

Non-Photorealistic Rendering

SIGGRAPH 99 Course 17
Monday 9th August 1999

Organizer

Stuart Green
LightWork Design Ltd.

Speakers

David Salesin
Microsoft Research and University of Washington

Simon Schofield
Slade School of Fine Art, London

Aaron Hertzmann
New York University

Peter Litwinowicz
RE:Vision Effects Inc.

Amy Ashurst Gooch
University of Utah

Cassidy Curtis
Pacific Data Images

Bruce Gooch
University of Utah



Abstract

In the history of computer graphics, the area of non-photorealistic rendering (NPR) has emerged relatively recently as a subject area in its own right. Its popularity is reflected in the conference programs of the last several SIGGRAPH events, in which a session in each has been set aside to cover the areas of 'Art, Illustration and Expression'. For the research community, NPR represents a gold mine of opportunity, with recent proponents having addressed a wide range of subject matter, including various artistic styles such as pen and ink, watercolor and pencil sketch.

One of the refreshing aspects of NPR is that it brings closer together the disciplines of art and science; its value is far less on the technical brilliance of the techniques but on the aesthetics of the results, and the scope to convey shape, structure and artistic expression. It is an area that requires artists and engineers to work together to solve new and challenging problems in computer graphics. The course will appeal to artists and technologists alike.

In this course proposal we have brought together a number of leading researchers in the field of NPR with artists and industrialists to provide participants with an excellent grounding in this exciting subject. The panel of eight speakers will provide coverage of the various strands of NPR research and applications, including 2D, 2½D and 3D approaches. The artist's perspective on NPR will be provided, incorporating a critique of different approaches and with reference to the classic techniques of fine art. The application of NPR to areas as diverse as Hollywood movie production and desktop consumer software programs will be covered.

Speaker Biographies

Stuart Green (Organizer)

Stuart Green is the Technical Director with LightWork Design Ltd., a UK-based software house specializing in 3D computer graphics. He received a PhD in Computer Science from the University of Bristol, UK, in 1989, where he researched the application of multiprocessors for solving photorealistic rendering problems. He is the author of "Parallel Processing for Computer Graphics", published by MIT Press. He is currently responsible for LightWork Design's Kazoo Technology Group, and the research and development program for a new line of software products that incorporate non-photorealistic rendering capabilities, released in April 1999. In his capacity as Technical Director of LightWork Design, he speaks regularly at courses and conferences on the subjects of photorealistic and non-photorealistic rendering.

David Salesin

David Salesin is a Senior Researcher at Microsoft Research and an Associate Professor in the Department of Computer Science and Engineering at the University of Washington. His research interests include non-photorealistic rendering, image-based rendering, realistic facial animation, and color reproduction. Dr. Salesin holds an ScB from Brown University and a PhD from Stanford. Prior to his graduate degree, he worked at Lucasfilm and Pixar, where he contributed computer animation for the Academy Award-winning short film, "Tin Toy," and the feature-length film, Young Sherlock Holmes. After his graduate degree, he spent a year as a Visiting Assistant Professor in the Program of Computer Graphics at Cornell. Dr. Salesin has received numerous research and teaching awards, including the NSF and ONR Young Investigator Awards, an Alfred P. Sloan Research Fellowship, an NSF Presidential Faculty Fellowship, a University of Washington Outstanding Faculty Achievement Award, a University of Washington Distinguished Teaching Award, and The Carnegie Foundation 1998-99 Washington Professor of the Year Award. Over the past several years, Dr. Salesin has published some 24 papers at SIGGRAPH, including 9 on the subject of non-photorealistic rendering.

Simon Schofield

In 1993 Simon Schofield received his doctorate from Middlesex University in "Non-photorealistic rendering". His research in the area led to the development of the Piranesi System at The Martin Centre, University of Cambridge Department of Architecture; a 3-D painting and interactive rendering system which incorporates many NPR features. Informatix Software International has subsequently marketed Piranesi. In 1996 Piranesi won the Designetics award at Niccograph 96, Tokyo. He then went on to develop Interactive Music software at the University of Westminster, and working with the artists collective AudioROM, won a BAFTA Interactive award for best overall design of interactive entertainment. He is currently a lecturer at the Slade School of Fine Art, University College London and computer graphic artist for Miller Hare Ltd., London.

Aaron Hertzmann

Aaron Hertzmann is a third-year PhD student at the Media Research Laboratory of New York University. Aaron received his BA in Computer Science and Art & Art History from Rice University in 1996 and his MS in Computer Science from NYU in 1998. He has worked as an intern at Interval Research Corporation and at NEC Research Institute. His current research concerns image-based rendering of human actors from example images and video, and non-photorealistic rendering of images and models. He presented a paper at SIGGRAPH 98 on processing images and video for a painterly effect.

Pete Litwinowicz

Pete Litwinowicz is co-founder of RE:Vision Effects Inc., a software and services company that provides tools for the moving image authoring market. RE:Vision's tools suite employs image processing, image-based rendering, computer vision, 2D and 3D animation technologies. Prior to founding RE:Vision, Pete received his MS from UNC Chapel Hill in 1987 and worked in Apple's research group from 1988 until 1997. Pete has been an active participant at SIGGRAPH, presenting four papers and sitting on the SIGGRAPH electronic theater jury in 1996. Pete has a particular interest in non-photorealistic techniques, and presented a paper entitled "Processing Images and Video for an Impressionist Effect" at SIGGRAPH 1997. Finally, with Pierre Jasmin, Pete co-created the "Motion Paintsm" system that was used to create 8 1/2 minutes of painterly effects for the motion picture "What Dreams May Come", which won an Academy Award for Best Visual Effects. Pete has extensive technical and artistic experience using, designing and developing NPR techniques.

Amy Ashurst Gooch

Amy Ashurst Gooch is a researcher for the Visual Simulation Group in the Department of Computer Science at the University of Utah. Amy earned her BS in Computer Engineering and an MS in Computer Science from the University of Utah. While working on her Masters degree in Computer Science at the University of Utah, she explored interactive non-photorealistic technical illustration as a new rendering paradigm. In addition to being a teaching assistant for the High School Summer Computing Institute and the Advanced Manufacturing class at the University of Utah, she presented a paper at SIGGRAPH 98. Her current research focuses on interactive environments and display algorithms for terrain modeling.

Cassidy Curtis

Cassidy Curtis is an animator at Pacific Data Images. He received his BA in Mathematics from Brown University in 1992, and worked as an intern for R/Greenberg Associates creating special effects for television commercials and high resolution print advertisements. Following that he worked as a production programmer and animator for Xaos, and as a Technical Director with PDI. His first film, *Bric-a-Brac* has appeared in several animation festivals, including SIGGRAPH 95. In 1996 he left industry for academia, becoming a researcher and instructor at the University of Washington. His research interests are in non-photorealistic rendering, and his first project, to simulate the artistic effects of watercolor painting, was presented as a paper at SIGGRAPH 97. He has teaching experience from his time as a Research and Teaching Assistant at Brown University, as a class tutor while at Pacific Data Images and as the primary instructor of the University of Washington annual computer animation class for undergraduates in art, music, and computer science. He recently returned to PDI, where he will be working on their next feature film, *Shrek*.

Bruce Gooch

Bruce Gooch is a graduate student in Computer Science at the University of Utah. He has worked for Ford and for Bacon and Davis Inc. conducting research in the areas of time dependent three-dimensional magnetic fields and corrosion-defect prediction in gas and oil pipelines. He also worked for a number of years as a graphic artist at the Eagle Publications group. Bruce earned his BS in Mathematics from the University of Utah. In addition to his SIGGRAPH 98 paper presentation, Bruce has given a department wide colloquium on Non-Photorealistic Rendering, and will be one of the featured speakers in this year's NSF Science and Technology Center lecture series on computer graphics. His current research focuses on using non-photorealistic rendering to communicate shape, structure, and material properties in automatically drawn technical and medical illustrations.

Course Syllabus

Time	Description	Pages
8.45 – 8.55	Welcome and course overview (Green)	1–1
8.55 – 9.15	Introduction to NPR (Green)	2–1
9.15 – 10.15	Beyond Realism: Aesthetics in Image Synthesis (Salesin) Computer-generated pen-and-ink illustration (8 minutes) Rendering parametric surfaces in pen-and-ink (2 mins) Interactive pen-and-ink illustration (6 minutes) Scale-dependent reproduction of pen-and-ink illustrations (6 minutes) Orientable textures for image-based pen-and-ink illustration (6 minutes) Computer-generated watercolor (6 minutes) Comic chat (6 minutes) Multiperspective panoramas for cel animation (6 mins) Computer-generated floral ornament (6 minutes)	3–1
10.15 – 10.30	Break	
10.30 – 11.30	NPR – The Artist's Perspective (Schofield) The Piranesi system - a short history (30 minutes) Some thoughts and prescriptions on the problems of Painterly Rendering (30 minutes)	4–1
11.30 – 12.00	Painterly Image Processing (Hertzmann) Still image processing (25 minutes) Video processing (5 minutes)	5–1
12.00 – 1.30	Lunch	

Time	Description	Pages
1.30 – 2.30	Image-Based Rendering and NPR (Litwinowicz) Processing Images and Video for an Impressionist Effect (20 minutes) "What Dreams May Come"... moving paintings in time (40 minutes)	6–1
2.30 – 3.00	Introduction to 3D NPR: Silhouettes and Outlines (Hertzmann) Image-Space Algorithms (10 minutes) Object-Space Algorithms (20 minutes)	7–1
3.00 – 3.15	Break	
3.15 – 3.45	Using NPR to communicate Shape (Ashurst Gooch) Use of Lines (12 minutes) Shading (12 minutes) Shadowing (6 minutes)	8–1
3.45 – 4.15	Non-Photorealistic Animation (Curtis) Defining a visual goal (10 minutes) Defining the problem space (10 minutes) Writing usable tools (5 minutes) Optimizing the pipeline (5 minutes)	9–1
4.15 – 4.45	Interactive NPR. (Gooch) Implementing Lines (10 minutes) Implementing Shading (10 minutes) Implementing Shadowing (10 minutes)	10–1
4.45 – 5.30	Kazoo – A case study in NPR (Green) Motivation (5 minutes) Design objectives (5 minutes) Rendering pipeline (10 minutes) Styles (10 minutes) Applications (10 minutes)	11–1

Table of Contents

Speaker Biographies.....	v
Course Syllabus.....	vii
Table of Contents.....	ix
CD-ROM Contents.....	xi
1 Welcome.....	1-1
2 Introduction to Non-Photorealistic Rendering.....	2-1
3 Beyond Realism: Aesthetics in Image Synthesis	3-1
4 Non-Photorealistic Rendering - The Artist's Perspective	4-1
5 Painterly Image Processing.....	5-1
6 Image-Based Rendering and Non-Photorealistic Rendering	6-1
7 Introduction to 3D NPR – Silhouettes and Outlines	7-1
8 Using Non-Photorealistic Rendering to Communicate Shape	8-1
9 Non-Photorealistic Animation	9-1
10 Interactive Non-Photorealistic Rendering.....	10-1
11 Kazoo – A Case Study in Non-Photorealistic Rendering	11-1
12 Bibliography.....	Bibliography-1

CD-ROM Contents

The CD-ROM folder for the NPR course contains the following items:

c17.pdf	This course notes document. Please note that the CD-ROM version does not include paper reprints.
kazoo\	A folder containing a Kazoo demonstration for Microsoft Windows.
install.exe	The Kazoo Viewer installer for Microsoft Windows 95, 98 and NT 4.0.
readme.txt	Readme for the Kazoo Viewer.
piranesi\	A folder containing a Piranesi demonstration for Microsoft Windows.
install.exe	The Piranesi demonstration installer for Microsoft Windows 95, 98 and NT 4.0.
readme.txt	Readme for the Piranesi demonstration.

1 Welcome

Welcome to the first SIGGRAPH course dedicated to the subject of non-photorealistic rendering (NPR). You may have seen elements of NPR techniques covered previously in other courses, such as those on advanced rendering and image-based rendering. You will also have seen the technical paper sessions dedicated to *Art, Illustration and Expression*. But this is the first time you will be able to hear prominent researchers and industrialists discuss techniques and trends on the subject.

Whether you are an artist or a technologist, this course will have something for you. The speakers will cover NPR from an artist's perspective, from the point of view of production animation, as well as covering the breadth of technical approaches that have been documented in the research literature.

The printed version of the course notes includes a compilation of recent papers on NPR. If you have access to the SIGGRAPH '99 Course Notes CD-ROM then be sure to check out the commercial NPR software for Windows provided in the **kazoo** and **piranesi** folders.

Stuart Green
stuart.green@lightwork.com
<http://www.lightwork.com>

2 Introduction to Non-Photorealistic Rendering

Stuart Green
LightWork Design Ltd.

From the first moments of the pioneering work in computer graphics some 30 years ago, the quest for realism has been an enduring goal. The term *photorealism* has been coined to denote techniques and art forms in which proponents strive to create synthetic images that are so lifelike they might be mistaken for photographs of real world scenes and objects. While advocates of photorealism have, over the years, conceived a wide range of methods and algorithms for synthetic image generation, the purest and dominant form of these has been *physically-based* techniques. These are inspired and driven by observations of the physical world, in which the interaction of light with the surfaces and objects in an environment, and the projection of an image on the film within a camera, are the key mechanisms to be emulated.

The techniques of ray tracing and radiosity have become established as powerful complementary tools in the photorealistic rendering toolbox. In both cases, the physical behavior of light is *simulated* within a virtual world to yield, in the best cases, fine examples of the hallmarks of photorealism.

Pursuit of photorealism through simulation is the most demanding of tasks. Our world is incredibly rich and diverse; the processes of nature are complex and often hard to predict precisely. Today, researchers can define synthetic environments that contain just the right kind of objects made from the right kind of materials, illuminated by the right kind of lights, and from these create a convincing image. But we are still not ready to respond to the challenge of the general case. Many researchers have focused on the diverse elements of the physical world, such as materials simulation, light interaction, performance of optical systems and of the chemical process that transforms a momentary exposure of light on the surface of photographic film into a permanent image. To solve the general photorealism problem there is still original research work left to do.

It is curious to note that often researchers set about a solution to these demanding physical simulations not because there is necessarily a clearly identified need for a solution, but because the technology can be applied to arrive at a solution. This is often the case of technologists with a solution in search of a problem. Certainly, there are plenty of viable applications of photorealism, including special effects in film and design visualization, however, the techniques applied are often empirical rather than physically-based – simulating reality is not so important as creating the *illusion* of reality in the mind of the observer.

Photorealism Defined

Before defining NPR, it is necessary to understand what is *really* meant by the term ‘photorealistic rendering’. **Photo** comes from the Greek *phos*, meaning light, or produced by light. **Realistic** means depicting or emphasizing what is real and actual, rather than abstract or ideal. **Rendering**, in this context, is traditionally regarded to mean a perspective drawing showing, for example, an architect’s idea of a finished building. In the computer graphics community, rendering is taken to refer to the process by which the representation of a virtual scene is converted into an image for viewing.

The art of photorealistic rendering had many proponents long before the birth of computer graphics. The work of the artist Johannes Vermeer (1632-1675) exhibits properties that are photographic in their quality. The perspective within the subject matter is very closely observed, and brush strokes are short and subtle and cannot easily be discerned in the finished painting. It is widely believed that Vermeer used a *camera obscura* to compose his paintings – a lens system and a projection screen was placed in front of the subject (an indoor scene) and the projected image was copied precisely. Despite whatever mechanical aids Vermeer may have used, his work stands as a most impressive example of photorealistic rendering at its best. However, some critics of his day were less appreciative of the work, accusing it of being cold, inartistic and prone to displaying ‘spiritual poverty’. Since the advent of photography, photorealism in art has become less fashionable, and has given way to more abstract and stylized forms of representation.

It is interesting that the criticisms of Vermeer’s work in the 17th century are equally true of modern practitioners of photorealistic computer graphics.

Certainly, the early attempts at photorealistic computer graphics, while technically impressive, were regarded as sterile and cold, being too perfect and lacking feeling. The efforts of numerous researchers over the last 30 years have placed a rich set of tools in the hands of the artist. Yet it is important to recognize that photorealistic rendering is not nor ever will be the panacea of the artist; it is simply one of many art forms at the artist’s disposal. An important skill of the artist is in choosing the right medium for each job, which will be guided by such considerations as aesthetic appeal and effectiveness of communicating the visual message.

Introducing NPR

A few years ago, the SIGGRAPH conference began to set aside a technical session dedicated to alternative forms of artistic expression. This research was often in marked contrast to that of the photorealistic rendering advocates, and became known as **non-photorealistic rendering** (NPR). It is rather absurd to describe a field of research and development after that which it is not, yet the term has endured and has become adopted by the computer graphics community to denote forms of rendering that are not inherently photorealistic. The terms *expressive*, *artistic*, *painterly* and *interpretative* rendering are often preferred by researchers of the field since they convey much more definitively what is being sought.

The term NPR is used throughout these course notes in deference to current popular terminology, but that term is itself a hindrance to the NPR movement. By analogy, it would be like categorizing the whole of fine art into ‘Impressionist’ and ‘Non-Impressionist’, and using the latter term to categorize all art forms by the fact that they are not in keeping with the Impressionist style. To do so has the effect of de-emphasizing and degrading other art forms. A richer vocabulary is needed to enable the art forms to develop through the written and spoken word as well as through the practice of the art itself.

So what is NPR? A definition along the lines of “a means of creating imagery that does not aspire to realism” is fundamentally flawed. For example, the images of Figure 2 fit this definition, but could they be regarded as examples of NPR? To strive for a definition of NPR is as pointless as defining “Non-Impressionist”. The field is in its infancy, and it is hard to be specific about what it is (and therefore more convenient to state what it is not). The richness and diversity of computer-assisted art forms is at least as



Figure 1: The Music Lesson by Vermeer.

wide as those of traditional art. Photorealism is but one form of representation; in time the term NPR will be replaced by others that are more specific to the branches of computer assisted art forms.

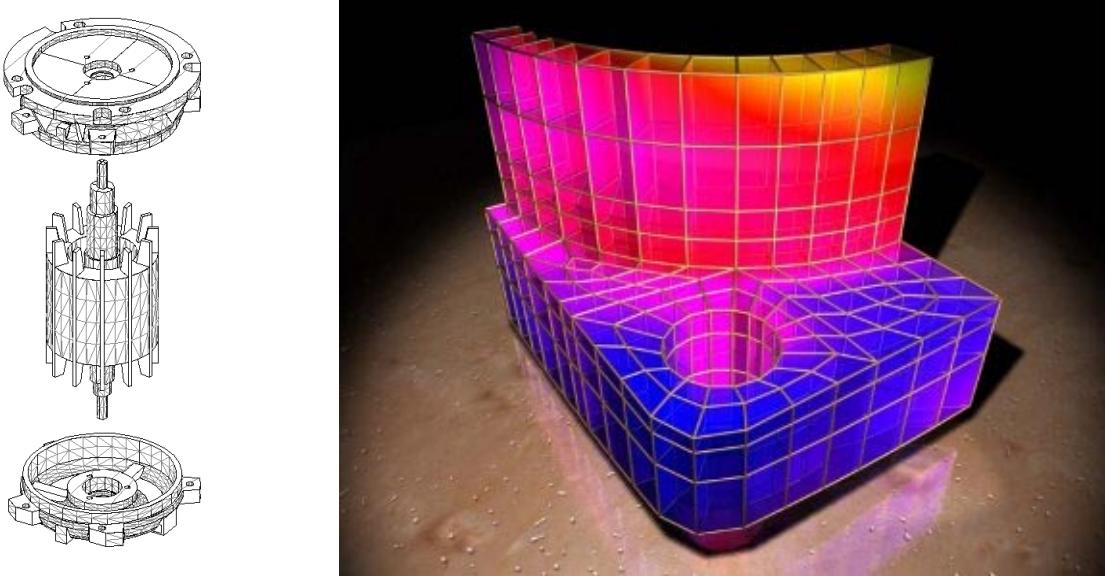


Figure 2: These images are not regarded as photorealistic. But are they NPR?

Not surprisingly, early proponents of NPR have focused their attention on *natural media emulation* – the reproduction of traditional art forms, such as styles of pen and ink, watercolor and oil on canvas. Natural media emulation can be regarded as one branch of NPR research. But NPR offers a much wider scope than this and the opportunity to experiment with new art forms that have not previously been popularized, either because they have not been ‘discovered’ or because those art forms would be impractical to create by hand.

In contrast to photorealism, in which the driving force is the modelling of physical processes and behavior of light, the processes of *human perception* can drive NPR techniques. This can be just as demanding as physical simulation but for different reasons – a technologist may find comfort in developing algorithms to reproduce a certain physical phenomenon that is objective and relatively predictable. Developing techniques to replace, augment or even assist the subjective processes of an artist requires a shared understanding of the use to which these techniques will be put, and the creative processes that pave the way. The finest examples of NPR work will be produced when artists and technologists work together to identify a problem and develop solutions that are sympathetic to the creative processes.

Comparing and Contrasting Photorealism and NPR

Table 1 provides a comparison of the trends of photorealism and NPR.

	Photorealism	NPR
<i>Approach</i>	Simulation	Stylization
<i>Characteristic</i>	Objective	Subjective
<i>Influences</i>	Simulation of physical processes	Sympathies with artistic processes; perceptual-based
<i>Accuracy</i>	Precise	Approximate
<i>Deceptiveness</i>	Can be deceptive or regarded as 'dishonest'; viewers may be misled into believing that an image is 'real'	Honest – the observer sees an image as a <i>depiction</i> of a scene
<i>Level of detail</i>	Hard to avoid extraneous detail; too much information; constant level of detail	Can adapt level of detail across an image to focus the viewer's attention
<i>Completeness</i>	Complete	Incomplete
<i>Good for representing</i>	Rigid surfaces	Natural and organic phenomena

Table 1: Comparing and Contrasting Photorealism and NPR

An Overview of NPR Research

Here we provide a short overview of the trends in NPR research in recent years. This is not intended to be exhaustive, but simply to give an indication of approaches that have been popular, and some examples of results. More details on the approaches are given in the later sections of this course.

Historically, NPR research can be regarded as having originated in early 2D interactive paint systems, such as Quantel Paintbox. These systems provided synthetic artist drawing objects, such as air brushes and pencil, and the user applied these to a canvas to create pixel-based effects. Researchers have developed these techniques further and two prominent areas emerged: 2D brush-oriented painting involving more sophisticated models for brush, canvas, strokes, etc., and 2D/2½D post-processing systems in which raw or augmented image data is used as the basis for image processing. A number of researchers have explored techniques that can be applied to photographic images to synthesize painterly renderings of those images.

One of the key approaches that separate branches of research in the field is the degree to which user intervention is required. Some researchers have favored automatic techniques that require no or very limited user input, while others use the computer to place strokes at the guidance of the artist. 2½D paint systems have been developed in which augmented image data is used to automate paint actions initiated by the artist on pre-rendered scenes.

A more recent trend of NPR research has been the adoption of 3D techniques. The classic 3D computer graphics rendering pipeline exposes a number of opportunities within which NPR techniques can be applied to manipulate data in both 3D and 2D forms. A number of researchers have focused on providing real time algorithms for NPR which afford stylized visualizations of 3D data that can be manipulated interactively. The view-independence of some of these approaches provides obvious benefits for the generation of animation sequences.

Figure 3, from [Teece98b], provides a historical summary of the emergence of NPR research.

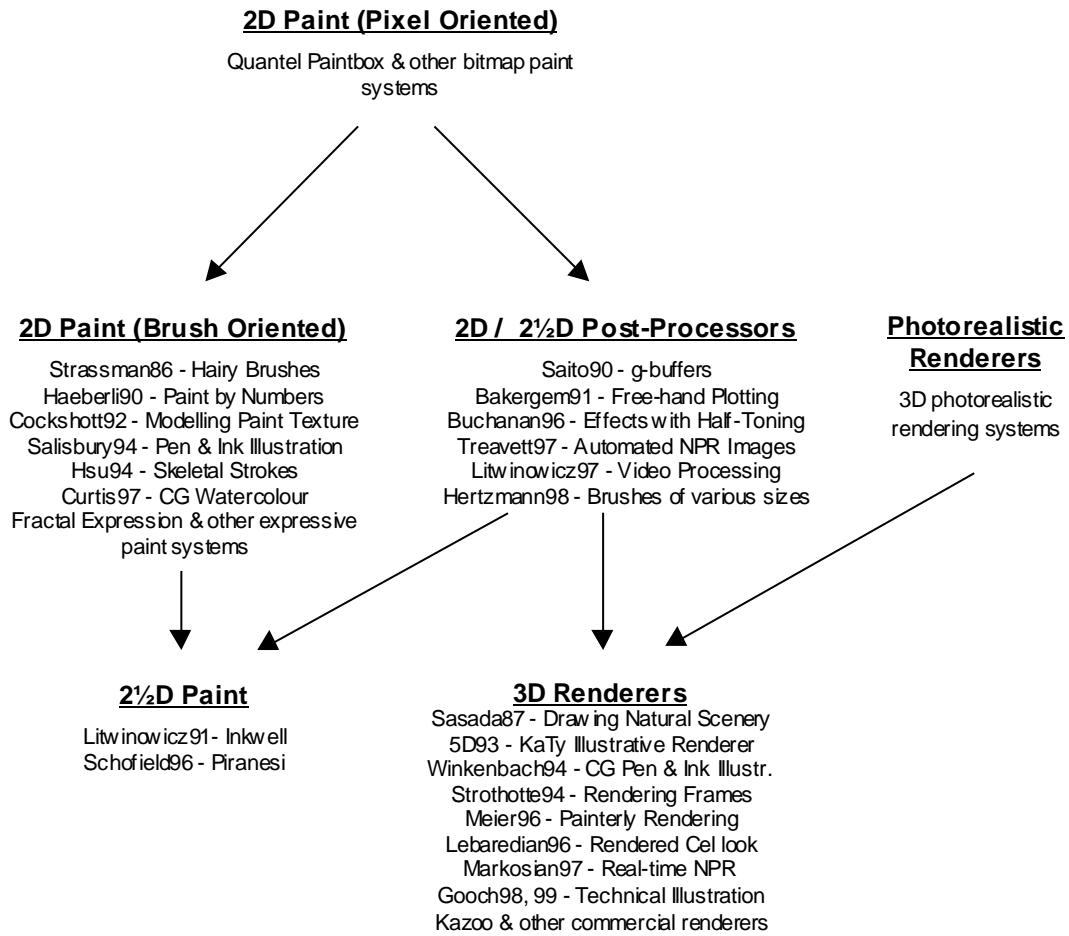


Figure 3: The Emergence of NPR, adapted from [Teece98b]

Taxonomy for NPR is provided by [Teece98b], in which different approaches are classified primarily according to whether they are 2D or 3D, and whether user intervention is required.¹ This provides four principle categories of NPR research as summarized in Table 2.

¹ Note that the term *interactive* is used with two different meanings in NPR research. It is used to convey the interactivity of the artist in directing the software system, and also to refer to rendering approaches that yield images in real time. For clarity, we use here the term *user intervention* for the former and *real time* for the latter.

	2D or 2½D	3D
No User Intervention	Saito90 – Rendering 3D Shapes Bakergem91 – Free-hand Plotting Buchanan96 – Effects with Half-Toning Litwinowicz97 – Video Processing Treavett97 – Automated NPR Images Hertzmann98 – Using brushes of multiple sizes	Sasada87 – Drawing Natural Scenery 5D93 – KaTy Illustrative Renderer Winkenbach94 – CG Pen & Ink Illustr. Strothotte94 – Rendering Frames Elber95 – Line Art Rendering Meier96 – Painterly Rendering Lebaredian96 – Rendered Cel Look Claes97 – Networked Rendering Markosian97 – Real-time NPR Gooch98 – Technical Illustration Gooch99 – Interactive Technical Illustration
User Intervention	Strassman86 – Hairy Brushes Haeberli90 – Paint by Numbers Litwinowicz91 – Inkwell Cockshott92 – Modelling Paint Texture Hsu94 – Skeletal Strokes Salisbury94 – Pen & Ink Illustration Schofield96 – Piranesi Curtis97 – CG Watercolour	Teece98a/b – 3D Expressive Painter

Table 2: A taxonomy of NPR systems, adapted from [Teece98b]

Teece further examines the system objectives of each work, and considers the following as being the secondary level differentiation in the published research:

- **Natural Media Emulation** – intended to mimic one distinctive artistic medium.
- **Image Enhancement** – the application of effects to increase the visual qualities of the final image.
- **Artistic Expression** – aim to give the user the greatest degree of control over image synthesis.
- **Animation** – focused on producing animated imagery rather than still images.

Some examples of work in the four primary categories are given below to provide an indication of the research areas that have been pursued.

Classification: 2/2½D, No User Intervention

This classification is illustrated by the work of Litwinowicz, in which ordinary video segments are transformed into animations that have the hand-painted look of an Impressionist effect [Litwinowicz97].



Figure 4: Processing Images and Video for an Impressionist Effect, from [Litwinowicz97]

Hertzmann has described an approach to hand painting an image using a series of spline brush strokes. A painting is built up as a series of layers of progressively smaller brushes [Hertzmann98].

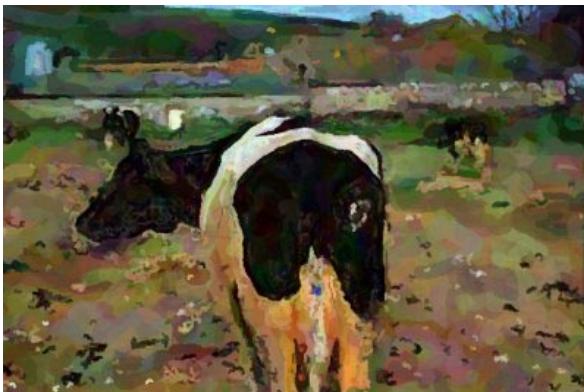


Figure 5: Painterly Rendering with Curved Brush Strokes of Multiple Sizes, from [Hertzmann98]

Classification: 3D, No User Intervention

The work of Winkenbach provides emulation of pen-and-ink illustration by rendering 3D geometry in conjunction with stroke textures [Winkenbach94].

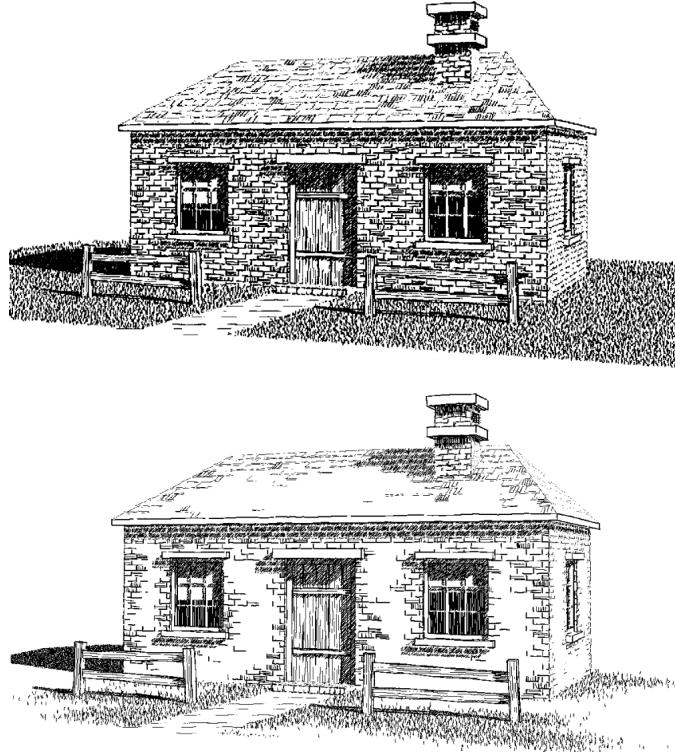


Figure 6: Pen-and-ink renderings from [Winkenbach94]

Gooch has developed a non-photorealistic lighting model that attempts to emulate the richness of hand-drawn technical illustration [Gooch98]. The lighting model uses luminance and changes in hue to indicate surface orientation, and gives a clearer picture of shape, structure and material composition than traditional computer graphics methods.

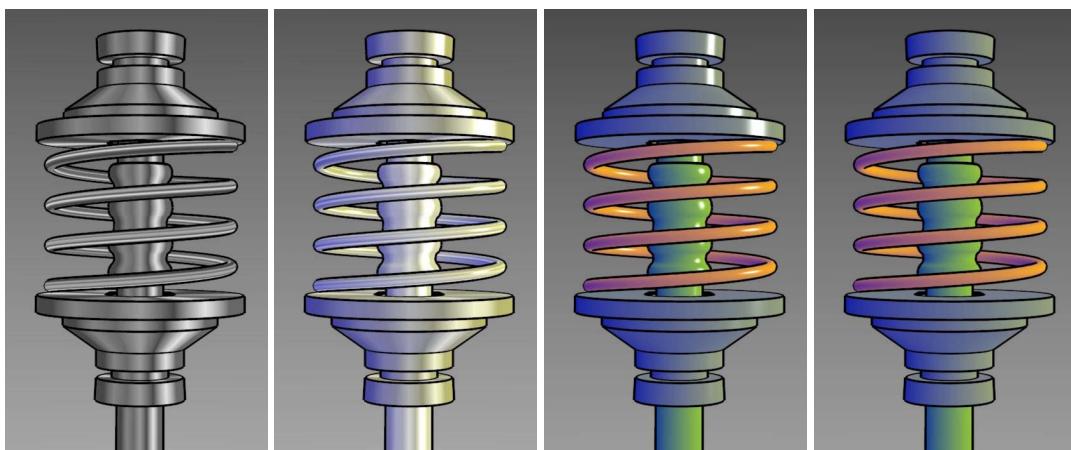


Figure 7: Non-Photorealistic Lighting Model for Automatic Technical Illustration, from [Gooch98]

Classification: 2/2½D, User Intervention

An example of this classification of approach is the work of Salisbury on interactive pen and ink illustration [Salisbury94], in which parameterized pen strokes are synthesized and combined to form stroke textures. The user controls the placement of these stroke textures which are applied to a 2D image that has been rendered from a 3D model.

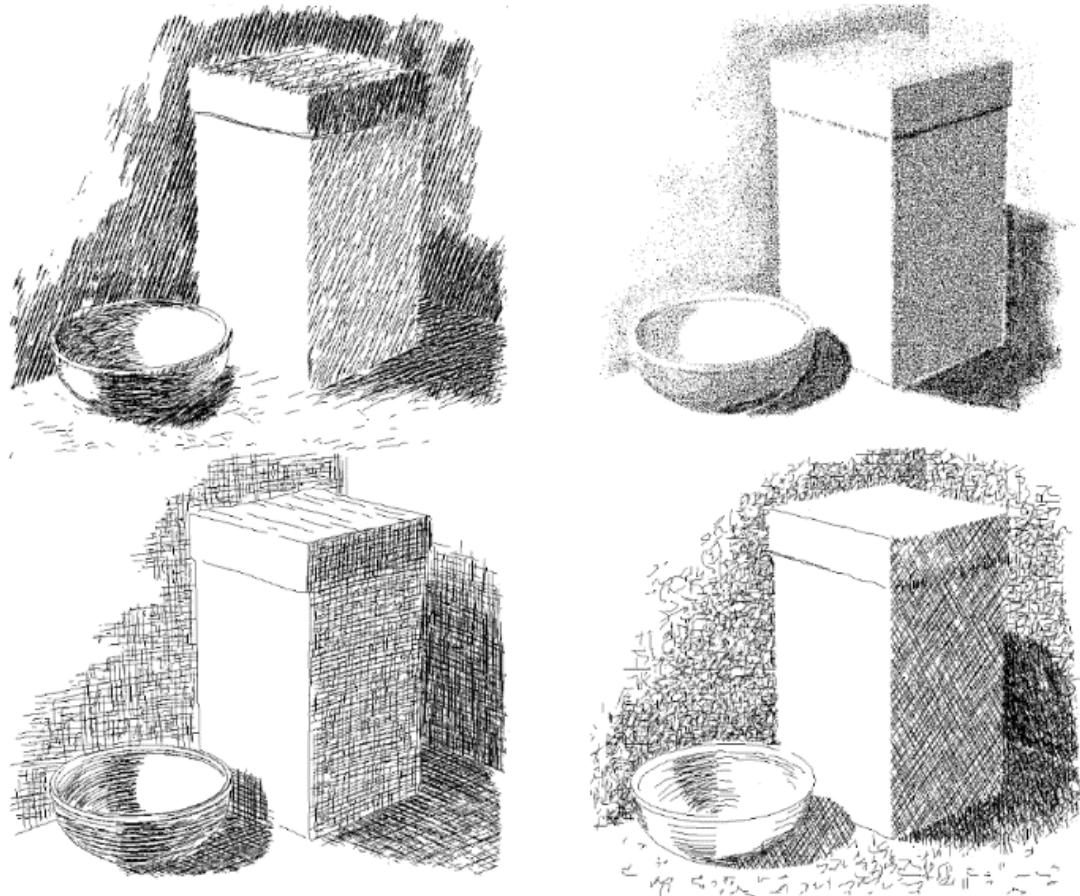


Figure 8: Interactive Pen-and-Ink Illustration, from [Salisbury94]

The work of Schofield focuses on interactive techniques designed to provide the artist with expressive freedom when working on augmented 2D images produced by a rendering package [Schofield94].

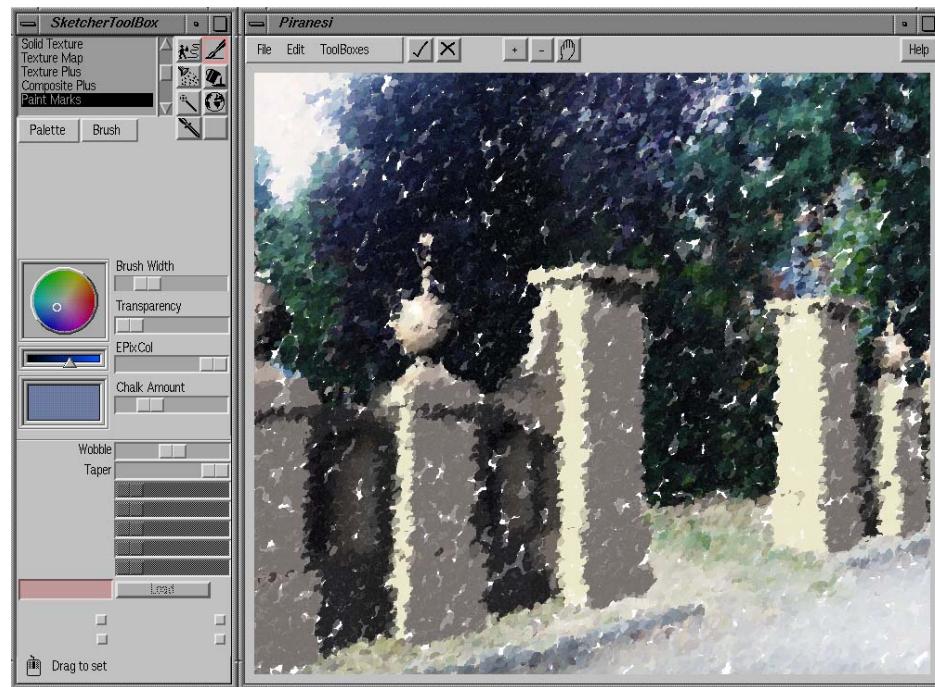


Figure 9: Schofield's Piranesi Painting System

Curtis has developed a method for automatic generation of watercolor effects using a water fluid simulation approach [Curtis97].



Figure 10: Computer Generated Watercolor, from [Curtis97]

Classification: 3D, User Intervention

In his thesis, Teece describes a new approach in the hitherto overlooked category of interactive 3D NPR [Teece98a, Teece98b]. In this system, the user interactively places brush strokes on the surfaces of 3D models, and these strokes can be subsequently replayed from various viewpoints. Once the strokes have been placed, animations of painted scenes can be generated automatically.

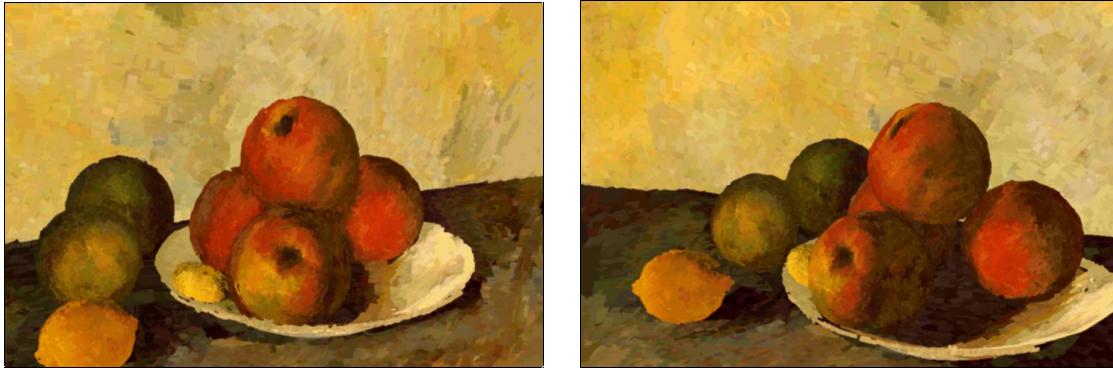


Figure 11: Two frames of an animation sequence from [Teece98b]

Summary

The area of NPR has emerged as a consequence of work to yield alternate rendering approaches from photorealistic computer graphics. The term *non-photorealistic rendering* is without a precise definition, and has come to be used as a blanket term for methods that are not driven by the pursuit of realism, but more usually by human perception and processes employed by the artist. Indeed, much of the research in recent years has focused on natural media emulation, to recreate effects such as pen-and-ink, watercolor and oil on canvas.

One of the refreshing aspects of NPR is that it brings closer together the disciplines of art and science; its value is far less on the technical brilliance of the techniques but on the aesthetics of the results, and the scope to convey shape, structure and artistic expression. It is an area that requires artists and engineers to work together to solve new and challenging problems in computer graphics

References

- [Bakergen91] W. D. van Bakergen and G. Obata, "Free-hand plotting - is it Live or is it Digital", CAD Futures 91, Vieweg, Wiesbaden.
- [Berman94] D. F. Berman, J. T. Bartell and D. H. Salesin, "Multiresolution Painting and Compositing", Computer Graphics (Proc. Siggraph), ACM SIGGRAPH, ACM Press, July 1994.
- [Buchanan96] John W. Buchanan, "Special Effects with Half-Toning", Eurographics '96 proceedings, Volume 15, Number 3, Blackwells Publishers, 1996.
- [Claes97] Johan Claes, Patrick Monsieurs, Frank Van Reeth and Eddy Flerackers, "Rendering Pen-drawings of 3D scenes on networked processors", WSCG '97 proceedings, Volume 1, February 1997.
- [Curtis97] Cassidy Curtis, Sean E. Andersen, Joshua E. Seims, Kurt W. Fleischer and David H. Salesin, "Computer-Generated Watercolour", Computer Graphics (Proc. Siggraph), ACM SIGGRAPH, ACM Press, August 1997.

- [Elber95] Gershon Elber, "Line Art Rendering via a Coverage of Isoparametric Curves", IEEE Transactions on Visualization and Computer Graphics, Volume 1, Number 3, September 1995.
- [Gooch98] Amy Gooch, Bruce Gooch, Peter Shirley, and Elaine Cohen. "A Non-photorealistic Lighting Model for Automatic Technical Illustration." Computer Graphics (Proc. Siggraph), ACM SIGGRAPH, July 1998.
- [Gooch99] Bruce Gooch, Peter-Pike Sloan, Amy Gooch, Peter Shirley, and Richard Riesenfeld. "Interactive Technical Illustration". Interactive 3D Conference Proceedings, April 1999.
- [Haeberli90] Paul Haeberli, "Paint by Numbers: Abstract Image Representations", Computer Graphics (Proc. Siggraph), Vol. 24, No. 4, ACM SIGGRAPH, ACM Press, August 1990.
- [Hertzmann98] Aaron Hertzmann, "Painterly Rendering with Curved Brush Strokes of Multiple Sizes". Computer Graphics (Proc. Siggraph), pages 453–460. ACM SIGGRAPH, July 1998.
- [Hsu94] Siu Chi Hui and Irene H. H. Lee, "Drawing and Animation using Skeletal Strokes", Computer Graphics (Proc. Siggraph), ACM SIGGRAPH, ACM Press, August 1994.
- [Lebaredian96] Rev Lebaredian, "Traditional Cel Animation Look with 3D Renderers", Siggraph 96 Visual Proceedings, ACM SIGGRAPH, ACM Press, 1996.
- [Litwinowicz91] Peter Litwinowicz, "Inkwell: A 2½-D Animation System", Computer Graphics (Proc. Siggraph), Vol. 25, No. 4, ACM SIGGRAPH, ACM Press, 1991.
- [Litwinowicz97] Peter Litwinowicz, "Processing Images and Video for An Impressionist Effect", Computer Graphics (Proc. Siggraph), ACM SIGGRAPH, ACM Press, 1997.
- [Markosian97] Lee Markosian, Michael A. Kowalski, Samuel J. Trychin, Lubomir D. Bourdev, Daniel Goldstein and John F. Hughes, "Real-Time Nonphotorealistic Rendering", Computer Graphics (Proc. Siggraph), ACM SIGGRAPH, ACM Press, 1997.
- [Meier96] Barbara Meier, "Painterly Rendering for Animation", Computer Graphics (Proc. Siggraph), 1996.
- [Saito90] Takafumi Saito and Tokiichiro Takahashi, "Comprehensible Rendering of 3D Shapes", Computer Graphics (Proc. Siggraph), Vol. 24, No. 4, ACM SIGGRAPH, ACM Press, August 1990.
- [Salisbury94] Michael P. Salisbury, Shaun E. Anderson, Ronen Barzel and David H. Salesin, "Interactive Pen-and-Ink Illustration", Computer Graphics (Proc. Siggraph), Vol. 28, No. 4, October 1994.
- [Salisbury96] Michael P. Salisbury, Corin Anderson, Dani Lischinski and David H. Salesin, "Scale-Dependent Reproduction of Pen-and-Ink Illustrations", Computer Graphics (Proc. Siggraph), ACM SIGGRAPH, ACM Press, 1996.
- [Salisbury97] Michael P. Salisbury, Michael T. Wong, John F. Hughes and David H. Salesin, "Orientable Textures for Image-Based Pen-and-Ink Illustration", Computer Graphics (Proc. Siggraph), ACM SIGGRAPH, ACM Press, August 1997.

- [Sasada87] Tsuyoshi T. Sasada, "Drawing Natural Scenery by Computer Graphics", Computer-Aided Design, Vol. 19, No. 4, May 1987.
- [Schofield94] Simon Schofield, "Nonphotorealistic Rendering", Doctoral Dissertation, Middlesex University, England, 1994.
- [Schofield96] Simon Schofield, "Piranesi: A 3-D Paint System", Eurographics UK 96 Conference Proceedings, 1996.
- [Strassman86] Steve Strassman, "Hairy Brushes", Computer Graphics (Proc. Siggraph), Vol. 20, No. 4, ACM SIGGRAPH, ACM Press, August 1986.
- [Teece98a] Daniel Teece, "3D Painting for Non-Photorealistic Rendering", SIGGRAPH '98 Conference Abstracts and Applications, ACM SIGGRAPH, ACM Press, July 1998.
- [Teece98b] Daniel Teece, "Three Dimensional Interactive Non-Photorealistic Rendering", PhD Thesis, University of Sheffield, England, 1998.
- [Treavett97] S.M.F. Treavett and M. Chen, "Statistical techniques for the automated synthesis of non-photorealistic images", Proc. 15th Eurographics UK Conference, March 1997.
- [Winkenbach94] Georges Winkenbach and David H. Salesin, "Computer-Generated Pen-and-Ink Illustration", Computer Graphics (Proc. Siggraph), Vol. 28, No. 4, ACM SIGGRAPH, ACM Press, 1994.
- [Winkenbach96] Georges Winkenbach and David H. Salesin, "Rendering Parametric Surfaces in Pen and Ink", Computer Graphics (Proc. Siggraph), ACM SIGGRAPH, ACM Press, 1996.

3 Beyond Realism: Aesthetics in Image Synthesis

David Salesin

Microsoft Research and University of Washington

Abstract

In many applications, such as automotive, industrial, architectural, and graphic design, and whenever effective communication is the goal, illustrations have certain advantages over photorealism. They convey information better by omitting extraneous detail, by focusing attention on relevant features, by clarifying, simplifying, and disambiguating shapes, and by showing parts that are hidden. Illustrations also provide a more natural vehicle for conveying information at different levels of detail. In many respects, illustrations are also more attractive: they add a sense of vitality difficult to capture with photorealism.

In this talk, I will discuss a variety of algorithms for creating non-photorealistic illustrations automatically, starting from continuous tone images, three-dimensional computer graphics models, or communications from an on-line “chat room” as input. Our early results, published in nine papers at SIGGRAPH over the last five years, include, among other things, support for:

- resolution-dependent pen-and-ink rendering, in which the choice of strokes used to convey both texture and tone is appropriately tied to the resolution of the target medium;
- the automatic “watercolorization” of source images;
- a system for representing on-line communications in the form of comics;
- an approach for simulating apparent camera motion through a 3D environment using a moving window over a single 2D background image; and
- the automatic ornamentation of regions of the plane with floral patterns.

The research I'll describe is joint work with a number of colleagues: Corey Anderson, Sean Anderson, Ronen Barzel, Cassidy Curtis, Adam Finkelstein, Kurt Fleischer, John Hughes, David Kurlander, Dani Lischinski, Mike Salisbury, Josh Seims, Tim Skelly, Craig Thayer, Georges Winkenbach, Michael Wong, Daniel Wood, and Doug Zongker.

