INTERACTIVE NON-PHOTOREALISTIC TECHNICAL ILLUSTRATION

by

Amy Gooch

A thesis submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computer Science

The University of Utah

December 1998

Copyright © Amy Gooch 1998

All Rights Reserved

THE UNIVERSITY OF UTAH GRADUATE SCHOOL

SUPERVISORY COMMITTEE APPROVAL

of a thesis submitted by

Amy Gooch

This thesis has been read by each member of the following supervisory committee and by majority vote has been found to be satisfactory.

	Chair: Peter Shirley
	Elaine Cohen
	Rich Riesenfeld

THE UNIVERSITY OF UTAH GRADUATE SCHOOL

FINAL READING APPROVAL

To the Graduate Council of the University of Utah:

I have read the thesis of Amy Gooch in its final form and have found that (1) its format, citations, and bibliographic style are consistent and acceptable; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the Supervisory Committee and is ready for submission to The Graduate School.

Date

Peter Shirley
Chair, Supervisory Committee

Approved for the Major Department

Robert Kessler
Chair/Dean

Approved for the Graduate Council

David S. Chapman
Dean of The Graduate School

ABSTRACT

Current interactive modeling systems allow users to view models in wireframe or Phong-shaded images. However, the wireframe is based on the model's parameterization, and a model's features may get lost in a nest of lines. Alone, a fully rendered image may not provide enough useful information about the structure or model features. Human technical illustrators follow certain visual conventions that are unlike Phong-shaded or wireframe renderings, and the drawings they produce are subjectively superior to conventional computer renderings. This thesis explores lighting, shading, and line illustration conventions used by technical illustrators. These conventions are implemented in a modeling system to create a new method of displaying and viewing complex NURBS models. In particular, silhouettes and edge lines are drawn in a manner similar to pen-and-ink drawings, and a shading algorithm is used that is similar to ink-wash or air-brush renderings for areas inside the silhouettes. This shading has a low intensity variation so that the black silhouettes remain visually distinct, and it has a cool-to-warm hue transition to help accent surface orientation. Applying these illustration methods produces images that are closer to human-drawn illustrations than is provided by traditional computer graphics approaches.

To Bruce,
for all of the love and support.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

Thanks to Bruce Gooch, Peter Shirley, Elaine Cohen, Rich Riesenfeld, Susan and Tice Ashurst, Georgette Zubeck, David Johnson, Bill Martin, Richard Coffey, Colette Mullenhoff, Brian Loss, Matt Kaplan, Tom Thompson, Russ Fish, Mark Bloomenthal, and the rest of the Alpha1 and SCI research groups, past and present. Thanks to Lee Markosian, Loring Holden, Robert Zeleznik, and Andrew Forsberg, from Brown University, for letting me collaborate on the JOT Project. Also, thanks to Jason Herschaft for the dinosaur claw model. This work was supported in part by DARPA (F33615-96-C-5621) and/or the NSF Science and Technology Center for Computer Graphics and Scientific Visualization (ASC-89-20219). All opinions, findings, conclusions, or recommendations expressed in this document are mine and do not necessarily reflect the views of the sponsoring agencies.

CHAPTER 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 realistic 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, conceived and refined over several centuries. Therefore, imitating some of these techniques and exploring the perceptual psychology behind technical illustration are good first steps in going beyond realistic rendering.

In this thesis I explore technical illustrations for a NURBS-based modeling system. A driving force for exploring technical illustration comes from viewing the wireframe representation of complex NURBS-based models, usually a mess of lines, as shown in Figure 1.1. More motivation for exploring illustration techniques is provided by comparing the two mechanical part images in Figure 1.2. The first image is a hand-tuned, computer

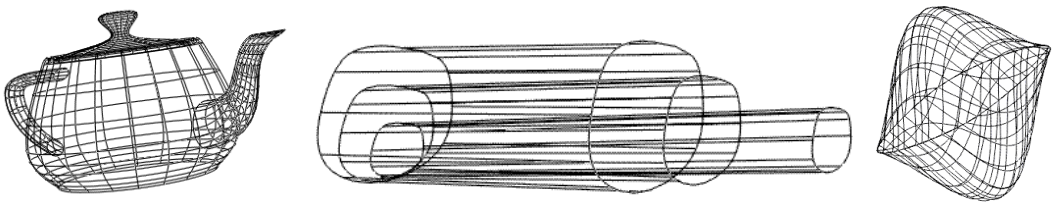
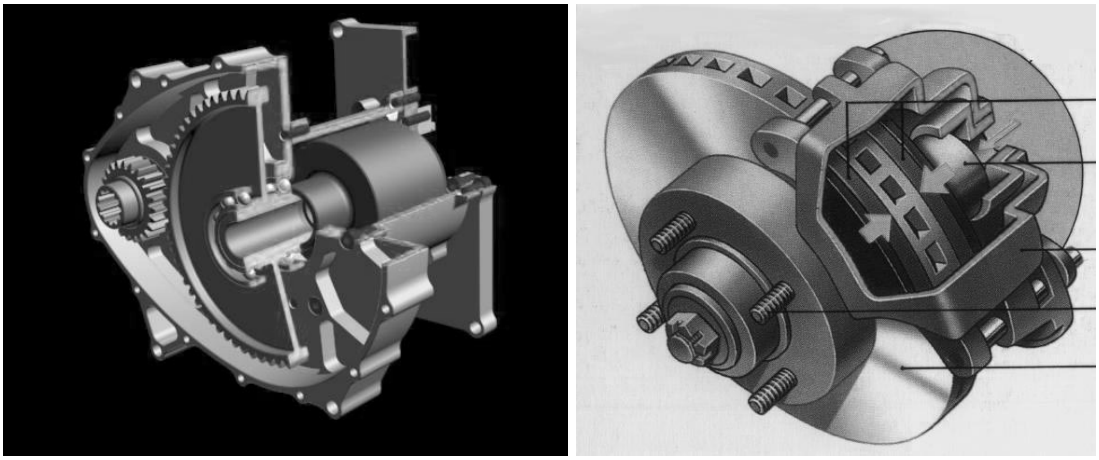


Figure 1.1. An few examples of a NURBS-based model displayed in wireframe. The large number of isolines makes distinguishing key features difficult.



(a) Hand-tuned Phong-rendered image of mechanical part by Dr. Sam Drake. It took him approximately six hours to hand tune this image.

(b) Illustration of a mechanical part using lines to separate parts and shading to convey material properties. Courtesy Macmillan Reference USA, a Simon & Schuster Macmillan Company [28].

Figure 1.2. Hand-tuned Phong-shaded image versus technical illustration.

generated, Phong-shaded part, which took about six hours for Professor Sam Drake to complete. The second image is an illustration from the book *The Way Science Works* by Macmillian [28]. The illustration uses lines to separate parts and shading to convey material properties. I would like to be able to automatically generate images with many of the characteristics of the illustration in Figure 1.2(b) for NURBS-based models.

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. In order to create illustration rules for technical illustration, I reviewed several books, e.g., [28, 25], and concluded that in addition to using color to differentiate parts [40], technical illustrators use black lines, as well as a specific type of shading which rarely includes shadows. These two-dimensional (2D) line illustration conventions can be imitated by drawing lines consisting of silhouettes, surface boundaries, and discontinuities. Adding shading completes the image and can be used to convey important nongeometric qualities of an object such as material properties.

Line, shading, and lighting techniques used by illustrators can convey a more accurate representation of shape and material properties of mechanical parts than traditional com-

puter graphics methods. These illustration techniques can improve or replace traditional representation of models such as wireframe or Phong-shaded. In Chapter 2, I review what has been done in computer graphics as well as some of the research on human recognition in perception studies. In Chapter 3, I describe the illustration conventions used for lines and shading. In Chapter 4, I discuss algorithms to imitate lines and shading of technical illustration. I will also discuss how these may or may not change for three-dimensional (3D) interactive illustrations. In Chapter 5, I will discuss the implementation details for 3D illustration, and in Chapter 6 I will draw some conclusions and propose some future research goals.

