seidel, “Implementing RenderMan–Practice, Problems and Enhancements”, Proceedings of Eurographics ’94. In *Computer Graphics Forum,* v13n3, 1994, pp. 443–454.

This paper describes the RenderMan implementation done at the University of Erlangen. Their implementation is based on a Monte-Carlo global illumination algorithm. They have added some extensions to the shading language to help guide the Monte-Carlo sampling.

[Slusallek98] Philipp Slusallek, Marc Stamminger, Wolfgang Heidrich, Jan-Christian Popp and Hans-Peter Seidel, “Composite Lighting Simulations with Lighting Networks”, *IEEE Computer Graphics and Applications,* v18n2, IEEE, March 1998, pp. 22–31.

They introduce *lighting networks,* using a graphical interface to connect *LightOp* blocks. Each LightOp handles one form of lighting interaction, and the network as a whole simulates the global illumination of a scene. With additional tags to indicate which LightOps apply to which surfaces, they can efficiently handle simple lighting simulations for some objects and more complex simulations for others.

[Snyder92] John M. Snyder and James T. Kajiya, “Generative Modeling: A Symbolic System for Geometric Modeling”, Proceedings of SIGGRAPH 92 (Chicago, Illinois, July 26–31, 1992). In *Computer Graphics*, v26n2, ACM SIGGRAPH, July 1992, pp. 369–378.

Generative modeling is a form of procedural modeling where models are created by sweeping a *generator* along some path. A simple language is used to describe both the generator and the path and the generator can change shapes as it sweeps along the path.

[Sun89] Sun Microsystems, *SunOS 4.1 Math library source code*, 1989.

This is the C source code for the math library distributed with SunOS version 4.1.

[Taylor93] Russell M. Taylor II, Warren Robinett, Vernon L. Chi, Frederick P. Brooks, Jr., William V. Wright, R. Stanley Williams and Eric J. Snyder, “The nanoManipulator: A Virtual-Reality Interface for a Scanning Tunneling Microscope”, Proceedings of SIGGRAPH 93 (Anaheim, California, August 1–6, 1993). In *Computer Graphics Proceedings*, Annual Conference Series. ACM SIGGRAPH, 1993, pp. 127–134.

The nanoManipulator is a teleoperation system for controlling a scanning tunneling microscope. The display used a procedural primitive on Pixel-Planes 5.

[Taylor94] Russell M. Taylor II, “The nanoManipulator, A Virtual-Reality Interface to a Scanning Tunneling Microscope”, *PhD Dissertation*, Department of Computer Science, University of North Carolina, Chapel Hill, 1994.

This dissertation includes the work covered in [Taylor93].

[ThinkingMachines89] Thinking Machines Corporation, *Connection Machine Model CM-2 Technical Summary*. Thinking Machines Corporation, Version 5.1, May 1989.

This is a high-level description of the CM-2 and the languages available for programming it.

[Trumbore93] Ben Trumbore, Wayne Lyttle and Donald P. Greenberg, “A Testbed for Image Synthesis”, Proceedings of SIGGRAPH 93 (Anaheim, California, August 1–6, 1993). *Developing Large-scale Graphics Software Toolkits seminar notes*, ACM SIGGRAPH, August 1993.

They have a collection of library routines called by user code to facilitate global illumination research.

[Upstill90] Steve Upstill, *The RenderMan Companion*, Addison-Wesley, 1990.

This book/user’s guide is the principle resource for learning the RenderMan scene description interface and shading language.

[vanWijk91] Jarke J. van Wijk, “Spot Noise: Texture Synthesis for Data Visualization”, Proceedings of SIGGRAPH 91 (Las Vegas, Nevada, July 28–August 2, 1992). In *Computer Graphics*, v25n4, ACM SIGGRAPH, July 1991, pp. 309–318.

This paper introduces spot noise. Spot noise is a form of band-limited noise constructed by a sparse convolution method [Lewis89], where kernel elements are randomly placed across the surface. Spot noise modifies the shape and orientation of each kernel spot based on the data to be represented.

[Watt92] Alan Watt and Mark Watt, *Advanced Animation and Rendering Techniques: Theory and Practice*, Addison-Wesley, 1992.

This is an *excellent* general graphics book including coverage of just about everything. For our purposes, there are important chapters on procedural texture mapping and modeling, shading languages and RenderMan, deformations, procedural animation, shadowing, shading, and parametric surfaces.

[Whitted81] T. Whitted and D. M. Weimer, “A software test-bed for the development of 3-D raster graphics systems”, Proceedings of SIGGRAPH 81 (Dallas, Texas, July 1981). In *Computer Graphics*, v15n3. ACM SIGGRAPH, August 1981, pp. 271–277.

They describe a generalized 3d scan line rendering program with support for C coded shaders. This is the earliest example of programmable shading I have. They can interpolate arbitrary parameters to be used for shading.

[Whitted82] T. Whitted and D. M. Weimer, “A Software Testbed for the Development of 3D Raster Graphics Systems”, *ACM Trans. on Graphics*, v1n1. ACM SIGGRAPH, January 1982, pp. 43–57.

This is a more detailed version of their SIGGRAPH 81 paper. Of particular personal interest, it has more detail on structure for the primitive half of the testbed. Primitives have a bounding box to determine the span when they are activated and may deposit new primitives for processing in later spans (ideal for subdivision algorithms).

[Williams78] Lance Williams, “Casting Curved Shadows on Curved Surfaces”, Proceedings of SIGGRAPH 78. In *Computer Graphics*, v12n3, ACM SIGGRAPH, August 1978, pp. 270–274.

This paper introduces shadow-mapping. In a preliminary pass, an image containing depths instead of colors is rendered from the point of view of each light source. During the final rendering pass, the surface locations are transformed into the same coordinate system as the depth buffer. This allows the surface depth to be directly compared to the stored depth. If the surface is farther from the light, it is in shadow.

[Williams83] Lance Williams, “Pyramidal Parametrics”, Proceedings of SIGGRAPH 83 (Detroit, Michigan, July 25–29, 1983). In *Computer Graphics*, v17n3, ACM SIGGRAPH, 1983, pp. 1–12.

Williams introduces MIP-mapping, the widely-used texture anti-aliasing algorithm. A MIP-map includes a pyramid of pre-filtered scaled images. During rendering, the filtered color is determined by interpolating between these. Also introduces reflection mapping as a way of using an image texture to simulate reflection in a scene.

[Wyvill85] Geoff Wyvill and Tosiyasu L. Kunii, “A Functional Model for Constructive Solid Geometry”, *The Visual Computer,* v1n1, July 1985, pp. 3–14.

They present a CSG system rendered with ray tracing. Primitives are defined with functional definitions. CSG operations are done on an octree representation, and ray tracing using the octree and arbitrary functional definitions. The CSG cells can be full, empty, point to a single object, or *nasty* (at the octree resolution limit with more than one object in the cell).