References

The following papers will be discussed in this presentation:

- Computer-generated pen-and-ink illustration**, with G. Winkenbach. Proceedings of SIGGRAPH 94, in *Computer Graphics* Proceedings, Annual Conference Series, 91-100, July 1994. Also available as Department of Computer Science and Engineering Technical Report TR 94-01-08, University of Washington, 1994.
- Rendering parametric surfaces in pen and ink**, with G. Winkenbach. Proceedings of SIGGRAPH 96, in *Computer Graphics* Proceedings, Annual Conference Series, 469-476, August 1996. Also available as Department of Computer Science and Engineering Technical Report TR 96-01-05, University of Washington, 1996.
- Interactive pen-and-ink illustration**, with M. Salisbury, S. Anderson, and R. Barzel. Proceedings of SIGGRAPH 94, in *Computer Graphics* Proceedings, Annual Conference Series, 101-108, July 1994. Also available as Department of Computer Science and Engineering Technical Report TR 94-01-07, University of Washington, 1994.
- Scale-dependent reproduction of pen-and-ink illustrations**, with M. Salisbury, C. Anderson, and D. Lischinski. Proceedings of SIGGRAPH 96, in *Computer Graphics* Proceedings, Annual Conference Series, 461-468, August 1996. Also available as Department of Computer Science and Engineering Technical Report TR 96-01-02, University of Washington, 1996.
- Orientable textures for image-based pen-and-ink illustration**, with M. Salisbury, M. Wong, and J.F. Hughes. Proceedings of SIGGRAPH 97, in *Computer Graphics* Proceedings, Annual Conference Series, 401-406, August 1997. Also available as Department of Computer Science and Engineering Technical Report TR 97-01-01, University of Washington, 1997.
- Computer-generated watercolor**, with C.J. Curtis, S.E. Anderson, J.E. Seims, and K.W. Fleischer. Proceedings of SIGGRAPH 97, in *Computer Graphics* Proceedings, Annual Conference Series, 421-430, August 1997.
- Comic chat**, with D. Kurlander and T. Skelly. Proceedings of SIGGRAPH 96, in *Computer Graphics* Proceedings, Annual Conference Series, 225-236, August 1996.
- Multiperspective panoramas for cel animation**, with D. Wood, A. Finkelstein, J. Hughes, and C. Thayer. Proceedings of SIGGRAPH 97, in *Computer Graphics* Proceedings, Annual Conference Series, 243--250, August 1997.
- Computer-generated floral ornament**, with M. Wong, D. Zongker. Proceedings of SIGGRAPH 98, in *Computer Graphics* Proceedings, Annual Conference Series, 423-434, July 1998.

4 Non-Photorealistic Rendering - The Artist's Perspective

Simon Schofield

Slade School of Fine Art, University College London

Introduction

This presentation is in two parts. In the first part we describe a phase of development in the Piranesi System [13,16] that focused on the automatic production of non-photorealistic renderings (NPR). This is a history of some failure as well as success – totally automated NPR imagery was eventually abandoned in favor of interactive techniques, but the process revealed some interesting limitations about NPR. In particular it highlighted the parts of the image making process the computer is good at and the parts we, the users, are good at. With this experience in mind, in the second part we present some more general thoughts on the problems experienced with painterly and expressive forms of NPR images, and try to offer some practical solutions.

Our thoughts are primarily from the perspective of an image-maker – someone who wants to make beautiful and compelling images – and so throughout holds aesthetic concerns over technical achievement. However we do not underestimate the technical difficulties of system development.

The Piranesi System - a short history

The Piranesi System was one of the outcomes of research into non-photorealistic rendering (NPR) conducted at Middlesex University Centre For Electronic Arts and, more substantially, The Martin Centre's CADLAB at Cambridge University Department of Architecture. Piranesi started its life as an NPR system based on a mark-making engine linked to a viewing pipeline. The initial intention was to create painterly animations; the image rendering was to be wholly automated after lengthy user-specification. Polygons from the original model were projected into screen space, their edges drawn and interior region rendered using marks. Marks were generated procedurally to emulate hand painted marks, besides some parameters to do with texture, they were very similar to those described by Haeberli [7] and Meier [11]. Filling was achieved using a scan line technique plus some stochastics displacement. Ray casting back into the 3D scene could determine any 3D data lost in the projection [15].

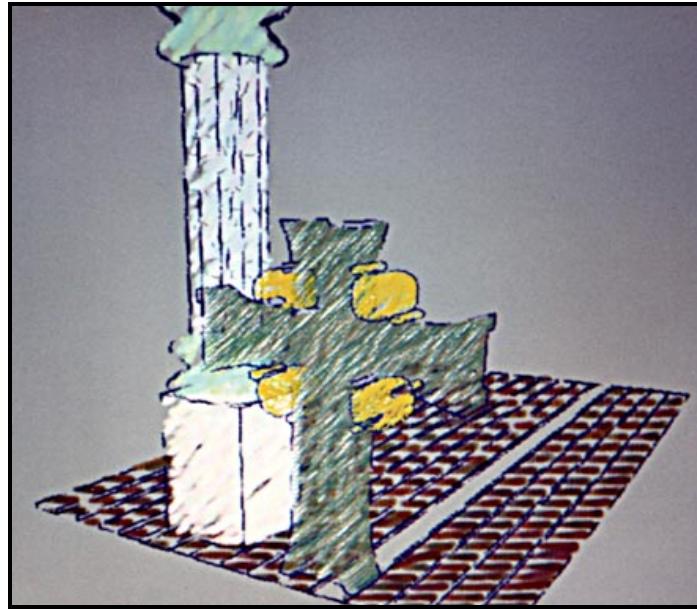


Image 1. An early (1989) NPR using scan-conversion technique to place marks

It soon became apparent that scan-line or ray casting techniques were highly inefficient in the context of this type of rendering. Marks were often placed in random positions, over arbitrary, often large, patches of the images in various directions. Often the same screen-location would be visited many times throughout the painting of an image. A more efficient and flexible solution was to pre-capture scene-data in a number of pre-rendered raster images, along the lines of Saito and Takahashi's "G-Buffer" [14]. These images then became "reference images" for NPR renderings.

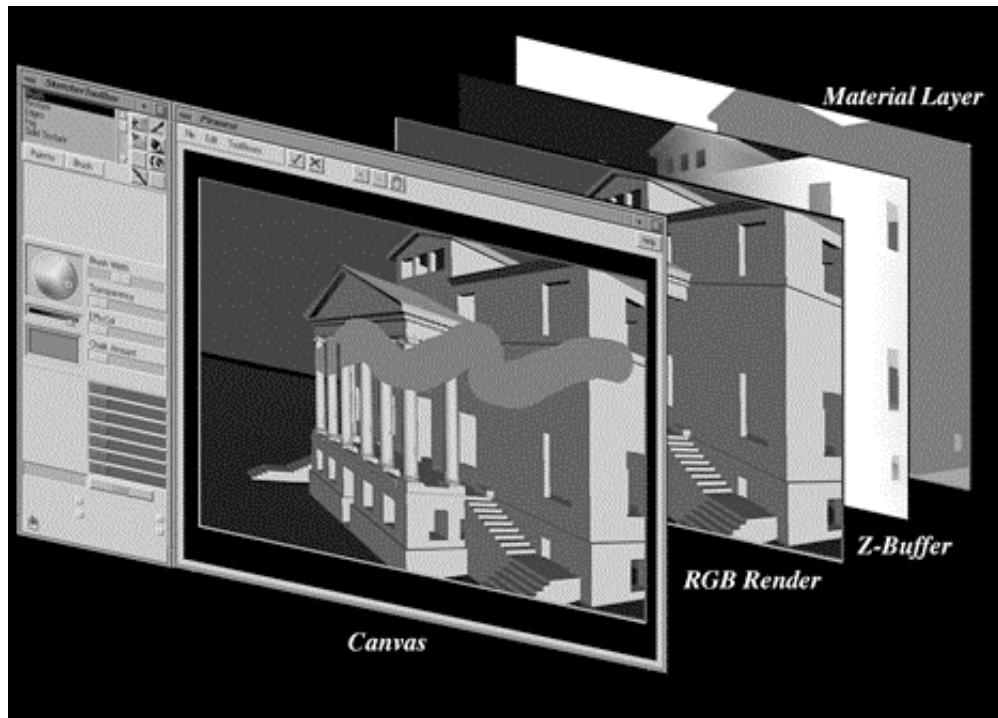


Image 2. A schematic of Piranesi system. An output image and three input reference images provided by a separate rendering system

The advantages of this technique were many: it freed the NPR part of the rendering process from standard viewing-pipeline operations. Also, once rendered the geometry pipeline could be forgotten, so that subsequent NPR operations were independent of model complexity.

Partly to save reduce file-size, and partly because we felt high geometric accuracy was not critical for NPR, we simplified Saito and Takahashi's G-Buffer to be a single Z-buffer. By preserving the projection details we could quickly determine eye and world space surface points on a per-pixel basis, and using small sampling kernels determine surface normal and planarity [16]. The lit scene and material descriptions were also rendered and stored along side the z-buffer in a data structure now known as the "Extended Pixel" or EPix file format. Modelling/viewing systems wishing to benefit from Piranesi's image enhancing capabilities need only to export this file format*.

The speed-gains on geometric calculations were such that geometrically enhanced real-time painting became possible on even a modest computer. We discovered an array of such enhancements to existing raster painting techniques, many of which could be used to augment the photorealism of the image. For instance, fog, solid textures and texture maps could be "brushed" on to the output image. What had begun its life as a batch rendering system gradually evolved into a geometry-enhanced paint system. We will revisit these later, but for the moment we will maintain a focus on NPR techniques.

Early NPR techniques with Piranesi

Our initial intention was to build a system where NPR styles and their specific application to regions of an output image could be user-defined at run-time to any degree of sophistication and saved back to file. The user could thereby define a range of personalised techniques that could be fashioned and tweaked over time, as opposed to using off-the-shelf styles. It was our intention that a style should contain two things. Firstly it should contain specific NPR painting capability such as painterly marks, hatching and edge enhancement. Secondly it should contain a set of arbitrary criteria representing the decisions made by the "artist", before any actual painting takes place. Hence, one could say, the final NPR style-graph embodied some "intelligence" as to *how* to paint the scene.

We loosely borrowed the structure of SGI's Inventor (now VRML) scene graph [18] as a way of dynamically building a description of NPR styles. The nodes in the stem of the "style-graph" were concerned with decision-making and analysing geometry and color. The "leaves", or end-nodes did the actual painting. As new nodes could be quickly fashioned and integrated, a wide range of techniques could be rapidly explored.

Once styles had been defined, an EPix file was imported from a modeller/viewer application. Re-rendering could then begin. A digital-dissolve algorithm selected random screen-points to render. The material of the scene at that point was determined and passed to the style-graph as part of a data structure. This structure (sometimes called an "action") traversed the style-graph, top down, right to left, and could be continuously modified by the nodes it encountered. The action traversed the graph until it found the branch set to render the particular material. Once found, child-nodes made calculations based on scene-geometry or lighting, or used stochastics to modify the final rendering effect. Embodied in the final end nodes of the graph were a range of painting techniques based on procedural marks, transformed bitmaps and vector shapes.

Once the "render" button was hit, the execution of an image was thereafter fully automated but for one detail; it had to be told when to stop. As most of the painting techniques worked on a greater than the pixel size, and were applied using stochastics, and that the end result was to be judged on purely aesthetic prejudices, there was actually no way of pre-determining completion of an image. Our technique was to keep an eye on the monitor and hit stop when we felt the image had cooked enough.

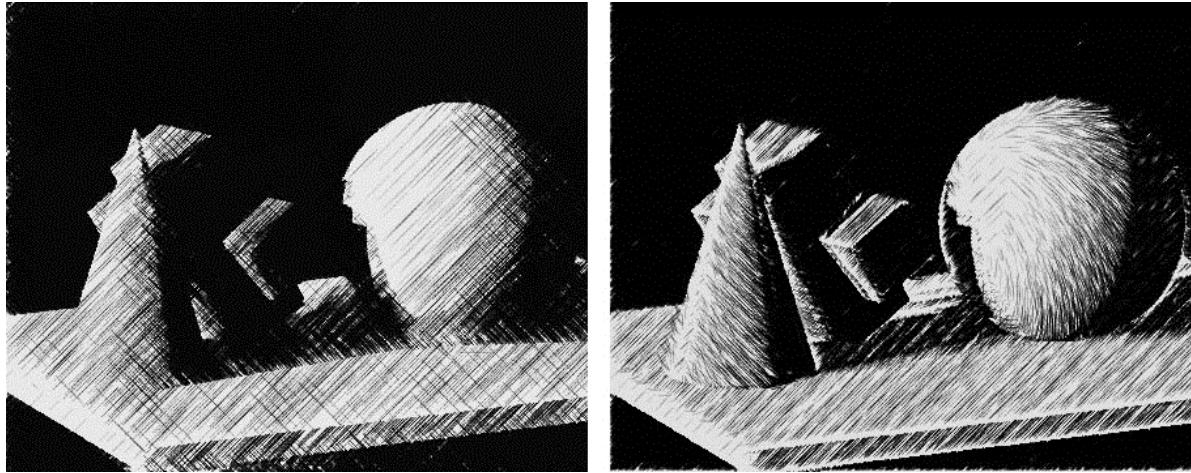


Image 3 and 4 furry balls. Mark density controlled by luminance in the reference image, and mark density and direction controlled by luminance and rate-of-change of luminance

Initial experiments were promising, particularly when using a single or limited number of styles applied in a blanket or "blind" fashion to the image [Images 3,4 and 5]. This seemed to indicate that certain aspects of hand drawing successfully decomposed into repetitive tasks that could be described to a computer. This was a surprise to us and in opposition to an often-held view held that drawing is a wholly virtuoso performance. Gombrich [5] notes that virtuoso artists like Giovanni Battista Piranesi (after whom we named our system), Rembrandt and Gustave Doré did indeed possess huge talent and vision. But it is also true that draftsmen such as these also collected catalogues of standardised methods of representation that were applied quite coldly, without necessarily the effort of the virtuoso, in quite specific and predictable contexts. It is at this level of intervention – at the sub-expressive gesture – that NPR techniques seem to best succeed.

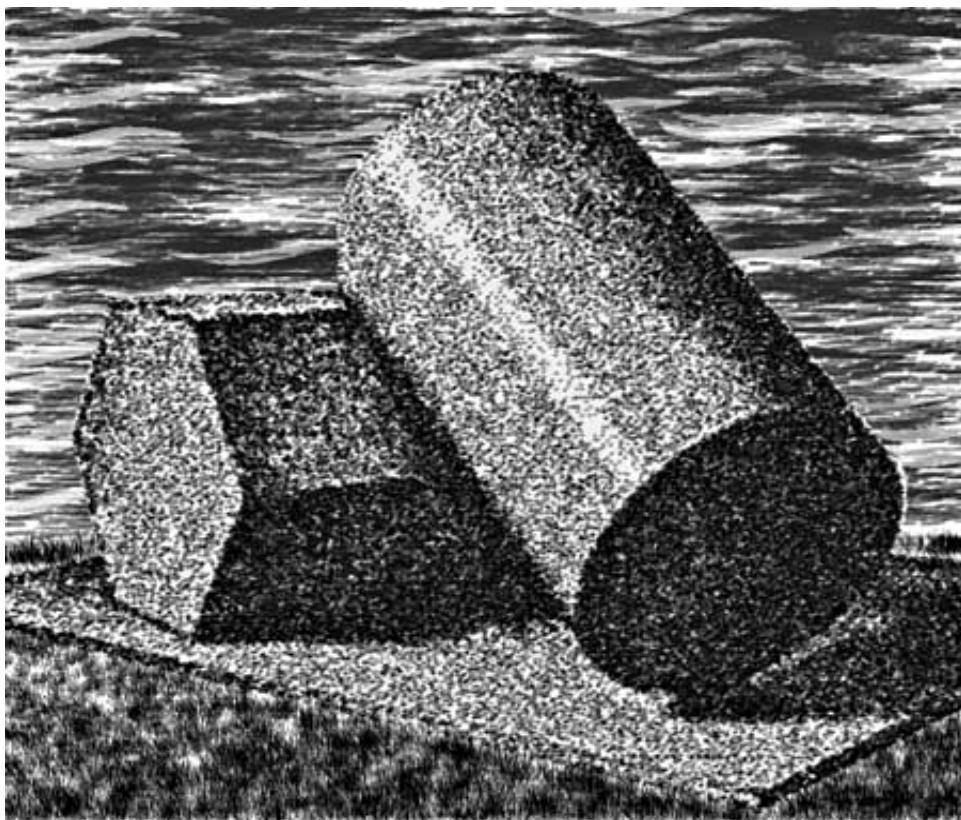


Image 5. Three techniques - working in reasonable harmony.

As the desired sophistication of our images increased, so too did the number of techniques applied to the output image. We began to use several techniques within a single render. Again to our surprise, rather than gaining a look of greater sophistication, the images began to betray a certain banality in execution, a general lack of expression and flatness of technique [Images 6, 7 and 8]. Whereas images using one or two techniques often looked like competent studies, images with more than four or five techniques began to look distinctly incompetent. Areas of differing technique were not suitably blended, resulting in a fragmented, disjointed look. Multiple styles also compounded the problem of surface coverage. Certain areas received too much painting, while others too little in relation to adjacent areas. It was not that the individual areas of differing styles were any less competent than before – the problem lay in their unharmonious distribution over the image. In other words, the problem lay, not so much in the painting of individual parts, but in the assemblage of the overall composition – how the parts worked together to create an overall effect.

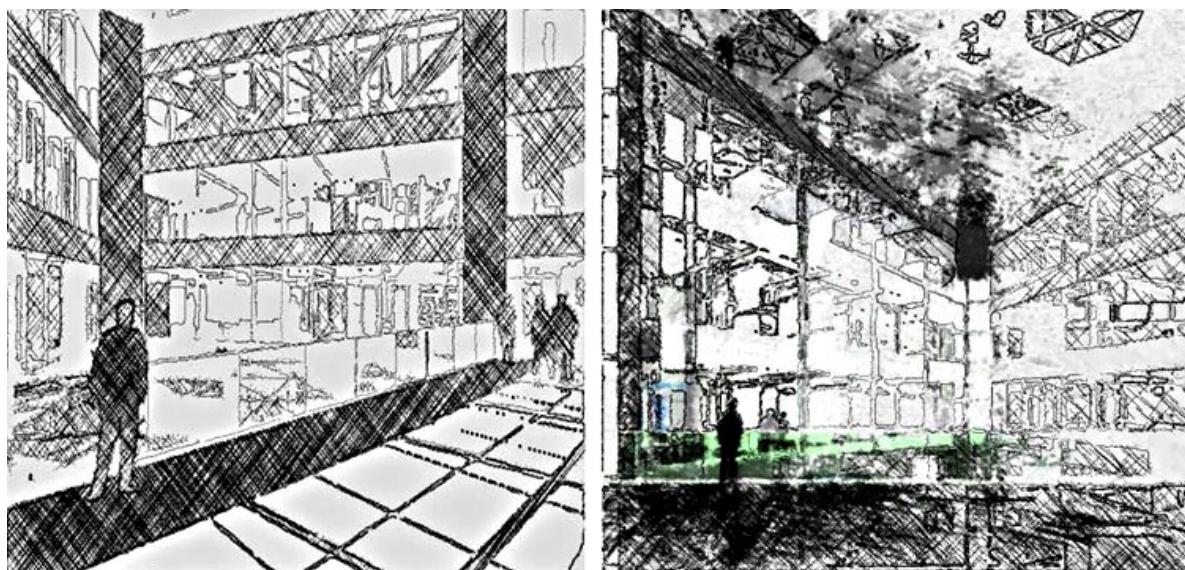


Image 6 and 7. Automated drawings of architectural spaces using three or four techniques. While parts of the images work well they begin to show a certain overall clumsiness.

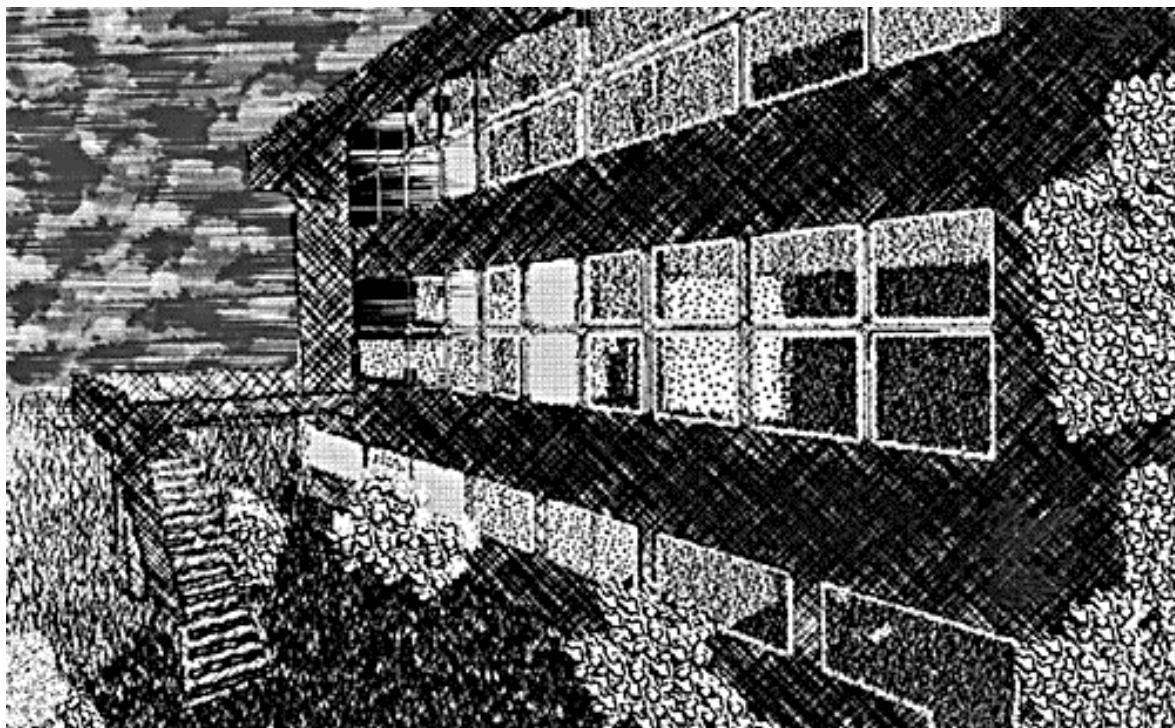


Image 8 A rendering of Corbusier's Villa Stein using seven techniques. Techniques battle with each other to create an unharmonious image.

We modified our approach by allowing the user to guide the focus of the painting by mouse-clicking into various material-regions of the image throughout the render process. Upon specifying a region, the digital dissolve effect was limited to that region's extents. Hence certain areas could be given more rendering effort than others. After a certain period, if no more mouse clicks were detected, painting automatically moved on other parts of the image. Using this approach we were able to harness multiple techniques to create a pleasing final image [Image 9]. However, by now we were conceding to the fact that good,

complex NPR imagery required continual user-input. We gradually abandoned the idea of automated NPR images (i.e. for animation) and began concentrating increasingly on the real time, single image construction via wholly interactive techniques.

As automatic, batch effects yielded to interactive techniques, the final form of the prototype Piranesi System emerged as more akin to an interactive paint system, than a rendering system. At the time of writing, some automatic painterly techniques still exist within the commercial system, and their popularity and usage will be determined by the user-community.



Image 9 "The Ideal Villas", an image in which the responsibility for rendering is shared equally between the computer and the user by guiding the focus of rendering effects.

Interactive paintings of non-photoreal images

It is beyond the scope of this paper to present all the painting techniques facilitated by our approach as many of them have little to do with NPR. For fuller coverage see Richens and Schofield 95 [13] and Schofield 96 [15]. However some of these techniques impact directly on the generation of NPR imagery, such as brushes which snap to set planes or normals, paint which fades or thickens in a certain world-space direction (fog, reverse fog, ground fog), edge enhancement and shape compositing. We show a selection of NPR images rendered using the fully interactive version of Piranesi.

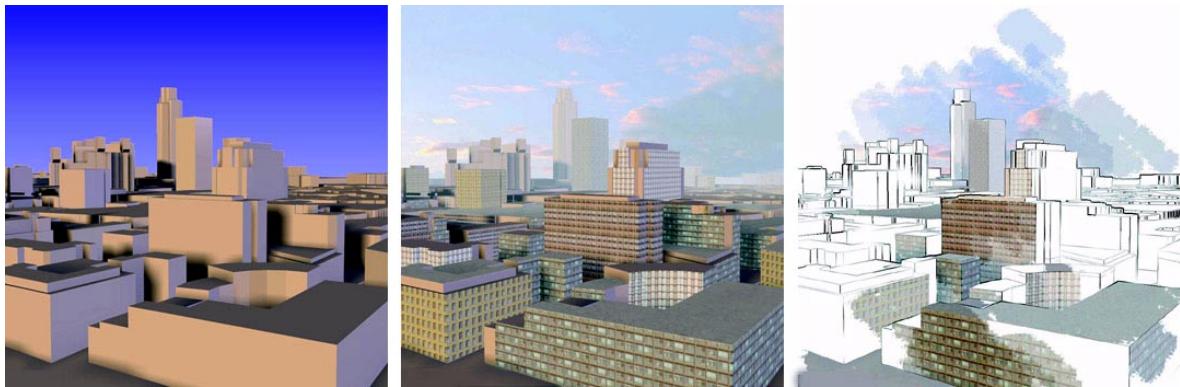


Image 10, 11, 12. Three stages in the painterly re-rendering of a model of London. i) The original image from the viewing/modelling system. ii) Urban textures brushed onto the model. iii) Re-rendered using a soft painterly brush that clips against geometric planes found in the scene.

Model courtesy Miller-Hare Ltd, London.

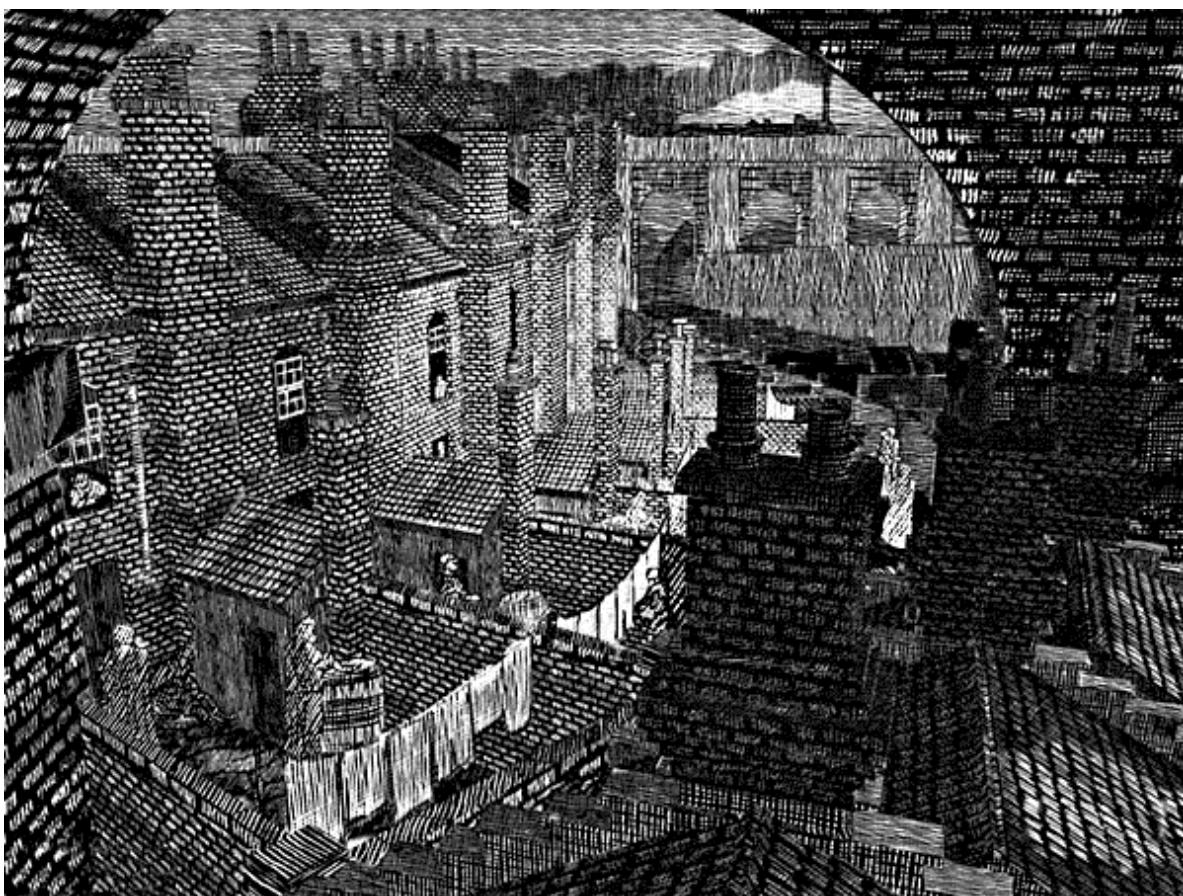


Image 13. An interactive rendering, based on one of Gustave Doré's drawings of London, in which a number of hand-drawn textures were interactively applied to the scene. Image courtesy of Paul Richens, Cambridge University Dept of Architecture.



Image 14. Lobby image. A range of techniques used to create a highly evocative image. Image courtesy of Paul Richens, Cambridge University Dept of Architecture.

Some thoughts and prescriptions on the problems of Painterly Rendering

NPR is growing and diversifying; styles now range from cartoon shading [4, 2], technical illustration techniques [4, 6, 14], through to highly expressive "hand crafted" forms of rendering [3, 7, 8, 10, 11, 17, 19, 20]. While the generation of these techniques is largely a technical matter, their acceptance as useful forms of image making by the larger community is wholly culturally determined. 'Toon shading and technical illustration both enjoy obvious use-value and aesthetic competence; they are both direct cousins of photorealism with the same user-control and determinism. As a consequence their usage has already been established within the industry. Painterly rendering on the other hand, present us with more cultural, aesthetic and ergonomic problems. We discuss the salient issues here adopting a knowingly polemic tone. We are not saying our opinion is categorically right because our own experimental results are too limited for any really definite conclusions. We are perhaps inviting others to prove us wrong.

Authenticity

If we credit the end user with being a serious image-maker, possessing a sophisticated appreciation of real painting, we can easily see why many current painterly NPR systems may strike a bad chord with them. Cries of "They're not real paintings... you can't really imitate a painter's skill...it's so phony!" were often heard during our days of painterly experiments, and were surprisingly difficult to counter [15].

To their credit, NPR developers tend not to overstate the claims of their work; no one has actually expressed the ability to simulate a painting perfectly, although it is sometimes stated as the goal. SIGGRAPH technical papers, the forum for most NPR publications to-date, tend to avoid any lengthy engagement with aesthetics *per se*. Aesthetics and the philosophy of representation are felt not to be an appropriate subject-focus for the technical paper's section, so tend to get underplayed, even if they are of real interest to the developer. Each development has been couched within economical and cautious

language about its relationship to real painting. Consequently there has been no real forum for the deeper discussion of the aesthetics of painterly NPR and its longer-term goals.

As a modest alternative to deeper discussion, painterly NPR techniques tend to be presented as striving towards a hand painted "look". Hertzman et al's opening comments are typically cautious, "*We present a new method for creating an image with a hand-painted appearance.*" or Peter Litwinowicz "*We describe a technique that transforms ordinary video into animations that have a hand painted look*". That is to say their images aim at being like paintings, while not actually being paintings. It is sometimes wise to admit one's own fraudulence rather than be accused of it by anyone else. This is by contrast to the position now enjoyed by photoreal CG, which is no longer considered as simply "imitating a photograph". As Negroponte observes, each new medium carries with it the unnecessary burden of its predecessor. We no longer criticise photoreal CG as being poor imitations of photographs (think of *Jurassic Park* or *Toy Story*). Photorealistic CG has left that legacy behind, and become simply another way of making images. The same cannot be said for much NPR imagery, which suffers from a terrible clutter of legacies. We seem presently unable to determine which of NPR's heirlooms we should keep and which ones we should throw out.

It is definitely interesting that the surface appearance of impressionism is so easily imitated and worth several SIGGRAPH papers! CG has long been comfortable with then notion of effect sufficing over deeper simulation. But as the field of NPR gathers momentum and develops a history, so too it gains seriousness. The *authenticity* of an image is a measure of its seriousness – that it is no longer a pale imitation of the genuine article, but something valid in itself. Authenticity is not a philosophical nicety enjoyed only by art theorists, it is an important determining factor between a successful and an unsuccessful image, and a mark of a medium's maturity. It is the determining factor between surface effect and deeper expression. We suspect that NPR developers wish not for their work to be thought of as a surface effects, but as the beginnings of a new medium with the ability to articulate serious visual ideas.

The delegation of responsibility for painting

Even if a potential user is not concerned with issues of authenticity, they will certainly be concerned with any new technique's capacity for articulating the user's desired expression, its flexibility and range.

As we indicated in the first section, a continual barrier to flexible expression encountered in painterly NPR systems lies in the propensity for such systems to make too many decisions on behalf of the user. Specifically, our concern begins when a computer is asked to take sole responsibility for large areas of the painting and so user-expression within an image. In the process of handing work over to the computer certain recurrent techniques are used. Stochastics [7, 11], cellular automata [1], and statistical analysis [7, 8] are commonly used within NPR imagery to imitate two things. (For the purposes of this paper we shall refer to these techniques simply as statistical techniques.) Firstly they are used in the imitation of the complexity of physical media – the way the pen might wobble, the pen responds to a paper's grain, the complexities of brush on canvas. Secondly such techniques are used to imitate the complex and often arbitrary decisions of the artist at work.

Statistical techniques used in the simulation of media - Good

We have no problem when statistical techniques are used to simulate physical properties of a medium in the raw. An artist enjoys the complexity of a chosen medium and often desires a certain loss of control. This can be seen quite plainly in the use of aquatint in etching, where the chemical bite of acid on copper plate creates uncontrollable patterns of grain. It is also witnessed in the use of scumbling in oil painting, wet washes in water-color, glazes in oil painting or charcoal on heavily textured paper. As the real-world artist neither quibbles with, nor fully controls, the outcome of these techniques, statistical methods may well be used to imitate them. The work of Strassmann [17], Curtis [3] and Cockshott [1] is heavily predicated by a desire to introduce this type of complex phenomena into the realm of CG in order to enrich the surface texture of the image.

If we simulate such phenomena on a computer are they still authentic? Consider a wobbly, grainy line drawn by pencil and a wobbly, grainy line drawn by a computer. The wobbliness and graininess of the computer line is just as real as that of the pencil line. Such phenomena transcend the specifics of a particular medium, so it seems perfectly reasonable for a computer to generate such outcomes.

Statistical techniques used to simulate expression - Bad

For us problems arise when statistical techniques are used to simulate a medium's usage by human hand and the decisions of the artist. Many extent systems exhibit such simulation. As described, we too were guilty of attempting to simulate the artist's decisions.

Aside from AI die-hards, its has become an established principal that computers are not inventive or expressive in themselves but are tools through which expression is articulated by users [12]. Hence a word processor handles individual characters but does not construct sentences, and a MIDI system processes musical note, but does not construct tunes. It is interesting that NPR developers often seem to overlook this principal and delegate the task of constructing a completed painting to the computer. The consequences of doing so produce results that are aesthetically similar to allowing computers to generate sentences or tunes - they tend to be either chaotic and unintelligible or flat and predictable.

Statistical techniques are highly inappropriate for imitating the decisions of the artist. The most-often seen form of NPR is the pseudo-impressionist painting - a technique which often relies on randomness to determine the placement, shape and color of marks across the picture plane. It is a constant surprise to see how effective this technique is. However, this technique's similarity to real impressionist painting is extremely trivial – it implicitly reduces the efforts of the most important artistic movement of the last century to experiments in controlled noise. The marks in any impressionist painting are highly specific, even more so than in earlier smoother types of painting, in that the very nature of there visibility made them of heightened concern to the artist.

Neither paintings, nor indeed clusters of marks, decompose into a set of algorithms. Paintings are not solutions to well posed problems; they are determined by whim, a desire to explore and articulate uncertainty. It is probably safe to say, that while we can understand and simulate a medium's physical phenomena, we do not understand and cannot simulate the way people will use that medium.

Winkenbach and Salesin's Pen and Ink work [19, 20] seem not to suffer as heavily from this problem as that of the "impressionist" systems. Their pen an ink renders are wholly algorithmic, yet posses a convincing hand drawn look without suffering from a seeming lack of authenticity. As discussed, we found that "blanket" or "blind" techniques, when applied overall to an image, often produced pleasing results, while techniques which simulated artists decisions, tended to fail. Etching, hatching and engraving methods can be, on a certain level, considered to be largely technical in their execution. Over smaller sections of an image, there may be very little consideration or decision given once a general approach to modelling form has been found. This does not mean that etchings or engravings are any less expressive than looser forms of painting, it simply means that the expressive decisions and virtuosity in these types of images lies elsewhere. In these cases, expression tends to lie in the overall composition, tonality and economy. Gustave Doré's genius lay not in his ability to engrave, which he often relied on others to do, but in his overall drawing, modelling and tone. It is still down to the user to provide these aspects when using Winkenbach and Salesin's systems.

Suggestions

We believe that systems attempting to imitate the creative decisions of the artist will necessarily produce images lacking in visual merit and authenticity. We suggest that the solutions to the problem of authenticity and the simulation of expression lies in passing over to the user those tasks which cannot be solved by the computer, while leaving the computer to attend to fine detail and repetitive tasks.

To take one example, Meier's [11] describes a painterly animation system that uses an algorithmic approach to initial mark-placement. Once locations have been determined in a key-frame, a world-space particle system is used to control the positioning of marks in subsequent frames of the animation, thereby avoiding temporal aliasing artifacts. We believe, (and we are sure this has occurred to Meier too) that such a system would benefit from an interactive front end that would allow the user to take full responsibility for mark placement. Meier's described system requires considerable user input to establish the parameters for a certain type of painting. We assume this involves a lengthy feed back cycle of trial-and-error before a satisfactory result is achieved. An interactive front end would avoid this lengthy process, and provide a more direct and subtle way of determining the final image. Meier's imaginative use of the particle system to facilitate frame-to-frame coherence could still be used to animate the scene, thereby providing a productive synergy between the expressivity and economy of the human user and the processing power of the computer.

Within Winkenbach and Salesin's systems, while individual marks are not such a problem, the added ability to interactively pull and push local tone, and the ability to locally modify the direction and internal texture of the hatch, would be an added bonus.

Once the user has regained responsibility for artistic decisions in such images –either in terms of individual mark placement, or in terms of manipulating local tone and texture – the image is once again humanised and can be considered to be truly expressive.

Conclusions

Through our experiments with the prototype Piranesi system we found that automated blanket or "blind" NPR techniques could be made to produce pleasing images. But as the techniques became less homogeneous we found manual intervention was increasingly necessary. It was difficult, if not impossible, to imitate the decisions of a real artist. This suggests that NPR systems that wish to go beyond "blanket" techniques may have to allow for continual user-input which will be in proportion to the desired sophistication of the output image.

In general we believe many painterly NPR images suffer from a lack of authenticity. While we do not object to the imitation of the physical media as such, we have reservations over systems that take larger, expressive decisions on behalf of the user. We believe that systems attempting to imitate the creative decisions of the artist will produce images lacking in visual merit and authenticity. We suggest that all these problems may be overcome by introducing more interactive front-ends to NPR systems. This will allow users to create truly expressive image, without the hindrance of presumptuous interference from the system itself.

Notes

The Piranesi System is now available for Windows from Informatix Software International, Cambridge, UK. More information is available at <http://www.informatix.co.uk>. A demonstration copy of Piranesi is available on the SIGGRAPH '99 Course Notes CD-ROM in the *piranesi* sub-folder.

*The Epix file format documentation is available from the above URL.

The author can be contacted at simon@aftpiranesi.u-net.com

References

- [1] Cockshott, T. (1992). Modelling the Texture of Paint. Eurographics 92 Proceedings CGF Vol. 11, Blackwell (pp C217-C226).
- [2] Corría, T. et al (1998) Texture Mapping for Cell Animation. In SIGGRAPH 98 Conference Proceedings.

- [3] Curtis, C. et al (1997). Computer Generated Watercolor. In SIGGRAPH 97 Conference Proceedings.
- [4] Dooley, D and Cohen, M. (1990) Automatic Illustration of 3D Geometric Models: Surface. IEEE Computer Graphics and Applications 13(9):307-14
- [5] Gombrich, E. (1960a). *Formula and Experience*. In *Art and Illusion*. Phaidon, London
- [6] Gooch, A. et al (1998) An NPR Lighting model for automatic Technical Illustration, in SIGGRAPH 98 proceedings
- [7] Haeberli, P. (1990). Painting by numbers: abstract image representation, Computer Graphics, vol. 24, no. 4, ACM SIGGRAPH, pp. 207-214.
- [8] Hertzman, A (1998) Painterly Rendering with Curved Brush Strokes of Multiple Sizes. In SIGGRAPH 98 Conference Proceedings
- [9] Lansdown, J and Schofield, S. (1995). Expressive rendering: an assessment and review of non-photorealistic techniques, Computer Graphics and Applications (May), IEEE.
- [10] Litwinowicz, P. (1997) Processing Images and Video for an Impressionistic Effect. In SIGGRAPH 97 Conference Proceedings
- [11] Meier, B. (1996) Painterly Rendering for Animation. in SIGGRAPH 96 Conference Proceedings
- [12] Richens, P. (1994). Does knowledge really help? In (eds.) G. Carrara and Y.E. Kalay, Knowledge-Based Computer-Aided Design, pp. 305-325. Elsevier, New York.
- [13] Richens, P & Schofield, S. (1995) Interactive Computer Rendering, Architectural Research Quarterly, Issue 1 Vol. 1, Emap, UK
- [14] Saito, T. and Takahashi, T. (1990). Comprehensible rendering of 3-D shapes, Computer Graphics, vol. 24, no. 4, ACM SIGGRAPH, pp. 197-206.
- [15] Schofield, S (1994). Non-photorealistic Rendering. PhD Thesis. Middlesex University. London.
- [16] Schofield, S. (1996) Piranesi: A 3-D Paint System, Eurographics UK Proceedings 1996 Vol. 2, Imperial College London
- [17] Strassmann, S. (1986). Hairy Brushes. Computer Graphics Vol. 20, ACM Press (pp 225-231).
- [18] Strauss, P., & Carey, R. (1992). An Object-Oriented 3-D Graphics Tool kit. In SIGGRAPH 92 Conference Proceedings
- [19] Winkenbach, G. and Salesin, D.H. (1994). Computer generated pen-and-ink illustration. In SIGGRAPH 94 Conference Proceedings
- [20] Winkenbach, G. and Salesin, D.H. (1996). Rendering Parametric Surface in Pen and Ink. In SIGGRAPH 96 Conference Proceedings

Painterly Image Processing

Aaron Hertzmann
Media Research Laboratory
Department of Computer Science
New York University

Processing Images

- Make a painting from a photograph



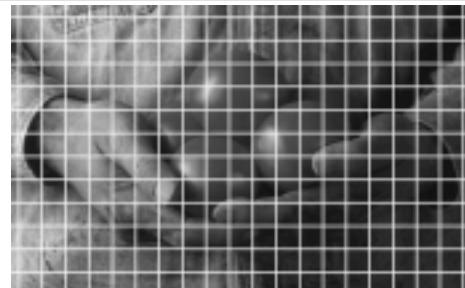
Brief History of Computer Painting

- Paint programs
 - 1970's: *Shoup, Smith, etc.*
 - 1980's - Present: *SuperPaint, MacPaint, Adobe Photoshop, Fractal Design Painter, Piranesi*

Brief History of Computer Painting

- Haebel's *Impressionist* (SIGGRAPH 90)
 - *Automatic color*
- “Artistic” Filters
 - *Automatic color and stroke placement*
 - *Photoshop; Litwinowicz, SIGGRAPH 97*

“Grid” Algorithms





Layering

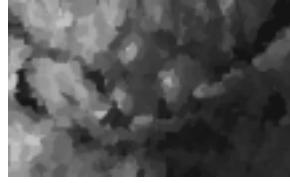
Coarse-to-fine painting

- Sketch with a large brush
- Refine with a small brush

Layering
<p>Brush Radius = 8 pixels</p>  <p>Reference Image</p>

Layering

Brush Radius = 8 pixels

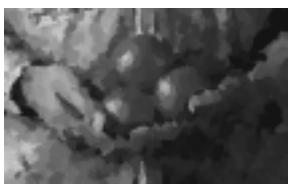


First Layer

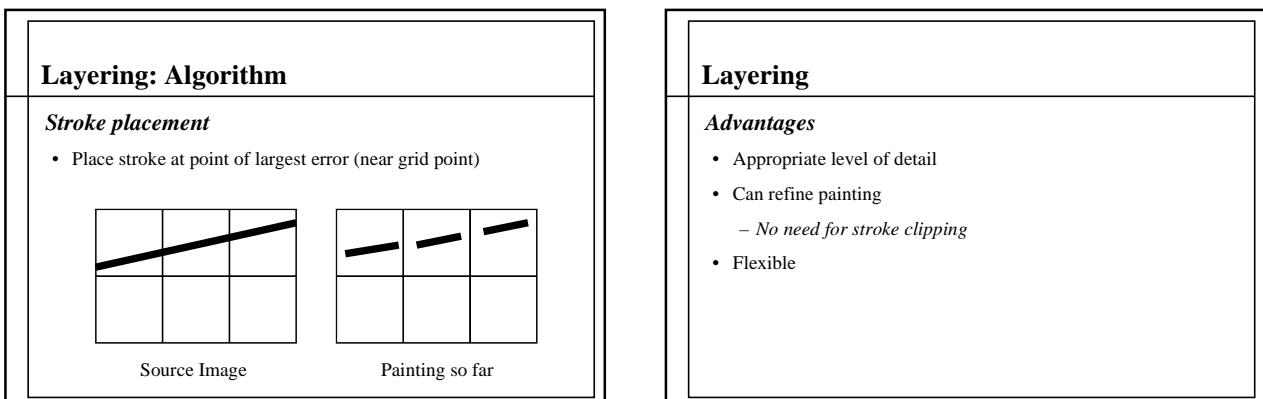
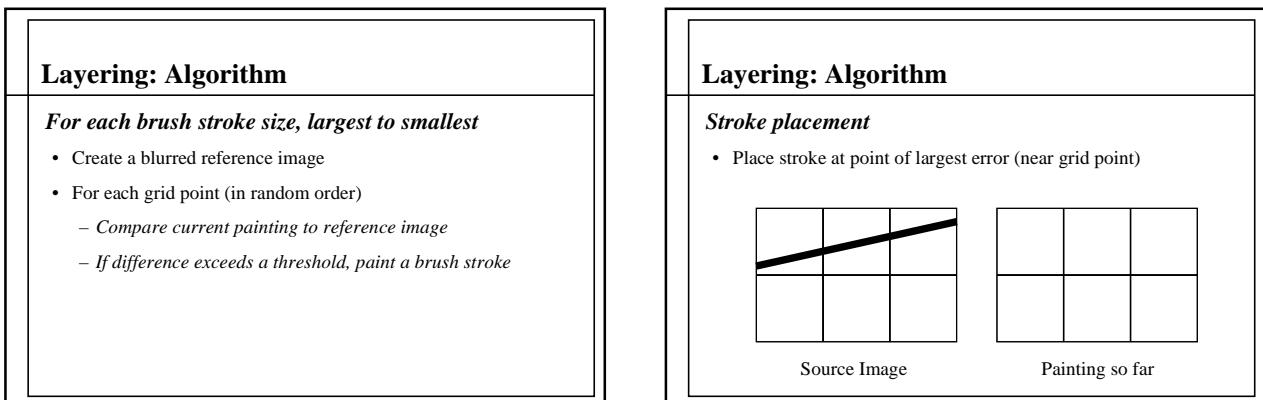
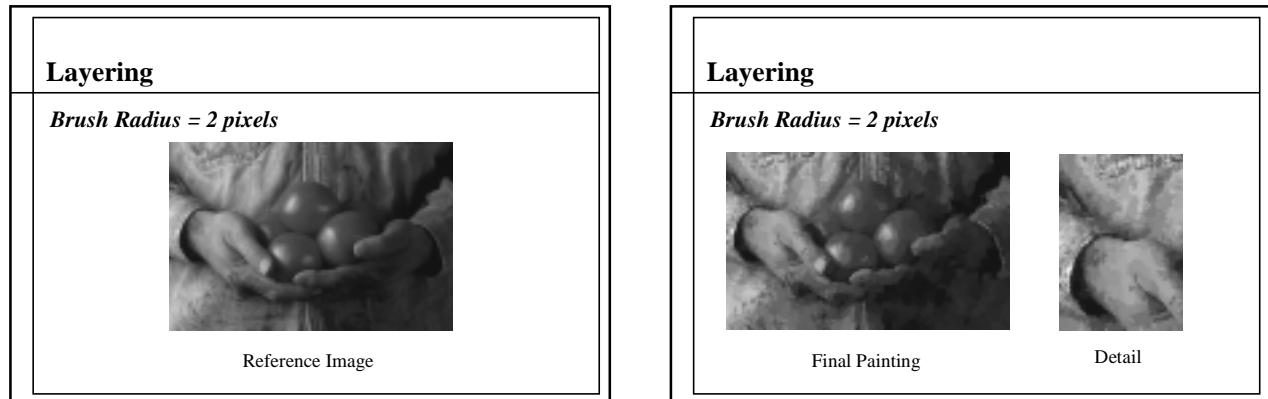
Layering
<p>Brush Radius = 4 pixels</p>  <p>Reference Image</p>

Layering

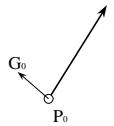
Brush Radius = 4 pixels

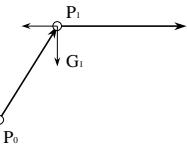


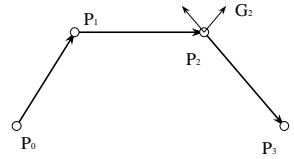
Second Layer

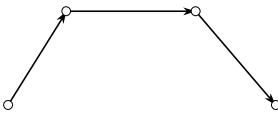


<h3>Brush Strokes</h3> <ul style="list-style-type: none"> • Short, regular brush strokes • Want long, brush strokes 	<h3>Brush Strokes</h3> <ul style="list-style-type: none"> • Constant color • Long, curved path – <i>cubic B-spline</i> 
---	--

<h3>Brush Strokes</h3> <p>Algorithm</p> <ul style="list-style-type: none"> • Place a control point • Move in the direction normal to the gradient, a distance R • Repeat 
--

<h3>Brush Strokes</h3> <p>Algorithm</p> <ul style="list-style-type: none"> • Place a control point • Move in the direction normal to the gradient, a distance R • Repeat 
--

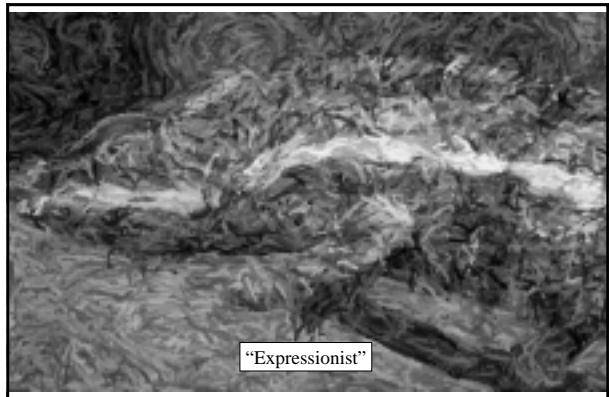
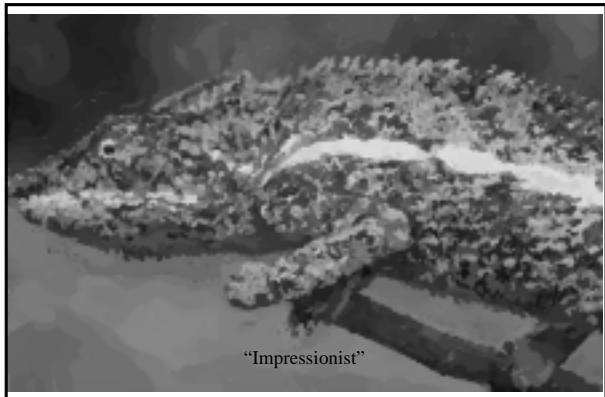
<h3>Brush Strokes</h3> <p>Algorithm</p> <ul style="list-style-type: none"> • Place a control point • Move in the direction normal to the gradient, a distance R • Repeat 
--

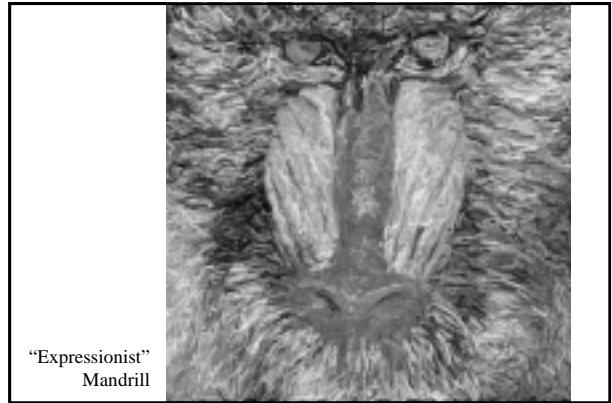
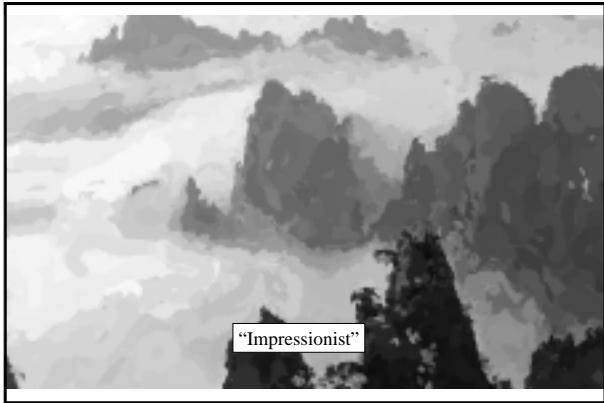
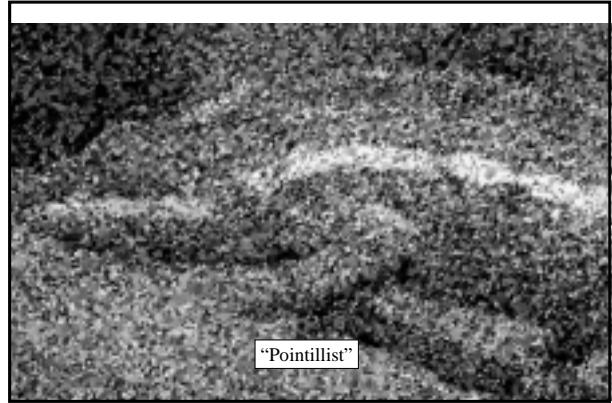
<h3>Brush Strokes</h3> <p>Termination</p> <ul style="list-style-type: none"> • Maximum number of control points reached <i>OR</i> • $\text{strokeColor} - \text{referenceImage}(x,y) > \text{canvas}(x,y) - \text{referenceImage}(x,y)$ 
--



Styles

- Many painting parameters
 - *Threshold*
 - *Stroke curvature, stroke length*
 - *Jitter color*
 - ...
- Encapsulate parameters as a “style”





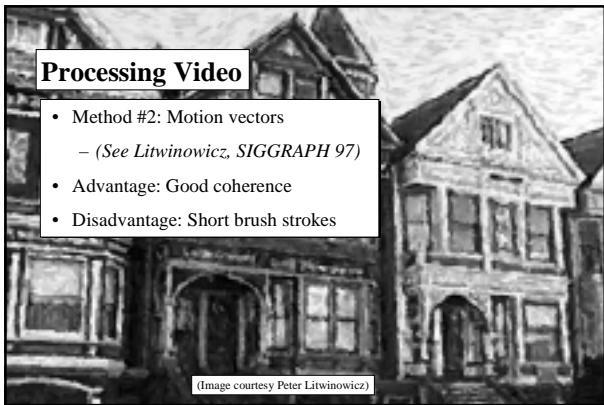
Processing Video

- Problem: Temporal Coherence (Flickering)

Processing Video

- Method #1: Paint over last frame
- Advantage: Very easy
- Disadvantage: Lingering artifacts

A black and white video frame showing a person sitting on a bench outdoors. A dog is standing next to them. The image is somewhat blurry and shows signs of temporal coherence issues.