CHAPTER 2

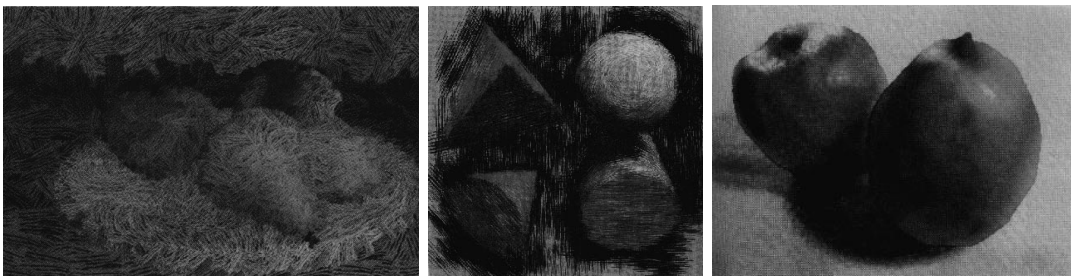
BACKGROUND

Non-photorealistic rendering (NPR) techniques vary greatly in their level of abstraction. In technical illustrations, shape information is valued above realism and aesthetics. Therefore a high level of abstraction, like the images in Figure 2.1, would be inappropriate. As summarized in Table 2.1, no work has been done which uses the ideas and techniques of technical illustrators to generate not only 2D technical illustrations but also to provide an interactive environment for viewing 3D models as 3D technical illustrations. A review of the papers involving NPR or other illustration techniques used in computer graphics reveals that most papers can be partitioned into two categories: those which generate only aesthetic images and those whose purpose is to convey shape and structure. The images in the latter category may themselves be aesthetically pleasing, but this is a side effect rather than a primary goal. These papers can also be further divided into those that generate a single image and those that are part of an interactive or animation system. Human perception and recognition studies [2, 4, 7] are another valuable source of information. These perception studies suggest an explanation of why line drawings, like technical illustrations, are enough for object recognition and provide a hint as to why they may also provide shape information.

2.1 Paint Programs and One-Shot Images

Creating sophisticated paint programs which generate single images and emulate techniques used by artists for centuries was the focus of research done by Meier [27], Haeberli [17], Curtis [10], Salisbury et al. [30, 31, 32], and Winkenbach et al. [38]. However, conveying shape and structure is not the goal of these images.

Meier [27] presents a technique for rendering animations in which every frame looks as though it were painted by an artist. She models the surfaces of 3D objects as 3D particle sets. The surface area of each triangle is computed, and particles are randomly distributed. The number of particles per triangle is equal to a ratio of the surface area



(a) Copyright 1996 Barbara Meier [27]. Used by permission.

(b) Copyright 1990 Paul Haeberli [17]. Used by permission.

(c) Copyright 1997 Cassidy Curtis [10]. Used by permission.

Figure 2.1. Non-photorealistic one-shot images with a high level of abstraction.

Table 2.1. Summary of non-photorealistic and other computer graphics papers.

Author	*Line Drawing*	*Shading*	*Automatic (Not user-driven)*	*3D Interactive*	*Applicable to Technical Illustration*	*Additional Illustration Rules**
Markosian [24]	√		√	√	√	
Dooley [11]	√	√			√	√
Saito [29]	√	√	√		√	
Driskill [12]			√	√		
Elber [13]	√		√		√	
Seligmann [34]		√	√		√	√
Land [22]	√		√		√	
Gooch [15]	√	√	√		√	
Zeleznik [42]		√		√	√	
Salisbury [32, 31]	√				√	
Salisbury [30]	√		√		√	
Winkenbach [38]	√		√		√	
Winkenbach [39]	√		√		√	
Meier [27]		√	√			
Haeberli [17]					√	
Litwinowicz [23]						
Curtis [10]		√				
This Thesis	√	√	√	√	√	

Note: "Line drawing" and "shading" categories are checked if the work uses shading and line drawing conventions similar to traditional illustration techniques. "Automatic" means that user intervention is not required and it is not a user-driven program; i.e., it is not like a paint program.

*Layout, cut-a-ways and object transparency, line style, etc.

of triangle to the surface area of the whole object. To maintain coherence from one shot of the object to the next, the random seed is stored for each particle. Particles are transformed into screen space and sorted with regard to the distance from viewer. The particles are then painted with 2D paint strokes, starting farthest from viewer, moving forwards, until everything is painted. The user determines artistic decisions like light, color, and brush stroke, similar to most paint programs. The geometric lighting properties of the surface control the appearance of the brush strokes.

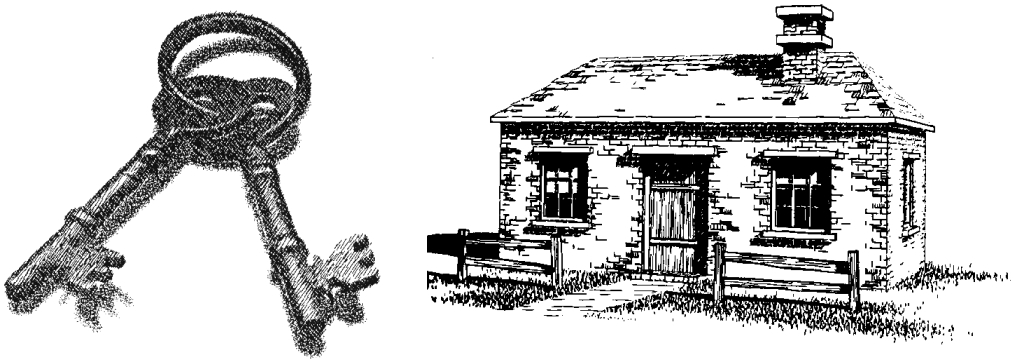
Haeberli [17] created a paint program which allows the user to manipulate an image using tools that can change the color and size of a brush, as well as the direction and shape of the stroke. The goal of his program is to allow the user to communicate surface color, surface curvature, center of focus, and location of edges, as well as eliminate distracting detail, to provide cues about surface orientation and to influence viewer's perception of the subject. Haeberli studied the techniques of several artists. He observed that traditional artists exaggerate important edges. Where dark areas meet light areas, the dark region is drawn darker and light region is drawn lighter. This causes the depth relationship between objects in a scene to be more explicit where they overlap. Haeberli also notes that artists use color to provide depth cues because, perceptually, green, cyan, blue (cool-colored) shapes recede, whereas red, orange, yellow, magenta (warm-colored) objects advance. He commented in his paper that he used these color depth cues and other techniques to enhance digital images before the paint begins, but he never provided any details on how these could be used algorithmically.

The computer-generated watercolor work by Curtis et al. [10] created a high-end paint program which generates pictures by interactive painting, or automatic image "watercolorization" or 3D non-photorealistic rendering. Given a 3D geometric scene, they generate mattes isolating each object and then use the photorealistic rendering of the scene as the target image. The authors studied the techniques and physics of watercolor painting to developed algorithms, which depend on the behavior of the paint, water, and paper. They provide information on watercolor materials and effects of dry-brush, edge darkening, backruns, granulation and separation of pigments, flow patterns, glazing, washes.

Salisbury et al. [32, 30, 31] designed an interactive system which allows users to paint with stroke textures to create pen-and-ink style illustrations, as shown in Figure 2.2(a). Using "stroke textures," the user can interactively create images similar to pen-and-

ink drawings of an illustrator by placing the stroke textures. Their system supports scanned or rendered images which the user can reference as guides for outline and tone (intensity) [32]. In their paper, "Scale-Dependent Reproduction of Pen-and-Ink Illustrations" [30], they gave a new reconstruction algorithm that magnifies the low-resolution image while keeping the resulting image sharp along discontinuities. Scalability makes it really easy to incorporate pen-and-ink style image in printed media. Their "Orientable Textures for Image-Based Pen-and-Ink Illustration" [31] paper added high-level tools so the user could specify the texture orientation as well as new stroke textures. The end result is a compelling 2D pen-and-ink illustration.

Winkenbach et al. [38] itemized rules and traditions of hand-drawn black-and-white illustrations and incorporated a large number of those principles into an automated rendering system. To render a scene, visible surfaces and shadow polygons are computed. The polygons are projected to normalized device coordinate space and then used to build a 2D BSP (binary space partition) tree and planar map. Visible surfaces are rendered, and textures and strokes applied to surfaces using set operations on the 2D BSP tree. Afterwards, outline strokes are added. Their system allows the user to specify where the detail lies. They also take into consideration the viewing direction of user, in addition to the light source. They are limited by a library of "stroke textures." Their process takes about 30 minutes to compute and print out the resulting image, as shown in Figure 2.2(b).



(a) Pen-and-Ink Illustration. Copyright 1996 Michael Salisbury [30]. Used by permission.