6 Image-Based Rendering and Non-Photorealistic Rendering

Pete Litwinowicz
RE:Vision Effects Inc.

“What Dreams May Come” – Motion Paintsm

One of the challenges encountered was to create a **painterly** style for an 8 1/2 minute sequence of moving imagery. This meant that what was "right" or what would work was much more subjective. Furthermore we needed to use and reconstruct the director and cinematographer's live-action compositions. For the first few months in development, we thought it was important to work out technical issues, such as:

- how do we track pixels?
- what were the brush strokes to look like?
- should we shade each brush as a 2D texture in 3D space?

As we started development (which started before post-production work started) we realized these questions couldn't be answered without a defined visual style. The algorithms and methods must be developed in tandem with the development of the look desired.

As our techniques developed we talked in painterly terms, which is not too surprising. We would first discuss perception issues, using phrases like “the painting needs a looser feel,” or “more like Monet,” or “more contrast,” etc. After these discussions we would then determine the various attributes of the brushes:

- color
- texture
- shape
- size
- density

The following challenges emerged:

- Each image had to look like a painting
- The moving sequences shouldn't feel flat or claustrophobic. After all, the audience was going to have to sit through 8 1/2 minutes of this world.

We developed a number of techniques and guidelines to meet our goals and challenges:

- 1) Brush strokes should have a size corresponding to the texture in the live action. (we don't want the actor standing next to what appears to be a 6 ft. brush stroke).

- 2) Objects were to be painted, but not made out of brush strokes in 3D space. That is, we didn't literally compose objects out of 3D brush strokes. Again, paint strokes were not floating 3D textures. We felt this seemed a little surreal, plus we didn't have the 3D data. The question remains, how did we handle occlusions and reveals? We solved this problem by giving each brush stroke a lifetime where it grew, stayed around a while, then shrank. In this way, brush strokes disappeared as objects went out of view. In static scenes, this meant that we had some animated life in the painting even while the scene itself remains still. Brush stroke aging not only solved technical problems, but also donated desired aesthetics.
- 3) Painting should happen in layers: e.g., shadows, mid-tones, and highlights. A group of strokes represented a highlight pass, etc. Each individual stroke was not painted with shadows, mid-tones and highlights (each stroke was not individually lit)!
- 4) Objects in the scene were to be painted with brush strokes. However, to give a sense of depth, atmospherics and sunbeams were rendered using traditional, more photorealistic techniques.

Influences

In WDMC we combined many different 19th century painters' styles. The major influences were Claude Monet, Vincent Van Gogh, and Casper David Friedrich and John Mallord William Turner.

Maintaining the director and cinematographer's compositions

We needed to recreate what the camera had caught on film. Most shots had no green screen (70 mile an hour winds) and none of the shots had motion control cameras. We used optical flow techniques to track pixels in the scenes. We had various interactive and batch fix-it tools to help clean up the motion.

We originally thought this wind motion would cause us many headaches. In fact, the wind was a boon because it added life to the scenes.

Maintaining depth and space for a non-claustrophobic experience

By maintaining brush size at roughly the texture scale it was supposed to represent, we avoided bringing the background flat against the foreground. Also by growing and shrinking strokes as objects appeared or disappeared, we were able to maintain some 3D nature of the underlying images. We also controlled the size of brush strokes with z- buffers... again; we didn't literally map strokes in a 3D sense, but had a z-transfer function. For scenes without reconstructed 3D objects, artists could paint artificial z-buffers. For a plane of water, a simple hand-painted gray scale ramp often sufficed for the input z buffer. We also introduced atmospheric elements in a more photorealistic way so as to maintain a greater sense of depth.

The Production Workflow

1. The art director would come up with a sample look for a shot.
2. Motion analysis would be run on the shots. Shots were "cleaned" up by a compositor before analysis; for instance, the actor would be removed and something resembling the rest of the background would be substituted in its place.

The rest of the workflow happened in an iterative and omnidirectional process. What follows are some sample scenarios:

1. The compositor would color "correct" the plates to be in line with the reference image.
2. The compositor and/or roto artists would key, or hand paint, mattes for objects, or portions of objects.
3. Mattes were made for regions that would have different brush stroke textures or sizes, or for shadow, mid-tone and highlight passes.
4. These mattes were used by the "motion paint" artists to generate paint layers... experimenting with brush stroke size, texture, color perturbations, density, and lifetime.
5. 3D equalizer was used on some scenes to recreate 3D camera info, based on tracking balls or other still objects in the scene.
6. 3D elements were produced... their mattes and motion data were often given to the motion paint artist as input.
7. Layers of sunbeams and mist were given directly to the compositor.
8. 2D elements were animated... their mattes and motion data were often given to the motion paint artists as input.
9. If a motion paint artists couldn't get a look because the input color or mattes were off, the compositor would be asked to regenerate mattes and image sequences.
10. Tracked motion would be fixed with interactive tools, both 2D and 3D in nature
11. The compositor took all the painted layers, graded them one more time, and created finished sequences.

Teams were built consisting of each type of artist because we had such a tight interactive loop between compositing, roto work, 2D and 3D animation and the motion paint artists. This team approach varies significantly from a more traditional, departmentalized approach where one team creates mattes, another does compositing and yet another does 2D and/or 3D animation.

Pierre Jasmin and I acted as software/technique problem solvers that helped each team as they had problems. We acted as cross-team support. Each shot had a unique set of problems and called for something just a bit different from all the other shots.

Combining live-action, 2D and 3D animation into a consistent style

We rendered everything in a consistent style by passing all elements through our paint system. There are some elements that we can't remember were live-action or 3D!

Tools we used and provided

One of the reasons computer vision technology has not reached a broader use lies within its roots. Computer vision sprang out of research that needs to work without user guidance. Instead of focusing on getting vision algorithms to work 100% of the time, we prefer to work on fix-it tools that allow the human to guide or repair the process when the automatic version doesn't work.

Tools we used and developed:

1. Traditional 2D roto/compositing software
2. Traditional 3D software
3. In-house particle system, using motion data from analysis of image sequences as well as motion data from 2D/3D systems. The particles were rendered as strokes with controls for orientation, size, texture, placement (via mattes), and density
4. In-house interactive 2D animation system. The system allowed artists to interactively place multipoint strokes in mid to foreground areas.
5. In-house tools for controlling orientation were provided (using 2D geometry or 3D geometry or image processing)
6. In-house tools for fixing or authoring motion were provided

Summary

Even though IBR techniques are far from perfect, we were successful at using them for transforming live-action into painterly sequences.

1. Painterly processing is very forgiving.
2. We painted landscapes. People or other articulated characters might have been much more unforgiving.
3. We provided interactive tools to fix mistakes generated by the vision techniques, instead of trying to make the vision techniques perfect.
4. Great artists used the tools to create extraordinary images.

Notes

The author may be contacted at pete@revisionfx.com, or visit the web site <http://www.revisionfx.com>.



1. Before



2. Motion Tracking Visualization



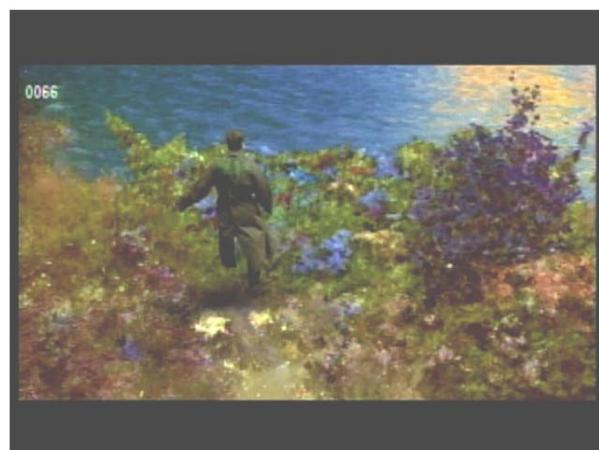
3. 3D Ground Model Test



4. Single Paint Layer



5. Multiple Paint Layers



6. Finished Frame

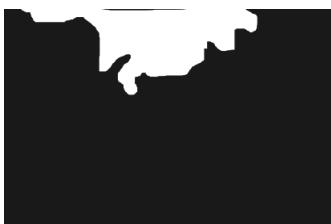
All Images © PolyGram Filmed Entertainment Distribution, Inc.



Original Picture



Final Painting



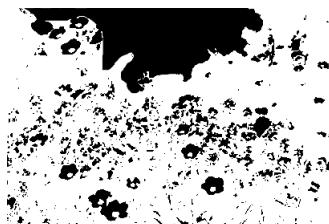
Background matte



Background layer



Green color source



Green layer matte



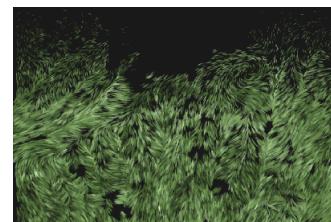
Orientation guides



Hand-drawn z-buffer



Green layer 1



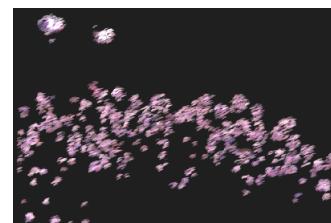
Green layer 2



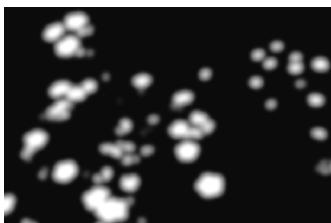
Pink flower layer matte



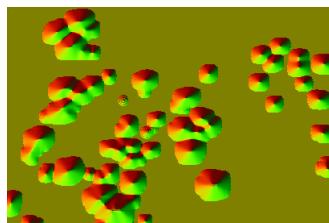
Pink layer color source



Pink layer



Red flower matte, blurred



Gradient of blurred matte,
for orientation



Red flower layer

NPR and IBR



RE:Vision Effects, Inc.
Peter Litwinowicz
pete@revisionfx.com
www.revisionfx.com
Motion Paintsm
and "What Dreams May Come"

NPR and IBR



Outline

- Goals
- Design and technical issues
- Step by step process
- Workflow and "Pipeline"
- Tools used and developed
- Summary

Goal



Create a painted world

- Use the director's and cinematographer's composition.
- Needed time coherency and a sense of depth.
- No motion control or green screens.
- Motion Paintsm system is the result.

Design and Technical Issues



Questions we faced

- How do we track objects in a scene and what information do we capture?
- Are objects made out of textures in 3D, or 2D renderings?
- How do we create a non-claustrophobic experience?

Design and Technical Issues



Influences, Claude Monet



Design and Technical Issues



Influences, Vincent Van Gogh



Design and Technical Issues

Influences, Caspar David Friedrich



Design and Technical Issues

Influences, John Mallord William Turner



Design and Technical Issues

We used a painter's vocabulary for brush strokes attributes

- Color
- Texture
- Shape
- Size
- Density
- Orientation



Process

Particle system

- An in-house particle system rendered particles with brush strokes.
- Images were used to control the massive number of strokes (IBR).
- In-house 2D animation system and a commercial available 3D system were used for key elements.



Process

First step

- Reference image given to us by the art director for each shot.
- This reference image was used to deduce:
 - Color palette for the shot
 - Brush textures to be used for which portions of the image.
 - Shape and size of brush strokes.
 - The density of brushes.



Process

Color

- Original plates were "colorized" from which brush strokes retrieved their color.
- The plates were processed to bring the sources into the palette of the reference still.
- Plates were processed in a painterly manner, not photo-realistic (e.g., a surface of water may start out blue or gray, but would end up, perhaps, green, red and yellow).

Process

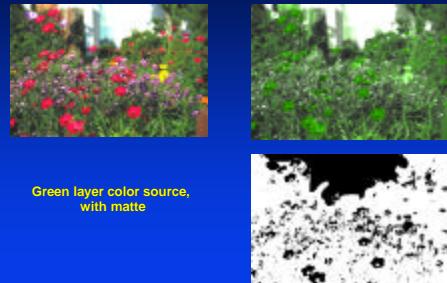
Texture, Shape and Size

- The reference image contained areas of common brush stroke texture, shape and size.
- Images were segmented via mattes into regions that were to be of common brush stroke attributes.
- These mattes were obtained via multiple techniques... the two most commonly used were keying and rotoscoping.

Background Layer



Green Layer



Pink Flower Layer



Red Flower Layer



Process

Lighting

- Lighting each brush stroke in a 3D sense could make the image very "speckly" and visually confusing.
- Often painted dark areas first, midtones next, and highlights last, with a brush stroke layer for each. This is similar to how a painter would work in a physical medium.
- As a result, we needed mattes to segment the image into these areas.

Process

Lighting

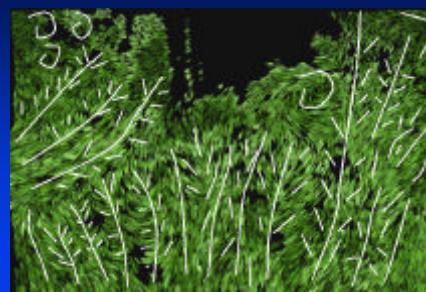
- In some scenes the lighting came directly from the live-action.
- In many cases we needed to relight the scene because most of the shooting occurred during overcast days.
- Scenes were relit using hand techniques, image processing and 3D techniques.

Process

Density and Size

- We controlled density and size with gray scale images.
- Often these control buffers were generated from some pseudo-3D information representing distance from the camera.
- Because we used images as input, these could be hand painted as well as generated from a 3D system or derived from image or motion analysis data.

Flow/orientation for green layer



Process

Motion Analysis and Synthesis

- The sequence was processed with optical flow to obtain motion vectors for each pixel for each frame.
- The motion vectors were QA'd by an animator and patched where the motion failed. (QA occurred by processing strokes with the motion data).
- The motion fields were patched with hand-tracked 2D and 3D data.
- For Van Gogh skies, flow was produced by drawing the flow field with our in-house 2D system.

Process

Motion Analysis and Synthesis, Camera Info

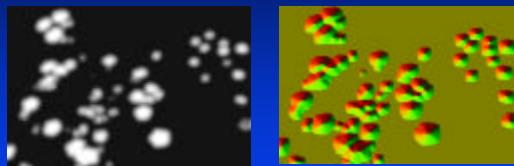
- No motion control rigs were used.
- Orange balls were placed in the scene and surveyed.
- Camera parameters were captured using 3D camera reconstruction software.
- 2D and 3D objects were modeled by animators and tracked through the scene when necessary.
- 3D models were also constructed to help control the painting process (brush strokes could follow their motion).

Process

Orientation obtained using:

- Image processing of color plates and mattes.
- Splines of the 2D animation system. Tangents of the splines were interpolated for the whole image. These splines could be tracked through scenes using the motion analysis.
- 3D objects and surfaces.

Orientation Buffer for Red Flowers



Process

Maintaining depth

- Sunbeams and fog were rendered with more photo-realistic techniques to help cue depth.
- Size of brush strokes roughly mimicked the scale of the textures being reproduced:
 - Larger in front and smaller in back
 - Smaller for high frequency detail, and larger for low frequency detail
 - No 6ft tall brush strokes next to the actors.

Process

Brush strokes have a “lifetime.”

- Use age of stroke to:
 - dissolve in or out.
 - grow or shrink.
- This helps us:
 - deal with reveals and obscurations.
 - give static images some life.

Process

Process

Combining elements

- All elements (2D, 3D and live-action) were processed with the same brush stroke rendering technique to give a consistent style.
- To this day there are differing opinions about some of the elements' sources.

Workflow

Data flow was omnidirectional

- Compositors supplied color processed plates and mattes to the particle system artist.
- Roto artists made mattes for the actors and key objects in a scene (mattes were used to control the motion paint process).
- “Clean plates” were made (sequences without the actors), which were used for brush stroke coloring and the motion analysis step.

Workflow

Data flow was omnidirectional

- If a motion paint artists couldn't get the desired control, he or she would ask for new mattes and control buffers from a compositor, 2D or 3D animator.
- Motion of synthetic objects could control the brush stroke motion. Conversely, motion analysis data often directed 3D object movement.
- Compositor took all the elements and integrated them into final images.

Workflow

Data flow was omnidirectional

- We had teams consisting of each type of artist (instead of individual departments of the same type of artist):
 - Compositor
 - Roto
 - 2D/3D animator
 - Motion Paint specialist

Tools

- Traditional 2D roto/compositing software (Cineon).
- Traditional 3D software (SoftImage).
- In-house particle system (P2).
- In-house interactive 2D animation system(animMate).
- In-house tools for orientation control.
- In-house tools for fixing or authoring motion.

Tools

- Visual look and NPR algorithms must be developed in tandem.
- Tools in hand are better than tools in development.
- Computer vision tools
 - *Most development focuses on making algorithms “perfect.”*
 - *We chose a different tactic by focusing on fix-it tools that allow animators to guide or repair the automatic processes.*

Summary

The big win of IBR techniques

- Wind was a bonus because it added life to each scene.
- Motion Paint scenes were “performance driven” where the performer was the landscape itself.
- Thousands of brushes could be animated that would have been difficult, if not prohibitively expensive, by hand.

Summary



We were successful at using computer vision even though the techniques are far from perfect...

- Painterly processing is very forgiving to mistakes.
- Provided interactive tools to fix mistakes generated by the vision techniques instead of trying to make the vision techniques perfect.
- Great artists created extraordinary images using the tools.

Introduction to 3D Non-Photorealistic Rendering: Silhouettes and Outlines

Aaron Hertzmann
Media Research Laboratory
Department of Computer Science
New York University
<http://www.mrl.nyu.edu/hertzmann/>

1 Introduction

In these notes, we survey some of the basic tools used for non-photorealistic rendering of 3D scenes. We will primarily focus on detecting outlines of object shape: silhouettes, boundaries, and creases. (Hatching and shading, which are also very important for communicating shape, are discussed elsewhere in the course notes.) The algorithms in this section can be divided into algorithms that operate in the image space (2D), and algorithms that operate in world space (3D).

2 Outlines in Image Space

The easiest way to detect silhouettes is to let your existing graphics packages do all the hard work. By rendering images in different ways and then post-processing, you can quickly produce pleasing results.

Image-space non-photorealistic rendering algorithms use rendered (and, thus, sampled) images, and are limited by the precision of the images. However, image space silhouette detection is sufficient for many applications. Exact algorithms that operate in 3D are described in Section 3.

Some of the simplest silhouette algorithms were introduced by Gooch et al. [12], and are described in Chapter 10 of the course notes. A related method by Raskar and Cohen [17] allows variable-length line segments; see their paper for more details.

2.1 Outlines with Edge Detection

A very simple way to generate a line drawing of a 3D scene would be to render the scene from the desired point of view, detect edges in the image, and display the edges. However, the edges of a

photograph do not typically correspond to the silhouette edges that we want to illustrate [19]. For instance, highly textured surfaces will generate many edges that are irrelevant to the object shape; likewise, no edge is detected between two overlapping objects with the same color.

2.1.1 Depth Map

A better way to generate silhouettes is to render the image, extract the depth map, and apply an edge detector to the depth map [18, 6, 5] (Figure 1(a,b)). The depth map is an image where the intensity of a pixel is proportional to the depth of that point in the scene. The idea behind this method is that the variation in depth between adjacent pixels is usually small over a single object, but large between objects. Edge detection is described in Appendix A. Most graphics packages provide some way of extracting the depth map from an image¹.

2.1.2 Normal Map

A problem with this method is that it does not detect the boundaries between objects that are at the same depth, nor does it detect creases. (In more formal terms, it can only detect C^0 surface discontinuities.) We can augment the silhouette edges computed with the depth map by using surface normals as well.

We will do this by using a normal map, which is an image that represents the surface normal at each point on an object. The values in each of the (R, G, B) color components of a point on the normal map correspond to the (x, y, z) surface normal at that point.

To compute the normal map for an object with a graphics package, we can use the following procedure [6]. First, we set the object color to white, and the material property to diffuse reflection. We then place a red light on the X axis, a green light on the Y axis, and a blue light on the Z axis, all facing the object. Additionally, we put lights with negative intensity on the opposite side of each axis. We then render the scene to produce the normal map. Each light will illuminate a point on the object in proportion to the dot product of the surface normal with the light's axis. (If negative lights are not available, then two rendering passes will be required.) An example is shown in Figure 1(c,d).

We can then detect edges in the normal map. These edges detect changes in surface orientation, and can be combined with the edges of the depth map to produce a reasonably good silhouette image (Figure 1(e)). These methods together detect C^0 and C^1 discontinuities in the image.

¹In OpenGL, for example, the depth map can be extracted by calling `glReadPixels` with the `GL_DEPTH_COMPONENT` argument.

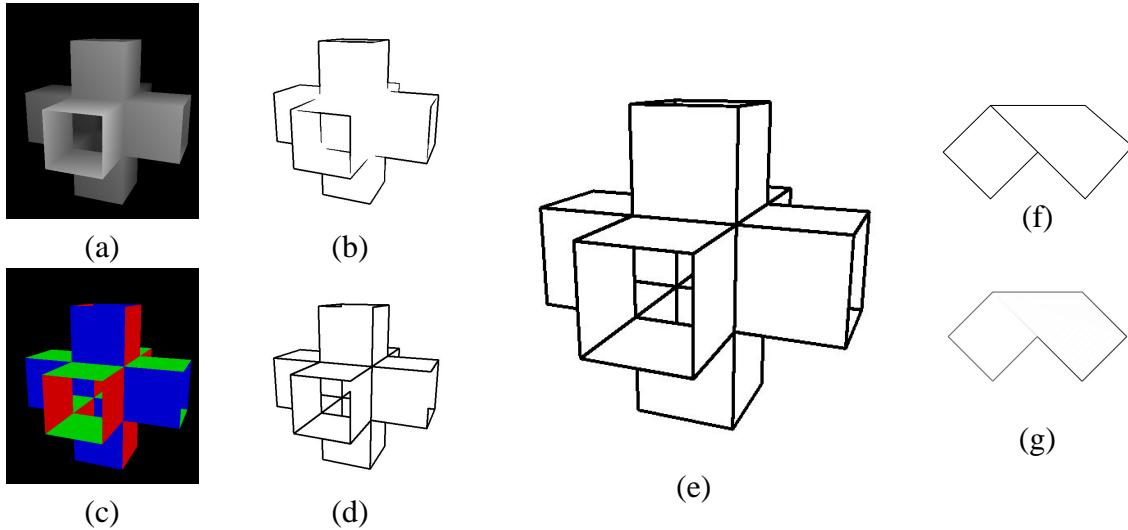


Figure 1: Outline drawing with image processing. (a) Depth map. (b) Edges of the depth map. (c) Normal map. (d) Edges of the normal map. (e) The combined edge images. (f) A difficult case: folded piece of paper (g) Depth edges. (See also the Color Plates section of the course notes.)

2.1.3 Other Types of Textures

We can generalize the above methods to render the image with any smoothly-varying surface function, and then detect edges in the resulting image. For example, we can texture-map the object with a smooth image. However, special care must be taken if the texture map is repeated on the surface [4]. Different methods, such as environment map textures and volumetric textures, can be explored for locating different classes of object lines.

2.2 Rendering

If we intend simply to render the image as dark lines, then these edge images are sufficient. If we wish to render the silhouettes as curves with some kind of attributes, such as paint texture or varying thickness, or if we wish to modify the line qualities, then we must somehow extract curves from the edge map. Methods for doing this are described by Curtis [5] and Corrêa et al. [4]. Saito and Takahashi [18] also discuss some heuristics for modifying the appearance of the edge images.

For many applications, the techniques described in this section are sufficient. However, they do suffer the fundamental limitation that important information about the 3D scene is discarded during rendering, and this information cannot be reconstructed from the 2D image alone. For example, a highly foreshortened surface will appear to be discontinuous. This means that you must manually tweak the image processing parameters for each model and scene. Furthermore, there

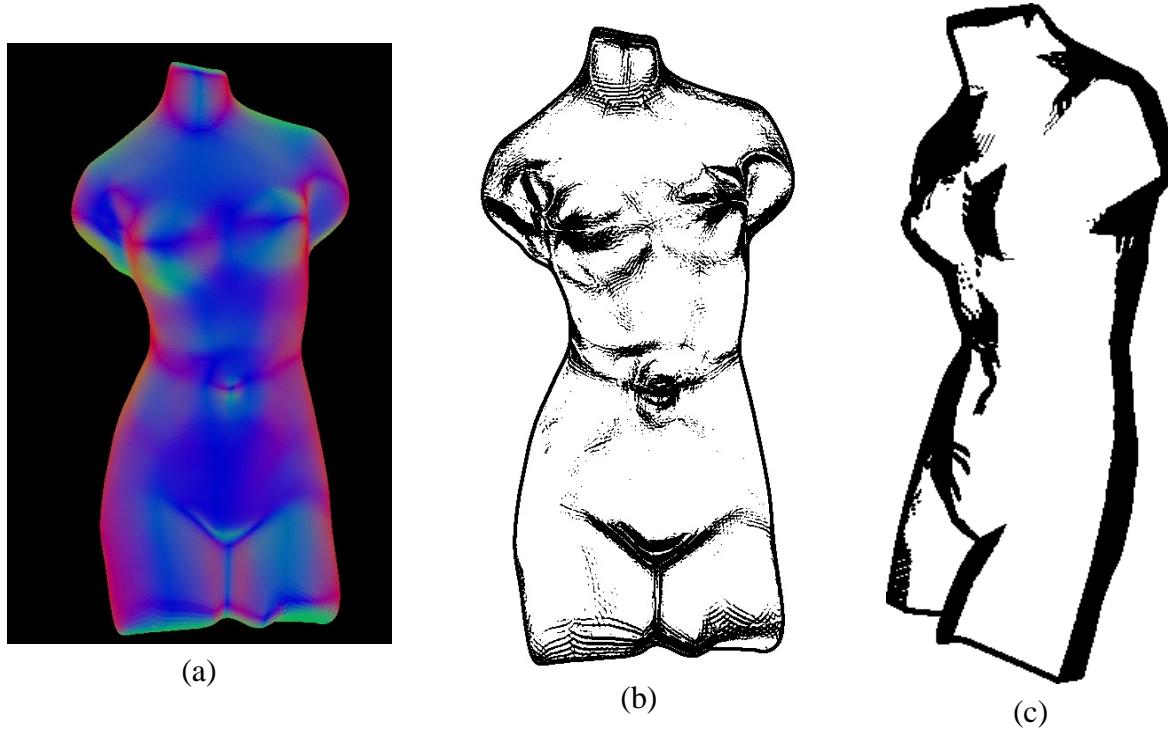


Figure 2: Serendipitous effects using edge detection. (a) Normal map of a subdivided Venus model. (b) Woodcut-style image generated from edges of the normal map. Sharp edges in the mesh generate hatch marks. (c) A sumi-e style image. Streak lines are due to sampling in the depth buffer. (See also the Color Plates section of the course notes.)

are fundamental limitations; Figure 1(f) shows a pathologically difficult case for these algorithms: none of the surface functionals or textures we have described will locate the corner of fold. In the next section, we describe algorithms that compute outlines precisely.

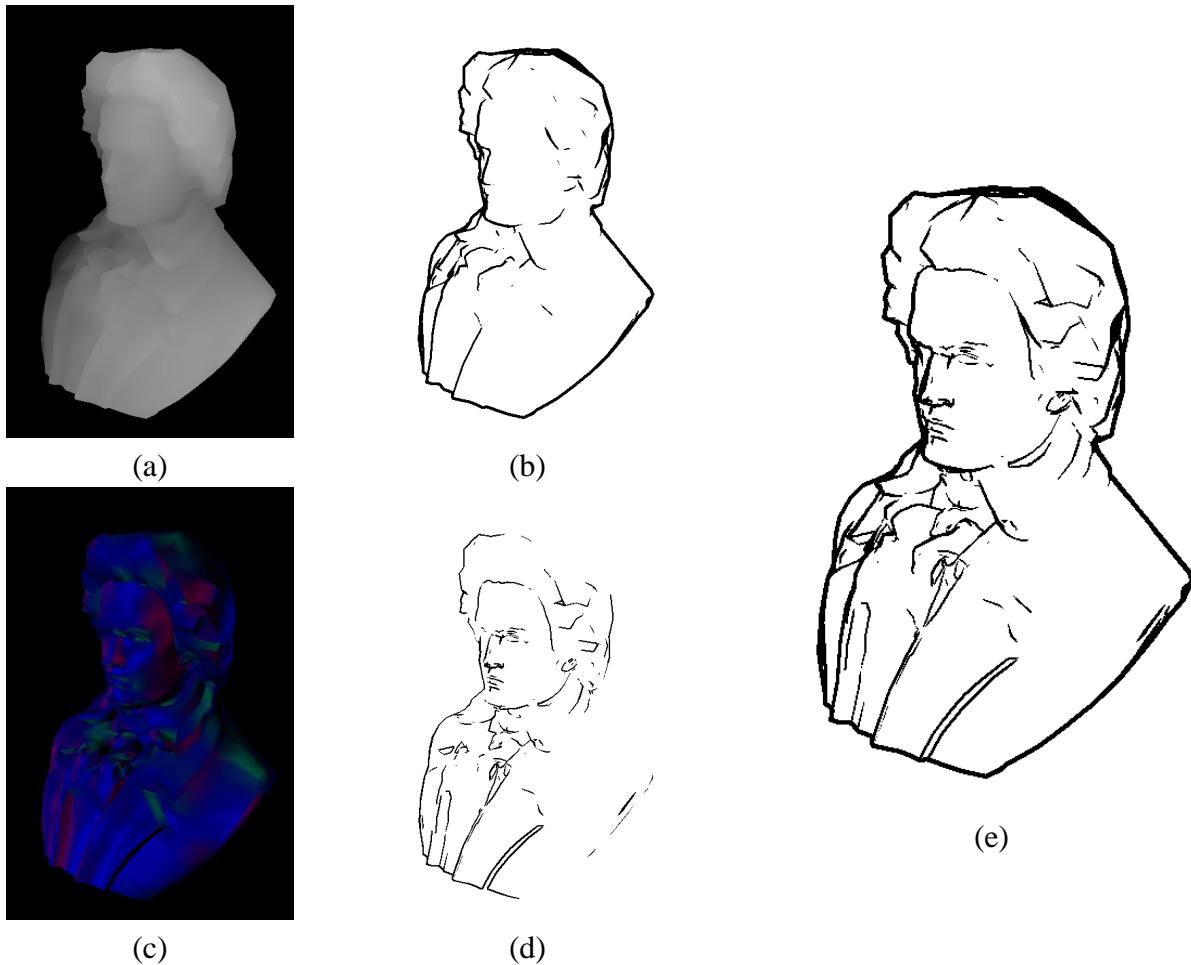


Figure 3: Outline detection of a more complex model. (a) Depth map. (b) Depth map edges. (c) Normal map. (d) Normal map edges. (e) Combined depth and normal map edges. (See also the Color Plates section of the course notes.)

3 Object Space Silhouette Detection

In this section, we will describe methods for finding silhouettes of models in 3D. These methods are more involved than the image space methods, but can produce curves with much higher precision. These curves are also suitable for additional processing; for example, they can be rendered with natural brush strokes.

There are several types of curves we are interested in:

- Silhouettes
- Surface boundaries
- Creases. A crease is a discontinuity on an otherwise smooth surface.
- Self-intersections. Some surfaces that arise in mathematics intersect themselves. Such surfaces are seldom used for typical graphics applications.
- Other surface curves, such as isoparametric curves.

We will primarily discuss detection of silhouettes. Detecting boundaries and creases is straightforward, and can be done in advance for each model. For discussion of isoparametric lines and other kinds of hatch marks, see [21, 7].

3.1 What is a Silhouette?

Before we proceed, we need to make formal the definition of silhouette. For polygonal meshes, the silhouette consists of all edges that connect back-facing (invisible) polygons to front-facing (possibly visible) polygons. For a smooth surface, the silhouette can be defined as those surface points \mathbf{x}_i with a surface normal \mathbf{n}_i perpendicular to the view vector (Figure 4):

$$\mathbf{n}_i \cdot (\mathbf{x}_i - \mathbf{C}) = 0 \quad (1)$$

where \mathbf{C} is the camera center. Note that this definition makes no mention of visibility; a point that is occluded by another object may still be a silhouette point. In almost every case, we are only interested in rendering the visible segments of silhouette curves, or in rendering the invisible sections with a different line quality. Hidden line elimination will be discussed in Section 3.4.

In this discussion, we will focus on perspective projection. For orthogonal projection, all view vectors are parallel, and the equation can be written as $\mathbf{n}_i \cdot \mathbf{v} = 0$, where \mathbf{v} is the view vector.

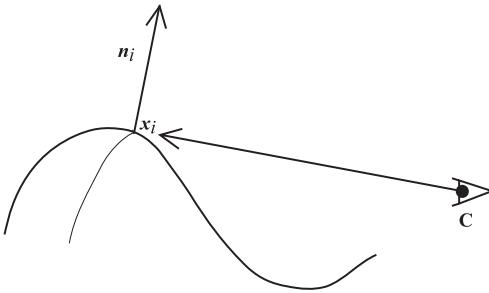


Figure 4: For smooth surfaces, the silhouette is the set of points for which the surface normal is perpendicular to the view vector.

3.2 Polygonal Meshes

For a polygonal mesh, the silhouettes are exactly the set of mesh edges that connect a front-facing polygon and a back-facing polygon. The usual method for computing the silhouette is to iterate over every mesh edge and check the normals of the adjacent faces. This loop must be performed every time the camera position changes.

Testing every mesh edge can be quite expensive. We can speed this up by testing only a few of the edges. Markosian et al. [15] describe a randomized algorithm that can find the majority of visible silhouette edges at interactive rates. Their algorithm is based on the observation that only a small fraction of mesh edges are visible silhouette edges [13]. Thus, we can randomly select a few edges for testing, rather than testing the entire mesh. Please see their paper for details about which edges to test. This algorithm is not guaranteed to find every silhouette edge, but it will find most of them.

For orthographic projection, it is possible to efficiently locate silhouettes using a Gaussian map [12, 2]. See Chapter 10 of the course notes for details.

3.3 Smooth Surfaces

In this section, we describe methods for computing silhouettes of smooth surfaces [11, 8]. It is often desirable to generate pictures of surfaces that are smooth, rather than just polyhedral. Two of the most popular surface representations are NURBS [10] and subdivision surfaces [20]. Both of these representations are based on polyhedral meshes; the actual smooth surface approximates (or interpolates) a mesh. Furthermore, it is possible to compute the surface normal and principle curvatures at any surface point². Our discussion here will focus on these surface representations. For discussion of implicit surfaces, see [3].

²Non-stationary subdivision surfaces are an exception. Little is known about the limit behavior of these surfaces.

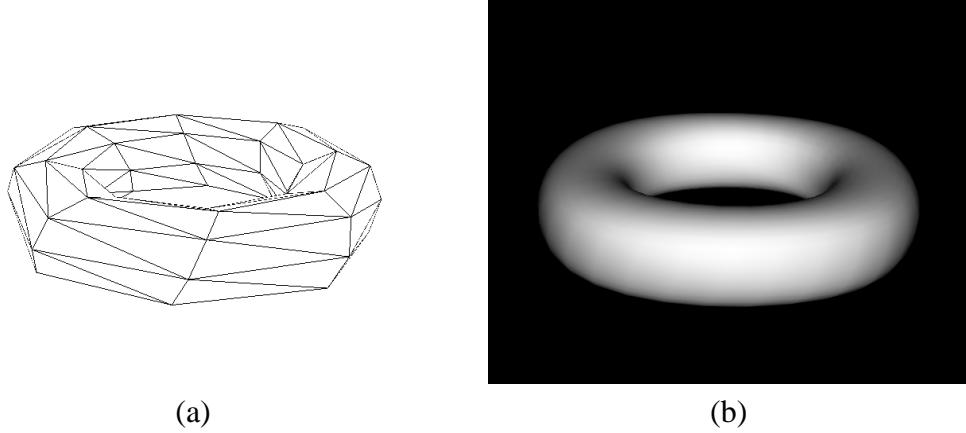


Figure 5: (a) A polygonal mesh (b) A smooth surface defined by the mesh

For simplicity, we will only discuss surfaces approximated by a triangular mesh (Figure 5). A quadrilateral mesh can be converted into a triangular mesh by splitting each quadrilateral into two triangles. Alternatively, these ideas can be extended to polygonal meshes; however, many more cases must be handled [11].

The first step is to compute the normalized dot product d_i of the normal \mathbf{n}_i of the smooth surface with the view vector at every mesh vertex:

$$d_i = \frac{\mathbf{n}_i \cdot (\mathbf{x}_i - \mathbf{C})}{\|\mathbf{n}_i\| \|\mathbf{x}_i - \mathbf{C}\|} \quad (2)$$

We then compute the sign of the dot product for each vertex:³

$$s_i = \begin{cases} +, & d_i \geq 0 \\ -, & d_i < 0 \end{cases} \quad (3)$$

Recall that our goal is to locate surface points with $d_i = 0$. Because these quantities vary smoothly over the surface, we can simply look at all pairs of adjacent vertices that have different signs. Given a mesh edge between points \mathbf{x}_i and \mathbf{x}_j , if $s_i \neq s_j$, then there must be a silhouette point on the edge. (This can be shown by noting that the surface normal and the view vector are both continuous over the surface; therefore, their dot product must also be continuous. There must exist a zero-crossing between a positive and a negative value of a continuous function.) We can approximate the position of the silhouette point by linearly interpolating the vertices:

$$\mathbf{x}' = \frac{|d_j|}{|d_i| + |d_j|} \mathbf{x}_i + \frac{|d_i|}{|d_i| + |d_j|} \mathbf{x}_j \quad (4)$$

³By treating 0's as positive, we avoid some complicated bookkeeping later on.

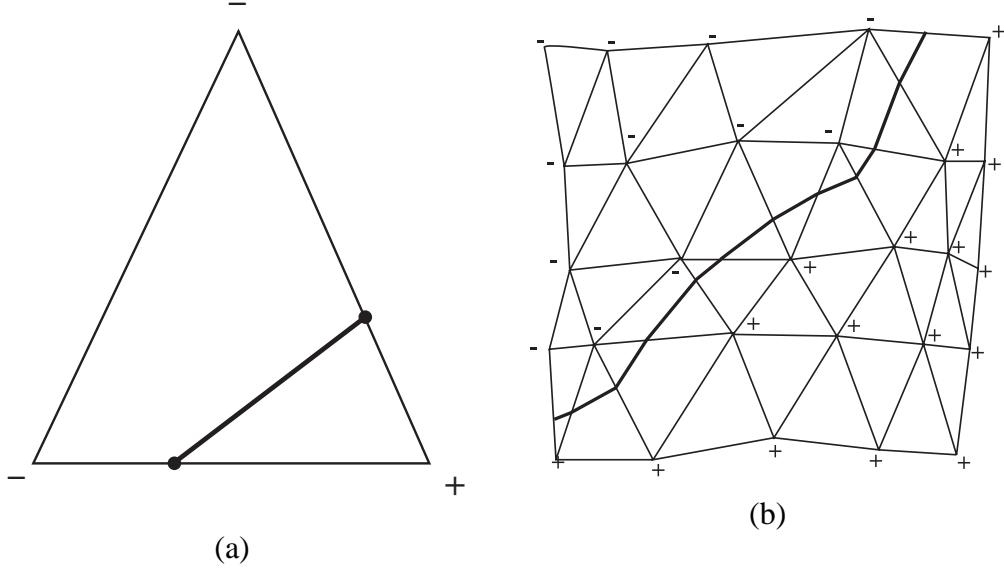


Figure 6: Approximating the silhouette curve for a smooth surface. The smooth surface is represented by a triangle mesh that is close to the surface. (a) Silhouette points are computed on each edge by linear interpolation. A line segment is created by connecting the points. (b) Line segments are connected to form a piecewise-linear curve.

We can then connect the silhouette points into silhouette curves. Suppose a mesh triangle contains sign changes. Since a triangle only contains three vertices, there is only one unique case (Figure 6(a)): one vertex has an opposite sign from the other two vertices. In this case, two triangle edges have a sign change, and the other does not. To approximate the silhouette in this triangle, we can simply compute silhouette points on the two edges, and connect them with a line segment. We then repeat this for every triangle with a sign change, and connect all pairs of line segments that share an end point (Figure 6(b)). This produces a piecewise-linear approximation to the silhouettes of the smooth surface.

If the mesh has large triangles, then the silhouette approximation will be very coarse. It is also possible for silhouettes of the smooth surface to be missed entirely by this method. The solution to this is to refine the mesh; in other words, to produce a mesh with smaller triangles that corresponds to the same smooth surface. The specific method of subdivision depends on the type of surface being used. Subdivision can be applied repeatedly, until the desired resolution is reached.

Furthermore, if we are only interested in mesh silhouettes, then we can use adaptive subdivision to refine the surface only near the silhouette. The obvious subdivision criteria is to subdivide every triangle that already contains a silhouette. However, this will miss silhouettes that are contained entirely within a single triangle. A sufficient condition is to subdivide a triangle if there is a sign

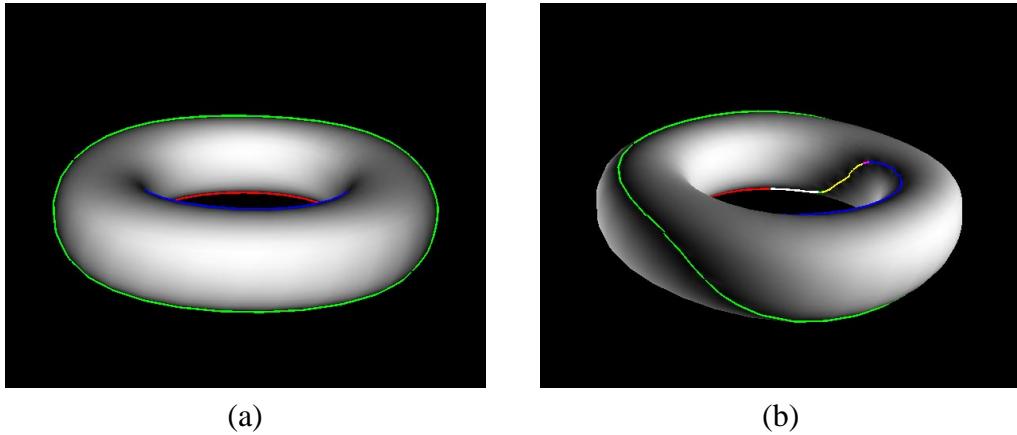


Figure 7: (a) Silhouette of a smooth surface. (b) Side view of the silhouette. Changes in visibility in this figure are due to cusps. (See also the Color Plates section of the course notes.)

change anywhere in the control mesh for a triangle. (Otherwise, the control mesh is 1-1 in the image plane, and cannot produce a silhouette.)

3.4 Visibility

Once we have curves of interest from a surface, we would like to determine which portions of these curves are visible. The visibility process is more or less the same for most types of curves (silhouette, boundary, crease, isoparametric line, etc.) Hidden line visibility is one of the oldest problems in computer graphics [1].

The basic algorithm for computing visibility is to break all curves at potential changes in visibility, producing a new set of curves where each curve is entirely visible or entirely invisible. We then determine the visibility of each new curve by ray tests. [1, 8, 15].

There are three situations where the visibility of a surface curve can change (Figure 8):

1. It passes under a silhouette, boundary or crease in the image plane.
2. It intersects a silhouette, crease, or self-intersection curve on the surface.
3. It is a silhouette or boundary and has a cusp.

Note that these are *potential* changes of visibility; for example, a curve completely hidden by another object will be entirely invisible, regardless of any of these cases.

The first case occurs when part of one object obscures another object. This case can easily be detected by projecting curves onto the image plane, and then computing all curve intersections. The intersections can be performed by the sweep-line algorithm [16] or by scan-converting the

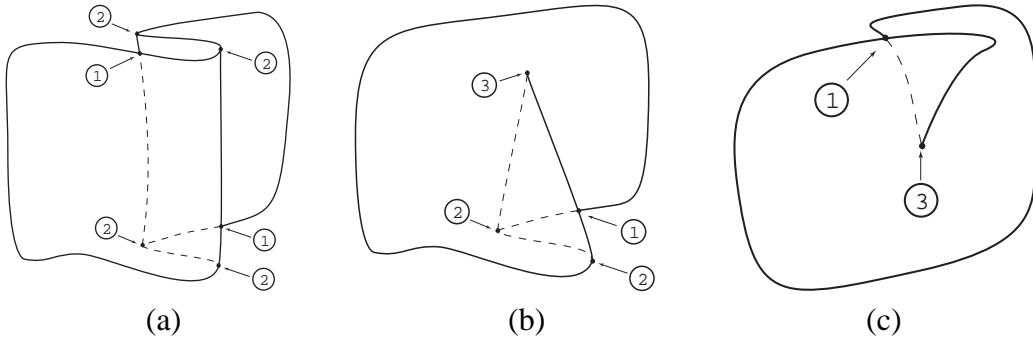


Figure 8: Examples of the three types of potential changes in visibility (See text).

edges of the curve into a grid, and computing intersections for all pairs of edges in each grid cell [15].

The second case is straightforward, and could be detected using the projection method above. However, if one of the curve terminates at another curve, then the intersection might be missed due to numerical precision errors, depending on the representation. It is probably more convenient to detect these cases on the surface.

The third case, a cusp, occurs when the projection of a curve contains a discontinuity. A silhouette or boundary curve may become invisible at a cusp. For polygonal meshes, cusps may only occur at mesh vertecies. A mesh vertex may only be a cusp if it is adjacent to both a front-facing silhouette edge and a back-facing silhouette edge, or if it lies on a boundary. An edge is front-facing if and only if the nearer of the adjacent faces is front-facing. For smooth surfaces, cusps occurs on silhouette curves wherever the projection of the curve has a C^1 discontinuity. Boundaries and creases may only have cusps at tagged corner vertices, i.e. vertices marked as discontinuous.

Once we isolate every potential change in visibility, we can split each curve into smaller curves. We then have a collection of curves, each of which is entirely visible, or entirely invisible. We can then test the visibility of each curve by performing ray tests.

To perform a ray test on a curve, we cast a ray in the direction of the view vector. If the ray intersects any surfaces before the curve, then the curve is invisible; otherwise, it is visible. The intersection test may be accelerated using conventional techniques, such as a BSP tree. For smooth surfaces, the ray test is complicated by the fact that we do not have an explicit surface representation, and so care must be taken when casting rays with respect to an approximating mesh. Kobbelt et al. [14] describe a more elaborate and reliable method for ray tests.

It is possible to avoid many ray tests by taking advantage of visibility coherence on the surface [15, 1, 8]. However, some visibility relationships for curves on smooth surfaces are not the same as for their piecewise-linear approximations, and such approximations should be used with caution.

Once the ray tests are complete, we have determined the visibility of every curve of interest,

and the curves may be rendered.

Acknowledgments

Portions of these notes describe joint work with Denis Zorin.

References

- [1] Arthur Appel. The Notion of Quantitative Invisibility and the Machine Rendering of Solids. In *Proc. ACM National Conference*, pages 387–393, 1967.
- [2] F. Benichou and Gershon Elber. Output Sensitive Extraction of Silhouettes from Polygonal Geometry. ftp://ftp.cs.technion.ac.il/pub/misc/gershon/papers/sil_extrac.ps.gz.
- [3] David J. Bremer and John F. Hughes. Rapid approximate silhouette rendering of implicit surfaces. In *Proc. The Third International Workshop on Implicit Surfaces*, June 1998.
- [4] Wagner Toledo Corrêa, Robert J. Jensen, Craig E. Thayer, and Adam Finkelstein. Texture Mapping for Cel Animation. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 435–446. ACM SIGGRAPH, Addison Wesley, July 1998. ISBN 0-89791-999-8.
- [5] Cassidy Curtis. Loose and Sketchy Animation. In *SIGGRAPH 98: Conference Abstracts and Applications*, page 317, 1998.
- [6] Philippe Decaudin. Cartoon-Looking Rendering of 3D-Scenes. Technical Report 2919, INRIA, June 1996.
- [7] Gershon Elber. Line Art Illustrations of Parametric and Implicit Forms. *IEEE Transactions on Visualization and Computer Graphics*, 4(1), January – March 1998. ISSN 1077-2626.
- [8] Gershon Elber and Elaine Cohen. Hidden Curve Removal for Free Form Surfaces. In Forest Baskett, editor, *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 95–104, August 1990.
- [9] Hany Farid and Eero P. Simoncelli. Optimally Rotation-Equivariant Directional Derivative Kernels. In *7th Int'l Conf Computer Analysis of Images and Patterns*, Kiel, Germany, September 1997.
- [10] Gerald Farin. *Curves and Surfaces for Computer-Aided Geometric Design: A Practical Guide*. Academic Press, Inc., Boston, third edition, 1993.

- [11] Amy Gooch. Interactive Non-Photorealistic Technical Illustration. Master's thesis, University of Utah, December 1998.
- [12] Bruce Gooch, Peter-Pike J. Sloan, Amy Gooch, Peter Shirley, and Richard Riesenfeld. Interactive Technical Illustration. In *Proc. 1999 ACM Symposium on Interactive 3D Graphics*, April 1999.
- [13] Lutz Kettner and Emo Welzl. Contour Edge Analysis for Polyhedron Projections. In W. Strasser, R. Klein, and R. Rau, editors, *Geometric Modeling: Theory and Practice*, pages 379–394. Springer Verlag, 1997.
- [14] Leif Kobbelt, K. Daubert, and Hans-Peter Seidel. Ray tracing of subdivision surfaces. In *Eurographics Rendering Workshop '98 Proceedings*, 1998.
- [15] Lee Markosian, Michael A. Kowalski, Samuel J. Trychin, Lubomir D. Bourdev, Daniel Goldstein, and John F. Hughes. Real-Time Nonphotorealistic Rendering. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 415–420. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.
- [16] F.P. Preparata and M.I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, 1985.
- [17] Ramesh Raskar and Michael Cohen. Image Precision Silhouette Edges. In *Proc. 1999 ACM Symposium on Interactive 3D Graphics*, April 1999.
- [18] Takafumi Saito and Tokiichiro Takahashi. Comprehensible Rendering of 3-D Shapes. In Forest Baskett, editor, *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 197–206, August 1990.
- [19] T. Sanocki, K. Bowyer, M. Heath, and S. Sarkar. Are real edges sufficient for object recognition? *Journal of Experimental Psychology: Human Perception and Performance*, 24(1):340–349, January 1998.
- [20] Peter Schröder and Denis Zorin, editors. *Subdivision for Modeling and Animation*. SIGGRAPH 99 Course Notes, 1999.
- [21] Georges Winkenbach and David H. Salesin. Rendering Parametric Surfaces in Pen and Ink. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 469–476. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.

A Edge Detection with Convolution

Hundreds of papers have been written about detecting edges in images. Fortunately, the kinds of images we are concerned with here — synthetic depth and normal maps — are amenable to reasonably simple edge detectors. In this section, we describe edge detection with the Sobel filter, and a simple variant.

The Sobel kernels are:

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad S_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad (5)$$

Let $I(x, y)$ be a grayscale image. Vertical and horizontal edge images of I are computed by discrete 2D convolution: $I_x(x, y) = I(x, y) \otimes S_x$, $I_y(x, y) = I(x, y) \otimes S_y$. (This operation is an approximation to differentiation of the image in the continuous domain.) Finally, to create an edge image, we compute the magnitude of the derivative, and then threshold the edges by some threshold T :

$$I_{mag}(x, y) = \sqrt{I_x^2(x, y) + I_y^2(x, y)} \quad (6)$$

$$Edge(x, y) = \begin{cases} 1 & I_{mag}(x, y) \geq T \\ 0 & I_{mag}(x, y) < T \end{cases} \quad (7)$$

The Sobel filter sometimes produces noisy results. For better performance, you can replace the S_x and S_y with the following “optimal” 5x5 filters F_x and F_y [9]:

$$\begin{aligned} p_5 &= [0.036470, 0.248968, 0.429123, 0.248968, 0.036470] \\ d_5 &= [-0.108385, -0.280349, 0.0, 0.280349, 0.108385] \\ F_x &= p_5^T d_5 \\ F_y &= d_5^T p_5 \end{aligned} \quad (8)$$

8 Using Non-Photorealistic Rendering to Communicate Shape

Amy Ashurst Gooch and Bruce Gooch
Department of Computer Science
University of Utah
<http://www.cs.utah.edu/>

8.1 Introduction

The advent of photography and computer graphics has not replaced artists. Imagery generated by artists provides information about objects that may not be readily apparent in photographs or real life. The same goal should apply to computer-generated images. This is the driving force behind non-photorealistic rendering. The term non-photorealistic rendering (NPR) is applied to imagery that looks as though it was made by artists, such as pen-and-ink or watercolor. Many computer graphics researchers are exploring NPR techniques as an alternative to photorealistic rendering. More importantly, non-photorealistic rendering is now being acknowledged for its ability to communicate the shape and structure of complex models. Techniques which have long been used by artists can emphasize specific features, expose subtle shape attributes, omit extraneous information, and convey material properties. These artistic techniques are the result of an evolutionary process, refined over centuries. Therefore, imitating some of these artistic methods and exploring the perceptual psychology behind the techniques of artists are good first steps in going beyond photorealistic rendering.

In these notes, we have gathered the ideas and methods from our previous work [11, 12, 13] to demonstrate how non-photorealistic rendering methods can be used to convey a more accurate representation of the shape and material properties of objects than traditional computer graphics methods. In order to demonstrate how non-photorealistic rendering can be used to communicate shape, we have explored computer-generated technical illustrations.

8.2 Technical Illustration

Human-drawn technical illustrations are usually stand-alone images from a single viewpoint presented on a non-stereo medium such as pen on paper. In this section we discuss the components of such illustrations that we use in a computer graphics context: line character, shading, and shadowing.

Examining technical manuals, illustrated textbooks, and encyclopedias reveals shading and line illustration conventions which are quite different than traditional computer graphics conventions. The use of these artistic conventions produces *technical illustrations*, a subset of non-photorealistic rendering. The illustrations in several books, e.g., [20, 23], imply that illustrators use fairly algorithmic principles. Although there are a wide variety of styles and techniques found in technical illustration, there are some common themes. This is particularly true when examining color illustrations done with air-brush and pen. The following characteristics are present in many illustrations:

- edge lines are drawn with black curves.
- matte objects are shaded with intensities far from black or white with warmth or coolness of color indicative of surface normal.
- a single light source provides white highlights.
- shadows are rarely included, but if they are used, they are placed where they do not occlude details or important features.
- metal objects are shaded as if very anisotropic.

These illustration characteristics result from a hierarchy of priorities. The edge lines and highlights are black and white, respectively, and provide a great deal of shape information themselves. Several studies in the field of perception [2, 4, 6, 26] have concluded that subjects can recognize 3D objects at least as well, if not better, when the edge lines (contours) are drawn versus shaded or textured images. Christou et al. [6] concluded in a perceptual study that “a few simple lines defining the outline of an object suffice to determine its 3-D structure”(p. 712). As seen in children’s coloring books, humans are good at inferring shape from line drawings. Lines can help distinguish different parts and features of an object and draw attention to details which may be lost in shading. Many illustrators use black edge lines to separate parts. Sometimes an illustrator might choose to use a white highlight line instead of a black edge line for interior silhouettes or discontinuities. Deciding which lines to draw and how to draw them is essential in imitating the conventions used in technical illustration. The above observations form only a subset of the conventions used by illustrators. We have concentrated only on the material property and shading aspects of illustration. Work done in computer graphics by Seligmann and Feiner [25] and Dooley and Cohen [7] concentrate on additional aspects of technical illustration like layout, object transparency, and line style. We have also drawn on the work of Markosian et al. [19], Elber [9], and Saito and Takahashi [24]. Markosian et al. [19] developed a real-time 3D interactive system for illustrating silhouettes and creases of non-self-intersecting polygon mesh-based models. Elber [9] provides algorithms for determining NURBS surface information by finding four types of curves: the surface boundary curves, curves along C^1 discontinuities in the surface, isoparametric curves, and silhouette curves. Saito and Takahashi [24] offer convincing pictures to show how 3D models enhanced with discontinuity lines, contour lines, and curved hatching can generate images which convey shape and structure. In Section 8.3, we will discuss the rules, properties, and types of lines needed to convey shape information as accomplished by the line drawings of technical illustrators.

When shading is added, in addition to edge lines, shape information can be maximized if the shading uses colors and intensities that are visually distinct from both the black edge lines and the white highlights. This means the dynamic range available for shading may be limited. Another important characteristic used in technical illustration is the conveyance of material property. Illustrators alternate bands of light and

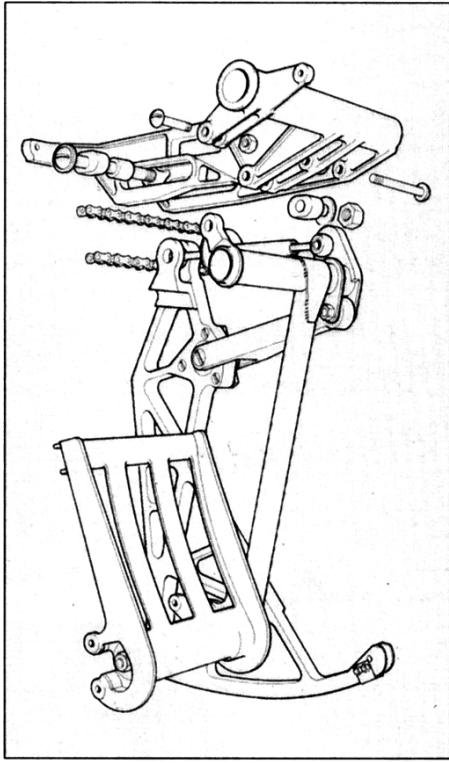


Figure 1: An example of the lines an illustrator would use to convey the shape of this airplane foot pedal. Copyright 1989 Macdonald & Co. (Publishers) Ltd. [20].

dark to represent a metallic object, similar to the real anisotropic reflections seen on real milled metal parts. These shading conventions will be investigated in detail in Section 8.4. Illustrators rarely use shadows in an illustration. Shadows are used only when they do not obscure details in other parts and will be discussed in Section 8.5.

8.3 Lines in Technical Illustration

Several researchers [1, 9, 11, 12, 13, 16, 19, 24] examined which lines should be drawn in a computer generated image to maximize the amount of information conveyed while minimizing the number of lines drawn. They observed that illustrators use edge lines, consisting of surface boundaries, silhouettes, discontinuities, and creases to separate individual parts and to suggest important features in the shape of each object. These static images represented edge lines with black lines of uniform weight.

To decide which lines to draw, we started by analyzing some examples from hand drawn technical illustrations. The illustration in Figure 1 consists of just enough lines to separate individual parts and to suggest important features in the shape of each object.

Most modeling systems display only a wireframe or a shaded image. A wireframe display is common because it can give a lot of information which is occluded by shading. However, a wireframe display of a complex model can be confusing due to the number of lines being displayed, as can be seen by comparing Figure 2 and 3. The wireframe of a Non-Uniform Rational B-Spline (NURBS) surface consists of isolines, which are parameterization dependent. Figure 4 demonstrates that changing which isolines are displayed can change the perception of the surface.

By drawing silhouettes, surface boundaries, discontinuities, and creases, one can imitate the lines drawn in technical illustrations without being parameterization dependent. An example of these different line types is provided in Figure 5. Silhouettes contain the set of points on a surface where $E \cdot n = 0$ or the angle between E and n is 90 degrees, given a point on a surface, σ , with E as the vector from the eye to σ , and n as the surface normal (Figure 6). Regions where the surface normal changes abruptly, C^1 discontinuities or creases, are also important in defining the shape of an object. For a polygonal model, a crease is determined by two front facing polygons whose dihedral angle is above some threshold. In an interactive system, the user should be allowed to control this parameter based on the model and the intent of the image. Sometimes surface boundaries also need to be drawn, but only in the case where there is not a surface connecting another surface or where the joint between surfaces changes abruptly. For example, the vertical boundary drawn in a dotted line in Figure 5 should not be drawn, since it is a shared surface boundary [15]. The calculations and implementation details necessary to create these line drawings will be addressed in Section 10 of the course notes.

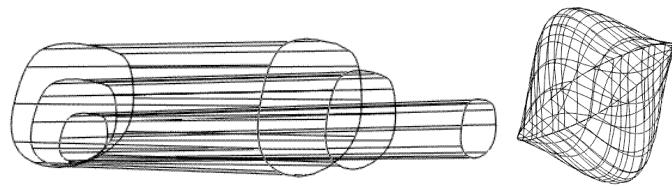


Figure 2: A few examples of a NURBS-based model displayed in wireframe. The large number of isolines makes distinguishing key features difficult.

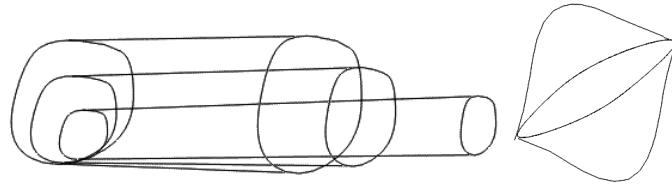


Figure 3: Comparing this figure to Figure 2, the edge lines displayed provide shape information without cluttering the screen.

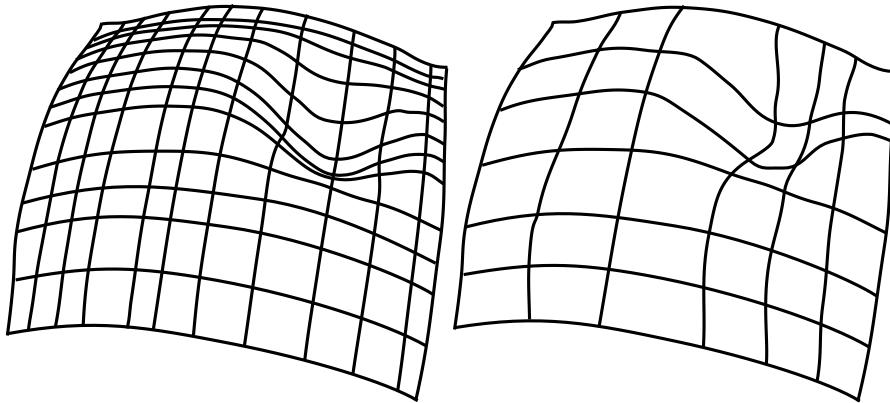


Figure 4: Changing which isolines are displayed can change the perception of the surface. The image on the right looks as if it has a deeper pit because the isolines go thru the maximum curvature point on the surface. Images courtesy of David Johnson.

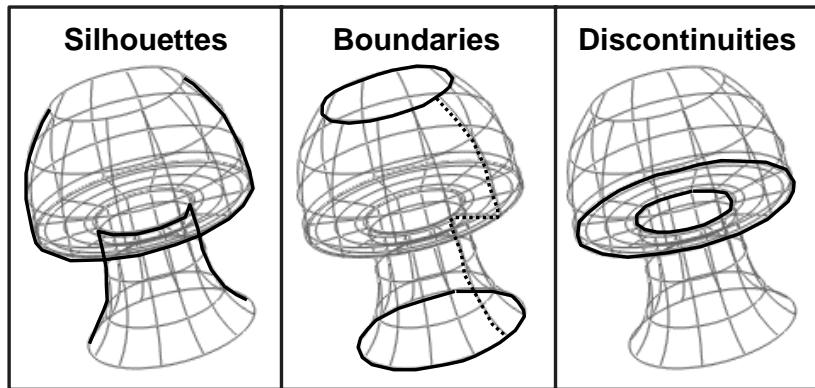


Figure 5: Illustrators use lines to separate parts of objects and define important shape characteristics. This set of lines can be imitated for geometric models by drawing silhouettes, boundaries, and discontinuities/creases, shown above (drawn over the wireframe representation).

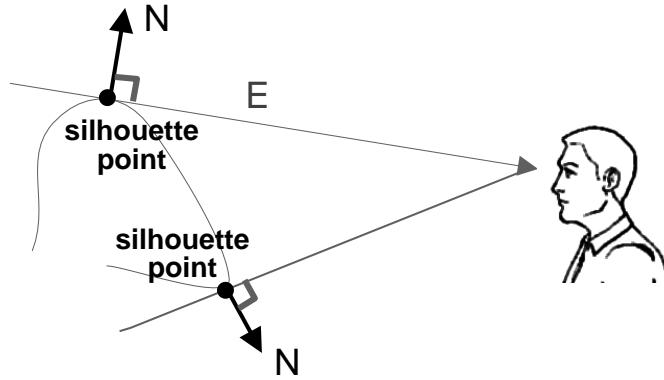


Figure 6: Definition of a silhouette: At a point on a surface, σ and given E as the eye vector and n as the surface normal, a silhouette point is defined as the point on the surface where $E \cdot n = 0$ or the angle between E and n is 90 degrees.

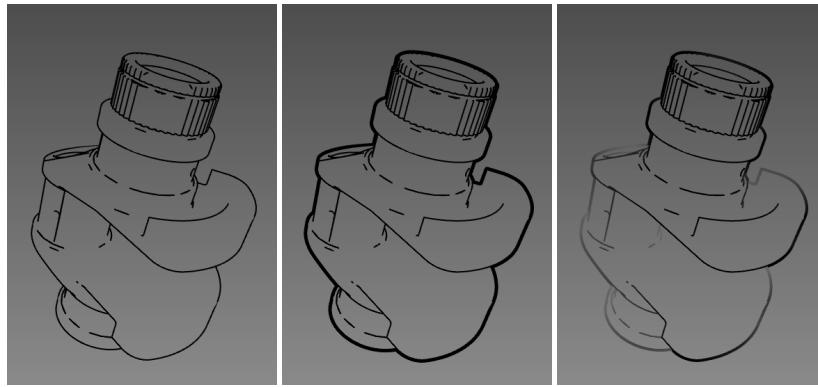


Figure 7: Three line conventions suggested by Martin [20]. Left: single weight used throughout the image. Middle: heavy line weight used for outer edges, other lines are thinner. Right: vary line weight to emphasize perspective.

8.3.1 Line Weight

There are many line weight conventions and an illustrator chooses a specific line weight convention dependent upon the intent of the 2D image. In the book *Technical Illustration*, Martin [20] discusses three common conventions, as shown in Figure 7:

- Single line weight used throughout the image
- Two line weights used, with the heavier describing the outer edges and parts with open space behind them
- Variation of line weight along a single line, emphasizing the perspective of the drawing, with heavy lines in the foreground, tapering towards the farthest part of the object.

One way of achieving the latter effect in raster graphics is to vary the line weight dependent upon the direction of the light source or in an user specified direction, giving a shadowed effect to the line. Most illustrators use bold external lines, with thinner interior lines, which aid in the perception of spaces [8].

Other less often used conventions include varying the line due to abrupt changes in the geometry (curvature based). A method for automatically generating these kind of edges is discussed in Section 10 of the notes, as well as by Raskar and Cohen [22]. However, for the purpose of technical illustration, most illustrators use bold external lines, with thinner interior lines.

8.3.2 Line Color and Shading

In almost all technical illustrations, lines are drawn in black. Occasionally, if the illustration incorporates shading, another convention may apply in which some interior lines are drawn in white, like a highlight. This technique may be the representation of the real white highlights as can be seen on edges of the mechanical part in Figure 8. By using this convention, lines drawn in black and white suggest a light source, and denote the model's orientation. For example, Figure 9 shows how an artist may use white for interior lines, producing a highlight.

Another example is shown in Figure 9, comparing an illustration produced by an artist and an image from our system in which white creases are drawn.

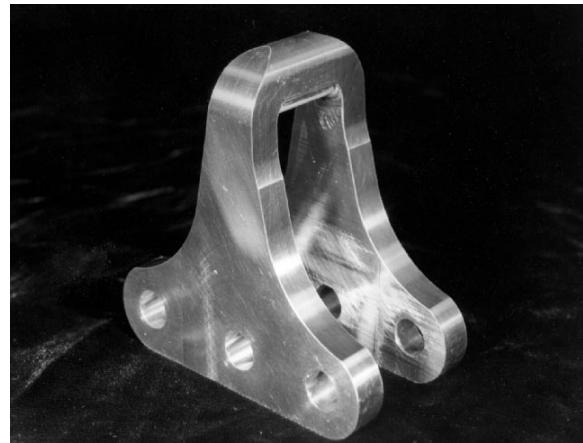


Figure 8: This photograph of a metal object shows the anisotropic reflections and the white edge highlights which illustrators sometimes depict.

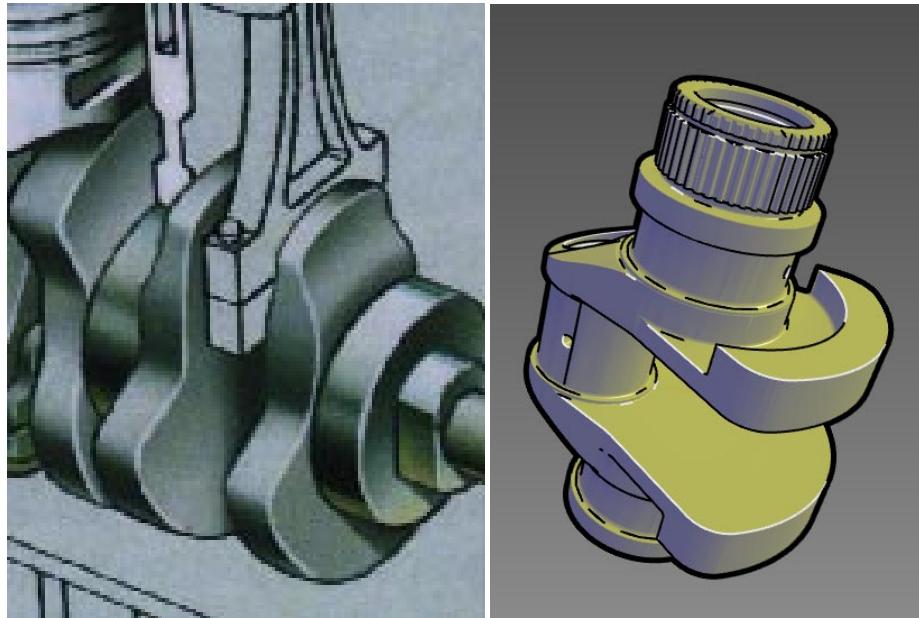


Figure 9: Left: Illustrators sometimes use the convention of white interior edge lines to produce a highlight. Courtesy of Macmillan Reference USA, a division of Ahsuog, Inc. [23]. Right: An image produced by our system, including shading, silhouettes, and white crease lines.

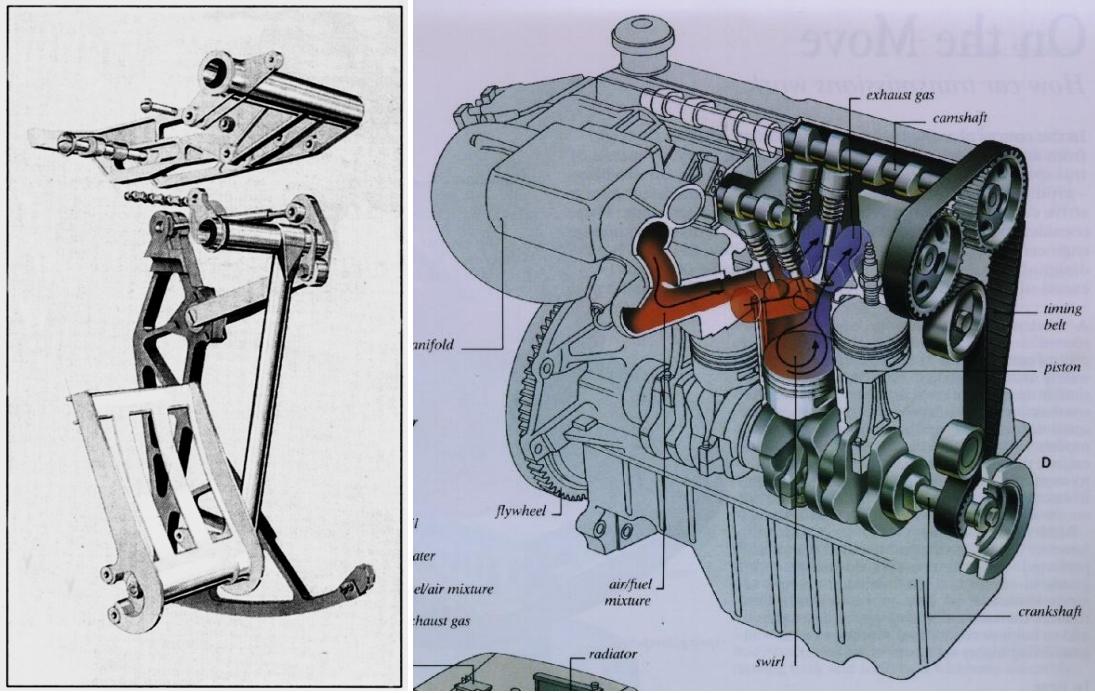


Figure 10: Illustrators combine edge lines with a specific type of shading. Shading in technical illustration brings out subtle shape attributes and provides information about material properties. Left: Compare this shaded image of airplane pedal to the line drawing in Figure 1. Copyright 1989 Macdonald & Co. (Publishers) Ltd. [20]. Right: Engine. Courtesy of Macmillan Reference USA, a division of Ahsuog, Inc. [23].

8.4 Shading

Shading in technical illustration brings out subtle shape attributes and provides information about material properties, as shown in Figure 10. Most illustrators use a single light source and technical illustrations rarely include shadows. In most technical illustrations, hue changes are used to indicate surface orientation rather than reflectance because shape information is valued above precise reflectance information. Adding a hue shift to the shading model allows a reduction in the dynamic range of the shading, to ensure that highlights and edge lines remain distinct. A simple low dynamic-range shading model is consistent with several of the principles from Tufte's recent book [27]. He has a case study of improving a computer graphics animation by lowering the contrast of the shading and adding black lines to indicate direction. He states that this is an example of the strategy of *the smallest effective difference*:

Make all visual distinctions as subtle as possible, but still clear and effective.

Tufte feels that this principle is so important that he devotes an entire chapter to it in his book *Visual Explanations*. Tufte's principle provides a possible explanation of why cross-hatching is common in black and white drawings and rare in colored drawings: colored shading provides a more subtle, but adequately effective, difference to communicate surface orientation. Based on observing several illustrations, surfaces with little or no curvature are generally flat or Phong-shaded in technical illustrations. Surfaces which have high curvature are shaded similar to the Phong shading model or are cool-to-warm shaded as in Gooch et al. [11], unless the surface has a material property such as metal. Illustrators apply different conventions to convey metallic surface properties, especially if the object has regions of high curvature like an ellipsoid.

8.4.1 Traditional Shading of Matte Objects

Traditional diffuse shading sets luminance proportional to the cosine of the angle between light direction and surface normal:

$$I = k_d k_a + k_d \max(0, \hat{\mathbf{l}} \cdot \hat{\mathbf{n}}),$$

where I is the RGB color to be displayed for a given point on the surface, k_d is the RGB diffuse reflectance at the point, k_a is the RGB ambient illumination, $\hat{\mathbf{l}}$ is the unit vector in the direction of the light source, and $\hat{\mathbf{n}}$ is the unit surface normal vector at the point. This model is shown for $k_d = 1$ and $k_a = 0$ in Figure 11. This unsatisfactory image hides shape and material information in the dark regions. Both

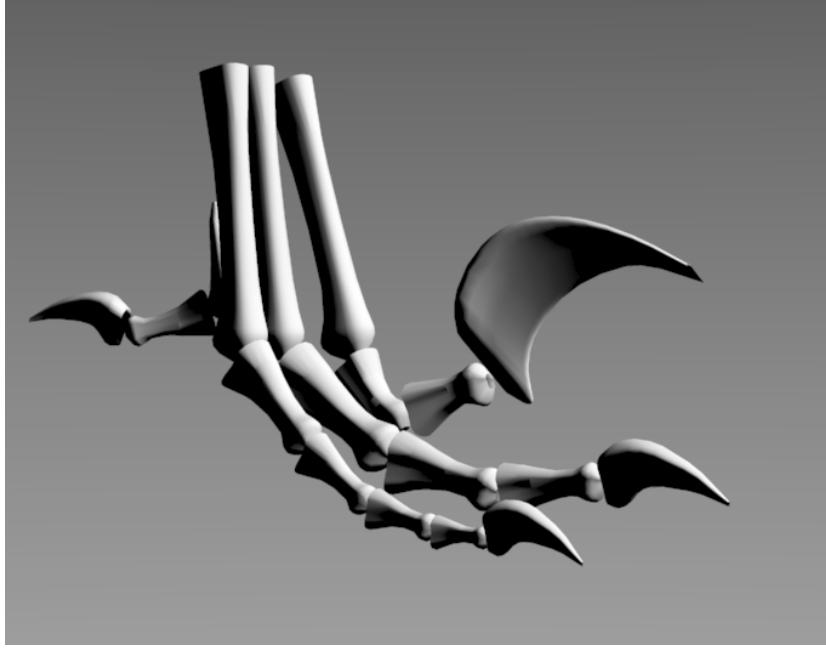


Figure 11: Diffuse shaded image using Equation 1 with $k_d = 1$ and $k_a = 0$. Black shaded regions hide details, especially in the small claws; edge lines could not be seen if added. Highlights and fine details are lost in the white shaded regions.

highlights and edge lines can provide additional information about the object. These are shown alone in Figure 12 with no shading. Edge lines and highlights could not be effectively added to Figure 11 because the highlights would be lost in the light regions and the edge lines would be lost in the dark regions.

To add edge lines to the shading in Equation 1, either of two standard heuristics could be used. First k_a could be raised until it is large enough that the dim shading is visually distinct from the black edge lines, but this would result in loss of fine details. Alternatively, a second light source could be added, which would add conflicting highlights and shading. To make the highlights visible on top of the shading, k_d could be lowered until it is visually distinct from white. An image with hand-tuned k_a and k_d is shown in Figure 13. This is the best achromatic image using one light source and traditional shading. This image is poor at communicating shape information, such as details in the claw nearest the bottom of the image, where it is colored the constant shade $k_d k_a$ regardless of surface orientation.

8.4.2 Tone-based Shading of Matte Objects

In a colored medium such as air-brush and pen, artists often use both hue and luminance (gray scale intensity) shifts. Adding black and white to a given color results in what artists call *shades* in the case of black and *tints* in the case of white. When color scales are created by adding gray to a certain color they are called *tones* [3]. Such tones vary in hue but do not typically vary much in luminance. Adding the complement of a color can also create tones. Tones are considered a crucial concept to illustrators and are especially useful when the illustrator is restricted to a small luminance range [18]. Another quality of color used by artists is the *temperature* of the color. The temperature of a color is defined as being warm (red, orange, and yellow), cool (blue, violet, and green), or temperate (red-violets and yellow-greens). The depth cue comes from the perception that cool colors recede whereas warm colors advance. In addition, object colors change temperature in sunlit scenes because cool skylight and warm sunlight vary in relative contribution across the surface, so there may be ecological reasons to expect humans to be sensitive to color temperature variation. Not only is the temperature of a hue dependent upon the hue itself, but this advancing and receding relationship is effected by proximity [5]. Gooch et al. used these techniques and their psychophysical relationship as the basis for their shading model.

The classic computer graphics shading model can be generalized to experiment with tones by using the cosine term ($\hat{\mathbf{l}} \cdot \hat{\mathbf{n}}$) of Equation 1 to blend between two RGB colors, k_{cool} and k_{warm} :

$$I = \left(\frac{1 + \hat{\mathbf{l}} \cdot \hat{\mathbf{n}}}{2} \right) k_{cool} + \left(1 - \frac{1 + \hat{\mathbf{l}} \cdot \hat{\mathbf{n}}}{2} \right) k_{warm}.$$

Note that the quantity $\hat{\mathbf{l}} \cdot \hat{\mathbf{n}}$ varies over the interval $[-1, 1]$. To ensure the image shows this full variation, the light vector $\hat{\mathbf{l}}$ should be perpendicular to the gaze direction. Because the human vision system assumes illumination comes from above [10], it is best to position the light up and to the right and to keep this position constant.

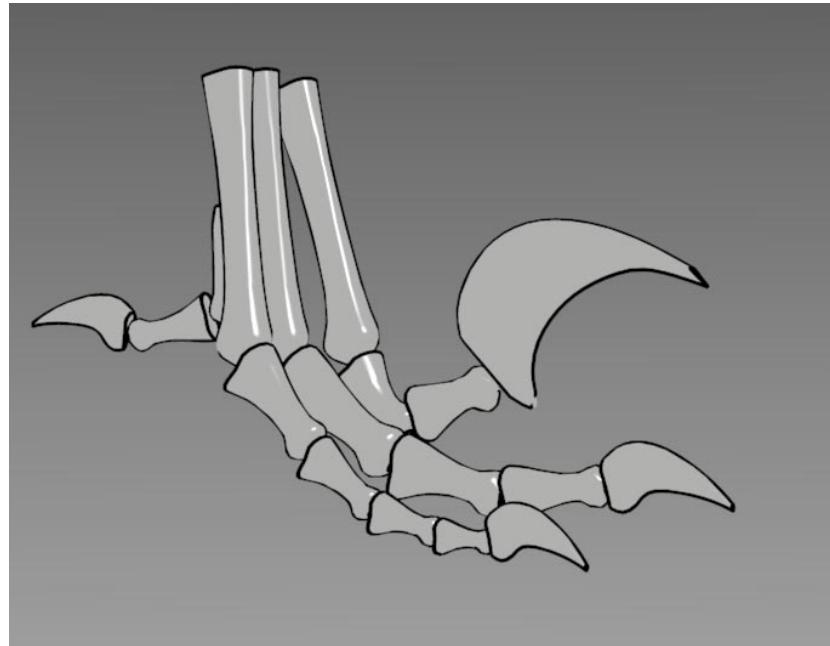


Figure 12: Image with only highlights and edges. The edge lines provide divisions between object pieces and the highlights convey the direction of the light. Some shape information is lost, especially in the regions of high curvature of the object pieces. However, these highlights and edges could not be added to Figure 11 because the highlights would be invisible in the light regions and the silhouettes would be invisible in the dark regions.

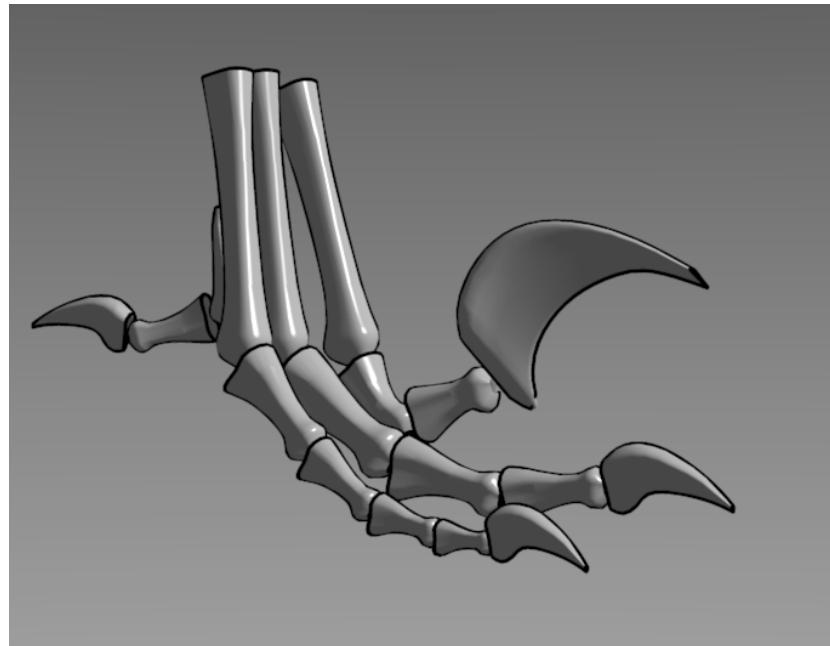


Figure 13: Phong-shaded image with edge lines and $k_d = 0.5$ and $k_a = 0.1$. Like Figure 11, details are lost in the dark gray regions, especially in the small claws, where they are colored the constant shade of $k_d k_a$ regardless of surface orientation. However, edge lines and highlights provide shape information that was gained in Figure 12, but could not be added to Figure 11.

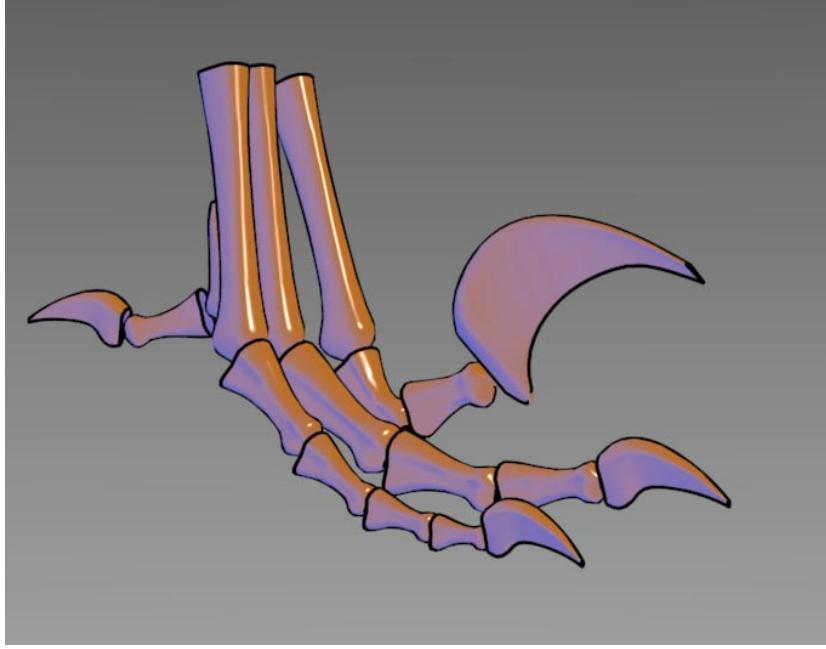


Figure 14: Approximately constant luminance tone rendering. Edge lines and highlights are clearly noticeable. Unlike Figures 11 and 13 some details in shaded regions, like the small claws, are visible. The lack of luminance shift makes these changes subtle. (See Color Plate).

An image that uses a color scale with little luminance variation is shown in Figure 14. This image shows that a sense of depth can be communicated at least partially by a hue shift. However, the lack of a strong cool-to-warm hue shift and the lack of a luminance shift makes the shape information subtle. The unnatural colors may also be problematic. The colors chosen for this hue shift must be picked with care. A red-green hue shift would be undesirable because of red-green color blindness. A blue-yellow hue shift is most common in many art forms and may be most natural because of yellow sun-light and shadows lit by the ambient blue sky. Blue and yellow, having a very large intensity shift, will also provide the desired luminance shift.

In order to automate this hue shift technique and to add some luminance variation to the use of tones, Gooch et al. examined two extreme possibilities for color scale generation: blue to yellow tones and scaled object-color shades. The final model is a linear combination of these techniques. Blue and yellow tones are chosen to insure a cool to warm color transition regardless of the diffuse color of the object.

The blue-to-yellow tones range from a fully saturated blue: $k_{blue} = (0, 0, b)$, $b \in [0, 1]$ in RGB space to a fully saturated yellow: $k_{yellow} = (y, y, 0)$, $y \in [0, 1]$. This produces a very sculpted but unnatural image and is independent of the object's diffuse reflectance k_d . The extreme tone related to k_d is a variation of diffuse shading where k_{cool} is pure black and $k_{warm} = k_d$. This would look much like traditional diffuse shading, but the entire object would vary in luminance, including where $\hat{l} \cdot \hat{n} < 0$. A compromise between these strategies will result in a combination of tone scaled object-color and a cool-to-warm undertone, an effect which artists achieve by combining pigments. The undertones can be simulated by a linear blend between the blue/yellow and black/object-color tones:

$$\begin{aligned} k_{cool} &= k_{blue} + \alpha k_d, \\ k_{warm} &= k_{yellow} + \beta k_d. \end{aligned} \quad (1)$$

Plugging these values into Equation 1 leaves four free parameters: b , y , α , and β . The values for b and y will determine the strength of the overall temperature shift, and the values of α and β will determine the prominence of the object color and the strength of the luminance shift. In order to stay away from shading which will visually interfere with black and white, intermediate values should be supplied for these constants. An example of a resulting tone for a pure red object is shown in Figure 15.

Substituting the values for k_{cool} and k_{warm} from Equation 1 into the tone Equation 1 results in shading with values within the middle luminance range as desired. Figure 16 is shown with $b = 0.4$, $y = 0.4$, $\alpha = 0.2$, and $\beta = 0.6$. To show that the exact values are not crucial to appropriate appearance, the same model is shown in Figure 17 with $b = 0.55$, $y = 0.3$, $\alpha = 0.25$, and $\beta = 0.5$. Unlike Figure 13, subtleties of shape in the claws are visible in Figures 16 and 17.

The model is appropriate for a range of object colors. Both traditional shading and the new tone-based shading are applied to a set of spheres in Figure 18. Note that with the new shading method objects retain their “color name” so colors can still be used to differentiate objects like countries on a political map, but the intensities used do not interfere with the clear perception of black edge lines and white highlights. One issue that is mentioned as people study these sphere comparisons is that the spheres look more like buttons or appear flattened. We hypothesize a few reasons why this may be so. The linear ramp of the shading may be too uniform and cause the spheres

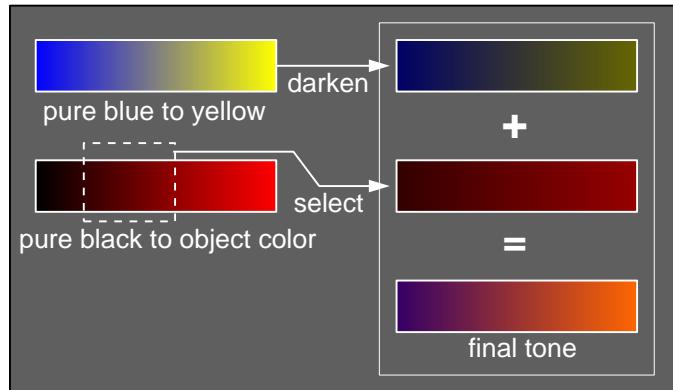


Figure 15: How the tone is created for a pure red object by summing a blue-to-yellow and a dark-red-to-red tone. (See Color Plate).

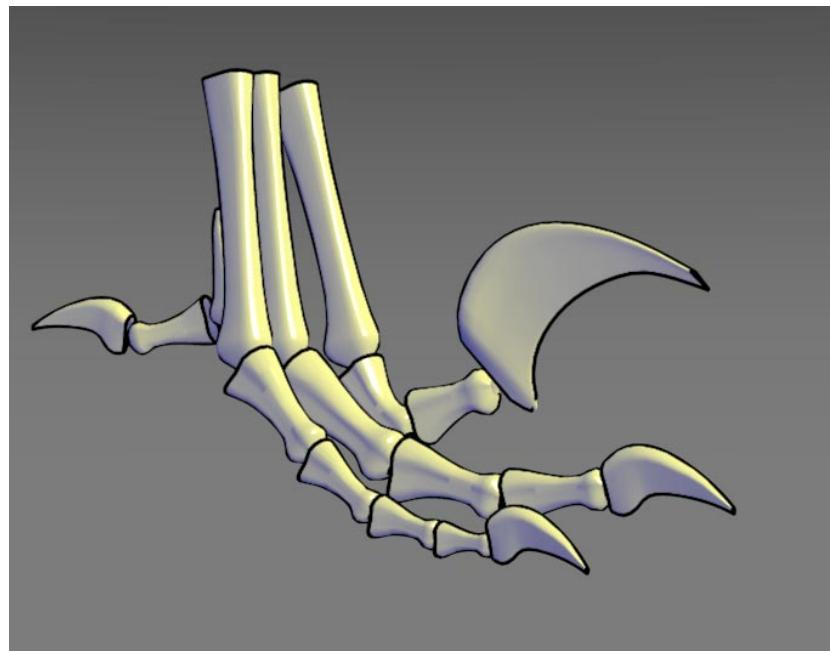


Figure 16: Luminance/hue tone rendering. This image combines the luminance shift of Figure 11 and the hue shift of Figure 14. Edge lines, highlights, fine details in the dark shaded regions such as the small claws, as well as details in the high luminance regions are all visible. In addition, shape details are apparent unlike Figure 12 where the object appears flat. In this figure, the variables of Equation 1 and Equation 1 are: $b = 0.4$, $y = 0.4$, $\alpha = 0.2$, $\beta = 0.6$. (See Color Plate).

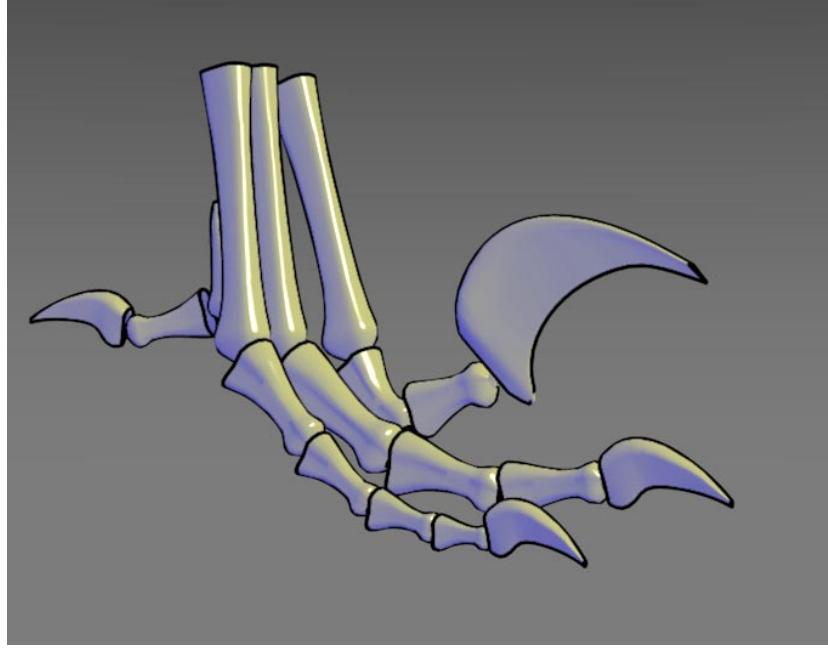


Figure 17: Luminance/hue tone rendering, similar to Figure 16 except $b = 0.55$, $y = 0.3$, $\alpha = 0.25$, $\beta = 0.5$. The different values of b and y determine the strength of the overall temperature shift, whereas α and β determine the prominence of the object color, and the strength of the luminance shift. (See Color Plate).

to flatten. The shading presented here is just a first pass approximation to the shading artists use and much improvement could be made. Another problem may be that the dark silhouettes around to the object may tie the spheres to the background. Figure 19 shows three sets of spheres, shaded the same but put against different gradations of background. The edge lines of the spheres on the darkest background fade a little bit and even seem to thin towards the light, due to the gradation of the background. In our opinion, the spheres set against the darkest background, where the edge lines loose some emphasis, seem to be a little more three dimension than the spheres with edge lines.

Figure 20 shows both the Phong-shaded spheres and the spheres with new shading without edge lines. Without the edge lines, the spheres stand out more. Spheres are not really the best model to test this new shading and edge lines. Edge lines are not really necessary on a sphere, since edge lines are used by illustrators to differentiate parts and discontinuities in a model, something that is not really necessary in a simple model like a sphere. However, it is a computer graphics tradition to test a shading model on the spheres.