(b) Pen-and-Ink Illustration. Copyright 1994 Georges Winkenbach [38]. Used by permission.

Figure 2.2. Computer-generated pen-and-ink illustrations.

2.2 One-Shot Images Conveying Shape

The research reviewed in the previous section concentrated on generating aesthetically pleasing images. The work done by Seligmann and Feiner [34], Winkenbach et al. [39], Saito et al. [29], Land et al. [22], Elber [13], and Dooley et al. [11] generated images in which the primary goal is to convey shape information. However, these techniques generate single images and do not allow user interaction.

Seligmann and Feiner [34] created a system based on the idea that an illustration is a picture that is designed to fulfill communicative intent. They assert that the purpose of illustration is to show an object's material, size, orientation, and, perhaps, how to use it. The purpose is not only to display geometric and material information but to also provide the viewer information about the object's features, physical properties, and abstract properties. "Illustration objects are generated based on both the representation of the physical object as well as the communicative intent"(p. 131), i.e., the images must convey the geometric characteristics as well as the purpose of each of the objects, such as which way to turn a dial on an image of a radio. Their "Intent-Based Illustration System" (IBIS) uses a generate-and-test approach to consider how the final illustration will look. For each illustration, there are several methods and several evaluators. By performing the permutations of the methods and then evaluating them by the "rules," IBIS automatically generates the image that would look "best."

Winkenbach et al. [39] renders a single pen-and-ink style image for parametric free-form surfaces, using "controlled-density hatching" in order to convey tone (intensity), texture and shape, as shown in Figure 2.3. Their paper provides a highly detailed algorithm on drawing lines (strokes) which gradually disappear in light areas of the surface or where too many lines converge. They use a planar map constructed from the parametric surfaces to clip strokes and generate outlines. The planar map is not constructed from 3D

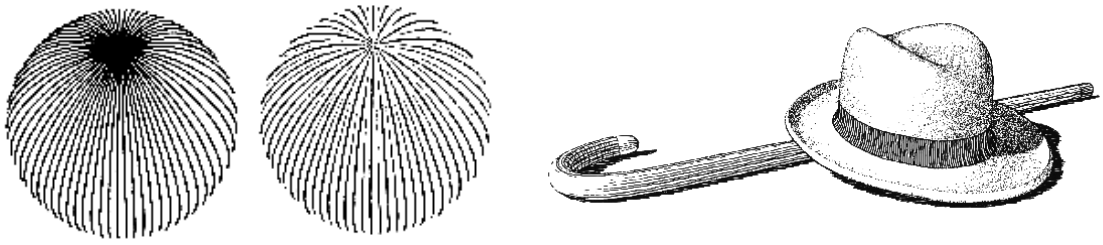


Figure 2.3. One-shot computer-generated pen-and-ink images conveying shape by Winkenbach. Copyright 1996 Georges Winkenbach [39]. Used by permission.

BSP Trees, but by the following method. They tessellate every object and then compute the higher-resolution piecewise linear approximations for all silhouette curves of meshed objects, similar to Elber and Cohen [13], whose work is discussed in Section 2.3. The planar map is created by determining which mesh faces are closest to the view. They then use 2D BSP trees to implement shadows [6].

Saito and Takahashi [29] 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, as shown in Figure 2.4. They propose "new rendering techniques to produce 3D images with enhanced visual comprehensibility," realized with 2D image processing. They construct a data structure called G-buffer, which preserves a set of geometric properties. If shapes and camera parameters are fixed, any combination of enhancement can be examined without changing the contents of the G-buffer.

Land and Alferness [22] present a method for rendering 3D geometric models as black and white drawings. They compute Gaussian and mean surface curvatures of objects and allow the user to threshold, combine, and modify these curvatures. They produce images that contain shape information that is independent of orientation or illumination. They state that, perceptually, humans are good at inferring shape from line drawings, "Lines which approximate lines of curvature may be particularly effective indicators for humans"(p. 2).

Elber [13] provides surface information with four types of curves: the surface boundary curves, curves along C^1 discontinuities in the surface, isoparametric curves, and silhouette curves, as shown in Figure 2.5(a). All of the above curves except silhouette curves are

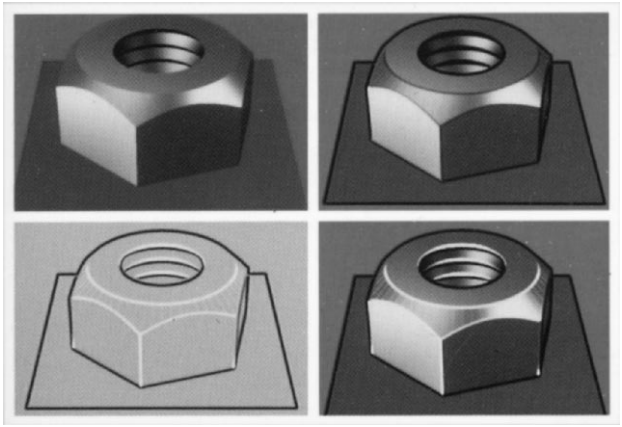
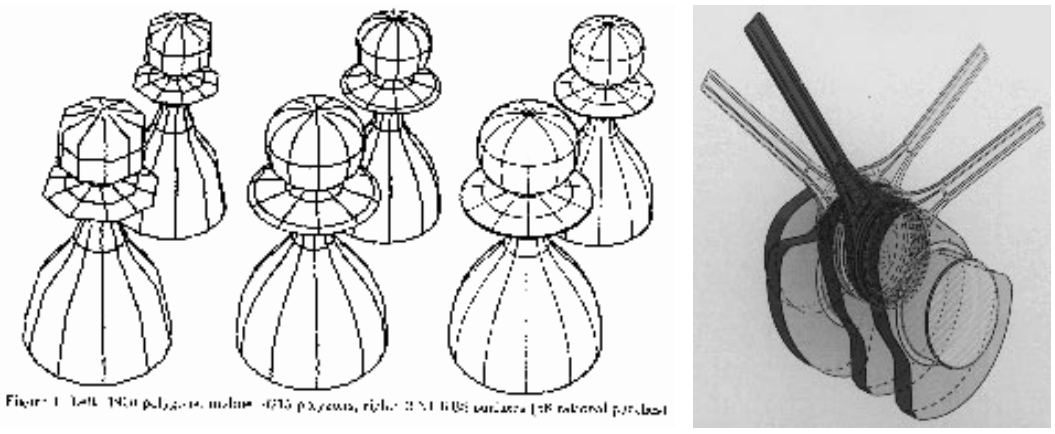


Figure 2.4. Another example of one-shot image conveying shape. Saito enhances a shaded model by drawing discontinuity and contour lines. Copyright 1990 Saito [29].

view-independent and only need to be calculated once per model. Silhouette curves are calculated by normalizing the view orientation so that the view is on the positive z-axis at infinity and the image is projected onto the plane z=0. Elber defines a silhouette point as a point on the surface whose normal has a zero z-component. The silhouette curve of the surface becomes the set of silhouette points forming a continuous curve. When a C^1 discontinuity is found in a surface, Elber divides the surface into two surfaces. Elber's methods cannot be applied directly in an interactive system, because the method uses costly ray-surface intersection calculations to determine visibility. I build upon his observations, using a different method to calculate silhouettes in order to achieve interactive rates.

Dooley and Cohen [11] created an illustration system which used display primitives, such as transparency, lines with variable width, and boundary/end point conditions, as shown in Figure 2.5(b). Visibility information is gathered by ray tracing, which helps to communicate structure and illuminate unnecessary details and clutter. By establishing a user-defined hierarchy of components, users can define not only what they want to see but how they want to see it. However, in their implementation the camera model is then generated and for the rest of the process remains fixed. Most of time is spent ray tracing to gather visibility information, which is done separately for lines and surfaces. After the information on lines and surfaces is put together, illustration primitives are created,



(a) Copyright 1990 Gershon Elber [13]. Used by permission.

(b) Copyright 1990 Debra Dooley [11]. Used by permission.

Figure 2.5. One-shot images conveying shape by Dooley and Elber.

placed in an image buffer, and read by a scan-line renderer. No measurements of the time it took to generate the illustrations were given. The result is a 2D illustrated image which cannot be manipulated like a 3D model.

2.3 Interactive Techniques

The batch-oriented systems presented previously lack the ability for the user to interactively change the viewpoint. There are only a few systems which allow the user to manipulate the viewpoint and the environment.