8.4.3 Shading of Metal Objects

Illustrators use a different technique to communicate the surface properties of metallic objects, as shown in the photograph in Figure 21. In practice illustrators represent a metallic surface by alternating dark and light bands. This technique is the artistic representation of real effects that can be seen on milled metal parts, such as those found on cars or appliances. Milling creates what is known as “anisotropic reflection.” Lines are streaked in the direction of the axis of minimum curvature, parallel to the milling axis. Interestingly, this visual convention is used even for smooth metal objects [20, 23]. This convention emphasizes that realism is not the primary goal of technical illustration.

To simulate a milled object, Gooch et al. [11] maps a set of 20 stripes of varying intensity along the parametric axis of maximum curvature. The stripes are random intensities between 0.0 and 0.5 with the stripe closest to the light source direction overwritten with white. Between the stripe centers the colors are linearly interpolated. An object is shown Phong-shaded, metal-shaded (without and with edge lines), and metal-shaded with a cool-warm hue shift in Figure 22. The metal-shaded object is more obviously metal than the Phong-shaded image and the metal-shaded object with edge lines provides more shape information. The cool-warm metal-shaded object is not quite as convincing as the achromatic image, but it is more visually consistent with the cool-warm matte-shaded model of Section 8.4.2, so it is useful when both metal and matte objects are shown together. In Section 10 of the course notes, we will discuss how these techniques may need to change in an interactive system.

8.4.4 Approximation to new model

The new shading model presented in Section 8.4.2 cannot be implemented directly in high-level graphics packages that use Phong shading. However, the Phong lighting model can be used as a basis for approximating our model. This is in the spirit of the nonlinear approximation to global illumination used by Walter et al. [28]. In most graphics systems (e.g., OpenGL) negative colors for the lights can be used. Then Equation 1 can be approximated by two lights in directions $\hat{\mathbf{l}}$ and $-\hat{\mathbf{l}}$ with intensities $(k_{warm} - k_{cool})/2$ and $(k_{cool} - k_{warm})/2$ respectively,

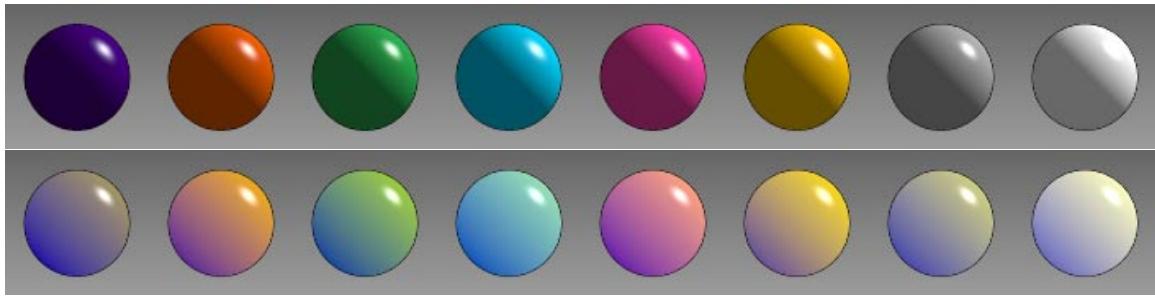


Figure 18: Comparing shaded, colored spheres. Top: Colored Phong-shaded spheres with edge lines and highlights. Bottom: Colored spheres shaded with hue and luminance shift, including edge lines and highlights. Note: In the first Phong-shaded sphere (violet), the edge lines disappear, but are visible in the corresponding hue and luminance shaded violet sphere. In the last Phong-shaded sphere (white), the highlight vanishes, but is noticed in the corresponding hue and luminance shaded white sphere below it. The spheres in the second row also retain their “color name.” (See Color Plate).

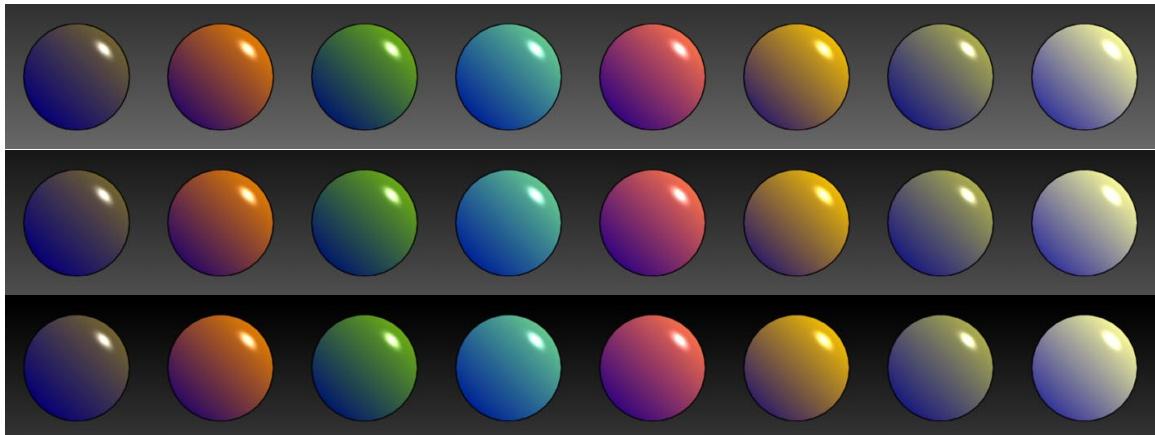


Figure 19: Tone and undertone shaded spheres with backgrounds getting darker.

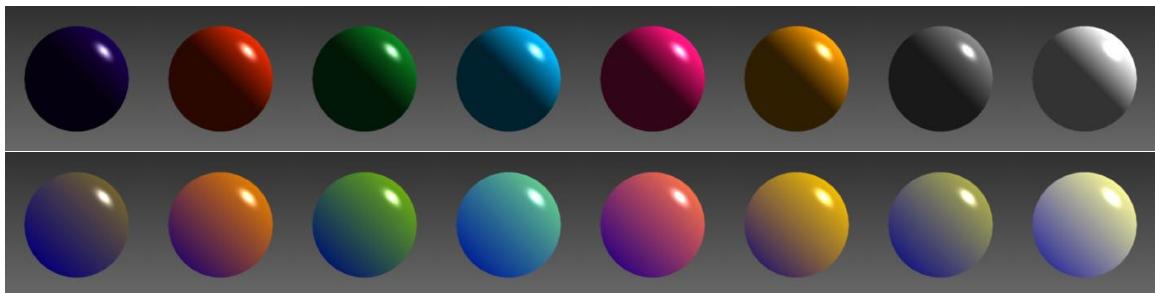


Figure 20: Shaded spheres without edge lines. Top: Colored Phong-shaded spheres without edge lines. Bottom: Colored spheres shaded with hue and luminance shift, without edge lines.



Figure 21: An anisotropic reflection can be seen in the metal objects in this photograph.

and an ambient term of $(k_{cool} + k_{warm})/2$. This assumes the object color is set to white. The Phong highlight should be turned off to remove the jarring artifacts caused by the negative blue light. Highlights could be added on systems with accumulation buffers [14].

C++ Code fragment for generating the two lights, using the OpenGL API:

```

GLfloat R_warm, G_warm, B_warm, R_cool, G_cool, B_cool;
R_warm=207/255.0; G_warm=207/255.0; B_warm=145/255.0;
R_cool=80/255.0; G_cool=80/255.0; B_cool=145/255.0;

GLfloat hi_diffuse[] = { (R_warm-R_cool)/2.0,
                         (G_warm-G_cool)/2.0,
                         (B_warm-B_cool)/2.0 };
GLfloat lo_diffuse[] = { (R_cool-R_warm)/2.0,
                         (G_cool-G_warm)/2.0,
                         (B_cool-B_warm)/2.0 };
GLfloat hi_position[] = { 1, 1, EYE, 1 };
GLfloat lo_position[] = { -1, -1, EYE, 1 };

GLfloat ambient[] = { 0.5, 0.5, 0.5 };

glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambient);

glLightfv(GL_LIGHT0, GL_DIFFUSE, hi_diffuse);
glLightfv(GL_LIGHT0, GL_POSITION, hi_position);
 glEnable( GL_LIGHT0 );

glLightfv(GL_LIGHT1, GL_DIFFUSE, lo_diffuse);
glLightfv(GL_LIGHT1, GL_POSITION, lo_position);
 glEnable( GL_LIGHT1 );

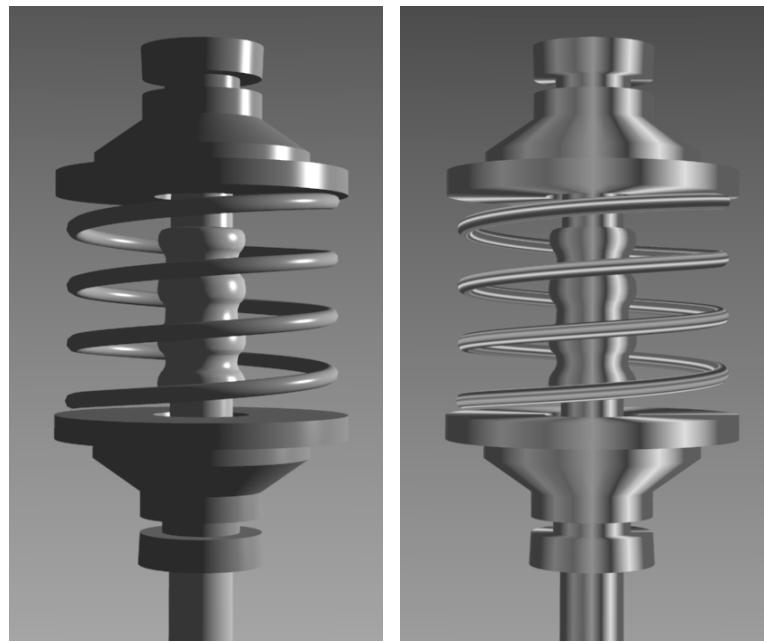
```

This approximation is shown compared to traditional Phong shading and the exact model in Figure 23.

A light source cannot be used for metals with a conventional API. However, either environment maps or texture maps can be used to produce alternating light and dark stripes.

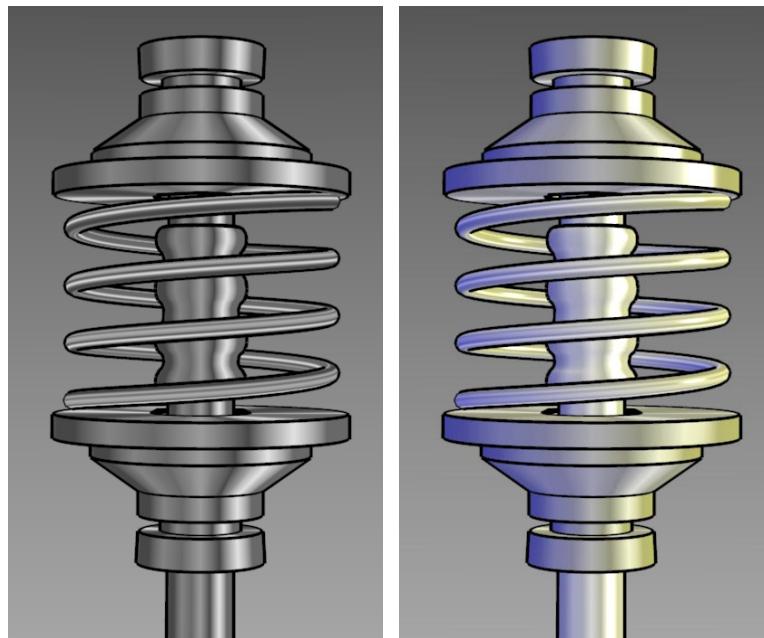
8.5 Shadowing

Illustrators only include shadows when they do not occlude detail in other parts of the object [20, 21, 23]. In 3D interactive illustrations, adding only a drop shadow on a ground plane, not the shadows that an object may cast onto itself, provide helpful visual clues without occluding important details on the object. It is probably not important that these shadows be highly accurate to provide valuable information about three-dimensional structure, especially the spatial layout of a scene [17, 29]. In Section 10 of the course notes, we will discuss the implementation details for adding drop shadows to an interactive illustration system.



(a) Phong-shaded object.

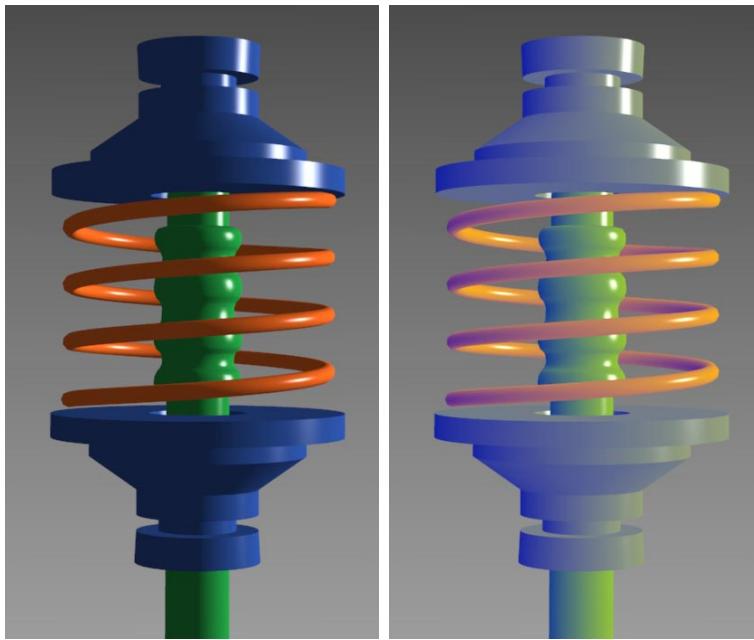
(b) New metal-shaded object without edge lines.



(c) New metal-shaded object with edge lines.

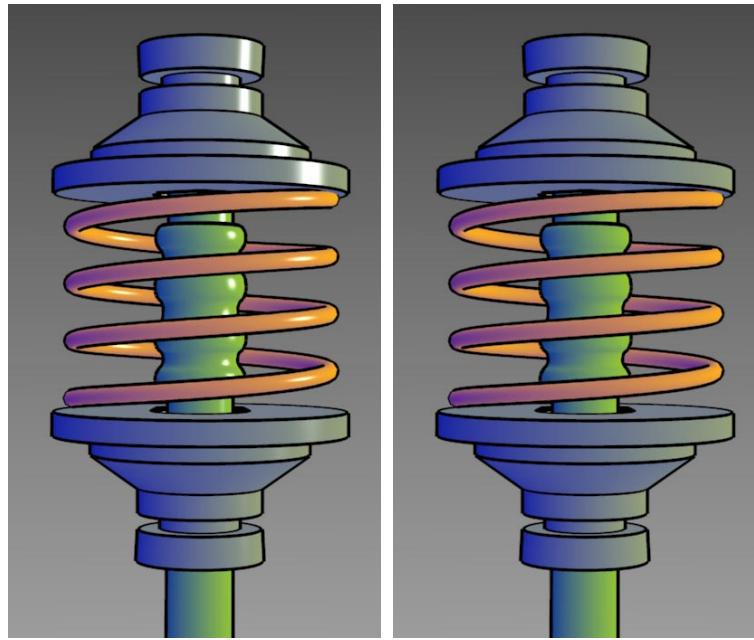
(d) Metal-shaded object with a cool-to-warm shift.

Figure 22: Representing metallic material properties. (See Color Plate).



(a) Phong shading model for colored object.

(b) New shading model without edge lines.



(c) New shading model: edge lines, highlights, and cool-to-warm hue shift.

(d) Approximation: Phong shading, two colored lights, and edge lines.

Figure 23: Comparison of traditional computer graphics techniques and techniques for creating technical illustrations. (See Color Plate).

8.6 Conclusion

Phong-shaded 3D imagery does not provide geometric information of the same richness as human-drawn technical illustrations. We have presented a non-photorealistic lighting model that attempts to narrow this gap. The model is based on practice in traditional technical illustration, where the lighting model uses both luminance and changes in hue to indicate surface orientation, reserving extreme lights and darks for edge lines and highlights. The lighting model allows shading to occur only in mid-tones so that edge lines and highlights remain visually prominent. In addition, we have shown how this lighting model is modified when portraying models of metal objects. These illustration methods give a clearer picture of shape, structure, and material composition than traditional computer graphics methods.

Acknowledgment

This work was done in collaboration with Peter-Pike Sloan, Peter Shirley, and Elaine Cohen.

Authors' Note

These notes should be read while looking at colored images. See the course notes on the SIGGRAPH 99 CD-ROM for the images if colored images do not accompany this document.

References

- [1] David Banks. Interactive Manipulation and Display of Two-Dimensional Surfaces in Four-Dimensional Space. *Symposium on Interactive 3D Graphics*, April 1992.
- [2] Irving Biederman and Ginny Ju. Surface versus Edge-Based Determinants of Visual Recognition. *Cognitive Psychology*, 20:38–64, 1988.
- [3] Faber Birren. *Color Perception in Art*. Van Nostrand Reinhold Company, 1976.
- [4] Wendy L. Braje, Bosco S. Tjan, and Gordon E. Legge. Human Efficiency for Recognizing and Detecting Low-pass Filtered Objects. *Vision Research*, 35(21):2955–2966, 1995.
- [5] Tom Browning. *Timeless Techniques for Better Oil Paintings*. North Light Books, 1994.
- [6] Chris Christou, Jan J. Koenderink, and Andrea J. van Doorn. Surface Gradients, Contours and the Perception of Surface Attitude in Images of Complex Scenes. *Perception*, 25:701–713, 1996.
- [7] Debra Dooley and Michael F. Cohen. Automatic Illustration of 3D Geometric Models: Surfaces. *IEEE Computer Graphics and Applications*, 13(2):307–314, 1990.
- [8] Betty Edwards. *Drawing on the Right Side of the Brain*. Jeremy P. Tarcher/Putnam, 1989.
- [9] Gershon Elber and Elaine Cohen. Hidden Curve Removal for Free-Form Surfaces. In *SIGGRAPH 90 Conference Proceedings*, August 1990.
- [10] E. Bruce Goldstein. *Sensation and Perception*. Wadsworth Publishing Co., Belmont, California, 1980.
- [11] Amy Gooch, Bruce Gooch, Peter Shirley, and Elaine Cohen. A Non-photorealistic Lighting Model for Automatic Technical Illustration. In *Computer Graphics*, July 1998. ACM Siggraph '98 Conference Proceedings.
- [12] Amy A. Gooch. Interactive non-photorealistic technical illustration. Master's thesis, University of Utah, December 1998.
- [13] Bruce Gooch, Peter-Pike Sloan, Amy Gooch, Peter Shirley, and Richard Riesenfeld. Interactive technical illustration. *Interactive 3D Conference Proceedings*, April 1999.
- [14] Paul Haeberli. The Accumulation Buffer: Hardware Support for High-Quality Rendering. *SIGGRAPH 90 Conference Proceedings*, 24(3), August 1990.
- [15] G. Heflin and G. Elber. Shadow volume generation from free form surfaces. In *Communicating with virtual worlds, Proceedings of CGI'93 (Lausanne, Switzerland)*, pages 115–126. Springer-Verlag, June 1993.
- [16] Victoria Interrante, Henry Fuchs, and Stephen Pizer. Enhancing transparent skin surfaces with ridge and valley lines. *Proceedings of Visualization '95*, pages 52–59, 1995.
- [17] D. Kersten, D. C. Knill, P. Mamassian, and I. Bulthoff. Illusory motion from shadows. *IEEE Computer Graphics and Applications*, 379(31), 1996.

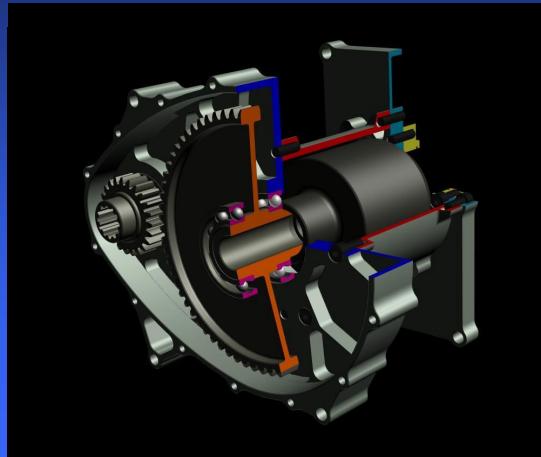
-
- [18] Patricia Lambert. *Controlling Color: A Practical Introduction for Designers and Artists*, volume 1. Everbest Printing Company Ltd., 1991.
 - [19] L. Markosian, M. Kowalski, S. Trychin, and J. Hughes. Real-Time Non-Photorealistic Rendering. In *SIGGRAPH 97 Conference Proceedings*, August 1997.
 - [20] Judy Martin. *Technical Illustration: Materials, Methods, and Techniques*, volume 1. Macdonald and Co Publishers, 1989.
 - [21] Scott McCloud. *Understanding Comics*. Tundra Publishing Ltd., Northhampton, MA, 1993.
 - [22] Ramesh Raskar and Michael Cohen. Image Precision Silhouette Edges. *Symposium on Interactive 3D Graphics*, April 1999.
 - [23] Tom Ruppel, editor. *The Way Science Works*, volume 1. MacMillan, 1995.
 - [24] Takafumi Saito and Tokiichiro Takahashi. Comprehensible Rendering of 3D Shapes. In *SIGGRAPH 90 Conference Proceedings*, August 1990.
 - [25] Doree Duncan Seligmann and Steven Feiner. Automated Generation of Intent-Based 3D Illustrations. In *SIGGRAPH 91 Conference Proceedings*, July 1991.
 - [26] Bosco S. Tjan, Wendy L. Braje, Gordon E. Legge, and Daniel Kersten. Human Efficiency for Recognizing 3-D Objects in Luminance Noise. *Vision Research*, 35(21):3053–3069, 1995.
 - [27] Edward Tufte. *Visual Explanations*. Graphics Press, 1997.
 - [28] Bruce Walter, Gun Alppay, Eric P. F. Lafortune, Sebastian Fernandez, and Donald P. Greenberg. Fitting Virtual Lights for Non-Diffuse Walkthroughs. In *SIGGRAPH 97 Conference Proceedings*, pages 45–48, August 1997.
 - [29] Leonard R. Wanger, James A. Ferwerda, and Donald P. Greenberg. Perceiving spatial relationships in computer-generated images. *IEEE Computer Graphics and Applications*, 12(3):44–58, May 1992.

Using NPR to Communicate Shape

Amy Gooch

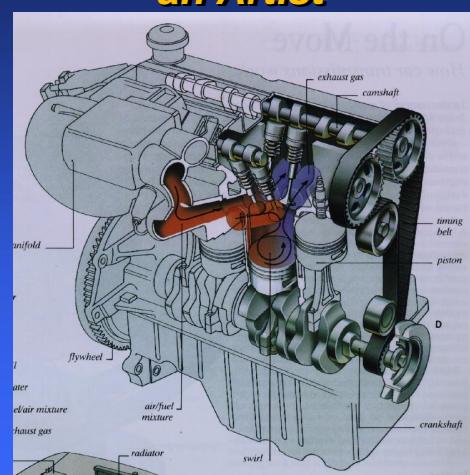
Motivation

*Hand-tuned,
Computer Generated*



Courtesy of Sam Drake

*Illustration by
an Artist*



From *The Way Science Works*,
Courtesy of Macmillan Reference USA.

Photograph



Illustration

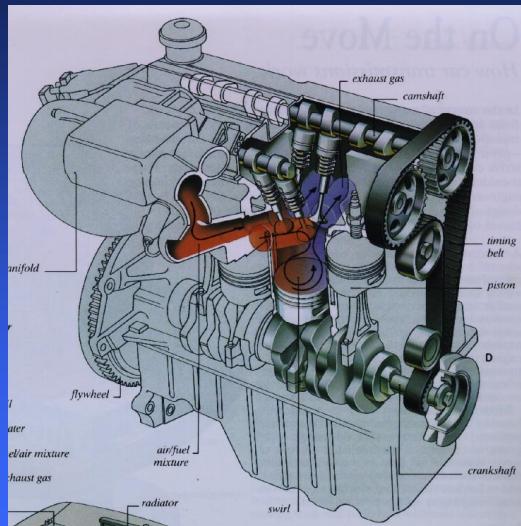


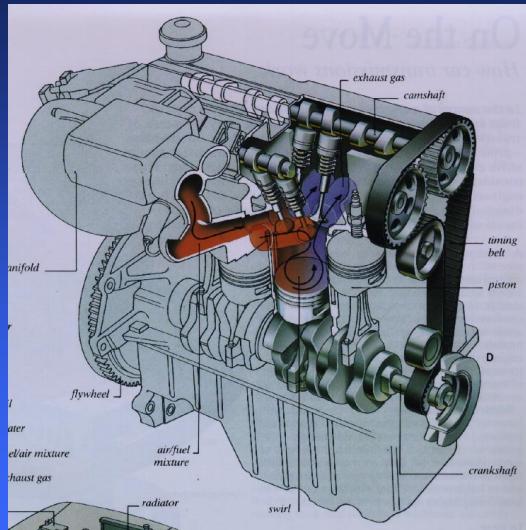
Image from "The Way Science Works".
Courtesy of Macmillan Reference USA, a division of Ahsuog, Inc.

Technical Illustration

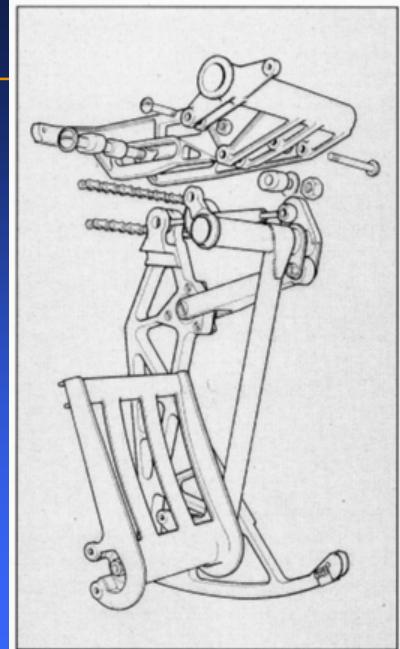
Shape information is most valued

- Edge lines
- Shading
- Shadows rarely included
- One light

Illustrators Use of Lines



From *The Way Science Works*,
Courtesy of Macmillan Reference USA.



From *Technical Illustration* by Judy Martin

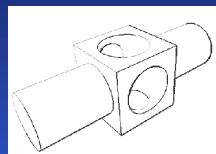
Level of Abstraction

- Abstraction is not just simplifying an image or eliminating detail
- It is focusing on specific details
- Technical Illustration occupies the middle ground of abstraction
 - *important 3D properties are accented*
 - *extraneous details are diminished or eliminated*

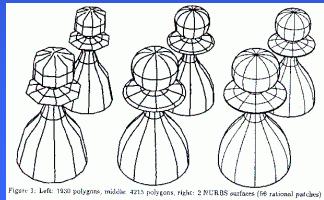
See Understanding Comics, by Scott McCloud, 1993

Background

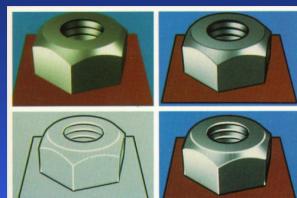
Markosian et al.
1997



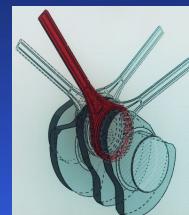
Elber
1990



Saito et al.
1990



Dooley et al.
1990



Winkenbach et al.
1996

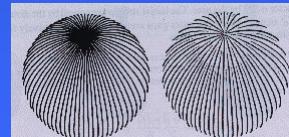
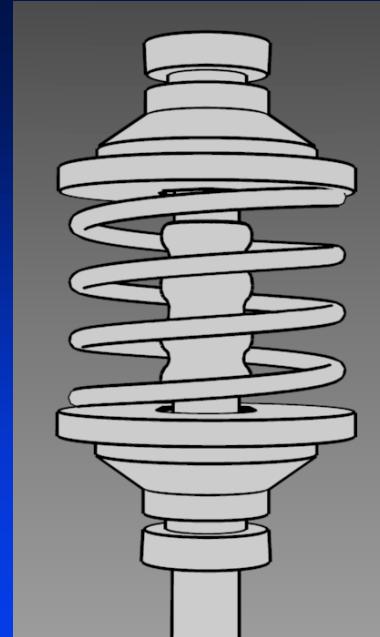
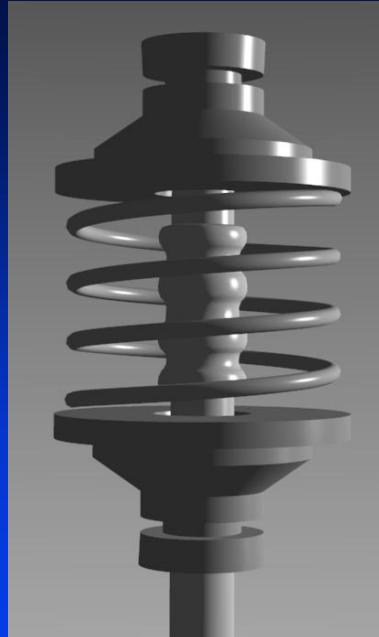


Figure 2: Left: 1920 polygons, middle: 4215 polygons, right: 2 NURBS surfaces (16 rational patches)

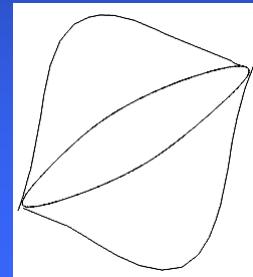
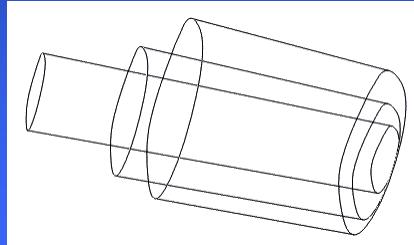
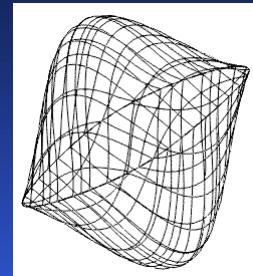
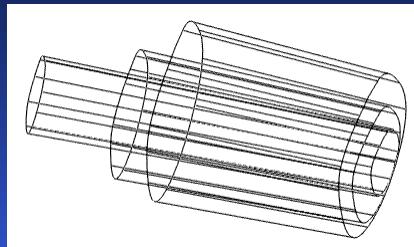
Perception

Recognition: shaded images vs. line drawings

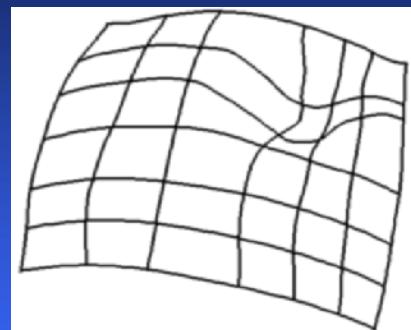
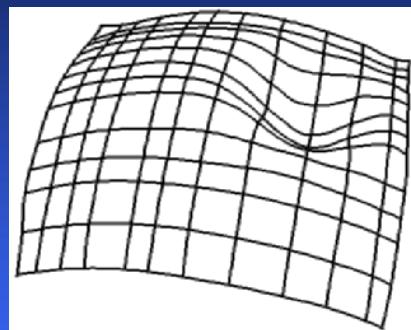
- Biederman et al., *Cognitive Psychology*, 1988
- Braje et al., *Vision Research*, 1995
- Christou et al., *Perception*, 1996
 - “*a few simple lines defining the outline of an object suffice to determine its 3D structure*”



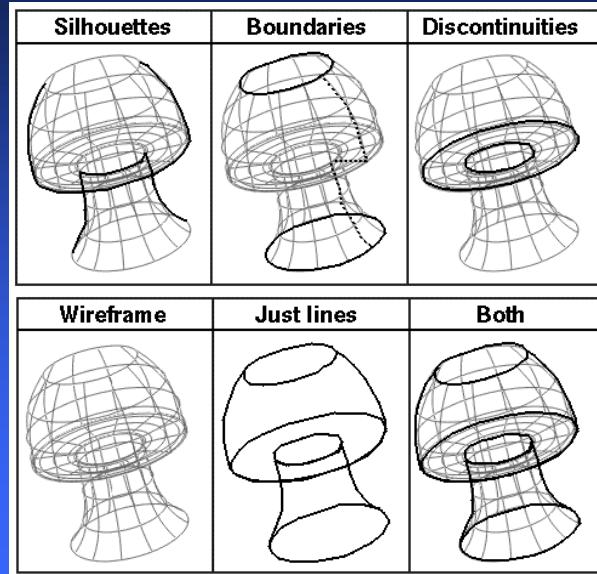
Wireframe versus Edge Lines



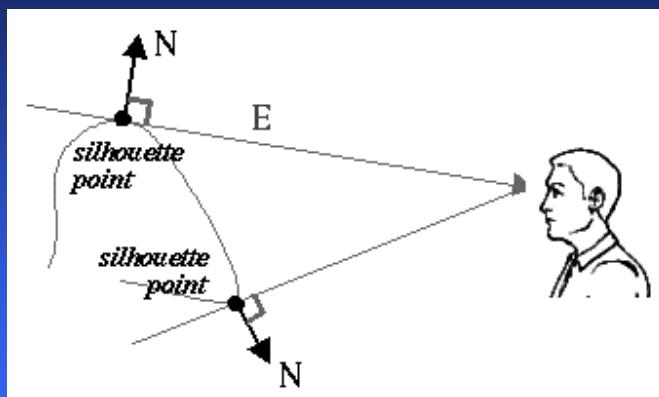
Changing Isolines



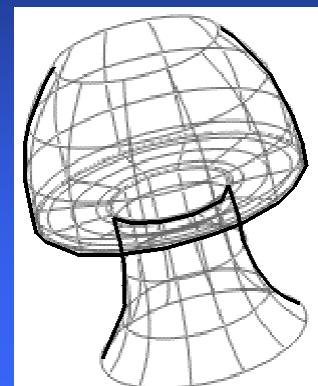
Imitate Illustrators Lines



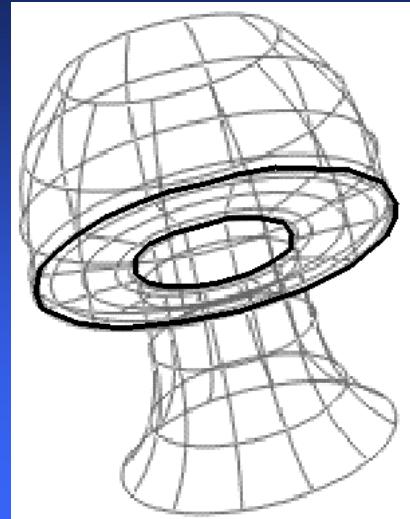
Define Silhouettes



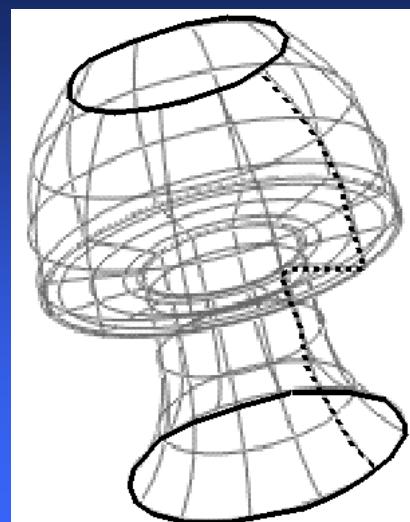
Example:



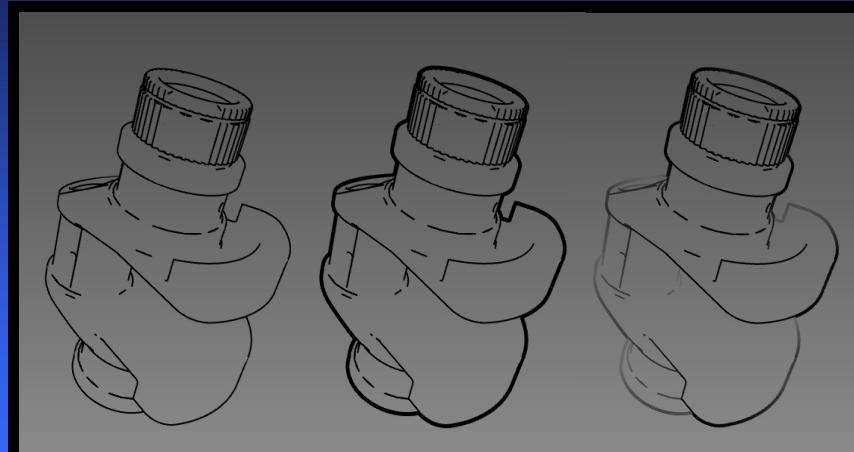
Discontinuities and Creases



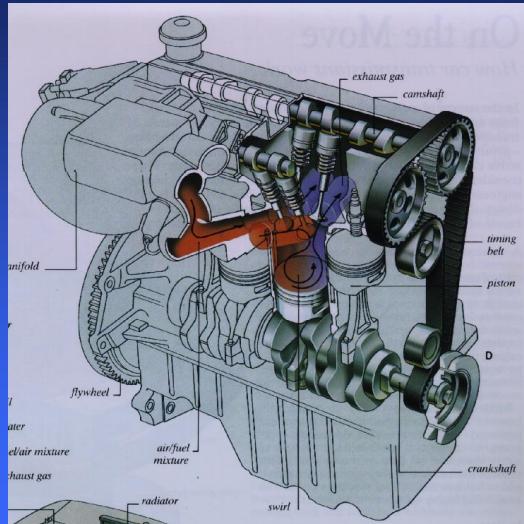
Surface boundaries



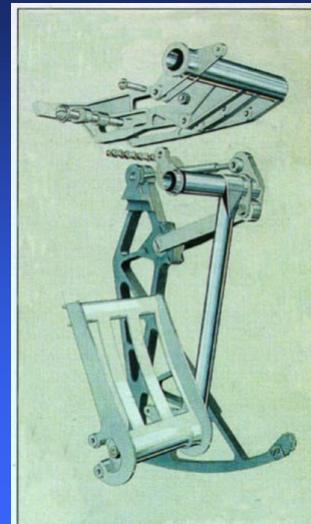
Line Width and Style



Shading in Technical Illustration



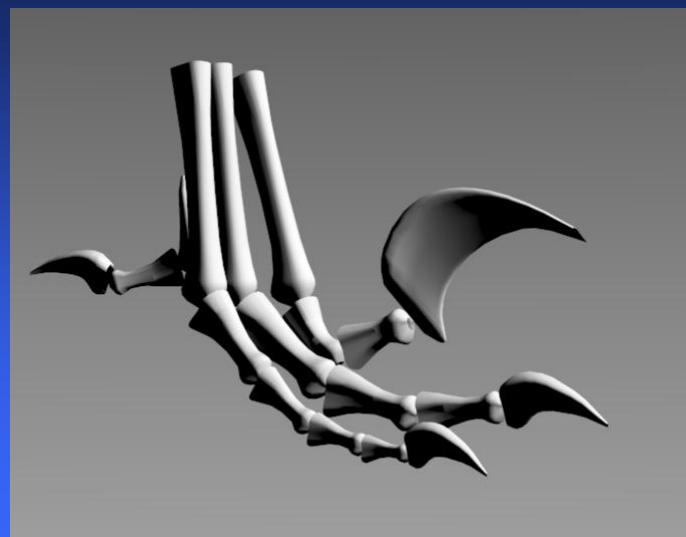
From *The Way Science Works*



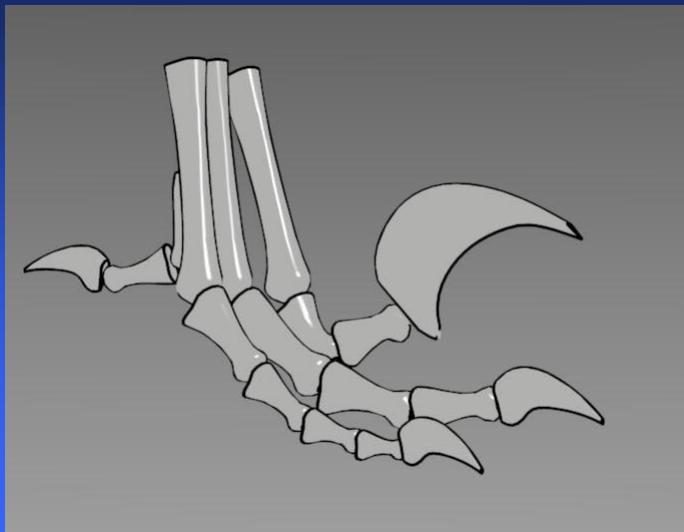
From *Technical Illustration by Judy Martin*

Diffuse shaded model

$$I = k_d k_a + k_d \max(0, L \cdot n) \text{ with } k_d=1 \text{ and } k_a = 0.$$



Just Highlights and Edge Lines



Hand-Tuned Phong Shaded Image



State of the Art

- Shading model is insufficient
- Lost shape information
 - *especially in the areas of subtle curvature
(ie. the small claws)*
- Not automatic, lots of hand-tuning

Smallest Effective Difference

***Tufte's design strategy of
the smallest effective difference :***

***"Make all visual distinctions as subtle
as possible, but still clear and
effective."***

From *Visual Explanations*, by Edward Tufte

Smallest Effective Difference

- To include shading in an image with black edge lines and white highlights
 - use a compressed dynamic range for shading
 - use colors visually distinct from black and white

Test your vision

In the next slide we provide a perceptual reason for using yellow and blue to provide depth.

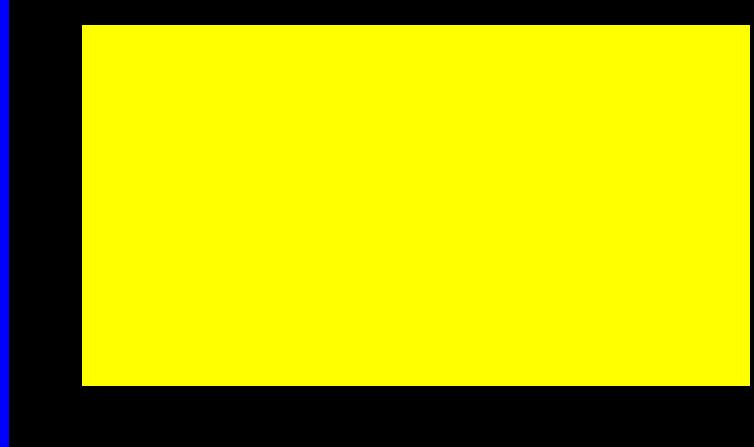
- Create a yellow box
- Within the yellow box draw a black box
- Within the black box draw a blue box
- Allow your eyes to relax & look at the boxes



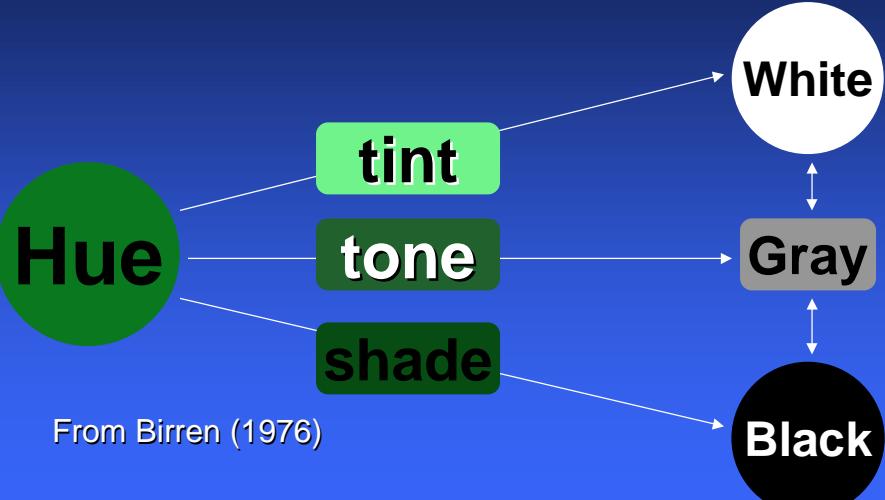
Test your perception

You should see that the blue box seems to fall to the background.

- Try reversing the blue & yellow boxes...



Tints, Tones, and Shades



Shading used by Artists

Complementary Shading

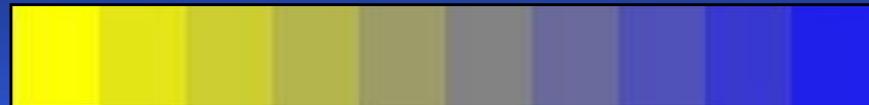


Image courtesy of Susan Ashurst

Using Color Temperature

Examples of Warm to Cool Hue Shift

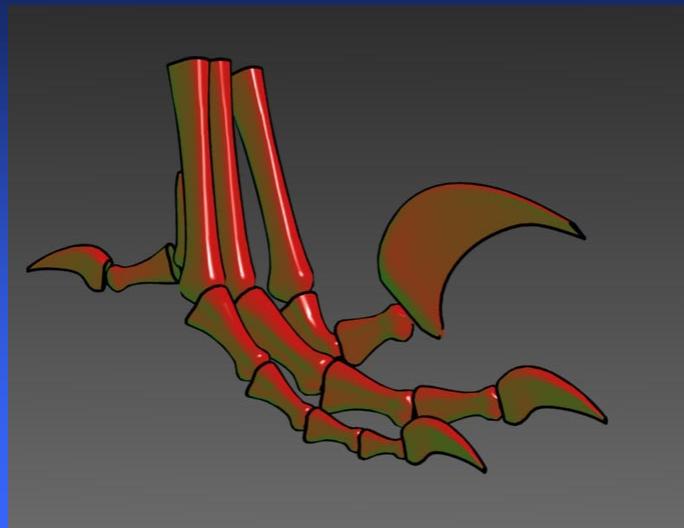
Yellow to Blue



Red to Green



Constant Luminance Tone Rendering



Creating Undertones

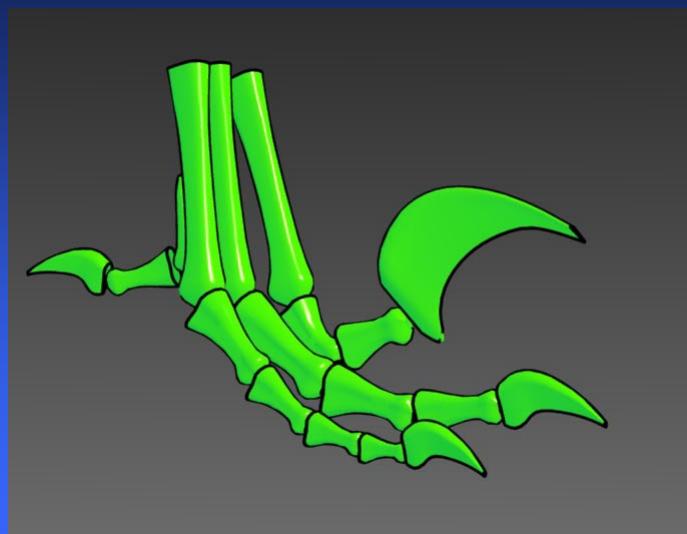
Warm to Cool Hue Shift



Green with Warm to Cool Hue Shift (undertone)



Model tone shaded with cool to warm undertones



Combining Tones with Undertones

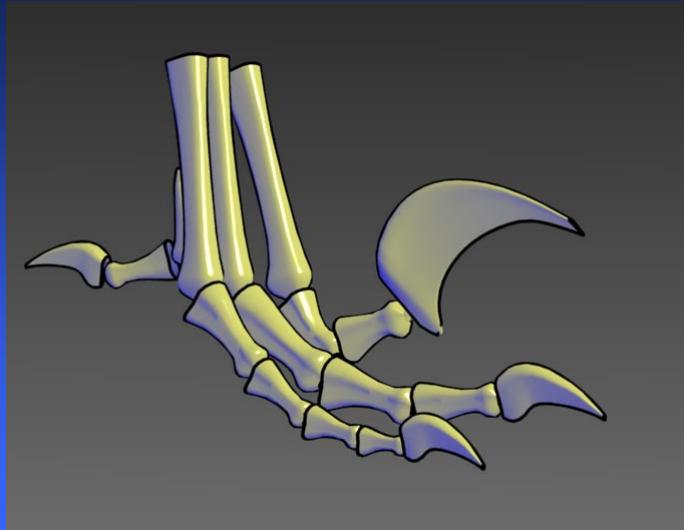
Green with Tone and Undertone



Model shaded with tones and undertones

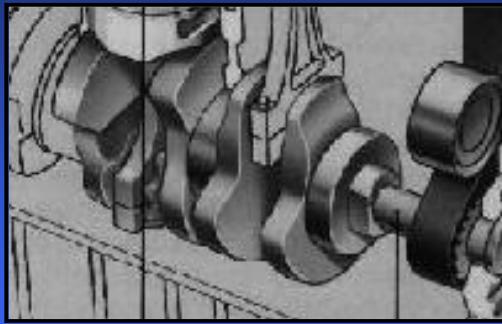


Undertones on Gray Model



Metallic Objects

Illustration



From *The Way Science Works*,
Courtesy of Macmillan Reference USA.

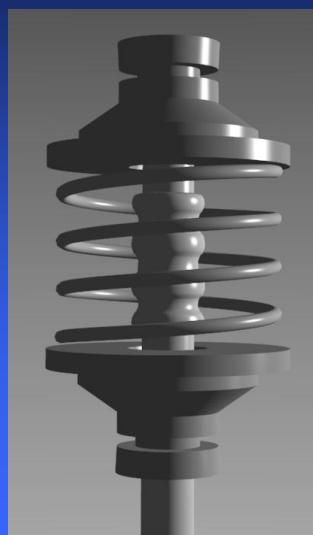
Photograph



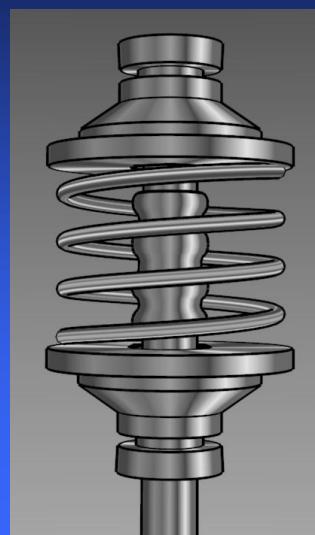
Courtesy of Sam Drake.