The Sketch system developed by Zeleznik et al. [42] uses gestures to create and manipulate polygonal shapes, "bridging the gap between hand sketches and computer-based modeling programs." The emphasis of their system is creating and editing polygonal objects.

Driskill [12] explored illustrating exploded views of assembly with minimal user intervention through a 3D, interactive display. She listed some basic illustration rules as they apply to exploded views; however, she was less concerned with how the model appeared, since her emphasis was conveying relationships between parts.

Markosian et al. [24] developed a real-time 3D interactive system for illustrating non-self-intersecting polygon mesh-based models, as seen in Figure 2.6. Their basic scheme is to use probabilistic identification of silhouette edges, interframe coherence of silhouette edges, and improvements and simplifications in Appel's hidden-line algorithm [1], a method based on the notion of quantitative invisibility which keeps track of front facing

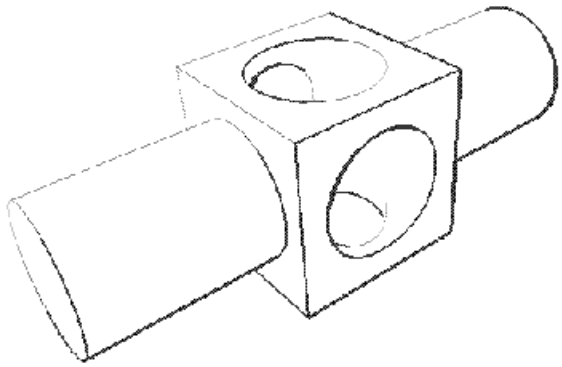


Figure 2.6. Image from a frame of Markosian et al. [24] real-time 3D interactive system for illustrating non-self-intersecting polygon mesh-based models. Copyright 1997 Lee Markosian. Used by permission.

polygons dependent upon the camera's position. However, using randomized checks for silhouettes causes problems with frame-to-frame coherence as well as introducing the risk of missing new silhouettes. They also had to add some extra tests to deal with silhouettes that cusp. They view these possible silhouette misses as less important in a real-time system. Using techniques based on economy of line, they convey information with few strokes, displaying silhouette edges and certain user-chosen key features, such as creases. In addition, they added options for sketchy lines or hatched shading strokes. The end result is a 3D interactive environment, where a single frame looks like an artist's sketch. They achieved their real-time interaction by using the silhouette calculated in the previous frame to guess at which lines are to be shown in the next. Their methods are only applicable for polygonal models and do not convey material properties.

2.4 Perception Studies

In computer graphics there has been very little work on quantifying claims like "Image 1 conveys more shape information than Image 2." However, perceptual psychologists have performed numerous studies and experiments, trying to learn about human recognition and the way we visually process information. Perception studies can help to provide a quantitative analysis instead of merely giving an educated hypothesis on the effectiveness of an image to convey shape information. Studies by Christou et al. [7], Braje et al. [4], and Biederman et al. [2] support the use of line drawings as an effective method for providing substantial shape information.

A study by Christou et al. [7] showed four scenes to subjects. Each scene was composed of a number of planar, cylindrical, and ellipsoidal surfaces. One scene contained shaded surfaces (similar to Phong shading); another scene with textured objects; a scene which only included contours, the line-drawn edges of the objects; and a scene with contours and textured objects. The subjects were asked to specify the surface attitude, the orientation of the local tangent plane at a point on a surface with respect to the viewing direction, at random points on in the scene. These tests showed that the subjects had improved performance in the scenes containing contours. They concluded, "a few simple lines defining the outline of an object suffice to determine its 3-D structure. The ecological significance of contours is clear. They delineate the different components of complex objects and the different parts of a scene"(p. 712).

Another recognition study by Braje et al. [4] found that humans fail to use much

of the information available to an ideal observer. Their conclusion was that human vision is designed to extract image features, such as contours, that enhance recognition, disregarding most of the other information available.

Biederman et al. [2] concluded that simple line drawings can be identified about as quickly and as accurately as fully detailed, textured, colored photos of the same object with the same viewpoint. The question they tried to answer was whether the presence of gradients made it easier to determine an object's 3D structure over that which can be derived solely by the depiction of an object's edges. They concluded, for example, that one could determine the curvature of a cylinder, planarity of a square, or volumetric characteristics of a nonsense object from a line drawing, without the presence of surface gradients. They noted that instruction materials for assembling equipment are more easily followed when the parts are drawn instead of photographed. Their opinion is that reproduced photographic images typically have insufficient contrast for determining the contours of components. Although it seems that one could modify a photograph to get the necessary contrast, there are other techniques, like cut-a-ways, that cannot be easily accomplished, if at all, with photography.

2.5 Background Summary

Non-photorealistic rendering techniques used in computer graphics vary greatly in their level of abstraction. Those that produce images such as watercolor or pen-and-ink are at a high level of abstraction which would be inappropriate for technical illustration. Using a medium level of abstraction like technical illustration helps to reduce the viewer's confusion by exposing subtle shape attributes and reducing distracting details. Adding interaction gives the user motion cues to help deal with visual complexity, cues that are missing in 2D images. A review of past research reveals that no one has created a 3D interactive environment that takes advantage of shape information given by line drawings and artistic shading, presenting the shape information without the confusion produced by the many lines in a wireframe representation or the limitations of Phong-shaded images.

CHAPTER 3

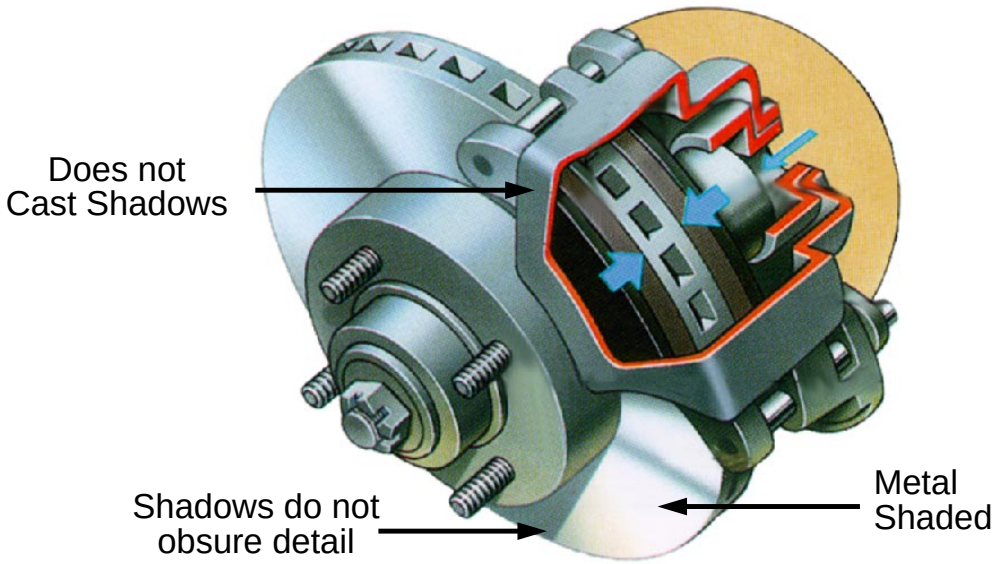
ILLUSTRATION TECHNIQUES

The illustrations in several books, e.g., [25, 28], 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, as shown in Figure 3.1:

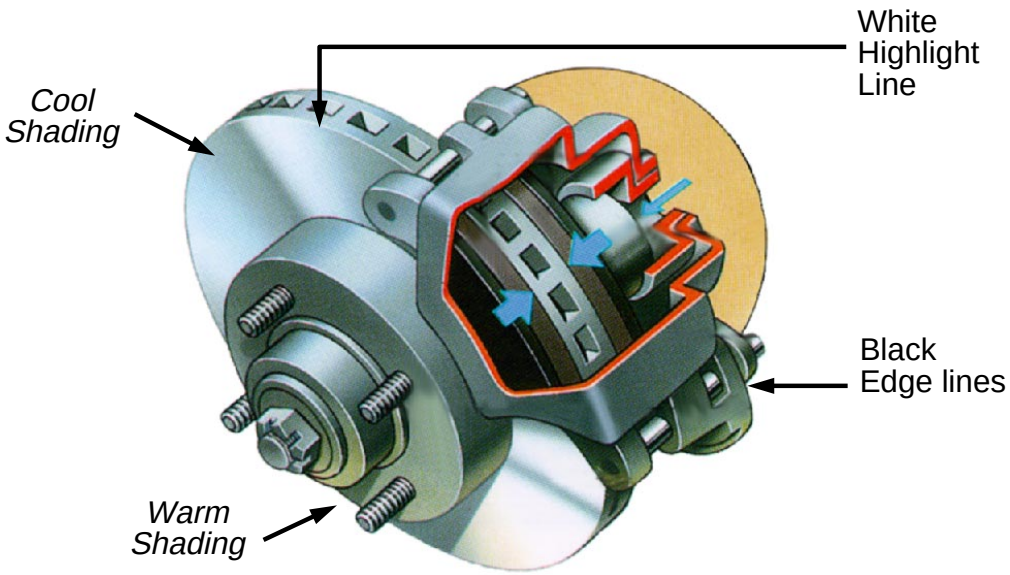
- edge lines are drawn with black curves.
- matte objects are shaded with intensities far from black or white with warmth or coolness of color indicative of surface normal.
- a single light source provides white highlights.
- shadows are rarely included, but if they are used, they are placed where they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

These form only a subset of the conventions used by illustrators. I have concentrated only on the material property and shading aspects of illustration. Work done in computer graphics by Seligmann and Feiner [34] and Dooley and Cohen [11] concentrate on additional aspects of technical illustration like layout, object transparency, and line style.

These 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, 8, 35] 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. [7] 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



(a)



(b)

Figure 3.1. Technical illustration conventions. Copyright 1995 Volvo Car UK Limited [28].

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. As demonstrated by Figure 3.1(a), 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. In Section 3.1, I will discuss the rules, properties, and types of lines needed to convey shape information like the line drawings of technical illustrators. In the next chapter I will discuss implementation details.

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. Figure 3.1(a) provides a good example of how an illustrator uses lights and artistic shading. In the figure, the light is up and to the right of the object and produces highlights as you would expect in a computer-generated Phong-shaded image. However, the illustrator also used cool shading for the upper part of the object with warm shading on the lower, bottom half of the object. This cool and warm shading is not dependent upon the light and may have been used to pull the eye from the cut-a-way to the bottom of the image. Illustrators rarely use shadows in an illustration. As shown in Figure 3.1(b), shadows are used only when they do not obscure details in other parts. Another important characteristic used in technical illustration is the conveyance of material property. Figure 3.1(b) shows how an illustrator alternates bands of light and 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 3.2.

3.1 Lines in Technical Illustration

To decide which lines to draw, I started by analyzing some examples from hand drawn technical illustrations. The illustration in Figure 3.2 consists of just enough lines to separate individual parts and to suggest important features in the shape of each object.

Most NURBS 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

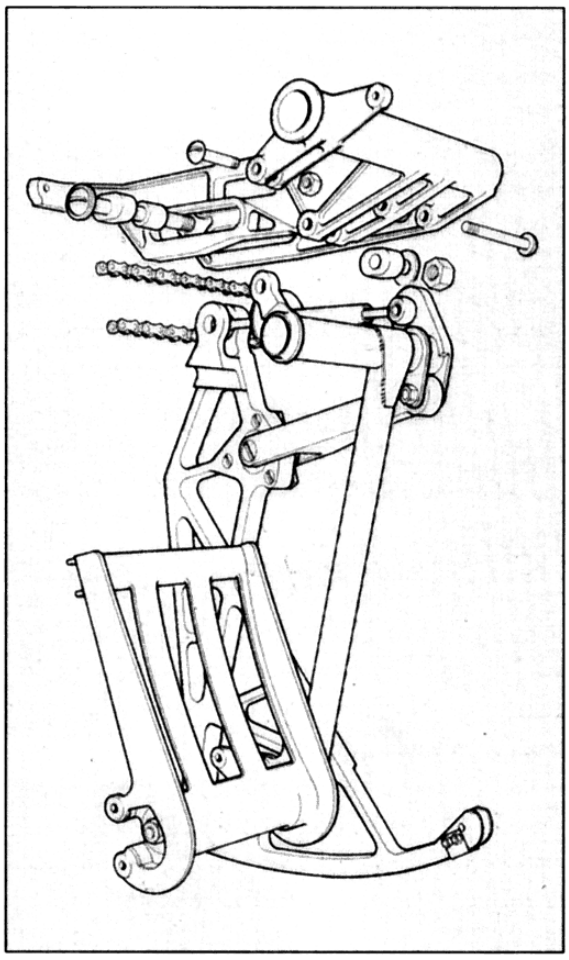


Figure 3.2. An example of the lines an illustrator would use to convey the shape of this airplane foot pedal. Copyright 1989 Macdonald & Co. (Publishers) Ltd. [25].

to the number of lines being displayed. The wireframe of a NURBS surface consists of isolines, which are parameterization dependent. Figure 3.3 demonstrates that changing which isolines are displayed can change the perception of the surface.

By drawing silhouettes, surface boundaries, and discontinuities for a NURBS-based model instead of isolines, one can imitate the lines drawn in technical illustrations without being parameterization dependent. An example of these three different line types is provided in Figure 3.4. Silhouettes contain the set of points on a surface where $E(u,v) \cdot n(u,v) = 0$ or the angle between $E(u,v)$ and $n(u,v)$ is 90 degrees, given a point on a surface, $\sigma(u,v)$, with $E(u,v)$ as the vector from the eye to $\sigma(u,v)$, and $n(u,v)$ as the surface normal (Figure 3.5). Regions where the surface normal changes abruptly, C^1 discontinuities, are also important in defining the shape of an object. Sometimes surface

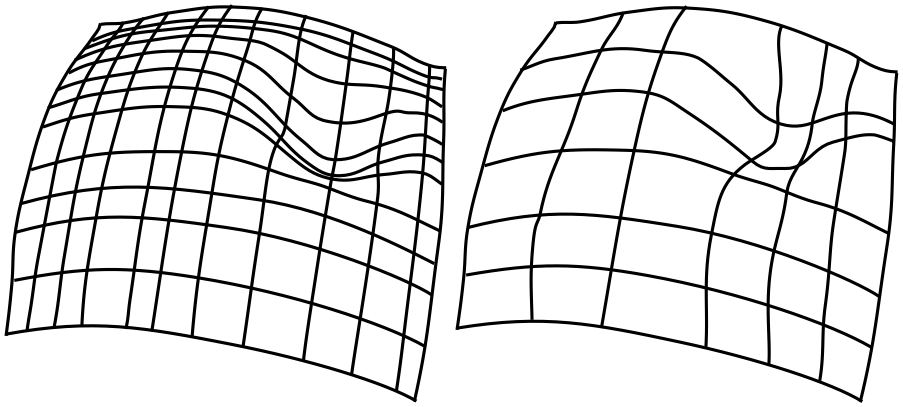


Figure 3.3. Changing which isolines are displayed can change the perception of the surface. The image on the right looks as if it has a deeper pit because the isolines go thru the maximum curvature point on the surface. Images courtesy of David Johnson.

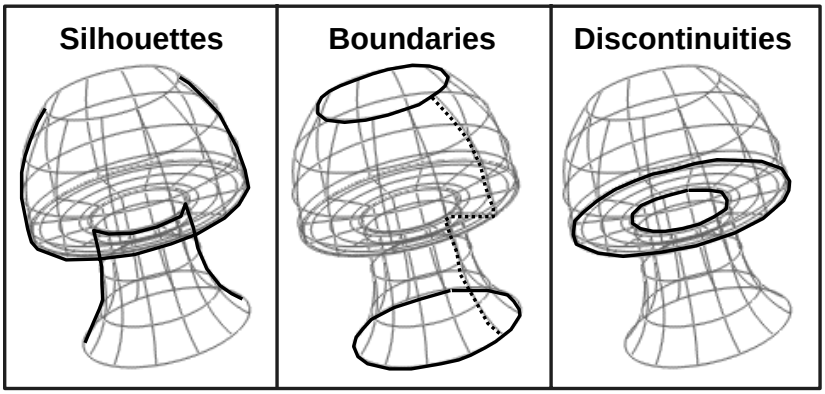


Figure 3.4. Illustrators use lines to separate parts of objects and define important shape characteristics. This set of lines can be imitated for NURBS models by drawing silhouettes, boundaries, and discontinuities, shown above (drawn over the wireframe representation).