Imitating Material Properties

Phong shaded

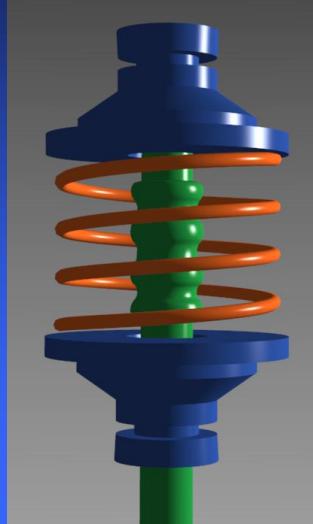


Metal shaded

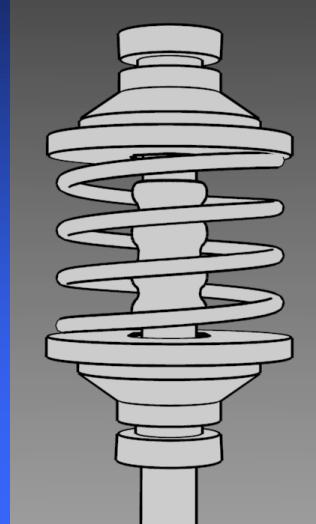


Results

Phong-shaded

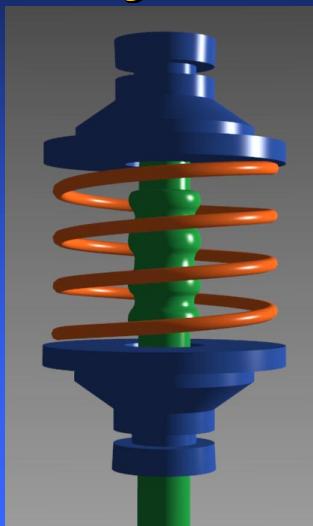


Edge Lines Only

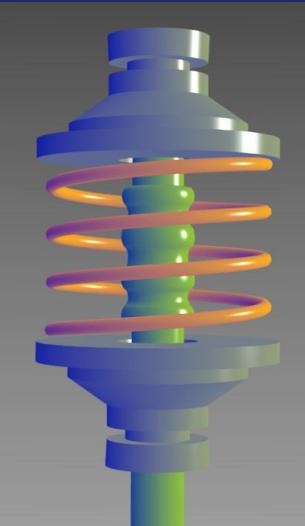


Results

Phong-shaded

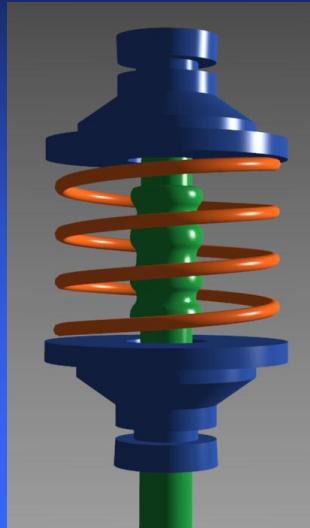


New Shading Without Edge Lines

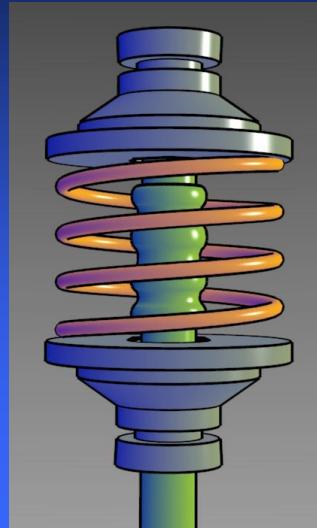


Results

Phong-shaded



New Shading With Edge Lines



Future Work

- Non-linear shading model
- Incorporate more illustration conventions
 - *different line weights and styles*
 - *object transparency and cut-a-ways*
 - *other material properties*
- Automate other illustration forms

Thanks

- Peter Shirley, Elaine Cohen
- University of Utah Computer Graphics Groups
- Dan Kersten, Dept. of Psychology, University of Minnesota
- Support provided in part by
 - DARPA (*F33615-96-C-5621*)
 - *NSF Science and Technology Center*



9 Non-Photorealistic Animation

Cassidy Curtis
Pacific Data Images

Abstract

A practical explanation of how to approach an animation production that calls for non-photorealistic imaging (NPI). To make things concrete, we will walk through the different key stages of NPI-based character animation, using some of the author's own projects as examples. The focus will be on the ways in which NPI requires alterations to the now-standard "traditional pipeline" for CG animation.

Audience: animators, technical directors, art directors, producers, production managers, and software developers.

Take-away knowledge: some guidelines for applying non-photorealistic techniques to an animation production, with lots of examples from real projects.

Introduction

Non-photorealistic animation may be quite new to the graphics community, but author Vladimir Nabokov seems to have anticipated the concept by decades. In his 1939 novel *Laughter in the Dark*, the main character, an art collector, is inspired by a "beautiful idea":

It had to do with colored animated drawings -- which had just begun to appear at the time. How fascinating it would be, he thought, if one could use this method for having some well-known picture, preferably of the Dutch School, perfectly reproduced on the screen in vivid colors and then brought to life -- movement and gesture graphically developed in complete harmony with their static state in the picture . . . and the colors . . . they would be sure to be far more sophisticated than those of animated cartoons. What a tale might be told, the tale of an artist's vision, the happy journey of eye and brush, and a world in that artist's manner suffused with the tints he himself had found! [NAB89]

It certainly is a beautiful idea, and one that still holds currency today. In the time since Nabokov's book, traditional animators have experimented with various techniques for bringing paintings to life. Alexander Petrov is known for the rich, hand-painted look of his oil-on-glass animations such as *The Cow* and *The Mermaid*. Another example is Joan Gratz, whose lighthearted film, *Mona Lisa Descending the Staircase* (1992) deftly morphs through dozens of works of modern art including Picasso's nudes and Van Gogh's self-portraits, always maintaining a painterly quality thanks to her innovative "claypainting" technique.

Computer animation is only beginning to catch up to the variety of styles found in traditional and experimental animation. Non-photorealistic rendering (NPR) is a step in the right direction, but like any tool, it can be used skillfully or clumsily. Nabokov's character articulates this perfectly:

...the designer would not only have to possess a thorough knowledge of the given painter and his period, but be blessed with talent enough to avoid any clash between the movements produced and those fixed by the old master: he would have to work them out from the picture -- oh, it could be done.

As recent work shows, it can indeed be done. But how? The existing literature on NPR techniques provides plenty of implementation details, but says little about how they might be used in production. The natural assumption is that an NPR algorithm can simply be tacked on to the end of the traditional CG pipeline as a post process, and that everything will work just fine. In my experience, this is almost never the case.

Animation requires a lot of design work and planning. Many of the most important decisions are made long before a finger touches a keyboard. If a project calls for a non-photorealistic look, it is essential to consider that look in every stage of pre-production and production, rather than trying to "fix it in post".

In this document, I hope to provide a few guidelines for using existing NPR techniques for animation, and for developing new techniques for specific projects. I will walk through some of the key stages of character animation production, and show examples of how an NPR technique can induce substantial changes in their planning and execution. The examples are drawn primarily from three short animations I have worked on: *The New Chair*, *Fishing*, and *Brick-a-Brac*.

Since each stage is typically the responsibility of a different person on a large production, each of the sections below is geared toward a different audience: art directors, technical directors, software developers, and producers. However, since there is often some overlap between these roles, it is worth reading beyond your own area to get a sense of the big picture.

Context

There are two terms used throughout this document: *non-photorealistic imaging* (NPI), which refers to the goal of creating images by any means that resemble some medium other than photography, and *non-photorealistic rendering* (NPR), an umbrella term comprising a set of digital techniques for achieving that goal. The distinction between the two is relevant in the context of animation production, since many animations these days employ a combination of digital and hand-drawn techniques.

A feature common to most current NPR methods is the use of some type of *screen-space marks* to construct an image. These marks may be small relative to the image being rendered, as in the case of pen-and-ink lines [SAL97] or oil paint brushstrokes [MEI96], or they may occupy nearly the entire image, as in the case of a watercolor wash [CUR97].

Screen-space marks, as their name implies, must obey rules relating to the two-dimensional space of the image. They may optionally also represent some aspect of a three-dimensional model, but they are fundamentally two-dimensional objects. An important distinction between screen-space marks and the two-dimensional polygons used in techniques like the Z-buffer is that the former are not necessarily projected from a three-dimensional original.

Section 1. Defining a visual goal

If computer animation is still a young medium, NPR is in its infancy. At this early stage, it's hard to know how any technique will look in motion because most have never been tried before. In this sense, every non-photorealistic animation project is a new experiment.

The greatest threat to the success of such an experiment is vagueness of purpose. I have seen many NPR animation projects revert to a traditional CG look, or fail entirely, simply because they were started without a clear visual goal in mind. Directors and animators can become locked in a pointless cycle of revisions, tweaking small parameters to little effect: "Can you make the lines thinner? Hmm, that's no good. Try making them thicker... No, that's still wrong. I guess this look just won't work after all." Such a situation is frustrating for everyone involved.

The key to avoiding such pitfalls is *art direction*. Art direction simply means determining how a project should look. Strong art direction depends mainly on two factors: vision and communication.

Vision

The art director has to have a vision. By this I don't mean some kind of supernatural revelation, but simply a clear mental image of what the finished product should look like, down to the tiniest detail.

For a non-photorealistic project, this means thinking of a certain class of details that aren't necessarily present in photorealistic CG. These are the specific qualities that make a painting look like a painting, a drawing like a drawing, and so on. They are also the details that give a particular image a "style". They include:

Texture of substrate:

Is the image drawn on paper, or painted on canvas, or scratched on the wall of a cave?
Every substrate has its own texture.

Type of medium:

Oil paint? Pencil? Watercolor?
Each medium has certain telltale signs that distinguish it from others. [CUR97]

Geometry of screen-space marks:

Are the marks short or long?
Fat or thin?
Curved or straight?

Character of marks:

Loose or tight?
Rough or smooth?
Calm or energetic?

Texture of marks:

Transparent or opaque?
Matte or glossy?
Smooth or granular?

Perceptual function of marks:

Do they represent outlines, highlights, or shadows?
Do they convey surface orientation or texture?
Do individual marks represent entire objects?

Semantic function of marks:

Do the marks express qualities not literally found in the subject, such as an emotional state?

In addition to these *static* qualities, the fact that the image will be animated brings on the need to consider its *dynamic* qualities. This is where things get most experimental: With current technology it's not yet possible to "sketch out" a flock of moving brushstrokes to see how they'll behave, so intuition and imagination are crucial. The dynamic qualities include:

Coherence of motion: If a mark persists from one frame to the next, it is said to be *coherent* over time. If its position is the same for consecutive frames, it will appear to stand still; if it changes, it will appear to move. The important question is not whether coherence is necessary, but what *kind* of coherence is desired.

- Coherence with the canvas?
- Coherence with an object's surface?
- Coherence with outlines, highlights or shadows?
- Or none at all?

Character of motion: Smooth or rough?

Perceptual or semantic function of motion: Does the mark's motion represent the motion of an object in the scene? Does it represent the motion of artist's hand? Or does it express some non-literal quality?

Change of other qualities over time: Is the size, shape, color, and orientation of each mark static over time, or can some of those qualities change? If so, do they change smoothly or abruptly? Does the change have a perceptual or semantic function?

These properties will all be plainly visible to the viewer, whether you've chosen them or not. For this reason, it's essential to consider them all, even the ones that don't seem relevant at first glance. Imagine what it would look like one way or the other, and decide which is more appropriate. When in doubt about a particular feature, it's always good to check it against the story content, and to ask: Does this feature fit the content at this point? What might fit better?

You need to consider these properties separately for each character, prop and environment in every scene. What works for one subject may not work for them all. If there are important distinctions between characters and background, or between scenes that have different moods, try to identify which properties make those distinctions clear.

What you're doing is essentially *aesthetic problem-solving*. Ultimately, your job will entail a balancing act between the rules of composition [GLA98] for a static image, and the rules of animation [THO81] and cinematography [CAL96]. But given the directive to achieve a certain look in every frame while allowing motion in the scene, there may be multiple solutions, or there may be none at all! In the latter case, it's necessary to go back and analyze the initial directive, and try to ferret out what's really behind it.

For example: in *Brick-a-Brac*, the paper texture was originally meant to make the animation look as if it had really been drawn on paper. To mimic the entire animation process faithfully, this would have meant changing the texture from frame to frame, since each image would have had to be drawn separately. However, doing this caused a distracting amount of noise when the results were played back. The richer I made the texture, the 'louder' the noise became -- but without the texture, it no longer looked 'real' enough. This was an unacceptable aesthetic compromise.

Looking back at my own motivation, I realized that my real goal was not to fool people into believing that the animation was done by hand, but rather to acquaint them with the characters in the story. So I chose

to keep the texture prominent but static from frame to frame. I felt like I had sacrificed the "handmade" illusion to protect the story. But much to my surprise, when I started showing the piece, many viewers still thought I had done it by hand.

This particular problem seems easy enough to resolve. But as the look of a piece becomes more complex, the coherence-related dilemmas tend to grow in number. The crucial step is to look at each such question as it comes up, and make an aesthetic choice that's consistent with and motivated by the story.

Communication

A clear vision in and of itself is not enough to get the job done. You have to communicate that vision to the rest of the team-- again, supplying all of the necessary details. Here are some suggestions:

Show pictures. Art books, magazines, and illustration annuals are terrific sources of reference material. Better still is to paint or draw the pictures yourself, if you have the skills to do so.

Show animations done using other techniques. Experimental animators have solved a huge number of aesthetic and technical problems, and there is a lot that you can learn by watching them frame by frame. *The Animated Film Collector's Guide* [KIL97] is a useful index for finding experimental short animations.

Point out the features that interest you in each piece of reference material. For example, you may want the line quality from a certain drawing, the color palette of a particular painter, and the motion style of a Joan Gratz animation.

Do tests using the software tools you already have, if time and budget allow. If new software gets developed for your project, use the early tests to illustrate what you do and don't want to see.

As a last resort, if no pictures are available, **use words.** This can help the artists on your team visualize what you mean, even if they can't see what you see.

Figures 1 and 2 are reference images from *The New Chair*. Figure 1 shows some drawings I used for line reference. My goal was to emulate the looseness and hastiness of these lines, if not their literal shape. Figure 2 (color) shows a collage I made to provide reference for texture and color palette.



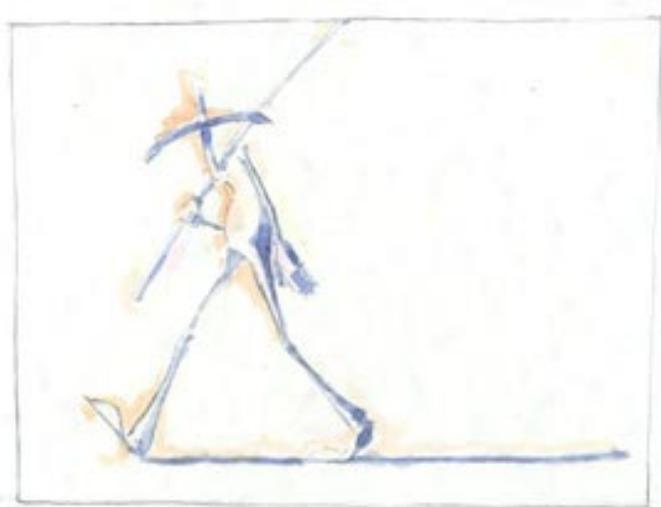
1. Line reference for *The New Chair*.



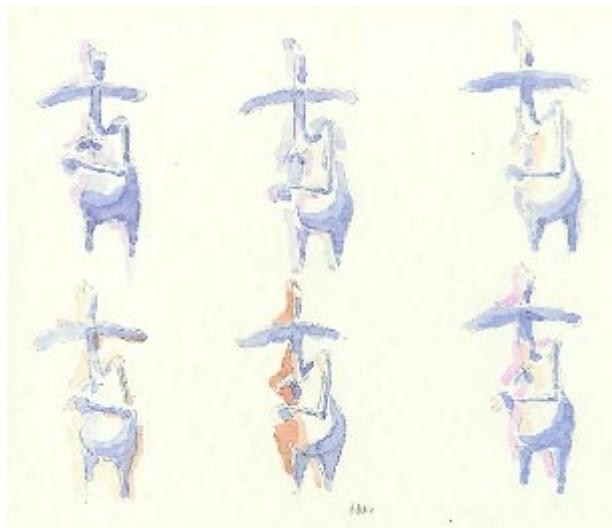
2. Color and texture reference.

Figures 3-5 (color) show some reference images from David Gainey's film *Fishing*, for which I provided a watercolor look. Gainey is an accomplished watercolorist, and was able to paint me a set of pictures that showed exactly how he wanted the finished product to look, in every respect. He pointed out the functions of the different marks he had used, showing how some conveyed shading while others acted to fill negative space. He also described how the color palette should change over the course of the film, to

convey the changing time of day. This is the best art direction one can hope for, and it is also the greatest challenge!



3. Hand-painted reference image by David Gainey for Fishing



4. Color reference for different scenes.



5. Line reference, with director's comments on the functions of the different marks.

On that note, it's important to prioritize (unless you have an infinite budget.) Choose which properties *define* the style, and which ones don't matter as much. In the case of *Fishing*, the high priorities included random variation in line thickness, and the use of color to indicate time of day. In *The New Chair*, it was most important to provide a way to vary the line style according to the character's mood.

A final word about the timing of this process: Don't think that it's necessary to have your vision worked out to the tiniest detail before the project begins. Obviously, the more decisions you can make in advance, the clearer the task will be for the rest of the team. But every production has its little surprises. You may find that the work your team produces pulls you in a direction that you couldn't have imagined ahead of time. Your vision should continue to become clearer and more refined throughout production until, hopefully, it is matched by the finished piece.

Section 2. Defining the problem space

Once a visual goal is clearly defined, someone, usually a lead technical director, has to translate that goal into something that a team of animators can accomplish. This may require developing some new software, or it may mean simply making clever use of tools you already have.

At this point, you should ask yourself the following questions:

Is CG really the right tool for the job?

The benefits of CG:

- Free inbetweening, thanks to keyframe animation.
- Free perspective, thanks to 3D.
- Unlimited revisions.
- An economy of scale. (Imagine painting a watercolor every frame!)

The drawback:

- Setup is costly. A long time will pass before you see a single image.

What visual properties are most important for the look? Try to get a sense of the art director's priorities. You need to be sure that you understand all of the terms, so study up on your art-speak. If you have no art background, a good way to begin catching up is by reading the notes from the 1998 SIGGRAPH course, "Art for Computer Graphicists". [GLA98]

How much variation is there from shot to shot, or from subject to subject? Which properties vary the most? Try to anticipate the range of styles you'll need to simulate.

What are the skills of the artists on your team? For example, if they have a lot of painting experience, perhaps a tool with a painting interface would be more effective than one that requires text input.

Is it possible to accomplish all of the visual goals using your current tools? If not, what new tools need to be developed? Would some additional software make the job substantially easier? Easier for whom?

The end product of this stage should be (a) a specification for any software tools that need to be developed, and (b) a rough idea of the pipeline through which each shot will be pushed into production.

Example 1: Fishing

For *Fishing*, the director's specification for the look could be broken down to the following criteria:

Static properties:

Substrate: rough watercolor paper.

Medium: transparent watercolor paint.

Geometry of marks: mostly long, thin lines, with a few wider brushstrokes.

Texture of marks: both hard and soft edges, very rough and messy.

Perceptual function: different marks may indicate body parts, shading and shadows, negative space, fish, or ripples in water.

Function of density of marks: indicates shading and/or depth.

Dynamic properties:

Coherence of paper: paper should move with environment, but should not change between frames.

Coherence of marks: marks should move with character, but outlines should change in an erratic way, so explicit coherence is not crucial.

Change in color: smooth transitions indicate changing time of day.

Was CG the right tool? Yes. The director's painting skills and craftsmanship were definitely up to the task of creating this animation by hand. However, he felt the end product would be much better if he could take advantage of the perspective and keyframe animation benefits of working in CG. Also, the project could then take place in an environment already streamlined for animation production, and make use of the skills of other CG animators.

Was additional software necessary to achieve the watercolor look? No. A heavy simulation approach to watercolor rendering [CUR97] would have been complete overkill in this case. The main benefit of such a simulation is the complex interaction between different flows of water and pigment within a reasonably-sized region. But in this case, the brushstrokes were all too narrow to show such effects. The edge-darkening effect could easily be faked using edge-detection and other filters available in a generic image processing toolkit, and the textures could be achieved with clever application of procedural noise.

Example 2: The New Chair

For *The New Chair*, my initial criteria were somewhat less clearly defined, because the project grew organically from what was originally a simple motion test. By the time it became a full-fledged animation project, however, I had come to some clear decisions about how I wanted it to look:

Static properties:

Substrate: crumpled paper.

Medium: ink lines, with some kind of color to fill shapes.

Geometry of marks: long, thin lines.

Texture of marks: solid black.

Character of marks: varies greatly between subjects.

Perceptual function of marks: all marks indicate silhouette edges, with different line styles indicating different surface textures.

Semantic function: style of marks can indicate mood of character (excited, angry, dizzy, etc.)

Dynamic properties:

Coherence of paper: paper should move with environment, but should not change between frames.

Coherence of marks: marks should move with character, but some looseness of motion is desired, so again explicit coherence is not crucial.

Change in line style: the style of the marks should change according to the main character's mood.

Was CG the right tool? Yes. Even if I had started from scratch, it would not have been possible for me to render this animation entirely by hand, because my drawing skills were not consistent enough to achieve the kind of subtle motions I wanted from the character. Working in a 3D keyframe-based system enabled me to refine the performance to the level I needed.

Was additional software necessary to achieve the look? Yes. It would not have been possible to achieve such a broad range of line styles using a simple image processing package. I needed some kind of tool that could create drawings in a wide variety of styles based on the 3D models generated by the animation package I was using.

Would this change the animation and rendering pipeline? Definitely. In Section 4, you will find a detailed explanation of what became easier and what became more difficult as a result.

Section 3. Writing usable tools

If you are going to need new software, it's important to keep it simple. It may be impossible to write a tool that automatically replicates the style of a Van Gogh. But if you can provide a simpler tool that lets a skilled user do the job pretty quickly, then you're on the right track.

The goal should be to reduce the tedium the users experience. This is not the same thing as reducing the amount of work they have to do! This is a very important distinction. If you eliminate work at the cost of making the interface obscure or confusing, you will cripple your users. But if you make the work engaging, it will go quickly and the end product will be vastly better.

The computer should make only the decisions that would be too tedious for a person to do by hand. Placing thousands of brushstrokes for every frame of a film is a task that clearly falls in this category. But the computer does not have to make all of the decisions. Of the properties listed in the previous chapter, certain ones lend themselves to automation, while others do not. The most abstract properties, such as the emotional expressiveness of a mark, are naturally the most difficult to automate. Luckily, they can also be the most fun to set by hand.

In cases where there are many dimensions of variation to the style, it may be difficult or even impossible to write an algorithm that is guaranteed to work under all possible inputs. If that's the case, don't bang your head against the problem for too long. If it's possible to screen for bad combinations of parameters, do so. But if that proves too complex a task, it's probably better to risk the occasional bad frame than to impose too narrow a restriction on the range of possible styles. A good TD can always find a way around a glitch.

In the case of *The New Chair*, the right implementation turned out to be a simple image filter that turned a depth map into a line drawing. (See Appendix A for a brief description of the technique.) The interface was text-based, a compromise between ease of development and ease of use. (Since the user in question was a programmer and not much of a painter, this was an appropriate way to go.) Figures 6 and 7 show examples of the simple formats used.

```
##  
## "shaky" drawing style:  
## thin, rough, multiple strokes--  
## makes things look like they're shivering.  
##  
precision    20  
erasure      0.25  
drag         0.4  
drunk        1.0  
min          1.0  
max          1.5  
rough        0.1
```

6. A text file describing a drawing style for the "Loose and Sketchy" filter.

```
##  
## Ergoman becomes angry at the chair.  
##  
path /home/cassidy/newchair/styles  
  
306 normal  
315 angry  
321 angry  
325 normal
```

7. An animation file for interpolating different styles.

Section 4. Optimizing the pipeline

Using NPR in production affects every stage of the process, including art direction, design, modeling, shading, lighting, and even motion. Depending on the technique used, some of these stages may be made more difficult, while others will become simpler or even disappear entirely. Consider this carefully before you start to budget your team's time.

Task	Difficulty	No longer important:	Still important:
Modeling	Easier	Crashing surfaces Smoothness of normals Texture stretching	Appearance of silhouette
Surfacing	Much easier	Texture of all objects	Texture of paper
Lighting	Easier	Separation of planes Rounding of surfaces	Color palette Ground shadows
Animation	A little easier	Intersecting objects The third dimension	Weight Acting Personality Etc...
Compositing	Harder		New concerns: Animation of line styles Combining different styles Hidden-line removal Ordering of planes Combining color with lines

8. The relative difficulty of various tasks, after the "Loose and Sketchy" process is introduced.

Figure 8 shows a synopsis of how the "Loose and Sketchy" process changed the production pipeline for *The New Chair*.

The task of modeling the main character and all of the props was reduced from potential weeks to a few days, because it was no longer necessary to worry about intersecting surfaces, parametric texture coordinates, or smoothness. For example, the main character consisted of a handful of surfaces of revolution jammed together in space in such a way that they would have a nice silhouette. (Figure 9.)

Surfacing became almost nonexistent: I spent a few hours refining the paper texture, and that was all. For a stylized 3D look it would have been necessary to do at the very least *some* work on every single surface, if only to give the viewer's eye something to attach to. And for a photorealistic look, it would have taken substantially longer.

Lighting was likewise simplified. Only one shadow-casting light was needed, to create ground shadows for the character and props. Two fill lights were added to get the balance right. The entire process took a matter of minutes. Had the animation had a traditional CG look, much more time would have been needed: for example, I would have had to add "kicker" highlights to separate foreground from background.

The task of animating the character was made easier by the new look in two different ways. First, since the objects were rendered in different layers (see Appendix B for why this was necessary), it was alright to let them intersect each other. (Notice that in Figure 9, the character's leg goes right through his desk!) Secondly, the look actually made the character more appealing to look at, and easier to identify with.

Without the distracting qualities imparted by poor photorealism, it much easier to evaluate and improve the performance.

Really, the only task that got more complex was compositing. Appendix B describes the compositing process in detail. Integrating several different line styles into a single image required managing many layers, and sometimes switching the order of the layers mid-shot. This made the shell scripts I used for rendering substantially longer and more complicated than they might have been otherwise.



9. Models used in The New Chair.

Conclusion

Computer animation is a truly limitless medium: any wild idea that you can imagine can be rendered digitally... but it's probably beyond your budget. For those of you in the production world, I hope these notes provide a bit of help in making your wilder ideas more feasible. For the researchers and developers among you, I hope you will consider collaborating with animators in your future projects. Your research will go much farther if it is driven by an animator's aesthetic goals.

Appendix A: Loose and Sketchy method

The "loose and sketchy" filter automatically draws the visible silhouette edges of a 3-D model using image processing and a stochastic, physically-based particle system. For input, it requires only a depth map of the model (Figure A1) and a few simple parameters set by the user. First, the depth map is converted into two images:

The *template image* (Figure A2), in which each pixel represents the amount of ink needed in its immediate neighborhood. This image is obtained by calculating the magnitude of the gradient of the depth map, thresholding to give binary values, and then blurring the result.

The *force field* (Figure A3), a vector field that pushes particles along the silhouette edges. Such a field can be obtained by calculating unit vectors perpendicular to the depth map's gradient.



A1. A depth map.



A2. The template image.

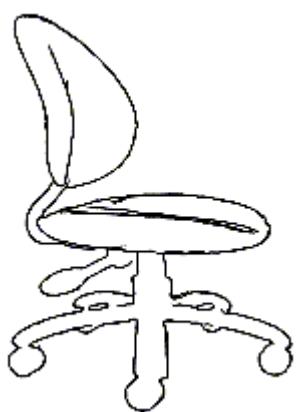


A3. The force field.

Next, particles are generated, one at a time, for a fixed number of particles. Each particle's initial position is chosen at random from within the template image, with a bias toward areas that need more ink. (No particles are ever born in areas that need no ink.) Acceleration at each timestep is based on the force field, with additional coefficients for randomness and drag. The particle is rendered onto the canvas as an antialiased line segment. If a particle wanders into an area that needs no more ink, it dies and a new one is born in another random place. The particle also erases the dark pixels from the template image as it travels, so that those edges will not be drawn again. (This aspect of the technique is similar to previous approaches to pen-and-ink rendering [SAL97, TUR96].)

Examples

By tweaking drag and randomness, a continuous gamut of styles can be generated ranging from a tightly controlled technical drawing (Figure A4) to a loose and gestural sketch (Figure A6). The higher the drag, the tighter the style. These parameters also affect the character of motion, which can range from near-perfect smoothness to a lively staccato. The looser styles are particularly appropriate for animated characters, giving them a bit of life even when they stand perfectly still. It is also worth noting that for animation, it is not necessary to exactly render all of the silhouette edges all of the time. An error in a single frame will usually go unnoticed, because each frame lasts only a fraction of a second.



A4. High drag, zero randomness.



A5. Moderate drag and randomness.



A6. Low drag and high randomness.

Figures A7-A11 show just a few of the other styles available by varying these and other parameters. Each of these images took only 10-60 seconds to compute.



A7.



A8.



A9.



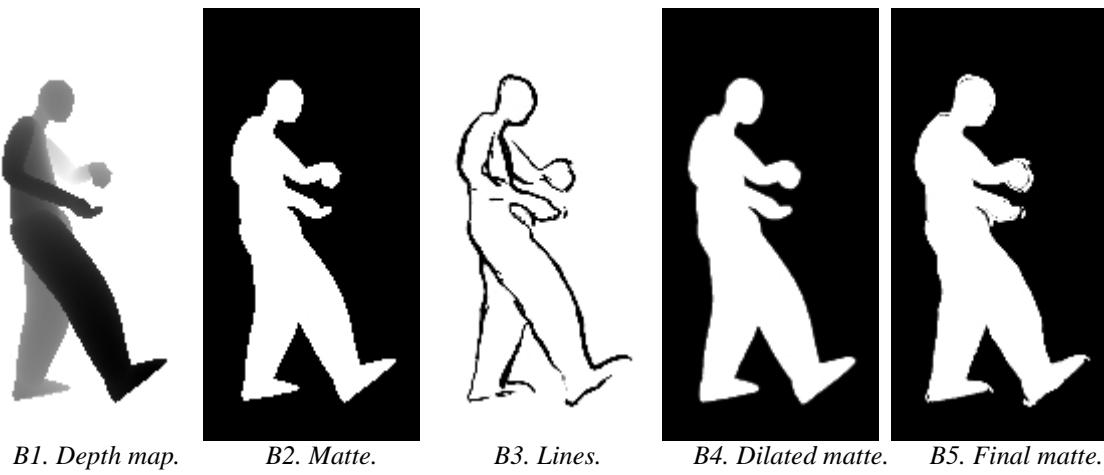
A10.



A11.

Appendix B: Compositing with Loose and Sketchy layers

1. For each object or group of objects that needs a separate style, create a *layer* that can be rendered separately. Then, for each layer, do the following:
 - a. Render a *depth map* (Figure B1) and a *matte* (Figure B2) for each layer.
 - b. Apply the loose and sketchy filter to the depth map, using a *style file* specific to that layer, to generate *lines* (Figure B3).
 - c. Dilate the matte by a few pixels, to cover for some sloppiness in the lines (Figure B4).
 - d. Combine the lines with the dilated matte, and let that be the matte for this layer. (Figure B5).
2. Once all layers have been rendered as line drawings with alpha mattes, composite them together in the proper order. The mattes will cause the upper layers to hide the lines of the lower ones. (Note that this order may change as objects change in proximity to the camera. In the case of "The New Chair", these changes in proximity were only relevant in certain scenes, such as when the character spins in the chair. In these situations, a shell script was modified to switch from one ordering to another based on the frame number.) (Figure B6)
3. Render the entire scene in full color. (Figure B7)
4. Desaturate and blur the colored scene. (Figure B8)
5. Multiply the blurry-color image by the fully-composited line image (Figure B9). Note that the colors appear to fill the lines completely despite the fact that the lines do not coincide with the edges of the objects. This works because our brains process color and edge information separately. Blurring the color image removes the high-frequency information that would cause the impression of a double edge.
6. Generate a paper texture using Perlin [PER85] and Worley [WOR96] noise (Figure B10). The paper texture changes from one environment to the next, but not from frame to frame, as this would be visually distracting.
7. Multiply the color and lines by the paper texture to produce the final image (Figure B11).

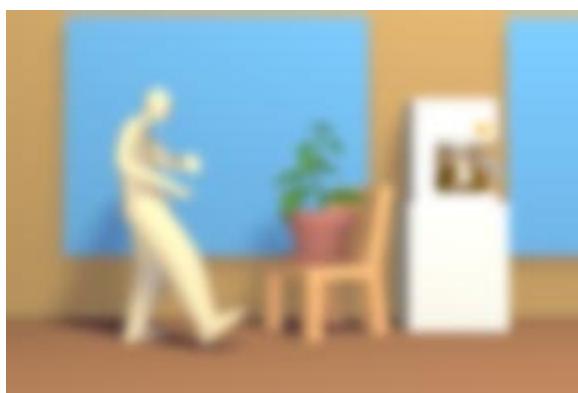




B6. Composition of all layers.



B7. Rendered color image.



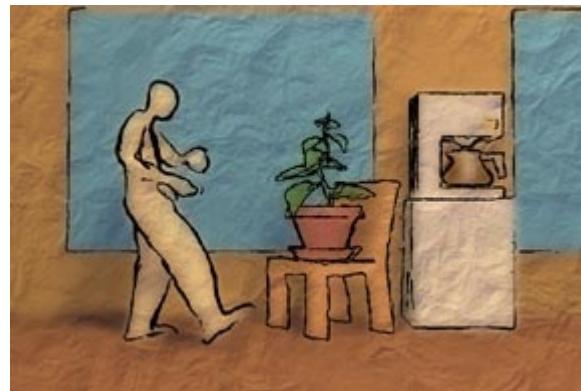
B8. Blurred color image.



B9. Combination of color and lines.



B10. Procedural paper texture.



B11. The final image.

References

- [CAL96] Sharon Calahan. "Pixel Cinematography", *SIGGRAPH 1996 Course #30*.
- [CUR97] Cassidy J. Curtis, Sean E. Anderson, Joshua E. Seims, Kurt W. Fleischer and David H. Salesin. Computer-Generated Watercolor, Proceedings of SIGGRAPH 1997, Computer Graphics Proceedings, Annual Conference Series, pp. 421-430 (August 1997). Addison Wesley. Edited by Turner Whitted. ISBN 0-89791-896-7.
- [KIL97] David Kilmer. *The Animated Film Collector's Guide*, John Libbey & Company, Sydney, Australia, 1997.
- [GLA98] Andrew Glassner, Barbara Kerwin, Jeff Callender, James Mahoney, and Mat Gleason. "Art for Computer Graphicists", *SIGGRAPH 1998 Course #30*.
- [MEI96] Barbara J. Meier. Painterly Rendering for Animation, Proceedings of SIGGRAPH 1996, Computer Graphics Proceedings, Annual Conference Series, pp. 477-484 (August 1996). Addison Wesley. Edited by Holly Rushmeier. ISBN 0-201-94800-1.
- [NAB89] Vladimir Nabokov. *Laughter in the Dark*, pp. 8-10, Vintage International Press, 1989
- [PER85] Ken Perlin. An Image Synthesizer, Computer Graphics (SIGGRAPH '85 Proceedings), 19 (3), pp. 287-296 (July 1985). Edited by B. A. Barsky.
- [SAL97] Michael P. Salisbury, Michael T. Wong , John F. Hughes and David H. Salesin. Orientable Textures for Image-Based Pen-and-Ink Illustration, Proceedings of SIGGRAPH 1997, Computer Graphics Proceedings, Annual Conference Series, pp. 401-406 (August 1997). Addison Wesley. Edited by Turner Whitted. ISBN 0-89791-896-7.
- [THO81] Frank Thomas and Ollie Johnston. *Disney Animation: The Illusion of Life*, Abbeville Press, New York, 1981.
- [TUR96] Greg Turk and David Banks. Image-Guided Streamline Placement, Proceedings of SIGGRAPH 1996, Computer Graphics Proceedings, Annual Conference Series, pp. 453-460 (August 1996). Addison Wesley. Edited by Holly Rushmeier. ISBN 0-201-94800-1.
- [WOR96] Steven P. Worley. A Cellular Texture Basis Function, Proceedings of SIGGRAPH 1996, Computer Graphics Proceedings, Annual Conference Series, pp. 291-294 (August 1996). Addison Wesley. Edited by Holly Rushmeier. ISBN 0-201-94800-1.

10 Interactive Non-Photorealistic Rendering

Bruce Gooch and Amy Ashurst Gooch
Department of Computer Science
University of Utah
<http://www.cs.utah.edu/>

10.1 Introduction

Most of the work in NPR has been static 2D images or image sequences generated by a batch process. In this part of the course notes we explore interactive NPR through the example of interactive technical illustration [4].

Work that has been done on computer generated technical illustrations has focused on static images and has not included all of the techniques used to hand draw technical illustrations. We present a paradigm for the display of technical illustrations in a dynamic environment. This display environment includes all of the benefits of computer generated technical illustrations such as a clearer picture of shape, structure, and material composition than traditional computer graphics methods. It also takes advantage of the three-dimensional interactive strength of modern display systems. This is accomplished by using new algorithms for real time drawing of silhouette curves, algorithms which solve a number of the problems inherent in previous methods. We incorporate current non-photorealistic lighting methods, and augment them with new shadowing algorithms based on accepted techniques used by artists and studies carried out in human perception.

An interactive NPR system needs the capability to interactively display a custom shading model and edge lines. In addition, this interaction must be possible for complex geometric models. In this part of the course notes we describe a variety of techniques for achieving these goals, and describe the tradeoffs involved in choosing a particular technique.

10.2 Making it Interactive

There are several new issues to address when creating 3D illustrations. Three-dimensional technical illustrations involve an interactive display of the model while preserving the characteristics of technical illustrations. By allowing the user to move the objects in space, more shape information may be available than can be conveyed by 2D images. Interaction provides the user with motion cues to help deal with visual complexity, cues that are missing in 2D images. Also, removing the distracting wireframe lines and displaying just silhouettes, boundaries, and discontinuities will provide shape information without cluttering the screen, as discussed previously in Section 8.

The question remains, “How do the 2D illustration rules change for a 3D interactive technical illustration?” Adapting the shading and line conventions presented previously in the course notes is fairly straightforward as long as the line width conventions have frame-to-frame coherence. The more interesting issues depend upon changing the viewer’s position versus moving the object. Since there are no protocols in traditional illustration, it may be best to model these 3D illustration conventions based on how you would move real object. This has an effect on how the light changes with respect to the object. The light position can be relative to the object or to the viewer. When looking at a small object in your hand, you turn the object and do not move your head, so the light stays in the same position relative to your eye. However when you move an object in a modeling program or when you look at a large part, the view

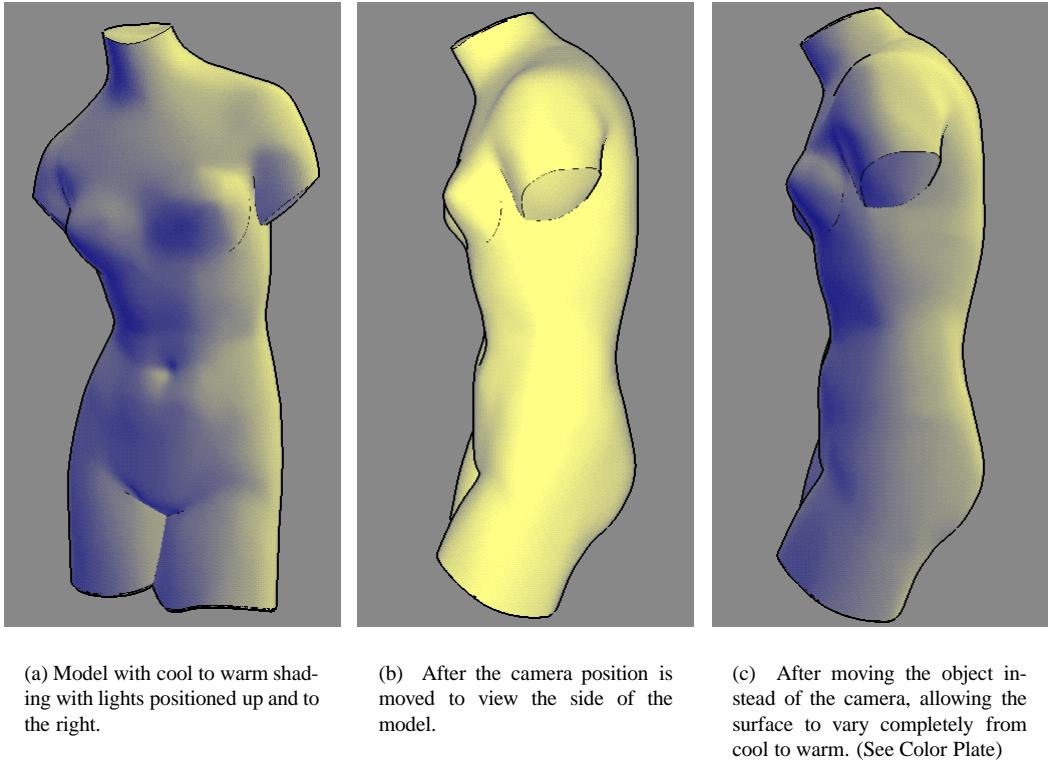


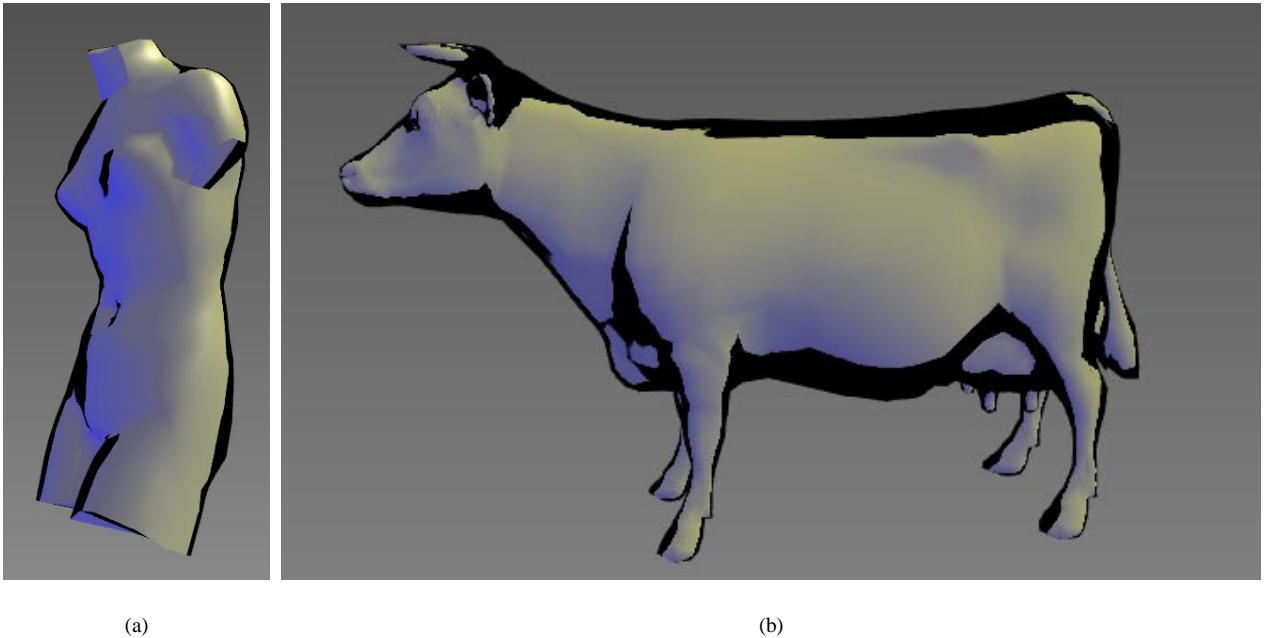
Figure 1: Frames from the NPR JOT Program [7], which uses Markosian et al.’s silhouette finding technique [8] and incorporates the OpenGL approximation to the shading model presented by Gooch et al [2].

point is actually moving, not the object. As shown in comparing Figure 1(b) and Figure 1(c), the shading model is used to its full advantage if the surface varies completely from cool to warm.

When illustrators light multiple objects, they may use a different shading across different objects, inferring that each object has its own light, which does not affect the other objects in the environment, similar to the “virtual lights” by Walter et al. [11]. For example, two objects in a scene may be lit differently to draw attention to different attributes of each object. If this was accomplished by adding two lights to the environment, the multiple highlights could be confusing.

Most material properties are semi-constant as the view direction or lighting changes. However the metal shading presented in Section 8 is the replication of the anisotropic reflection [5] due to the surface of the object and the reflection of the environment. When a real metal part is rotated in your hand, the banding does not stick to the object, but remains constant since the environment is not changing. However, in a non-photorealistic interactive environment it may be too jarring to have the metal shading changing abruptly. Using a metal texture might be more appropriate and a metal texture in an interactive environment would still convey the material property.

Another notion is to allow the relative size of the object to control the motion of the viewer, the object, and the light source in an interactive 3D illustration. In the end, allowing the user to choose whether the object moves or the eye point changes, as well as having control over the lights, may help the viewer gain the most shape information.



(a)

(b)

Figure 2: Adding the silhouettes to the environment map instead of calculating silhouettes from the geometry produces interesting artistic effects. (See Color Plate)

10.3 Displaying Important Edges and Silhouettes

Markosian et al. [8] have published real-time software algorithms for drawing edges. Their SIGGRAPH paper is included in these notes, and these methods were used to render Figure 1. Raskar and Cohen [10] have explored the uses of hardware to generate edge lines, similar to the methods we will discuss. We defer to their paper, included at the end of this section, for the details on their algorithms. Interactive technical illustration has also been explored for (Non-Uniform Rational B-Spline) NURBS surfaces [3], which we will not discuss here. To draw silhouettes, we have implemented several methods of extracting and displaying edge lines from polyhedral models, which we discuss in Section 10.3.1 and 10.3.2.

Methods for generating edge lines for polyhedral models can be roughly broken down into two categories. The first assumes no prior knowledge or preprocessing of the model and heavily leverages commodity graphics hardware. The second set of methods use preprocessing of the model and are purely software algorithms. Both hardware and software algorithms clearly have a place. The set of hardware methods are useful because of ease of implementation. The software methods are useful due to their flexibility and lower computational complexity. All of the software methods assume that the models are either manifold or manifold with boundary.

We can also extract boundary curves, edges adjacent to only a single face, and creases, that is, the edge between two front facing polygons whose dihedral angle is above some threshold. The user is provided the option to draw these edges, dependent upon the model and intent of the image. The computation and drawing of creases is discussed in Section 10.4.

10.3.1 Hardware Methods

Using multi-pass rendering [1] there are several ways to extract silhouettes. The algorithm presented in the SIGGRAPH 1998 OpenGL Course does not capture internal silhouette edges and requires four passes of rendering. Below we provide algorithms which require two or three rendering passes and capture internal silhouettes.

In a polyhedral model, a silhouette is an edge that is connected to both a front facing and a back facing polygon. The following is pseudo code for the basic algorithm:

```
draw shaded front faces
draw front faces in line mode:
    setting only stencil
draw back faces in line mode:
    setting color if stencil was set
    decrementing stencil if drawn
```

To draw lines over polygons, the `PolygonOffset` extension (or `PolygonOffset` function in GL 1.1) [9] is needed. This function effectively modifies the depth values of the first pass based on the slope of the triangles and a bias factor. This technique can create something similar to a silhouette, effectively a halo. The depth values are pushed forward instead of back to allow lines to be rasterized over faces. Then wide lines are drawn. Where there are large discontinuities in depth (silhouettes and boundaries), only part of the line is drawn. This method requires only two passes instead of the three listed above, but can be fairly sensitive to the parameters of the polygon offset function. Using OpenGL hardware makes the implementation simple, however, it limits the thickness of the edge lines.

Another hardware technique is to add the edge lines to a shading environment map as a preprocess. However, as shown in Figure 2(a), the lines lack crispness, and if the model varies greatly in curvature, there may be large black regions. In order to include silhouettes on the feet of the cow in Figure 2(b), we have to set the threshold low enough to draw lines in these high curvature regions. This causes regions which have relatively low curvature to be filled in with black. Although this effect produces some interesting, artistic results, it may be inappropriate for technical illustration.

10.3.2 Software Methods

A straightforward way to draw silhouettes is to explicitly test every edge in the model. We compute an edge structure based on the face normals of the model, which are also used for back face culling as in Zhang et al. [12]. An edge is a silhouette edge if and only if:

$$(\vec{n}_1 \cdot (\vec{v} - \vec{e})) (\vec{n}_2 \cdot (\vec{v} - \vec{e})) \leq 0,$$

where \vec{v} is a vertex on the edge, \vec{e} is the eye point, and \vec{n}_i are the outward facing surface normal vectors of the two faces sharing the edge. This situation only occurs when one face is front facing and the other is back facing. While this computation is simple, it can potentially become a bottleneck with large models. Since we have to shade (or prime the z buffer for hidden surface elimination) this computation can be done in parallel while the model is being rendered.

We use a more complex preprocess and search algorithm when classifying edges becomes a bottleneck. This algorithm is similar in spirit to Zhang et al. [12], but requires looking at arcs on the Gauss map instead of points. The Gauss map of an edge on a polyhedral model is a great arc on the sphere of orientations (Figure 3). Under orthographic projection, a plane through the origin in this sphere defines the view.

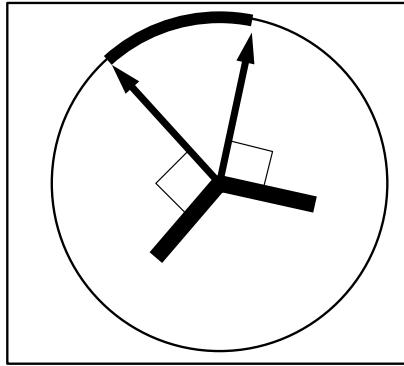


Figure 3: The arc in a Gauss map seen in 2D. The two bold line segments are faces that share a vertex. The orientations of their normals can be represented as points on the circle. The arc between those two points represents all orientations swept out between the two normals. In 3D the same reasoning applies, and the arc is an arbitrary segment on a great circle.