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 3.4 should not be drawn, since it is a shared surface boundary [18]. The calculations and implementation details necessary to create these line drawings will be addressed in Chapter 4.

3.1.1 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*,

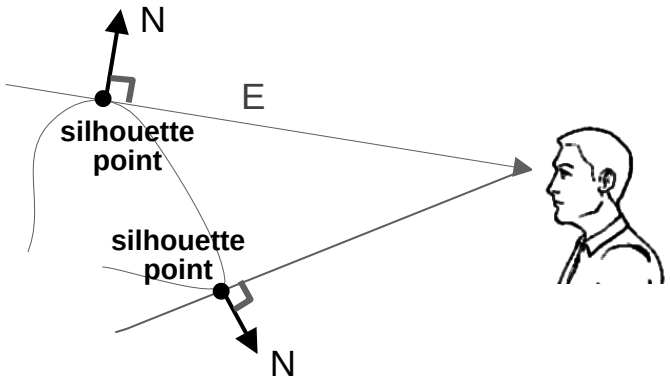


Figure 3.5. Definition of a silhouette: At a point on a surface, $\sigma(u, v)$ and given $E(u, v)$ as the eye vector and $n(u, v)$ as the surface normal, a silhouette point is defined as the point on the surface where $E(u, v) \cdot n(u, v) = 0$ or the angle between $E(u, v)$ and $n(u, v)$ is 90 degrees.

Martin [25] discusses three common conventions, as shown in Figure 3.6:

- 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.

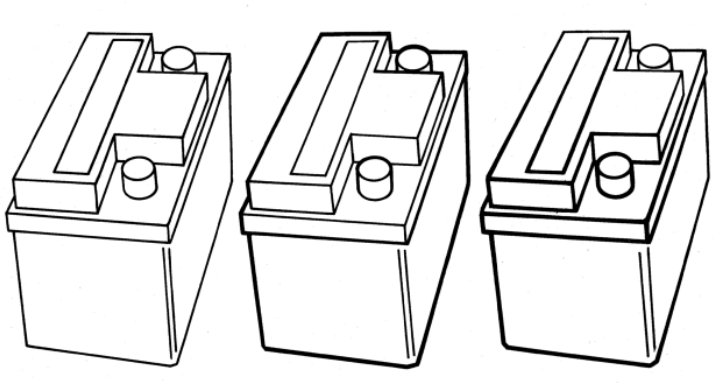


Figure 3.6. Three line conventions suggested by Judy Martin [25]. Left: single line weight used throughout the image. Middle: heavy line weight used for out edges and parts with open space between them. Right: vary line weight to emphasize perspective. Copyright 1989 Macdonald & Co. (Publishers) Ltd. [25].

Other less often used conventions include varying the line weight dependent upon the direction of the light source, giving a shadowed effect or varying the line due to abrupt changes in the geometry (curvature based). However, most illustrators use bold external lines, with thinner interior lines.

3.1.2 Line Color and Shading

In almost all 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 3.7. By using this convention, lines drawn in black and white suggest a light source, and denote the model's orientation. For example, Figure 3.8 shows how an artist may use white for interior lines, producing a highlight.

3.2 Shading in Illustrations

Shading in technical illustration brings out subtle shape attributes and provides information about material properties, as shown in Figure 3.9. 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

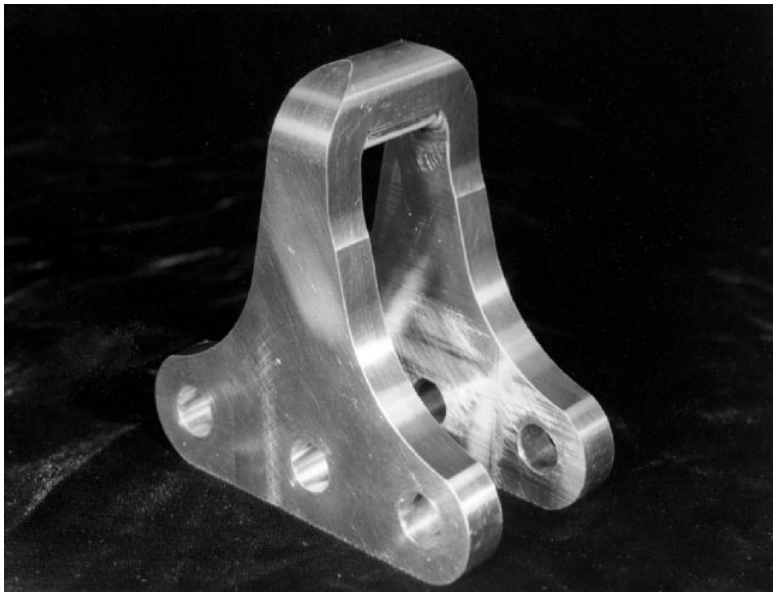


Figure 3.7. This photograph of a metal object shows the anisotropic reflections and the white edge highlights which illustrators sometimes depict.

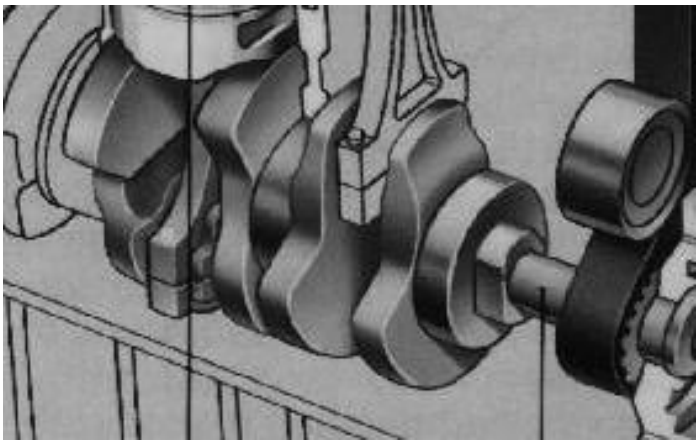


Figure 3.8. Illustrators sometimes use the convention of white interior edge lines to produce a highlight [25].

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 [36]. 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. [15], 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. In the next chapter I will discuss the algorithms used for shading in computer graphics and why it is inadequate for technical illustration. I will also discuss the shading algorithms used by Gooch et al.

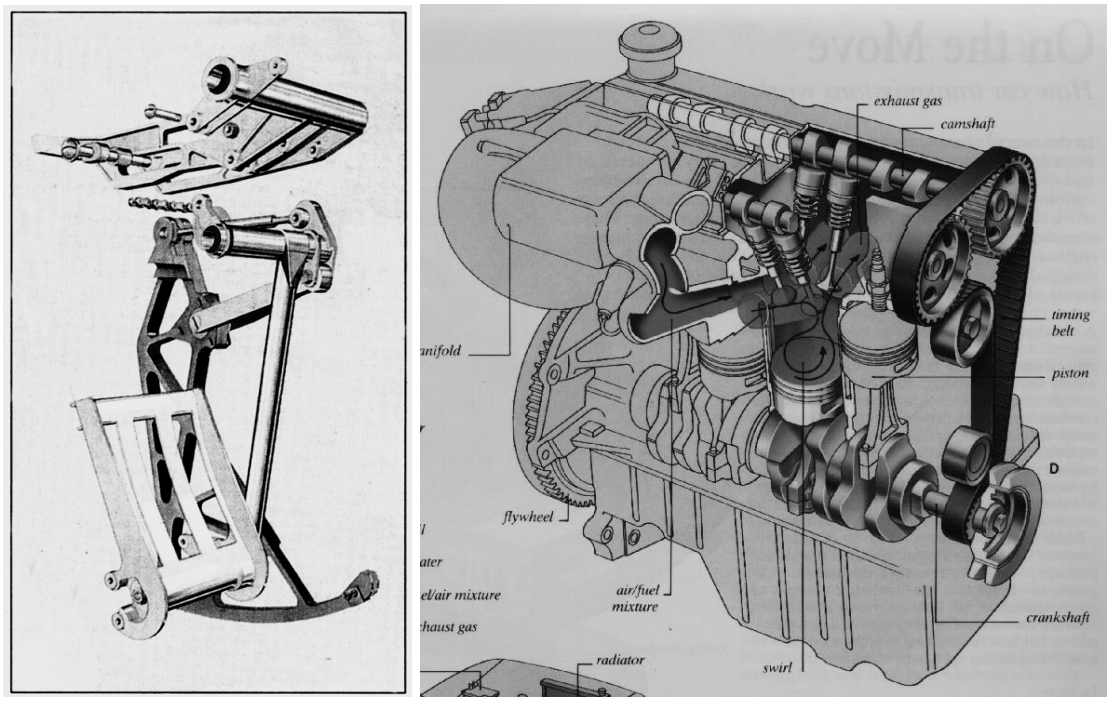


Figure 3.9. Illustrators combine edge lines with a specific type of shading. Shading in technical illustration brings out subtle shape attributes and provides information about material properties. Left: Compare this shaded image of airplane pedal to the line drawing in Figure 3.2. Copyright 1989 Macdonald & Co. (Publishers) Ltd. [25]. Right: Engine. Courtesy Macmillan Reference USA, a Simon & Schuster Macmillan Company [28].

for matte and metal objects.

3.3 Illustration Summary

Technical illustration can be imitated in computer graphics by using black edge lines, a single light source, tone and undertone shading with highlights as presented in Gooch et al., and no shadows. For a NURBS-based model, displaying silhouettes, surface boundaries, and discontinuities provides shape information similar to that of traditional technical illustrations. In the next chapter, I will discuss some algorithms for finding these silhouettes, boundaries, and discontinuities using geometric information of NURBS surfaces. I will also mention some of the other possibilities for generating edge lines for polygonal objects, like the work of Markosian et al., as well as some image processing techniques.

CHAPTER 4

ALGORITHMS FOR ILLUSTRATION

One of the most important issues to address when trying to create illustrations is how to calculate the edge lines. Edge lines for polygonal objects can be generated interactively using the techniques of Markosian et al. [24]. In order to calculate edge lines for higher-order geometric models, like NURBS, the surfaces would have to be tessellated to apply Markosian's technique. On high-end systems, image-processing techniques [29] could be made interactive. In Section 4.1, I will discuss how silhouettes, surface boundaries, and discontinuities can be calculated for NURBS surface, as well as suggest some other techniques for calculating edge lines. After calculating edge lines, the illustrations are completed by considering a new shading model and material properties presented by Gooch et al. [15], summarized in Section 4.2. In Section 4.3, I will discuss the considerations that need to be made to create 3D technical illustrations.

4.1 Algorithms for Finding Edge Lines

Using the geometric information intrinsic to NURBS allows some precalculations. Surface normals are view-independent and can be precalculated, given that it is known which normals one will need. As stated in Section 3.1, in order to imitate the edge lines used in technical illustration for a NURBS model, surface boundaries and discontinuities, as well as silhouettes, need to be drawn. In Section 4.1.1, I will discuss how to find boundaries and discontinuities for NURBS surfaces. In Section 4.1.2, I will define some algorithms for finding silhouettes on NURBS surfaces.

4.1.1 Algorithms for Finding Boundaries and Discontinuities

Surface boundaries and discontinuities are view-independent and only need to be calculated once per model. Boundaries can be found easily from the surface implementation. As discussed in Section 3.1 and Figure 3.4, not all boundaries should be drawn. I define unshared boundaries to mean those surface boundaries which are not shared by any

other surface [18]. Only "unshared" boundaries should be drawn, or in the cases where the joint between two surface boundaries changes abruptly. Discontinuities are due to knot multiplicities and are very simple to extract since they fall along isolines.

4.1.2 Algorithms for Finding Silhouettes for NURBS

Silhouettes are the only view-dependent part. A brute force method will be at interactive so long as the number of surfaces and the amount of silhouette testing are kept reasonable. Defining the bounds on reasonable depends on machine and program speed as well as the number of control points for the NURBS model.

I have explored three methods for finding silhouettes for a given viewpoint. I will define the methods as Mesh Method, Tessellated-Mesh Method, and Srf-Node Method.

4.1.2.1 Some Definitions

Let:

$$\begin{aligned} \sigma &\equiv \text{the surface} \\ \sigma(u,v) &\equiv \text{a point on the surface at parametrics values} u, v \\ E(u,v) &\equiv \text{vector from the eye point to } \sigma(u,v) \\ n(u,v) &\equiv \text{the normal at } \sigma(u,v) \\ \theta &\equiv \text{the angle between the vectors } E(u,v) \text{ and } n(u,v) \\ m_{i,j} &\equiv \text{control point of the mesh indexed by } i,j \end{aligned}$$

Given $E(u,v)$ and $n(u,v)$, a silhouette point is defined as the point on the surface where $E(u,v) \cdot n(u,v) = 0$ or the angle between $E(u,v)$ and $n(u,v)$ is 90 degrees, as shown in Figure 3.5.

Linear interpolation is done only in one parametric dimension, u or v, keeping the other constant. Given two surface points at parametric values t_1 and t_2, such that $t_1 = (t_1, v_o)$ and $t_2 = (t_2, v_o)$, θ_i can be defined by $n(t_i)$, the normal at t_i, and $E(t_i)$, the eye vector, as seen in Equation 4.1.

$$\theta_i = \arccos\left(\frac{E(t_i) \cdot n(t_i)}{\|E(t_i) \cdot n(t_i)\|}\right).$$

Given θ_1 and θ_2 and the corresponding parametric values, t_1 and t_2, linear interpolation will give an approximate t_* where the angle is 90 degrees or $\frac{\pi}{2}$:

$$t_* = t_2 - (t_2 - t_1)\frac{(\theta_2 - \frac{\pi}{2})}{(\theta_2 - \theta_1)}.$$

Linear interpolation is further explained by Figure 4.1

4.1.2.2 Mesh Method

The Mesh Method relies upon the control mesh of a surface, σ, to supply information on where a silhouette could and could not be. Due to the variation diminishing properties of the control mesh, one can rule out where there cannot be a silhouette point on the surface. If there is a silhouette in the control mesh, then there may be a silhouette point on the surface of the object. However, finding silhouettes is not easy and requires maintaining some large data structures. For every mesh point, $m_{i,j}$, one needs a control point data structure which contains u, v, surface point $\sigma(u, v)$, normal $n(u, v)$, mesh indices i and j, and the sign, δ, of $E(u, v) \cdot n(u, v)$. A 2D marching-cube data structure is necessary for holding the silhouette points and assembling them into silhouette curves. The 2D marching-cube data structure contains four control points and their (u,v) values, as well as a list of possible silhouette points (four are possible between the mesh points with four additional points possible at the mesh points).

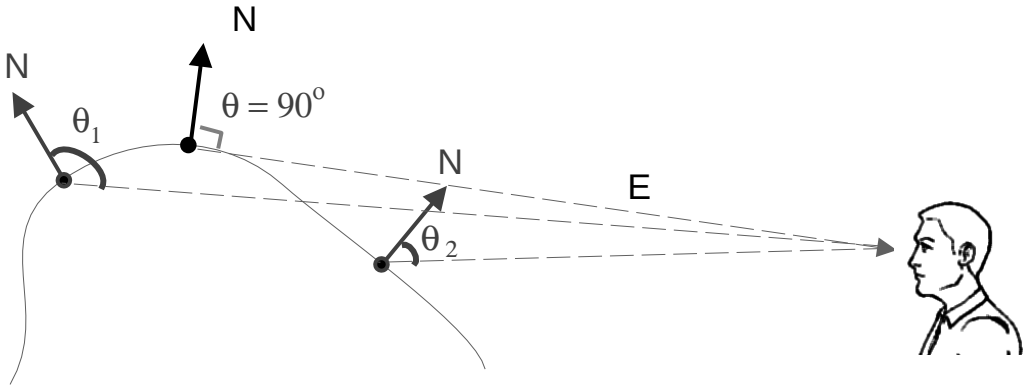


Figure 4.1. Interpolating silhouettes: After two neighboring surface points with different δ's are found, the point where $E(u, v) \cdot n(u, v) = \delta = 0$ can be found by linearly interpolating in u or v with the two angles as calculated in Equation 4.1. Note: $\theta_1 = E_1 \cdot n_1 > 0$ and $\theta_2 = E_2 \cdot n_2 < 0$.

The algorithm is as follows. First find the normals at each of the control mesh points. For every mesh point, there are up to four possible normals that need to be calculated, $n_{1,3}$, $n_{1,2}$, $n_{4,3}$, $n_{4,2}$ as can be seen in Figure 4.2. This calculation only needs to be done once per surface; the rest of the calculations needs to be made every time the viewpoint changes.