All of the faces on one side of the plane are front facing, and on the other side they are back facing. If the “arc” corresponding to an edge is intersected by this plane, it is a silhouette edge. To search for such edge/plane intersections, we store the arcs in a hierarchy on the sphere to quickly cull edges that can not be silhouettes. We have implemented a decomposition of the sphere starting with a platonic solid (octahedron or icosahedron) and all successive levels are four to one splits of spherical triangles. An arc is stored at the lowest possible level of the hierarchy. This makes silhouette extraction logarithmic in the number of edges for smooth models where the arcs tend to be short. One problem with this hierarchy is that the edges of the spherical triangles on the sphere interfere with the arcs and limit how far they can be pushed down the hierarchy. The probability of being stored in a leaf node that can contain an arc of a given length decreases as the size of the triangles shrink because the boundaries of these spherical triangles become denser as you recurse. An ad hoc solution to this problem is to use multiple hierarchies, whose spherical triangles are different, and store an arc in the hierarchy with the spherical triangle with the smallest area that contains it. A more attractive alternative would be to use “bins” on the sphere that overlap and/or making data dependent hierarchies.

Under perspective viewing, the region you have to check grows, based on planes containing the object and intersecting the eye. Building a spatial hierarchy over the model as in [12] would minimize this effect. One advantage of any software approach is that it makes it easier to implement different styles of line drawing.

10.4 Line Styles

As discussed in Section 8, line width can appear to change by either shading the lines based on the surface orientation, or by using OpenGL 1D texture mapping hardware to shade lines. A 1D texture map can convey the distance between a surface(edge) and a light in the scene.

Fat boundary lines can be drawn with either the software or hardware methods. These lines are drawn after the rest of the model has been drawn (shading, creases, silhouettes). While the earlier phases are drawn, they set a stencil bit, indicating that the given pixel has been draw for this frame. Finally, the boundary silhouettes are drawn over again with wider lines. In hardware this requires a full traversal of the front or back faces, while using software extraction algorithms only require a traversal of the silhouette edges which have been previously computed. All of these algorithms are more efficient than the methods mentioned in the OpenGL course [1] because it required four rendering passes while these algorithms

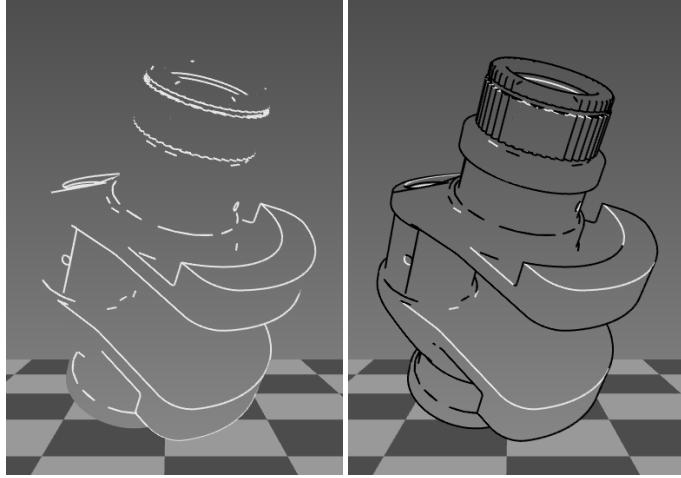


Figure 4: All creases are drawn in white (Left), and then all of the silhouette lines are drawn in black (Right), overlapping the creases.

Model	Faces	Edges	Naive	Gauss	Num Sil
S Crank	24999	35842	.027	.0198	3873
L Crank	169999	254941	.165	.096	12469
Sphere	78804	117611	.082	.016	273

Table 1: Model information and timings, in seconds on 195Mhz R10k for *naive* and *hierarchical silhouette extraction* methods under an orthographic view.

require only one extra pass, and that pass may only be of the silhouette edges.

Creases are extracted independent of the view and are drawn as white lines. After adding shading and silhouettes, only the creases that are connected to two front facing faces, and are not already silhouettes, are visible. To emulate the look of illustrations the creases need to be drawn with the same thickness as the silhouettes, as shown in Figure 4.

One problem when rendering rasterized wide lines is the “gaps” where the lines do not overlap. A solution to this is to render the end of the lines with large points, effectively filling in the gaps. There is much less of a performance loss with the software extraction methods, since they only need to redraw the actual silhouettes, not the entire model.

10.5 Discussion

Silhouette finding using specialized graphics APIs like OpenGL is simple to implement and not as dependent on “clean” models. However it is less flexible and does not allow the user to change line weight. The software methods we discussed are more complex and depend on “clean” models which must have shared vertices, otherwise internal boundaries can not be checked for silhouettes. However the software methods provide more flexibility and, potentially, better performance.

Table 1 presents the information of two extreme cases. These cases are based on orthographic views. Under perspective projection some form of bounding volume hierarchy would have to be employed [12] to increase the efficiency. Both the simplified and the finely tessellated versions of the crank shaft model have many sharp features, while the sphere has very small dihedral angles.

The current implementation of the hierarchy method uses an icosahedron with four levels of subdivision which generates 1280 faces. On the sphere this method is extremely efficient. When using overlapping

Level	No Overlap	Overlap
0	12294	3138
1	4844	2221
2	5978	3569
3	4666	5943
4	9704	22615

Table 2: Hierarchy method showing the number of edges stored at each level on a Gaussian sphere for 25k-polygon crank shaft model for non-overlapping and overlapping bins.

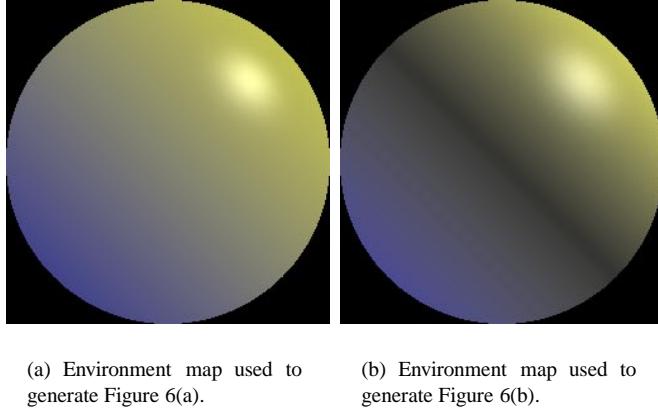


Figure 5: Shaded sphere images used for environment maps. (See Color Plate)

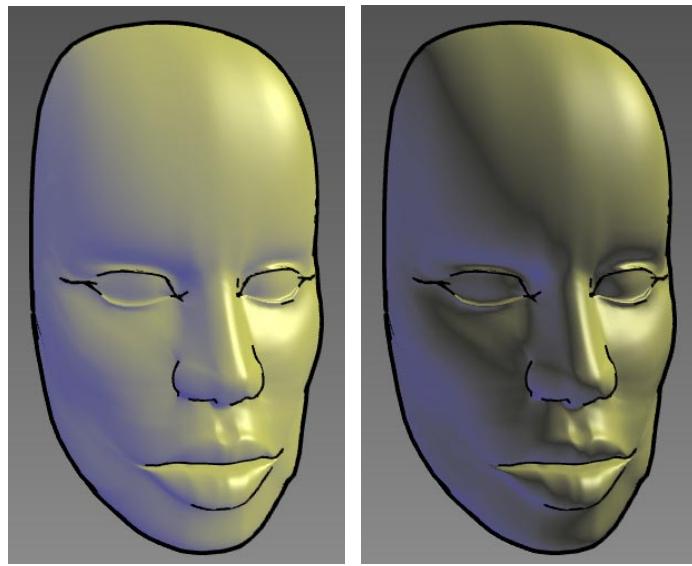
bins, all of the edges are stored in the leaf nodes. When using non-overlapping bins only 84% of the edges are in the leaf nodes and 2132 are on level zero. Table 2 shows the number of edges stored at every level of the hierarchy for non-overlapping and overlapping hierarchies. The overlapping method did a much better job, even on the simplified crank model.

Parallelizing the silhouette extraction with the rest of the rendering can cause the extraction time to be negligible. A separate thread can extract silhouettes while the polygons are being rendered to shade the model or initialize the Z buffer. This parallelization takes only three-thousandths of a second for the sphere and five one-hundredths on the large crank shaft model. If you are using software visibility algorithms this technique would probably prove to be more effective.

10.6 Shading

There are several ways to apply NPR shading models using hardware [1, 2]. In our “Interactive Technical Illustration” paper [4], we chose to use environment maps because they provide the most flexibility in the shading model. This effectively allows us to evaluate a lighting model at every normal/reflection direction in the visible hemisphere in eye-space.

Shading is rendered using one of three modes. The first two are the diffuse and metallic shading models [2] presented in Section 8. In its simplest form the diffuse cool to warm shading model interpolates from a cool (blue-green) to a warm (yellow-orange) color based on the surface normal. This cool-to-warm diffuse shading is shown in Figure 6(a). The third method is an adaptation of this cool to warm shading, simulating the more dramatic shading effects sometimes used by artists. Figure 6(b) illustrates the effect achieved when the reflected light from the left of the object produces a back-splash of light opposite the



(a) Shading by Gooch et al.

(b) Shading with splash back

Figure 6: The dark banding in the light splash back model can communicate more curvature information and works well on organic models. (See Color Plate)

direct lighting source. This is accomplished by modifying the model of [2] with a simple multiplier:

$$(\alpha |\cos \theta| + (1 - \alpha))^p,$$

where α and p are free parameters which, for this image, are set to 0.76 and 0.78, respectively.

We evaluated the whole shading equation in a Phong environment map. In using an environment map as shown in Figure 5, all normals in eye-space are mapped to a 2D texture. This shading only is valid in eye-space, but it is possible to extend these to view-independent environment maps [6]. The cool-to-warm and light “splashback” terms are a function of the light direction and could be implemented with this representation. However, the Phong term would have to be computed for each view even though a single light source could be implemented as a single 1D texture map instead of a full 2D texture map.

10.6.1 Metal Shading

The metal shading technique we use assumes a principle direction of curvature and striping occurs in an orthogonal direction. We first compute a table of random intensities where sample is: $b + (r * a)$, where the base b is -0.1, r is a random number in [0,1] and a is 1.4. This causes the distribution be to biased towards white and black. We then filter each element in the table with each of its neighbors using a 1-5-1 weighting scheme and clamp this value to be in the range of [0,1]. We make these values periodic so there is some coherence which will remain smooth as they wrap around the model.

The table is then resampled into a 1D texture map. The texture map is used as a cosine distribution because it is indexed via a dot product. The resampling makes sure the bands are uniformly distributed on a cylinder. We then render the model with this texture map. The texture matrix computes the dot product with a fixed axis orthogonal to the principle curvature direction, and remap the value into [0,1]. This technique can be scaled in order to change the spacing of the stripes.

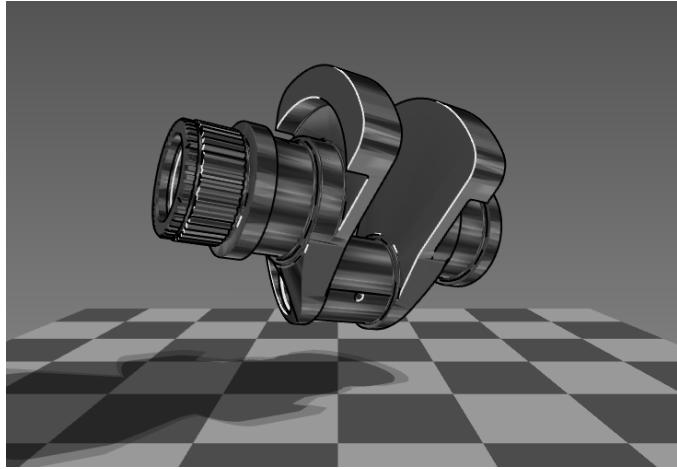


Figure 7: Metal-shaded object with shadow and ground plane. White creases and black silhouette lines are also drawn.

By itself, this texture does not look convincing, therefore we add Phong highlights computed by lighting a texture map in eye space with several Phong light sources oriented in the directions of a icosahedron's vertices, shown in Figure 7. A fairly large specular power, empirically around 30-50, seemed to work best with a specular coefficient of about 0.3.

10.6.2 Shadowing

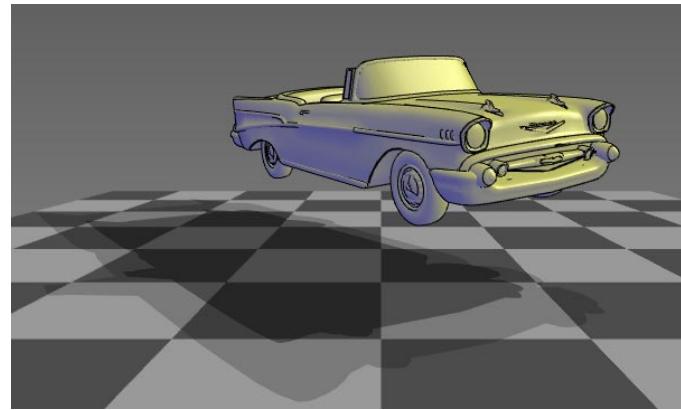
We draw shadows in one of three modes: a shadow with a hard umbra and a hard penumbra, a single hard shadow, and a soft shadow, as shown in Figure 8. Both of the later two modes approximate a spherical light source at a fixed distance from the center of the model in the direction of the light source used for shading.

The simplest and fastest method to draw simple shadows is to explicitly draw an umbra and penumbra. We draw two hard shadows, one from the center of the spherical light source back in the direction used for shading, and the other forward.

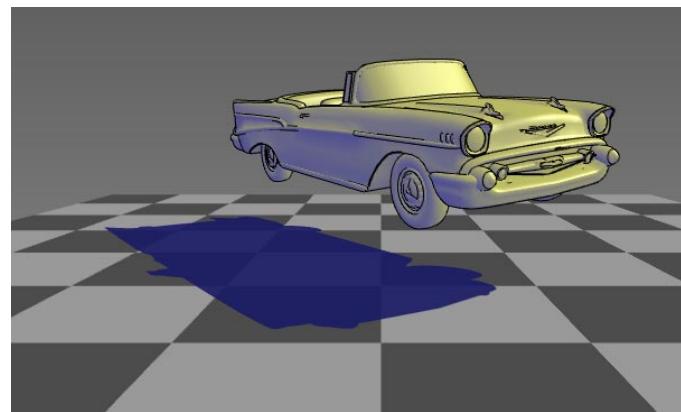
Soft shadows are problematic to render both accurately and efficiently, so we use an approximation to gain speed. Instead of using the conventional method to simulate an area light source, i.e., sampling the area light source and accumulating the point approximations, we project multiple shadows from the center of the approximation sampling a 1D direction, the ground plane's normal. This is done by projecting the same shadow onto a stack of planes, then translating the shadows to the ground plane and accumulating them, as shown in Figure 9.

Note that with this method, each “sample” is a perspective remapping of the first, intersected on a different plane. We could render a single shadow, copy it into texture memory and then remap it correctly to accumulate the other samples. This is much faster than projecting multiple jittered samples since there is a lower depth complexity for rasterization and a much lower burden on the transformation if the texture mapping method were used.

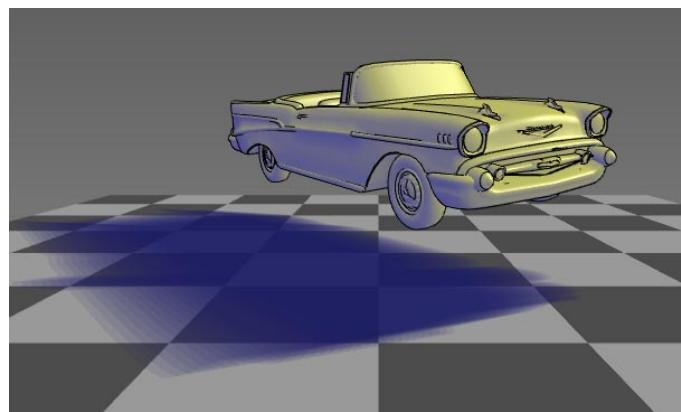
This method assumes that the silhouette from different points on the spherical light source is the same, i.e., the projection is the same. The planes coming out of the receiver will not correctly model contact. However, you can render only the lower planes if contact occurs resulting in a less realistic shadow, but one without distracting spillover.



(a) Hard penumbra and hard umbra.



(b) Single hard, colored shadow.



(c) Colored soft shadow.

Figure 8: Shadows provide valuable information about three-dimensional structure, especially the spatial layout of the scene. (See Color Plate)

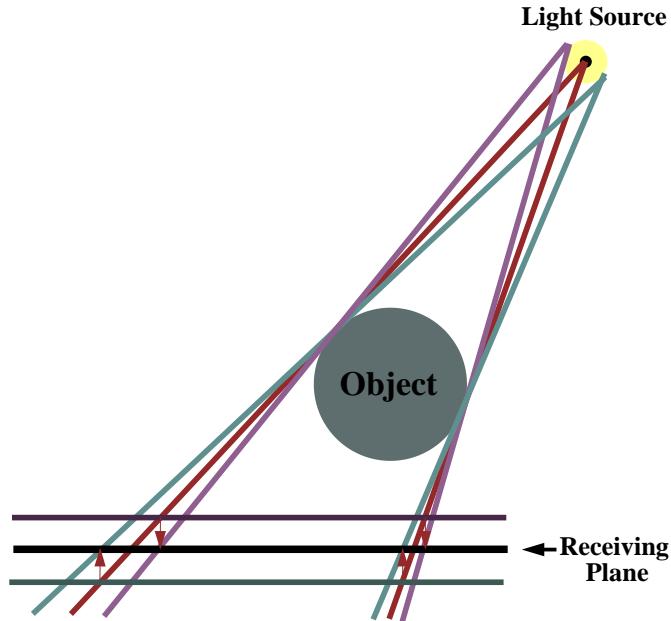


Figure 9: Drawing the shadow of a sphere with a spherical light source directly onto a ground plane below, traditionally each sample will render an ellipse. To get an accurate representation of the penumbra, this surface of the spherical light source needs to be sampled in 2 dimensions. With our method, each shadow is a concentric circle, requiring less samples to get the same results.

10.7 Conclusion

One of the largest goals in computer graphics has been realistic image synthesis. However, in a number of situations, an image which highlights particular information is valued above realism. For example, an automobile repair manual uses illustrations to remove unnecessary details and to draw attention to specific features.

Many computer-generated images have to be hand-tuned and they still convey shape poorly. The goal of this research was to use the techniques explored by illustrators for centuries to automatically generate images like technical illustrations and to be able to interact with these illustrations in three dimensions.

Acknowledgment

This work was done in collaboration with Peter-Pike Sloan, Peter Shirley, Elaine Cohen, and Rich Riesenfeld.

Authors' Note

These notes should be read while looking at colored images. See the course notes on the SIGGRAPH 99 CD-ROM for the images if colored images do not accompany this document.

References

- [1] David Blythe, Brad Grantham, Scott Nelson, and Tom McReynolds. *Advanced Graphics Programming Techniques Using OpenGL*. http://www.sgi.com/Technology/OpenGL/advanced_sig98.html, 1998.

-
- [2] Amy Gooch, Bruce Gooch, Peter Shirley, and Elaine Cohen. A Non-photorealistic Lighting Model for Automatic Technical Illustration. In *Computer Graphics*, July 1998. ACM Siggraph '98 Conference Proceedings.
 - [3] Amy A. Gooch. Interactive non-photorealistic technical illustration. Master's thesis, University of Utah, December 1998.
 - [4] Bruce Gooch, Peter-Pike Sloan, Amy Gooch, Peter Shirley, and Richard Riesenfeld. Interactive technical illustration. *Interactive 3D Conference Proceedings*, April 1999.
 - [5] Wolfgang Heidrich. A model for anisotropic reflections in open gl. In *SIGGRAPH 98 Conference Abstracts and Applications*, page 267, July 1998.
 - [6] Wolfgang Heidrich and Hans-Peter Seidel. View-independent environment maps. In *Eurographics/SIGGRAPH Workshop on Graphics Hardware*, pages 39–45, September 1998.
 - [7] Michael A. Kowalski and Lee Markosian et al. Art-based rendering of fur, grass, and trees. In *SIGGRAPH 99 Conference Proceedings*, August 1999.
 - [8] L. Markosian, M. Kowalski, S. Trychin, and J. Hughes. Real-Time Non-Photorealistic Rendering. In *SIGGRAPH 97 Conference Proceedings*, August 1997.
 - [9] Jackie Neider, Tom Davis, and Mason Woo. *OpenGL Programming Guide*. Addison-Wesley Publishing Company, 1993.
 - [10] Ramesh Raskar and Michael Cohen. Image Precision Silhouette Edges. *Symposium on Interactive 3D Graphics*, April 1999.
 - [11] Bruce Walter, Gun Alppay, Eric P. F. Lafortune, Sebastian Fernandez, and Donald P. Greenberg. Fitting Virtual Lights for Non-Diffuse Walkthroughs. In *SIGGRAPH 97 Conference Proceedings*, pages 45–48, August 1997.
 - [12] H. Zhang and K. Hoff III. Fast backface culling using normal masks. In *Proc. 1997 Symposium on Interactive 3D Graphics*, pages 103–106, April 1997.

Interactive Non-photorealistic Rendering

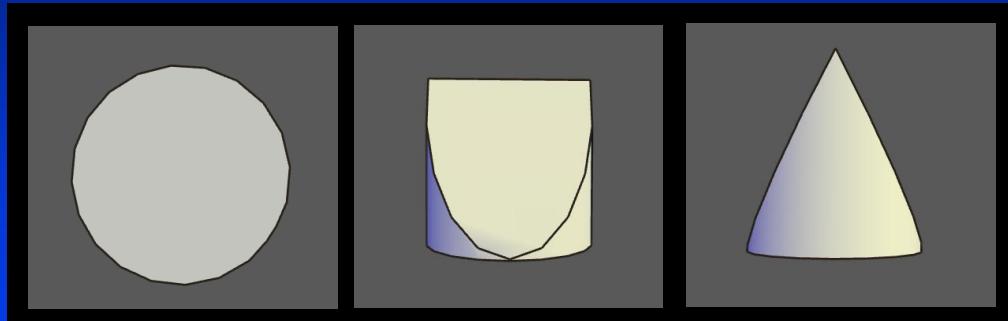
Bruce Gooch

Static Technical Illustration

Bottom

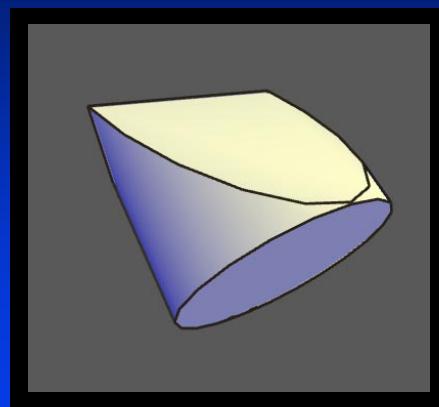
Front

Side



Dynamic Technical Illustration

Interactive Video Clip



3D Illustration

Roll that Video...

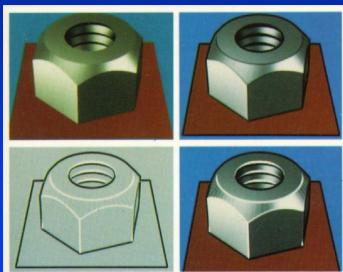
***Video Clip of other interactive NPR systems,
and Interactive Technical Illustration.***

Background

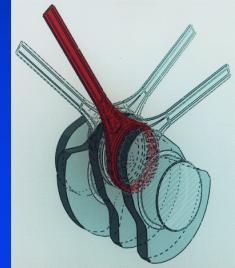
Teece 1998



*Saito et al.
1990*

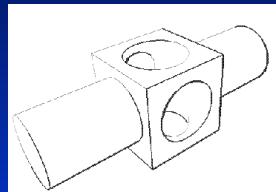


*Dooley et al.
1990*

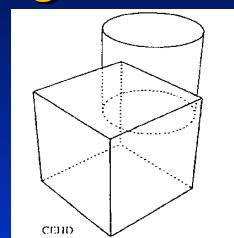


Background

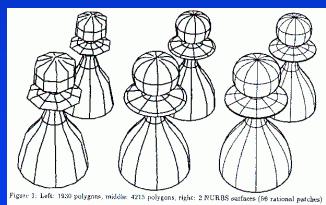
Markosian et al. 1997



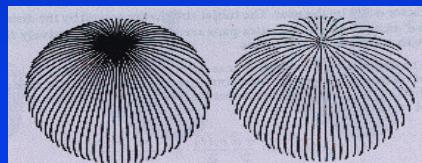
Rossignac et al. 1992



Elber 1990



Winkenbach and Salesin 1996



Implementing Lines

- Data Structures and Preprocessing
- Deterministic Software Algorithms for Drawing Silhouettes
- OpenGL Hardware methods for drawing silhouettes
- Implementing line styles

Calculating Edge Lines for Interaction

Polygonal

- Markosian et al. (Siggraph 97)
- Gooch et al. (I3D 99)

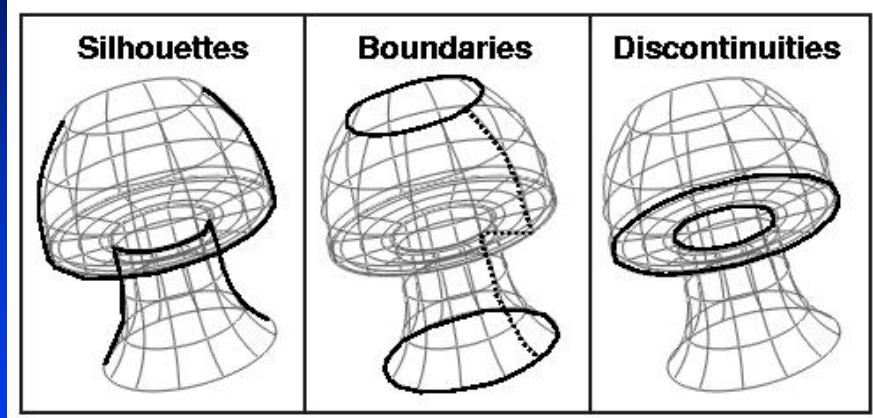
NURBS

- Amy Gooch, Master Thesis

Depth Buffer

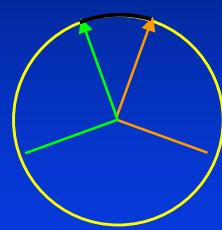
- Cassidy Curtis

Line Definitions

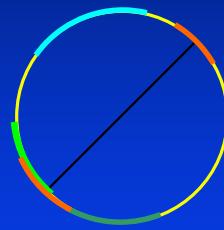


Software based silhouettes

Similar to Zhang97, but use gauss map of edges instead of faces



Edge



Orthographic



Perspective

Software based silhouettes cont.

Build hierarchy over the sphere, two methods

One:

Store arcs at level in hierarchy that completely contains them. Overlapping bin problems.

Two:

Have single discrete rep, store arcs in all bins they intersect.

Software silhouettes cont.

Pros: One traversal gathers all silhouettes

Cons: Need better way to tie in spatial information for perspective, requires “clean” models, more complex coding.

Note:

Brute force in parallel with shading works well.

Hardware based silhouettes

Similar to Rashkar99 and Rosignac92

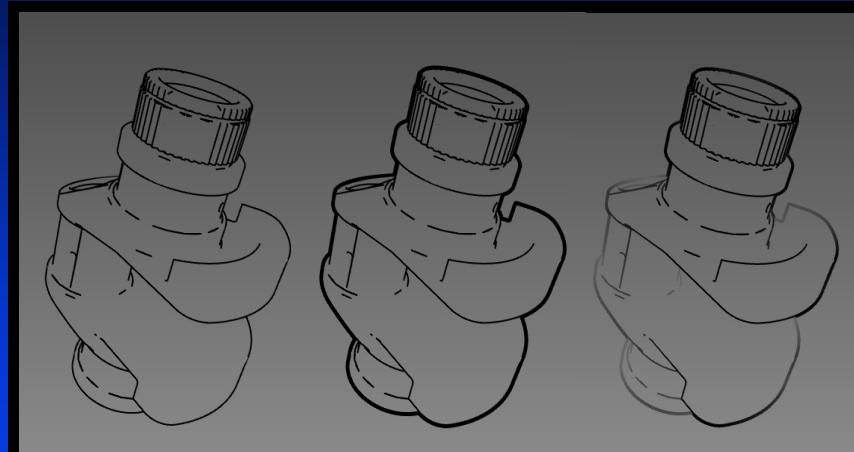
Several variations, most different from above:

***Draw front/back facing separately, pixels
that were both are silhouettes.***

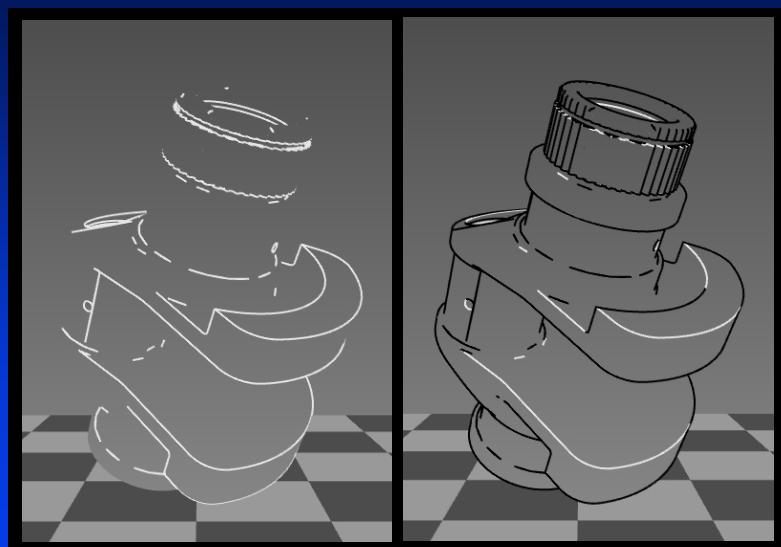
Pros: Simple, doesn't require "clean" meshes

***Cons: "thresholds", inefficient if edges need to
be drawn multiple times***

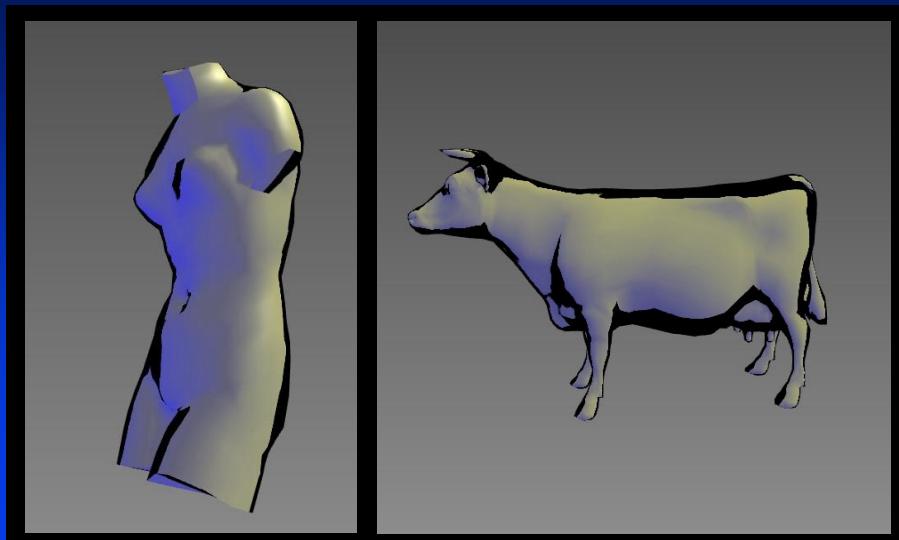
Line Width and Style



Creases Drawn in White



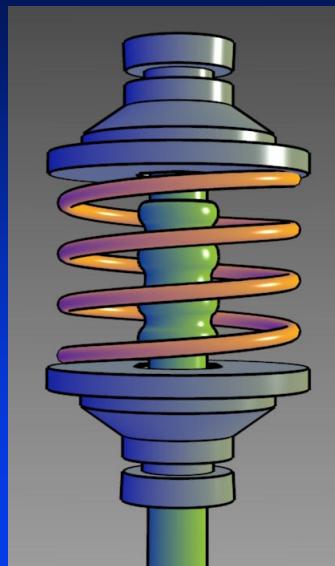
Curvature Based Silhouettes



Implementing Shading

- Interpolating between cool and warm colors
- Using a tone relationship with an object color
- Light Splash-back model
- Metal shading hack

New Shading



New Shading Formula

$$I = \left(\frac{1 + \hat{L} \cdot \hat{N}}{2} \right) k_{cool} + \left(1 - \frac{1 + \hat{L} \cdot \hat{N}}{2} \right) k_{warm}$$

Creating Undertones

Warm to Cool Hue Shift



Green with Warm to Cool Hue Shift (undertone)



Combining Tones with Undertones

Green with Tone and Undertone

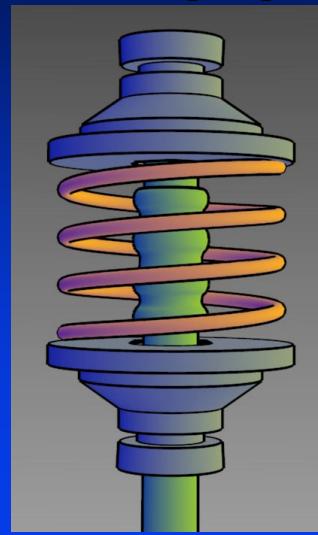


Model shaded with tones and undertones



OpenGL Approximation

Without highlights

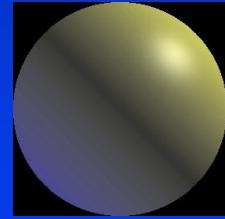
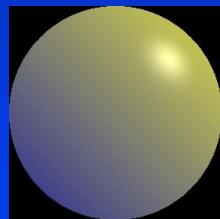


Light RGB Intensities:

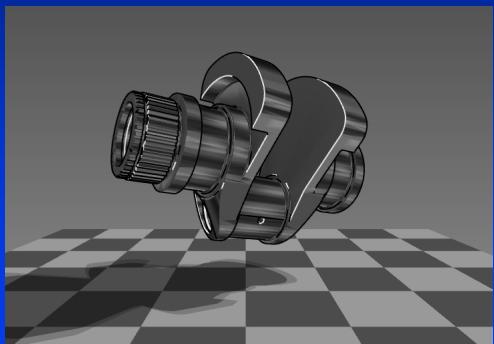
$$L_1 = (0.5, 0.5, 0);$$

$$L_2 = (-0.5, -0.5, 0);$$

Shading using environment maps

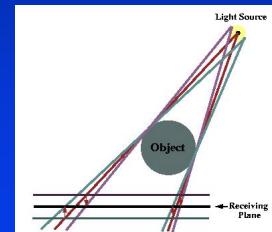


Imitating Material Properties

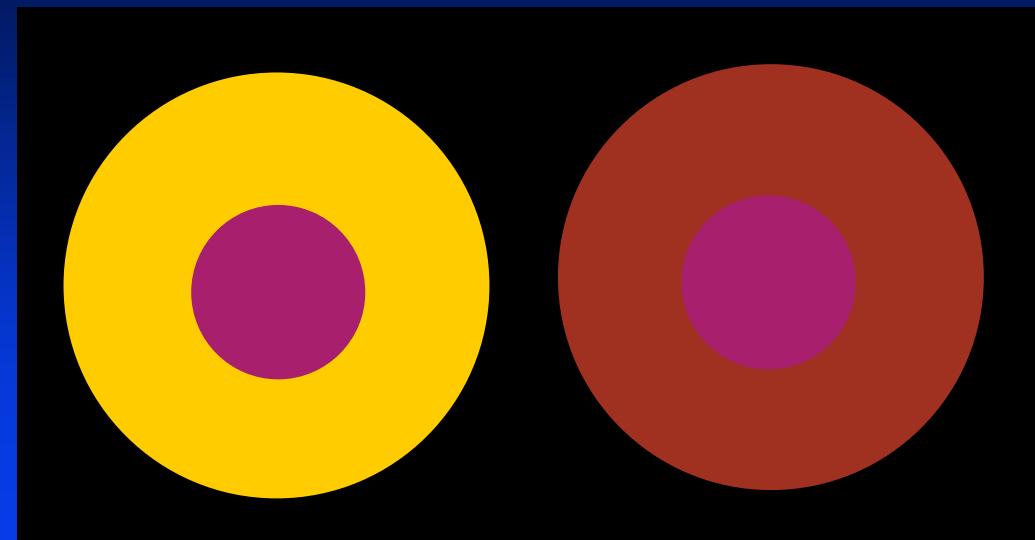


Implementing Shadows

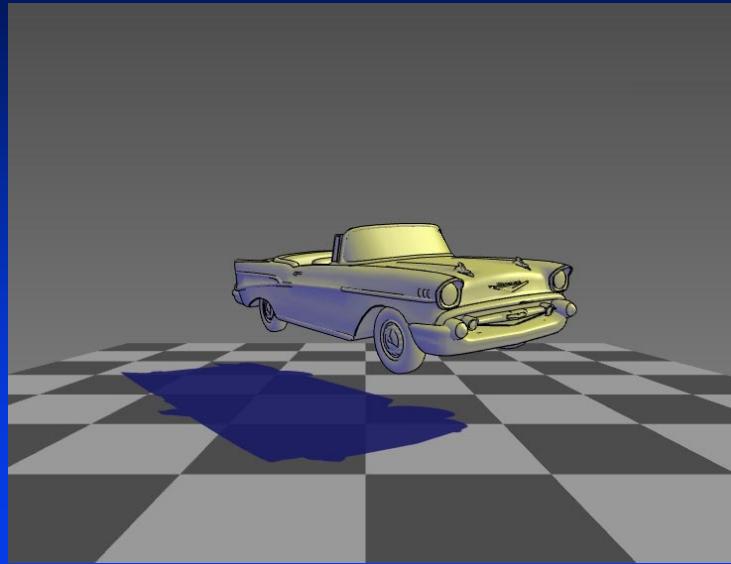
- Hard penumbra
- Hard umbra and hard penumbra
- Soft shadows



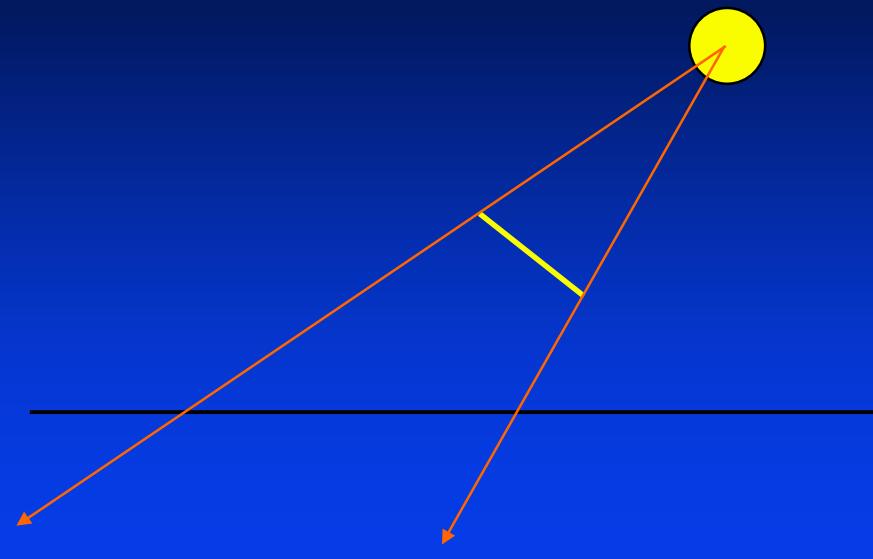
Colored Shadows



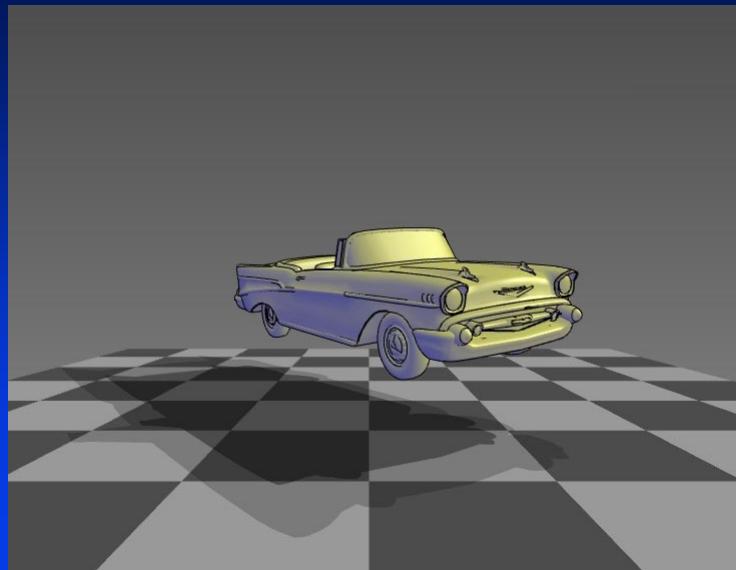
Colored Hard Shadow (Umbra)



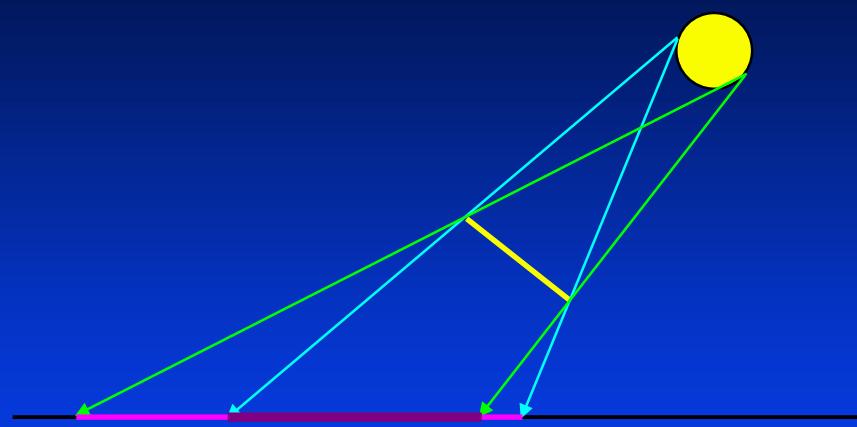
Shadow Hack



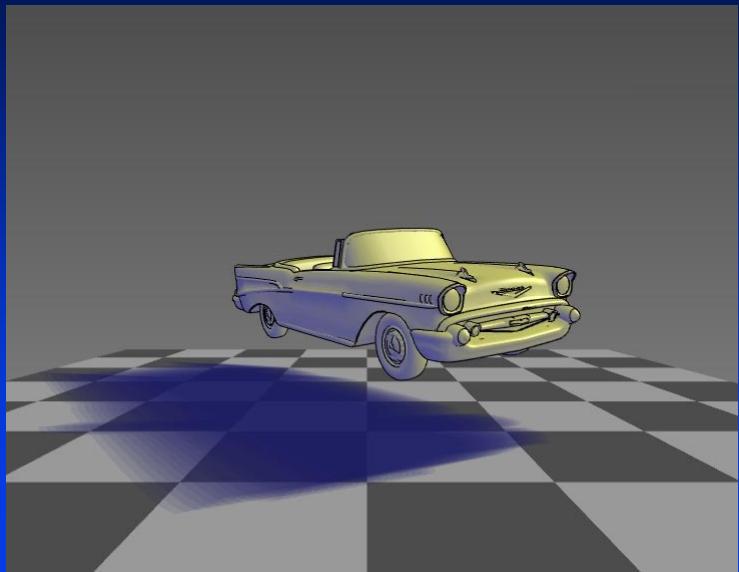
Hard Umbra and Penumbra



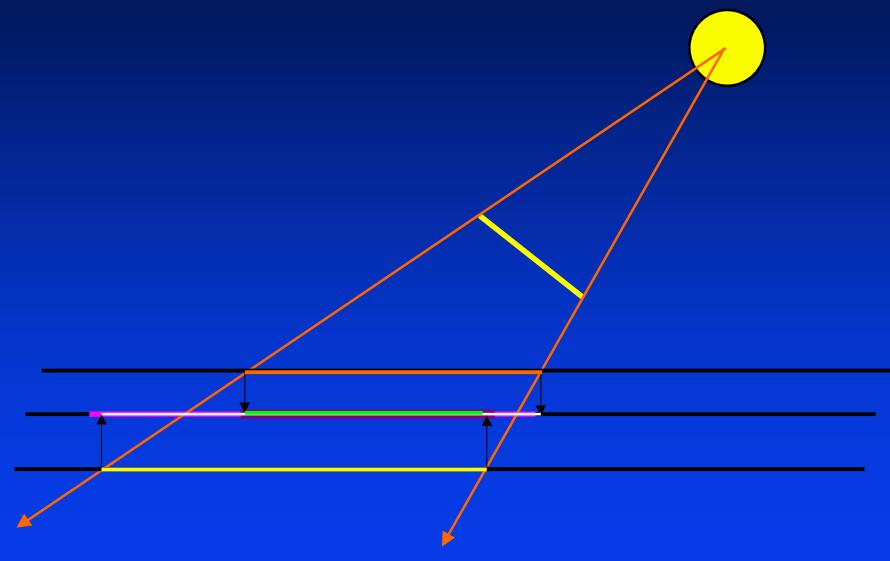
Shadow Hack - traditional



Soft Colored Shadow



Shadow Hack - new method



Shadow Hack

Generate single texture, other shadows are just projective re-mapping (single polygon.)

Fine for floating objects, doesn't properly deal with contact.

3D illustration

Moving the object

vs

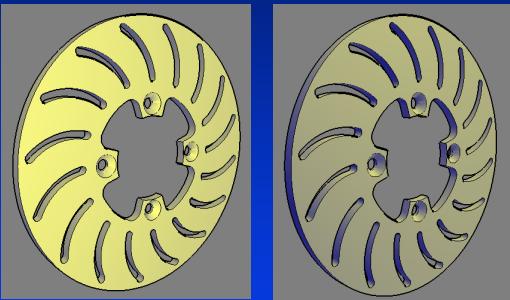
Changing the view point



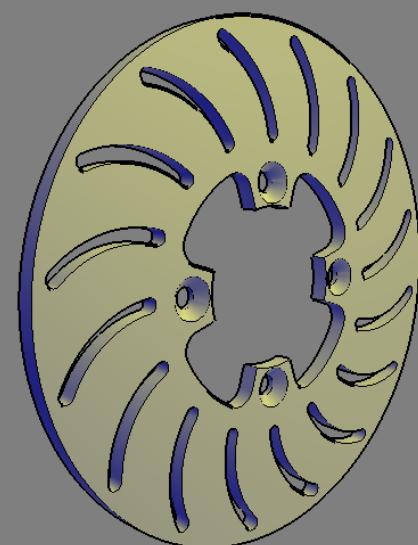
Object-oriented Lights

vs

Viewer-oriented Lights



3D illustration



Thanks

- Peter-Pike Sloan
- Michael Cohen, Ramesh Raskar, John Snyder
- Peter Shirley and Rich Riesenfeld
- University of Utah Computer Graphics Groups



11 Kazoo – A Case Study in Non-Photorealistic Rendering

Stuart Green
LightWork Design Ltd.

Introduction

LightWork Design is a software developer specializing in toolkits for rendering and visual simulation. Its flagship product, LightWorks, provides the industry standard photorealistic rendering functionality to many of the world's leading 3D design packages, particularly in the Computer Aided Design (CAD) and Architecture Engineering and Construction (AEC) markets. LightWorks is integrated with applications that create 3D data. Its Application Programming Interface (API) provides interfaces to the geometric data types found in 3D design systems, including polygonal, parametric and implicit surfaces.

Over the last 10 years, the company has driven its product development by the ever-increasing demands of 3D professionals, particularly in industrial and graphic design market segments. The LightWorks rendering engine provides a rich feature set of functionality, including advanced scan line techniques, ray tracing, radiosity and atmospheric rendering. The LightWorks rendering libraries allow 3D application developers to integrate advanced rendering functionality within their products.

Over the past few years LightWork Design has observed a growing desire within various market segments to provide forms of visualization other than photorealism. Photorealistic computer graphics has its place, but the visual messages conveyed are often too precise and sterile for certain applications. A designer often has the need to convey work in progress, which is incomplete and imprecise, or to express a design idea to inspire a client. Realistic rendering is not well suited to this.

As an example, consider the point of sale systems used for design of kitchens and bathrooms in retail stores. These operate by observing prescribed rules that describe how units connect together. The designer then places the units within the space plan, and the rules are applied to snap the location of the units to the appropriate boundaries. This allows an assembly of predefined 3D objects to be designed very quickly and accurately. This assembly can then be provided to a rendering system from which photorealistic images can be generated. An example is given in Figure 1.



Figure 1: Photorealistic kitchen, image courtesy of Planit International

When selling a kitchen, it is usual to invest more in presenting the design for a client who is prepared to pay for high value units and appliances. Consequently, the services of an artist will often be employed to create a stylized depiction of the design, as illustrated in Figure 2. This rendering evokes quite different visual messages from Figure 1. The stylization conveys aesthetics and a sense of romanticism that is missing from the photorealistic image, and for this reason the expressive rendering can be a more effective sales tool.



Figure 2: An artist's impression of a kitchen, courtesy of Planit International

LightWork Design has developed the **Kazoo™** family of software for providing non-photorealistic rendering (NPR) styles to application programs. Here, we describe the design objectives of Kazoo and how these have been addressed. The Kazoo Toolkit offers a configurable rendering pipeline that provides a software infrastructure within which a wide range of existing and new NPR styles can be developed.

Design Objectives

When designing Kazoo, the following objectives were addressed:

Configurability – A key objective was to provide a fully configurable system and allow multiple alternate styles to be implemented. The range of NPR styles that could be accommodated should not be restricted to specific algorithms and approaches, but rather the software infrastructure of Kazoo should provide building blocks to simplify the implementation of new styles and algorithms.

Support of a 3D rendering pipeline – Researchers in the field of NPR have proposed a range of different algorithms for generating synthetic images from both 3D and 2D (image) data. Since 2D image data can readily be generated from 3D data sets using a renderer, a clear objective was that Kazoo would accommodate a 3D graphics pipeline that is capable of photorealistic rendering.

Automatic rendering – Kazoo is targeted primarily for integration with applications of which users are typically *not* artists. Therefore, an objective was to provide a framework within which styles can be developed that require little or no intervention by the user. Thus, the objective was to support techniques that create the *appearance* of artistic styles in an automated fashion. Applications for computer graphics artists were not excluded from consideration, but the emphasis was placed on users without strong artistic skills. The objective was to provide these users with tools to generate stylistic images quickly and easily.

Ease of use – By focusing on non-artist users, an emphasis was placed on ease and simplicity of use. While it was anticipated that each style would expose controls to allow customization of its behavior, these controls were to be provided using simple and intuitive user interfaces and language.

Building blocks – In considering a range of current NPR styles, it is clear that some basic functionality recurs across different approaches. The following basic capabilities were identified:

- *Support for photorealism* – Photorealistic rendering is regarded as an expressive style, and it was believed that Kazoo should support photorealism.
- *Stroke generation* – the means to evaluate continuous outline strokes from 3D surfaces.
- *Real time rendering* – Support for hardware-assisted polygon-based rendering.
- *Post processing* – Support of image post processing capabilities.

Computer platforms – The Windows on Intel and Power Macintosh platforms were to be supported.

Ease of Integration – Kazoo was to be developed as a suite of software technologies, including a toolkit, and simplicity of integration was regarded as an important objective.

Configurable Rendering Pipeline

To address the stated objectives, Kazoo was designed to accommodate two configurable rendering pipelines. These are a *shading pipeline* and a *stroke pipeline*. The former accepts 3D surface geometry and yields raster image data. The latter also accepts 3D surface geometry and yields both 2½D resolution-independent stroke data (curves) or image data. A *Kazoo style* is implemented using one or both of these pipelines configured to deliver the required result. When both pipelines are used, this may be in parallel (when usually the stroke output is superimposed on the shading output) or sequentially (when usually the output of the stroke pipeline drives the input of the shading pipeline) as illustrated in Figure 3. There is no limit to the number of passes through the pipelines that can be performed for the generation of a single image.

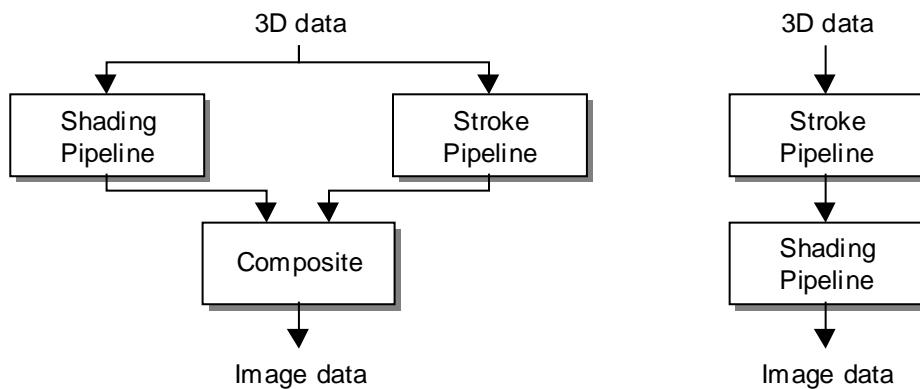


Figure 3: Alternative configurations of the Kazoo rendering pipelines

The Kazoo shading pipeline is an enhanced photorealistic renderer, with each stage being configurable such that it may be substituted by purpose-written software by the style writer. The pipeline is illustrated in Figure 4. Input to the pipeline is 3D data represented as polygons, implicit surfaces and parametrically defined surfaces. A style may begin by optionally performing some deformation to the geometry, for example, to give a false perspective. This distorted geometry is then passed to the visibility-processing

step to yield a collection of surfaces for shading. These surfaces are divided into multiple sample points within the visible region beneath each pixel. The sample points are shaded by a series of steps that treat orthogonal characteristics of the surfaces, including their illumination and reflectivity. Finally a post-processing step provides a framework within which image-based effects can be implemented. This step, as indeed are most others, is multi-pass in that repeated occurrences of the step can be applied as required.

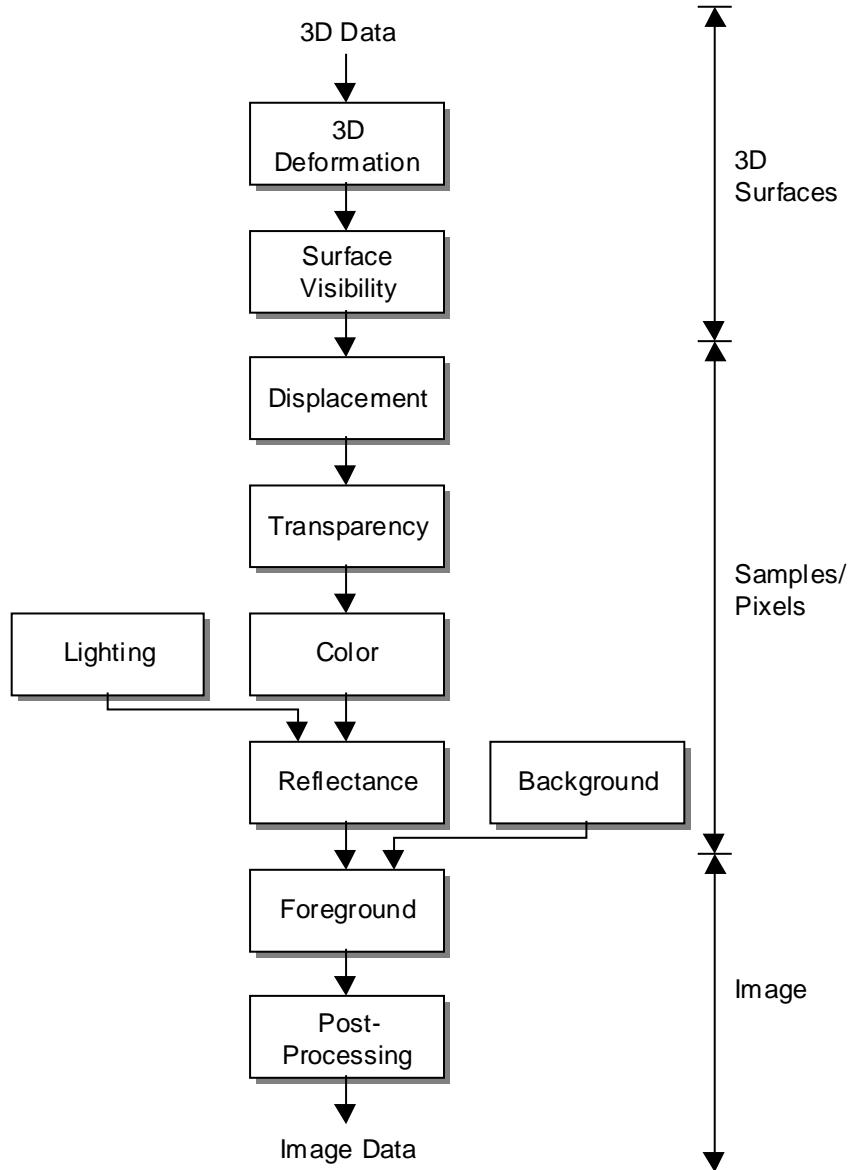


Figure 4: The Kazoo shading pipeline

The Kazoo stroke pipeline also begins with 3D surface data that may be optionally deformed (see Figure 5). But it differs from the shading pipeline in that the visibility processing is designed to determine visible edges (boundaries and silhouettes) that are classified against surfaces. A stroke generation stage then evaluates these edges to yield continuous 2½D strokes. Intuitively, a stroke is a single line that would be placed on the canvas by an artist. The stroke generation combines the geometric properties of the edges and applies a series of heuristics to arrive at a candidate stroke drawing sequence. Note that at this point the strokes can be regarded as 2½D because their appearance is view dependent.

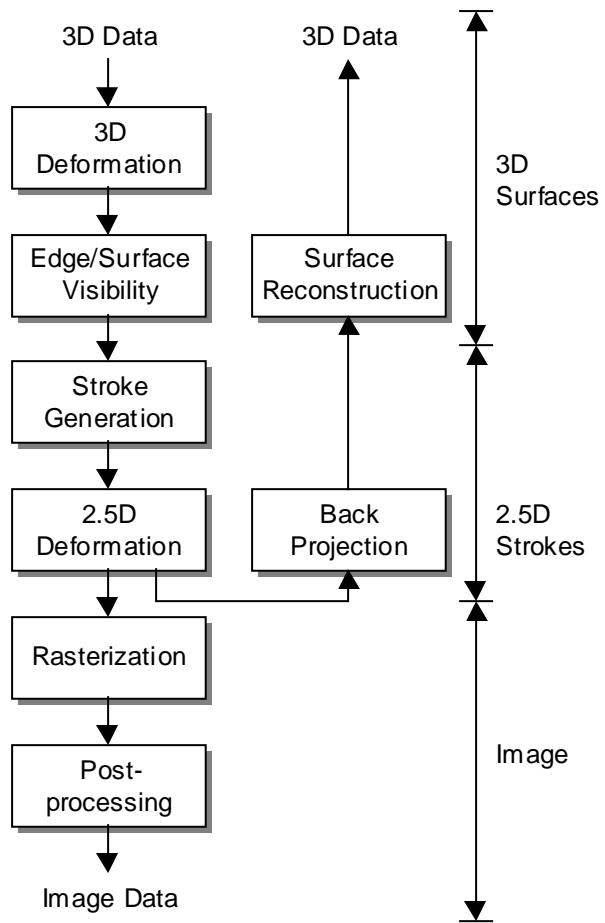


Figure 5: The Kazoo stroke pipeline

Having evaluated a collection of strokes, the next stage in the pipeline is to optionally deform the strokes, for example, to provide randomness or to distort them. After deformation, the strokes can be either rasterized then post-processed as in the shading pipeline, or alternatively they can be used directly as resolution-independent strokes. Although the strokes are at this point recorded as a 2½D data set, sufficient geometric interaction is maintained with them so that 3D surfaces can be reconstructed through a back-projection stage. The resulting 3D data can be passed through one of the pipelines again for further processing.

The operation of the pipelines is illustrated by the implementation of a ‘chalk’ style, as illustrated by the steps of Figure 6. This style invokes the stroke pipeline, then feeds the reconstructed surfaces into the shading pipeline. The steps involved are as follows:

- The original incoming 3D data, in this case a cylinder (implicit surface).
- Deformations are applied to the 3D data to distort the geometry, or create a false perspective projection.
- A hidden surface algorithm is applied to yield visible surfaces.
- Strokes are constructed from the boundaries and silhouettes of the visible surfaces. A stroke incorporates its direction in the 2½D coordinate system.

- (e) The strokes are back-projected and then a surface reconstruction is applied. In this case, the strokes are converted into brush-like regions whose appearance is a function of the length and curvature of the stroke.
- (f) The reconstructed surfaces are presented to the shading pipeline, within which a procedural texture pattern is applied in image space to create the appearance of the undulations of the canvas. This final step yields image data.

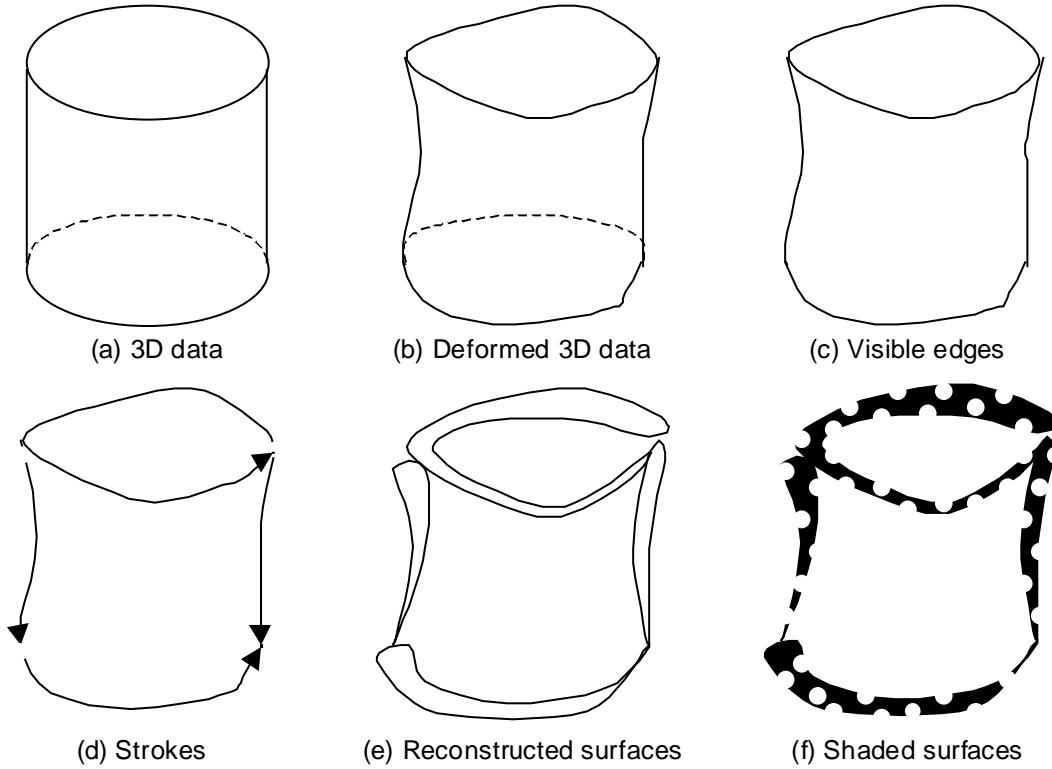


Figure 6: Illustration of a simple 'chalk' style

This is one example of how the pipelines can be used to implement a simple style. The configurability of each stage of the pipelines means that a wide range of interesting styles can be implemented very quickly and easily.

Kazoo Styles

An implementation of a particular path through the pipelines, including any custom code, is called a Kazoo *style*. A style is designed to provide a particular visual effect, and will normally expose a series of controls that may be used to customize the appearance of the style. Styles are collected together and delivered in the form of a *Style Pack*. Figure 7 illustrates a range of Kazoo styles. Three styles are described below.



Figure 7: A selection of Kazoo styles

Cartoon

The **Cartoon** style invokes first the shading pipeline using a simple two- or three-tone color quantization, and then superimposes upon this the result of passing the geometry through the stroke pipeline in which simple lines of constant width are rasterized. The result is simple cartoon effect. Examples are provided in Figure 8.

Hand Drawn

This style invokes only the stroke pipeline. Two frequencies of randomness are applied to the placement of the strokes in the image to mimic the movement of an unsteady hand. Each stroke is rasterized with a simple uniform taper so that the stroke has zero width at its end. Examples are given in Figure 9.

Stipple

This style is implemented as a simple post-processing step in the shading pipeline to create a stipple pattern. The appearance of the pattern is a function of the brightness of the surface. Examples are given in Figure 10.



Figure 8: Sample Cartoon images

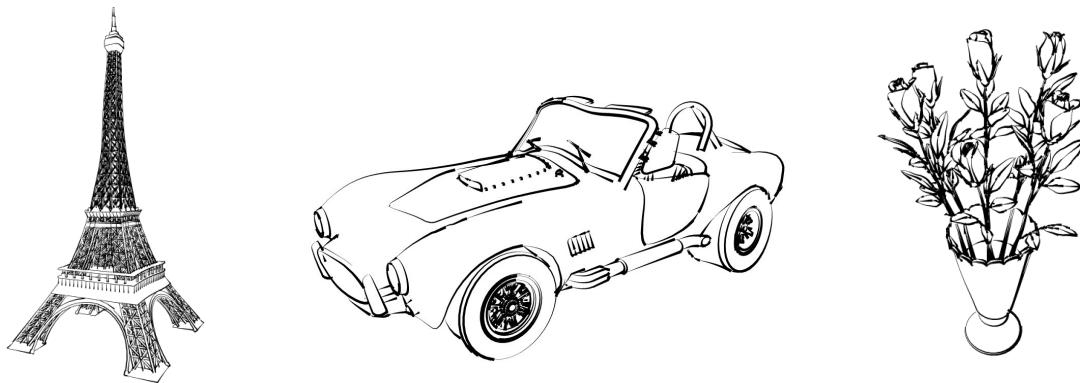


Figure 9: Sample Hand Drawn images

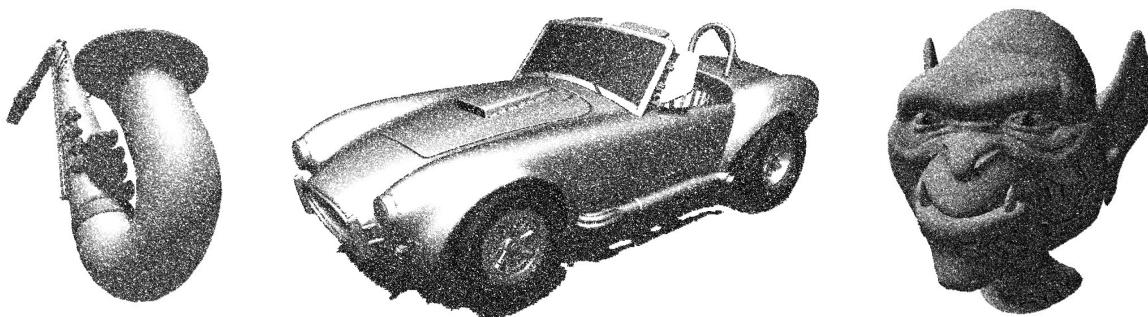


Figure 10: Sample Stipple images

Style Controls

The style editing dialogs for four Kazoo styles are illustrated in Figure 11. These styles are designed for non-artistic users who require very simple access to the styles. Consequently the user interfaces are basic and intuitive.

Where styles incorporate the stroke pipeline, a simple *Line Width* control is provided in the form of a drop-down list, from which a number of pre-defined settings can be selected.

The names of the controls are chosen to convey intuitive interpretations of their actions. For example, a **Hand Drawn** style provides *Wobbliness* and *Curliness* settings controlled by a simple slider.

Figure 12 illustrates the effect of changing some of the controls for three of the styles. In Figure 12(a) the line width of the **Cartoon** style is adjusted. Note that the availability of explicit, connected strokes in the image plane means that high quality, crisp boundaries can be drawn accurately. In Figure 12(b) the *Curliness* control of the **Hand Drawn** style is adjusted. Again, the explicit representation of continuous strokes allows perturbations of the lines to be performed while maintaining the integrity of the strokes. Adjustments to the *Size* control of the **Mosaic** style is shown in Figure 12(c). Finally, the textured appearance of a **Soft Pencil** style is adjusted in Figure 12(d).

Summary

Kazoo provides, in the form of a software toolkit, an effective environment for the development of NPR techniques. Emphasis has been placed on styles that can be automated to allow artistic renderings to be produced from 3D data by non-artistic users. Styles may also be implemented for Kazoo that address the more demanding requirements of professional 3D artists.

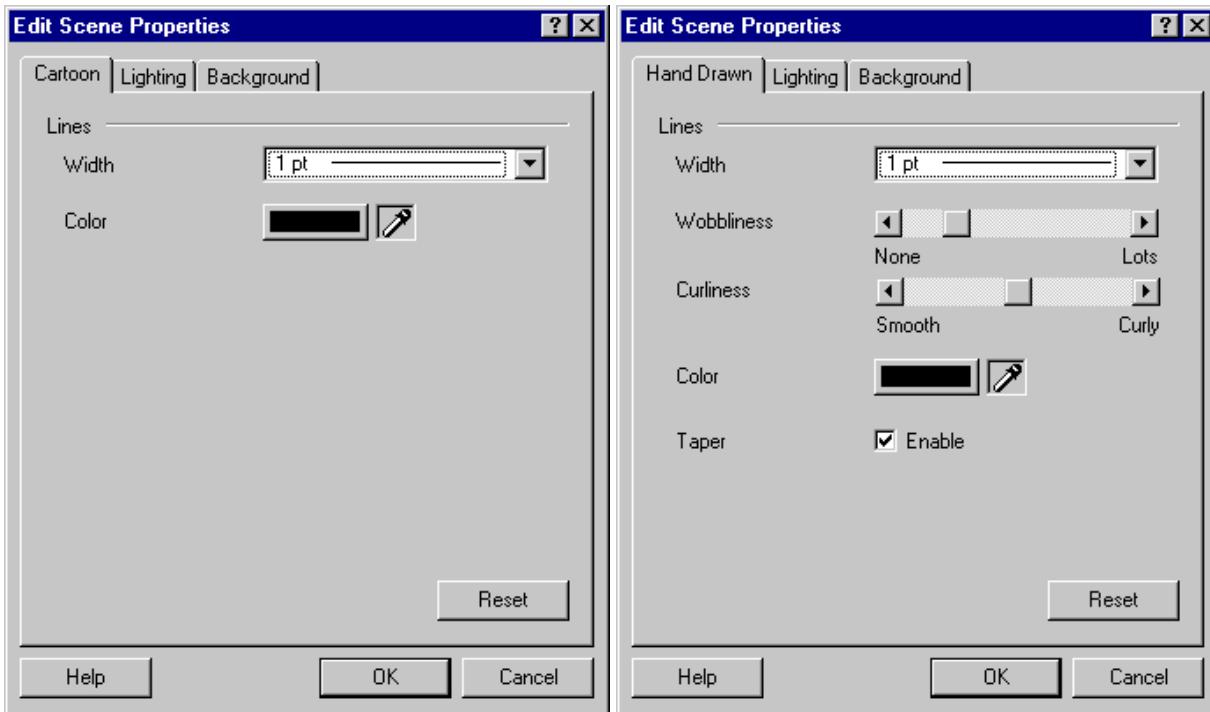
Notes

A version of Kazoo for Microsoft Windows is supplied in the **kazoo** sub-folder of the SIGGRAPH 99 Course Notes CD-ROM. Further details on Kazoo can be obtained from <http://www.kazoo3d.com>.

The 3D models used in this chapter are supplied by Zygote Media Inc.

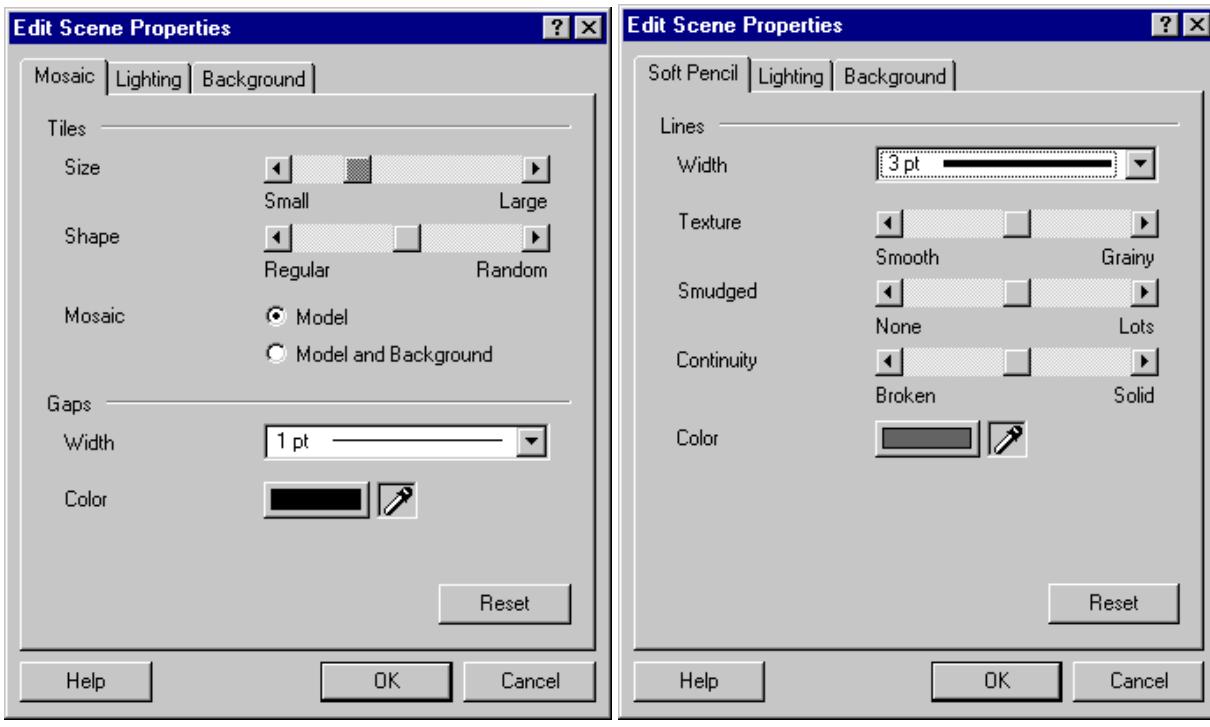
Kazoo is a trademark of LightWork Design Ltd.

Techniques described here are protected by patent law in the USA and other countries.



(a) Edit **Cartoon** dialog

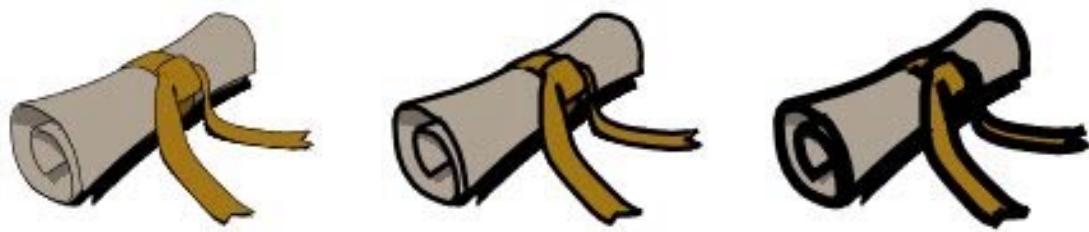
(b) Edit **Hand Drawn** dialog



(c) Edit **Soft Pencil** dialog

(d) Edit **Mosaic** dialog

Figure 11: Example Kazoo style controls



(a) **Cartoon** line width



(b) **Hand Drawn** curliness



(c) **Mosaic** tile size



(d) **Soft Pencil** Texture (smooth to grainy)

Figure 12: Example style controls

Bibliography

- [1] Arthur Appel. The Notion of Quantitative Invisibility and the Machine Rendering of Solids. In *Proc. ACM National Conference*, pages 387–393, 1967.
- [2] David Banks. Interactive Manipulation and Display of Two-Dimensional Surfaces in Four-Dimensional Space. *Symposium on Interactive 3D Graphics*, April 1992.
- [3] Michael F. Barnsley. *Fractals Everywhere*. Academic Press, 1988.
- [4] Michael F. Barnsley, Arnaud Jacquin, Francois Malassenet, Laurie Reuter, and Alan D. Sloan. Harnessing Chaos for Image Synthesis. *Computer Graphics (Proc. Siggraph)*, ACM SIGGRAPH, ACM Press, 22(4), August 1988.
- [5] F. Benichou and Gershon Elber. Output Sensitive Extraction of Silhouettes from Polygonal Geometry. ftp://ftp.cs.technion.ac.il/pub/misc/gershon/papers/sil_extrac.ps.gz.
- [6] Deborah F. Berman, Jason T. Bartell, and David H. Salesin. Multiresolution Painting and Compositing. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 85–90. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.
- [7] Irving Biederman and Ginny Ju. Surface versus Edge-Based Determinants of Visual Recognition. *Cognitive Psychology*, 20:38–64, 1988.
- [8] E. A. Bier and K. R. Sloan. Two Part Texture Mapping. *IEEE Computer Graphics and Applications*, 6(9), 1986.
- [9] Faber Birren. *Color Perception in Art*. Van Nostrand Reinhold Company, 1976.
- [10] J. F. Blinn and M. E. Newell. Texture and Reflection in Computer Generated Images. *Communications of the ACM*, 19:542–546, 1976.
- [11] David Blythe, Brad Grantham, Scott Nelson, and Tom McReynolds. *Advanced Graphics Programming Techniques Using OpenGL*. http://www.sgi.com/Technology/OpenGL/advanced_sig98.html, 1998.
- [12] Bet Borgeson. *Color Drawing Workshop*. Watson-Guptill Publications, New York, 1984.
- [13] Wendy L. Braje, Bosco S. Tjan, and Gordon E. Legge. Human Efficiency for Recognizing and Detecting Low-pass Filtered Objects. *Vision Research*, 35(21):2955–2966, 1995.
- [14] W.L. Braje, D. Kersten, M.J. Tarr, and N.F. Troje. Illumination and Shadows Influence Face Recognition. *ARVO 96*, 1996.
- [15] David J. Bremer and John F. Hughes. Rapid Approximate Silhouette Rendering of Implicit Surfaces. In *Proc. The Third International Workshop on Implicit Surfaces*, June 1998.
- [16] Tom Browning. *Timeless Techniques for Better Oil Paintings*. North Light Books, 1994.
- [17] John W. Buchanan. Special Effects with Half-toning. *Computer Graphics Forum*, 15(3):97–108, August 1996. Proceedings of Eurographics '96. ISSN 1067-7055.
- [18] Brian Cabral and Leith (Casey) Leedom. Imaging Vector Fields Using Line Integral Convolution. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 263–272, August 1993.
- [19] Chris Christou, Jan J. Koenderink, and Andrea J. van Doorn. Surface Gradients, Contours and the Perception of Surface Attitude in Images of Complex Scenes. *Perception*, 25:701–713, 1996.
- [20] Johan Claes, Patrick Monsieurs, Frank Van Reeth, and Eddy Flerackers. Rendering Pen-drawings of 3D scenes on networked processors. *WSCG '97 proceedings*, 1, February 1997.
- [21] Tunde Cockshott. *Wet and Sticky: A novel model for computer based painting*. PhD thesis, University of Glasgow, 1991.
- [22] Tunde Cockshott and David England. Wet and Sticky: Supporting Interaction with Wet Paint. *People and Computers VI: Proceedings of HCI '91*, University Press Cambridge, August 1991.
- [23] Tunde Cockshott, John Patterson, and David England. Modelling the Texture of Paint. In A. Kilgour and L. Kjelldahl, editors, *Computer Graphics Forum*, volume 11, pages 217–226, 1992.
- [24] E. F. Codd. *Cellular Automata*. Academic Press Inc., 1968.
- [25] Harold Cohen. The Further Exploits of Aaron, Painter. *Stanford Humanities Review*, 4(2):141–158, 1995.
- [26] Microsoft Corporation. Impressionist.

-
- [27] Wagner Toledo Corrêa, Robert J. Jensen, Craig E. Thayer, and Adam Finkelstein. Texture Mapping for Cel Animation. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 435–446. ACM SIGGRAPH, Addison Wesley, July 1998. ISBN 0-89791-999-8.
- [28] Richard Coutts and Donald P. Greenberg. Rendering with Streamlines. In *SIGGRAPH 97: Visual Proceedings*, page 188, 1997.
- [29] Cassidy Curtis. Loose and Sketchy Animation. In *SIGGRAPH 98: Conference Abstracts and Applications*, page 317, 1998.
- [30] Cassidy J. Curtis, Sean E. Anderson, Joshua E. Seims, Kurt W. Fleischer, and David H. Salesin. Computer-Generated Watercolor. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 421–430. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.
- [31] Viewpoint Datalabs. Liveart 98.
- [32] Douglas DeCarlo, Dimitris Metaxas, and Matthew Stone. An Anthropometric Face Model using Variational Techniques. *SIGGRAPH 98 Conference Proceedings*, 1998.
- [33] Philippe Decaudin. Cartoon-Looking Rendering of 3D-Scenes. Technical Report 2919, INRIA, June 1996.
- [34] Fractal Design. Painter.
- [35] Debra Dooley and Michael F. Cohen. Automatic Illustration of 3D Geometric Models: Surfaces. *IEEE Computer Graphics and Applications*, 13(2):307–314, 1990.
- [36] Elena Driskill. *Towards the Design, Analysis, and Illustration of Assemblies*. PhD thesis, University of Utah, Department of Computer Science, Salt Lake City, Utah, September 1996.
- [37] Betty Edwards. *Drawing on the Right Side of the Brain*. Jeremy P. Tarcher/Putnam, 1989.
- [38] Gershon Elber. Gridless Halftoning of Freeform Surfaces via a Coverage of Isoparametric Curves. Technical Report 9507, Center for Intelligent Systems Report, Israel Institute of Technology, March 1995.
- [39] Gershon Elber. Line art rendering via a coverage of isoparametric curves. *IEEE Transactions on Visualization and Computer Graphics*, 1(3):231–239, September 1995. ISSN 1077-2626.
- [40] Gershon Elber. Line Art Illustration of Freeform Surfaces. Technical Report 9702, Center for Intelligent Systems Report, Israel Institute of Technology, January 1997.
- [41] Gershon Elber. Line Art Illustrations of Parametric and Implicit Forms. *IEEE Transactions on Visualization and Computer Graphics*, 4(1), January – March 1998. ISSN 1077-2626.
- [42] Gershon Elber and Elaine Cohen. Hidden Curve Removal for Free Form Surfaces. In Forest Baskett, editor, *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 95–104, August 1990.
- [43] Hany Farid and Eero P. Simoncelli. Optimally Rotation-Equivariant Directional Derivative Kernels. In *7th Int'l Conf Computer Analysis of Images and Patterns*, Kiel, Germany, September 1997.
- [44] Gerald Farin. *Curves and Surfaces for Computer-Aided Geometric Design: A Practical Guide*. Academic Press, Inc., Boston, third edition, 1993.
- [45] Jean-Daniel Fekete, Érick Bizouarn, Éric Courtnarie, Thierry Galas, and Frédéric Taillefer. TicTacToon: A Paperless System for Professional 2-D animation. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 79–90. ACM SIGGRAPH, Addison Wesley, August 1995. held in Los Angeles, California, 06-11 August 1995.
- [46] Margaret Fleck, David Forsyth, and Chris Bregler. Finding Naked People. *1996 European Conference on Computer Vision*, 2:592–602, 1996.
- [47] David A. Forsyth and Margaret M. Fleck. Identifying nude pictures. *IEEE Workshop on the Applications of Computer Vision*, pages 103–108, 1996.
- [48] Michael Goldberg Gail Manchur(Editor). *Alberto Vargas: The Esquire Years Vol 1*. Collectors Press, New York, 1984.
- [49] Robert W. Gill. *Basic Rendering*. Thames and Hudson Ltd, London, 1991.
- [50] E. Bruce Goldstein. *Sensation and Perception*. Wadsworth Publishing Co., Belmont, California, 1980.
- [51] E. Gombrich. Formula and Experience. *Art and Illusion*, 1960.

-
- [52] Amy Gooch, Bruce Gooch, Peter Shirley, and Elaine Cohen. A Non-photorealistic Lighting Model for Automatic Technical Illustration. In *Computer Graphics*, July 1998. ACM Siggraph '98 Conference Proceedings.
- [53] Amy A. Gooch. Interactive Non-photorealistic Technical Illustration. Master's thesis, University of Utah, December 1998.
- [54] Bruce Gooch, Peter-Pike Sloan, Amy Gooch, Peter Shirley, and Richard Riesenfeld. Interactive Technical Illustration. *Interactive 3D Conference Proceedings*, April 1999.
- [55] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The Lumigraph. In *Computer Graphics Proceedings, Annual Conference Series, 1996*, pages 43–54, 1996.
- [56] Roger L. Gould. “Hercules:” The 30-Headed Hydra. In *SIGGRAPH 97: Visual Proceedings*, page 213, 1997.
- [57] Qinglian Guo. Generating Realistic Calligraphy Words. *IEEE Trans. Fundamentals*, E78-A(11):1556–1558, November 1995.
- [58] Arthur L. Guptill. *Drawing with Pen and Ink*. Reinhold, New York, 1961.
- [59] Paul Haeberli. The Impressionist. <http://reality.sgi.com/grafica/impression/>.
- [60] Paul Haeberli. The Accumulation Buffer: Hardware Support for High-Quality Rendering. *SIGGRAPH 90 Conference Proceedings*, 24(3), August 1990.
- [61] Paul E. Haeberli. Paint By Numbers: Abstract Image Representations. In Forest Baskett, editor, *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 207–214, August 1990.
- [62] Pat Hanrahan and Paul E. Haeberli. Direct WYSIWYG Painting and Texturing on 3D Shapes. In Forest Baskett, editor, *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 215–223, August 1990.
- [63] Pat Hanrahan and Wolfgang Krueger. Reflection from Layered Surfaces due to Subsurface Scattering. *SIGGRAPH 93 Conference Proceedings*, August 1993.
- [64] P. S. Heckbert. A Survey of Texture Mapping Techniques. *IEEE Computer Graphics and Applications*, 6(11), November 1986.
- [65] G. Heflin and G. Elber. Shadow Volume Generation from Free Form Surfaces. In *Communicating with virtual worlds, Proceedings of CGI'93 (Lausanne, Switzerland)*, pages 115–126. Springer-Verlag, June 1993.
- [66] Wolfgang Heidrich. A Model for Anisotropic Reflections in OpenGL. In *SIGGRAPH 98 Conference Abstracts and Applications*, page 267, July 1998.
- [67] Wolfgang Heidrich and Hans-Peter Seidel. View-independent Environment Map. In *Eurographics/SIGGRAPH Workshop on Graphics Hardware*, pages 39–45, September 1998.
- [68] K. P. Herndon, A. VanDam, and M. Gleicher. Workshop Report: The Challenges of 3D Interaction. *SIGCHI Bulletin*, 26(4), 1994.
- [69] Aaron Hertzmann. Painterly Rendering with Curved Brush Strokes of Multiple Sizes. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 453–460. ACM SIGGRAPH, Addison Wesley, July 1998. ISBN 0-89791-999-8.
- [70] Siu Chi Hsu and Irene H. H. Lee. Drawing and Animation Using Skeletal Strokes. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 109–118. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.
- [71] Biederman I. and Kalocsai P. Neurocomputational bases of object and face recognition. *Philosophical Transactions of the Royal Society London: Biological Sciences*, 352, April 1997.
- [72] Victoria Interrante, Henry Fuchs, and Stephen Pizer. Enhancing Transparent Skin Surfaces with Ridge and Valley Lines. *Proceedings of Visualization '95*, pages 52–59, 1995.
- [73] D. Kersten, D. C. Knill, P. Mamassian, and I. Bulthoff. Illusory motion from shadows. *IEEE Computer Graphics and Applications*, 379(31), 1996.
- [74] Lutz Kettner and Emo Welzl. Contour Edge Analysis for Polyhedron Projections. In W. Strasser, R. Klein, and R. Rau, editors, *Geometric Modeling: Theory and Practice*, pages 379–394. Springer Verlag, 1997.
- [75] Leif Kobbelt, K. Daubert, and Hans-Peter Seidel. Ray tracing of subdivision surfaces. In *Eurographics Rendering Workshop '98 Proceedings*, 1998.
- [76] Michael A. Kowalski and Lee Markosian et al. Art-Based Rendering of Fur, Grass, and Trees. In *SIGGRAPH 99 Conference Proceedings*, August 1999.

-
- [77] ViewPoint Data Labs. *LiveArt 98*. Orem, UT, 1998.
- [78] Patricia Lambert. *Controlling Color: A Practical Introduction for Designers and Artists*, volume 1. Everbest Printing Company Ltd., 1991.
- [79] John Lansdown and Simon Schofield. Expressive Rendering: A Review of Nonphotorealistic Techniques. *IEEE Computer Graphics and Applications*, 15(3):29–37, May 1995.
- [80] Rev Lebaredian. Traditional Cel Animation Look with 3D Renderers. *Siggraph 96 Visual Proceedings, ACM SIGGRAPH, ACM Press*, 1996.
- [81] Yuencheng Lee, Demetri Terzopoulos, and Keith Waters. Realistic Modeling for Facial Animation. *SIGGRAPH 95 Conference Proceedings*, 1995.
- [82] Informatix Software International Limited. Piranesi.
- [83] Peter Litwinowicz. Inkwell: A 2-D Animation System. *Computer Graphics (Proc. Siggraph), ACM SIGGRAPH, ACM Press*, 25(4), 1991.
- [84] Peter Litwinowicz. Processing Images and Video for an Impressionist Effect. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 407–414. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.
- [85] LightWork Design Ltd. Kazoo.
- [86] Harrington Mann. *The Technique of Portrait Painting*. Seeley Service & Co, Ltd., London, 1933.
- [87] Lee Markosian, Michael A. Kowalski, Samuel J. Trychin, Lubomir D. Bourdev, Daniel Goldstein, and John F. Hughes. Real-Time Nonphotorealistic Rendering. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 415–420. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.
- [88] Judy Martin. *Technical Illustration: Materials, Methods, and Techniques*, volume 1. Macdonald and Co Publishers, 1989.
- [89] Maic Masuch, Stefan Schlechtweg, and Bert Schönwälder. dali! – Drawing Animated Lines! In O. Deussen and P. Lorenz, editors, *Simulation und Animation '97*, pages 87–96, Erlangen, Ghent, 1997. SCS Europe.
- [90] Maic Masuch, Lars Schumann, and Stefan Schlechtweg. Animating Frame-to-Frame Consistent Line Drawings for Illustrative Purposes. In Peter Lorenz and Bernhard Preim, editors, *Simulation und Animation '98*, pages 101–112, Erlangen, Ghent, 1998. SCS Europe.
- [91] Scott McCloud. *Understanding Comics*. Tundra Publishing Ltd., Northhampton, MA, 1993.
- [92] Pamela McCorduck. *AARON's CODE: Meta-Art, Artificial Intelligence, and the Work of Harold Cohen*. W. H. Freeman and Co., New York, 1991.
- [93] Barbara J. Meier. Painterly Rendering for Animation. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 477–484. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.
- [94] Jackie Neider, Tom Davis, and Mason Woo. *OpenGL Programming Guide*. Addison-Wesley Publishing Company, 1993.
- [95] Jose M. Parramon. *The Book of Color*. Watson-Guptill Publications, New York, NY, 1993.
- [96] PartNet. <http://www.partNet.com/>. 423 Wakara Way Suite 216 Salt Lake City, Utah 84108, 1998.
- [97] Ken Perlin and Luiz Velho. A Wavelet Representation for Unbounded Resolution Painting. Technical report, New York University, New York, 1992.
- [98] Ken Perlin and Luiz Velho. Live Paint: Painting With Procedural Multiscale Textures. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 153–160. ACM SIGGRAPH, Addison Wesley, August 1995. held in Los Angeles, California, 06-11 August 1995.
- [99] Binh Pham. Expressive Brush Strokes. *CVGIP*, 53(1):1–6, January 1991.
- [100] Bui-Tuong Phong. Illumination for Computer Generated Images. *Communications of the ACM*, 18(6):311–317, June 1975.
- [101] F.P. Preparata and M.I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, 1985.
- [102] Ramesh Raskar and Michael Cohen. Image Precision Silhouette Edges. In *Proc. 1999 ACM Symposium on Interactive 3D Graphics*, April 1999.

-
- [103] P. Richens. Does knowledge really help? *Knowledge-Based Computer-Aided Design*, pages 305–325, 1994.
 - [104] Paul Richens and Simon Schofield. Interactive Computer Rendering. *Architecture Review Quarterly*, 1, Autumn 1995.
 - [105] Barbara Robertson. Different Strokes. *Computer Graphics World*, December 1997.
 - [106] Barbara Robertson. Brushing Up on 3D Paint. *Computer Graphics World*, April 1998.
 - [107] Tom Ruppel, editor. *The Way Science Works*, volume 1. MacMillan, 1995.
 - [108] Takafumi Saito and Tokiichiro Takahashi. Comprehensible Rendering of 3-D Shapes. In *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 197–206, August 1990.
 - [109] Michael P. Salisbury, Sean E. Anderson, Ronen Barzel, and David H. Salesin. Interactive Pen-And-Ink Illustration. In *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 101–108. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.
 - [110] Michael P. Salisbury, Michael T. Wong, John F. Hughes, and David H. Salesin. Orientable Textures for Image-Based Pen-and-Ink Illustration. In *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 401–406. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.
 - [111] Mike Salisbury, Corin Anderson, Dani Lischinski, and David H. Salesin. Scale-Dependent Reproduction of Pen-and-Ink Illustrations. In *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 461–468. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.
 - [112] T. Sanocki, K. Bowyer, M. Heath, and S. Sarkar. Are real edges sufficient for object recognition? *Journal of Experimental Psychology: Human Perception and Performance*, 24(1):340–349, January 1998.
 - [113] Tsuyoshi T. Sasada. Drawing Natural Scenery by Computer Graphics. *Computer-Aided Design*, 19(4), May 1987.
 - [114] Simon Schofield. *Non-photorealistic Rendering*. PhD thesis, Middlesex University, England, 1994.
 - [115] Simon Schofield. Piranesi: A 3-D Paint System. *Eurographics UK 96 Conference Proceedings*, 1996.
 - [116] Peter Schröder and Denis Zorin, editors. *Subdivision for Modeling and Animation*. SIGGRAPH 99 Course Notes, 1999.
 - [117] Dorée Duncan Seligmann and Steven Feiner. Automated generation of intent-based 3D illustrations. In Thomas W. Sederberg, editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 123–132, July 1991.
 - [118] Alvy Ray Smith. Paint. In Beatty and Booth, editors, *IEEE Tutorial on Computer Graphics*, pages 501–515. IEEE Computer Society Press, second edition, 1982.
 - [119] Alvy Ray Smith. Digital Paint Systems Historical Overview. Microsoft Corporation, May 1997.
 - [120] Hajime Sorayama. *Sorayama Hyper Illustrations*. Books Nippan, Tokyo, 1989.
 - [121] Steve Strassmann. Hairy Brushes. In David C. Evans and Russell J. Athay, editors, *Computer Graphics (SIGGRAPH '86 Proceedings)*, volume 20, pages 225–232, August 1986.
 - [122] Christine Strothotte and Thomas Strothotte. *Seeing between the pixels: pictures in interactive systems*. Springer-Verlag, Berlin, 1997.
 - [123] Ivan Sutherland. Sketchpad: A Man-Machine Graphical Communication System. In *Proc. AFIPS Spring Joint Computer Conference*, pages 329–346, Washington, D.C., 1963. Spartan Books.
 - [124] Adobe Systems. Adobe Photoshop.
 - [125] Daniel Teece. 3D Painting for Non-Photorealistic Rendering. In *SIGGRAPH 98: Conference Abstracts and Applications*, page 248, 1998.
 - [126] Daniel Teece. *Three Dimensional Interactive Non-Photorealistic Rendering*. PhD thesis, University of Sheffield, England, 1998.
 - [127] Bosco S. Tjan, Wendy L. Braje, Gordon E. Legge, and Daniel Kersten. Human Efficiency for Recognizing 3-D Objects in Luminance Noise. *Vision Research*, 35(21):3053–3069, 1995.
 - [128] Xaos Tools. Paint Alchemy.
 - [129] S. M. F. Treavatt and M. Chen. Statistical Techniques for the Automated Synthesis of Non-Photorealistic Images. In *Proc. 15th Eurographics UK Conference*, March 1997.

-
- [130] Edward Tufte. *Visual Explanations*. Graphics Press, 1997.
 - [131] Greg Turk and David Banks. Image-Guided Streamline Placement. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 453–460. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.
 - [132] W. D. van Bakergen and G. Obata. Free-hand plotting - is it Live or is it Digital. *CAD Futures 91, Vieweg, Wiesbaden*, 1991.
 - [133] Bruce Walter, Gun Alppay, Eric P. F. Lafontaine, Sebastian Fernandez, and Donald P. Greenberg. Fitting Virtual Lights for Non-Diffuse Walkthroughs. In *SIGGRAPH 97 Conference Proceedings*, pages 45–48, August 1997.
 - [134] Leonard R. Wanger, James A. Ferwerda, and Donald P. Greenberg. Perceiving Spatial Relationships in Computer-Generated Images. *IEEE Computer Graphics and Applications*, 12(3):44–58, May 1992.
 - [135] Colin Ware. Color Sequences for Univariate Maps: Theory, Experiments, and Principles. *IEEE Computer Graphics & Applications*, 8(5):41–49, 1988.
 - [136] Ruth E. Weiss. BE VISION, a Package of IBM 7090 FORTRAN Programs to Drive Views of Combinations of Plane and Quadric Surfaces. *Journal of the ACM*, 13(4):194–204, April 1966.
 - [137] Jane Wilhelms and Allen Van Gelder. Anatomically Based Modeling. *SIGGRAPH 97 Conference Proceedings*, 1997.
 - [138] Lance Williams. Pyramidal Parametrics. In Peter Tanner, editor, *SIGGRAPH 83 Conference Proceedings*, pages 1–11, July 1983.
 - [139] Lance Williams. Performance-Driven Facial Animation. *Computer Graphics*, 24(4), August 1990.
 - [140] Lance Williams. Shading in Two Dimensions. *Graphics Interface '91*, pages 143–151, 1991.
 - [141] Stephen Wilson. Artificial intelligence research as art. *SEHR*, 4(2), 1995.
 - [142] Georges Winkenbach and David H. Salesin. Computer-Generated Pen-And-Ink Illustration. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 91–100. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.
 - [143] Georges Winkenbach and David H. Salesin. Rendering Parametric Surfaces in Pen and Ink. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 469–476. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.
 - [144] Michael T. Wong, Douglas E. Zongker, and David H. Salesin. Computer-Generated Floral Ornament. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 423–434. ACM SIGGRAPH, Addison Wesley, July 1998. ISBN 0-89791-999-8.
 - [145] Daniel N. Wood, Adam Finkelstein, John F. Hughes, Craig E. Thayer, and David H. Salesin. Multiperspective Panoramas for Cel Animation. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 243–250. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.
 - [146] Lee H. Wurm, Gordon E. Leffe, Lisa M. Isenberg, and Andrew Luebker. Color Improves Object Recognition in Normal and Low Vision. *Journal of Experimental Psychology: Human Perception and Performance*, 19(4):899–911, 1993.
 - [147] Chris Yessios. Computer Drafting of Stones, Wood, Plant and Ground Material. *Computer Graphics, ACM Press*, 13, 1979.
 - [148] Yizhou Yu and Jitendra Malik. Recovering Photometric Properties of Architectural Scenes from Photographs. *SIGGRAPH 98 Conference Proceedings*, July 1998.
 - [149] Robert C. Zeleznik, Kenneth P. Herndon, and John F. Hughes. SKETCH: An Interface for Sketching 3D Scenes. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 163–170. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.
 - [150] H. Zhang and K. Hoff III. Fast backface culling using normal masks. In *Proc. 1997 Symposium on Interactive 3D Graphics*, pages 103–106, April 1997.
 - [151] Song Chun Zhu and David Mumford. GRADE: Gibbs Reaction And Diffusion Equations — a framework for pattern synthesis, image denoising, and removing clutter. In *Proc. ICCV 98*, 1998.