Next, classify each mesh normal based on the sign, δ, of $E(u,v)\cdot$n(u,v). There are four signs per mesh point. For example, a 4x3 control mesh can be visually represented and stored in a table like Figure 4.3.

To define which set of signs signal a possible silhouette, I looked at the combinations of δ's stored in the table. The trick is in determining what constitutes a possible silhouette. This method creates a large number of sign group variations which can indicate possible silhouettes, as can be seen by looking at the combinations of pluses and minuses around each mesh point in Figure 4.4(a). The implementation of this method involves a large set of case statements, looking at the mesh and the relative signs to determine where silhouettes may be.

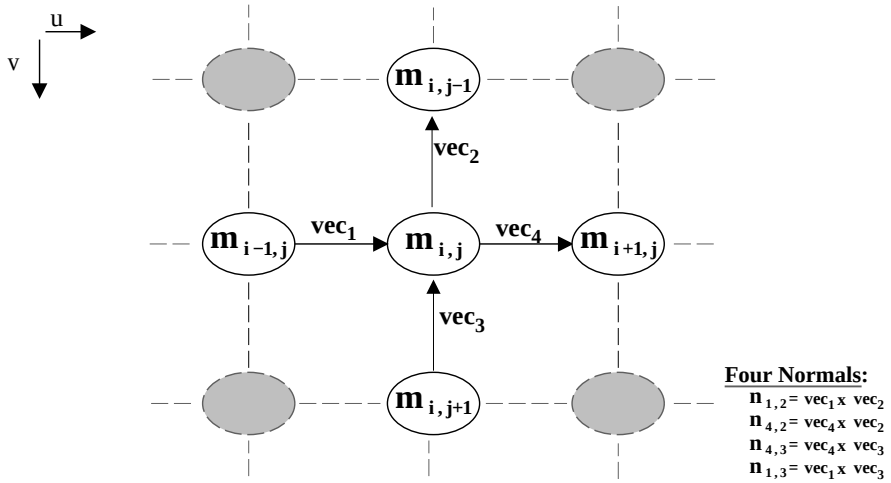


Figure 4.2. Calculating the mesh normals: The four mesh normals which correspond to $m_{i,j}$ are $n_{1,3}$, $n_{1,2}$, $n_{4,3}$, $n_{4,2}$, where for example $n_{1,3} = vec_1 \times vec_2$, with $vec1 = m_{i,j} - m_{i-1,j}$ and $vec2 = m_{i,j-1} - m_{i,j}$.

$\delta_{0,0}$ $\delta_{0,1}$ $\delta_{1,0}$ $\delta_{1,1}$	$\delta_{0,2}$ $\delta_{0,3}$ $\delta_{1,2}$ $\delta_{1,3}$	$\delta_{0,4}$ $\delta_{0,5}$ $\delta_{1,4}$ $\delta_{1,5}$
$\delta_{2,0}$ $\delta_{2,1}$ $\delta_{3,0}$ $\delta_{3,1}$	$\delta_{2,2}$ $\delta_{2,3}$ $\delta_{3,2}$ $\delta_{3,3}$	$\delta_{2,4}$ $\delta_{2,5}$ $\delta_{3,4}$ $\delta_{3,5}$

Figure 4.3. Envision the mesh method as a table of signs, where δ can be +, -, or 0.

Comparisons need to be made in both the u ($m_{i,j}$ and $m_{i+1,j}$) and in the v ($m_{i,j}$ and $m_{i,j+1}$) directions.

First check for $\delta_{i,j} = 0$. If $\delta_{i,j} = 0$ then interpolate based on the parametric values associated with $m_{i-1,j}$ and $m_{i+1,j}$ to get the silhouette point on the surface, if there is one.

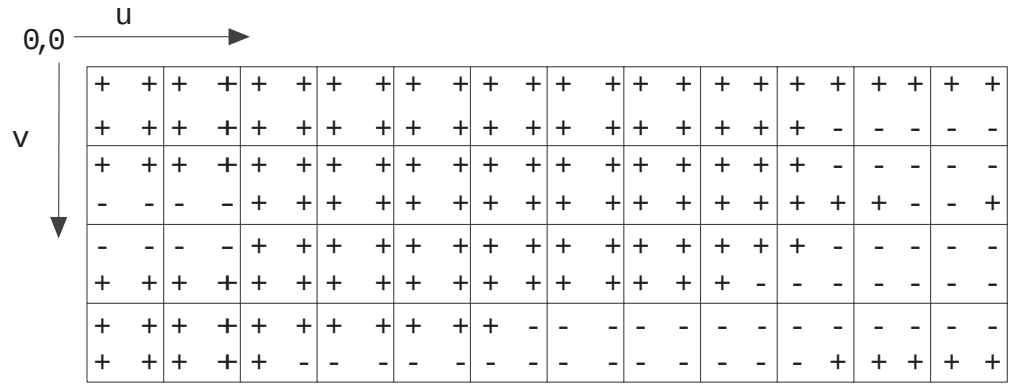
Next, check for changes between the mesh points in the u and v directions, i.e., $m_{i,j}$ and $m_{i+1,j}$, as well as $m_{i,j}$ and $m_{i,j+1}$. For example, this would mean looking at the two groups: $m_{1,1}$ ($\delta_{1,1}$, $\delta_{1,2}$, $\delta_{2,1}$, $\delta_{2,2}$) and $m_{2,1}$ ($\delta_{1,3}$, $\delta_{1,4}$, $\delta_{2,3}$, $\delta_{2,4}$) in Figure 4.3.

There are four sign comparisons made per box in the 2D marching cube data structure: for example, ($\delta_{0,2}$ and $\delta_{0,3}$), ($\delta_{1,2}$ and $\delta_{1,3}$), ($\delta_{0,2}$ and $\delta_{1,2}$), ($\delta_{0,3}$ and $\delta_{1,3}$).

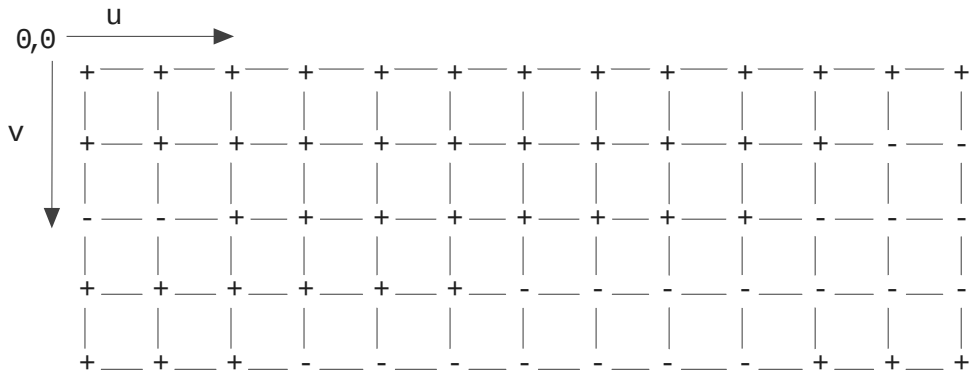
If a sign change is found, then the linear interpolation described in Section 4.1.2.1 will provide a silhouette point at u,v. The silhouette points are stored in the 2D marching-cube structure. Silhouette points are turned into silhouette curves by traveling though the marching cube data structure, connecting points to form edges. The top image in Figure 4.4 provides a visualization of the $\delta_{i,j}$ and Figure 4.5 the approximate silhouette lines for the Mesh Method and the Srf-Node Method. Figure 4.6 shows the results of the Mesh Method on a surface. Using the Secant Method or Newton's Method, these edges can be refined.

4.1.2.3 Tessellated-Mesh Method

A variation on the Mesh Method is the Tessellated Mesh Method. In order to simplify the number of possible sign combinations, I tessellated the control mesh. The control mesh of a surface is a set of bilinear patches. I split each of those bilinear patches into triangles by choosing the diagonals to be in the direction of minimum curvature across each bilinear patch. Then there is only one normal per triangle or two normals per bilinear patch. However, checking for silhouettes with these normals only tells one where a silhouette *may* be. After choosing the area that may have silhouettes, you then have to find the corresponding point on the surface and find the nearby silhouette point and curve if it exists. A winged-edge data structure can keep track of all of these data and is useful for turning silhouette points into silhouette curves. The only part left is to refine these jaggy lines as seen in Figure 4.7 into silhouette curves. I did not proceed past finding the approximate silhouette curves because this method was too slow.

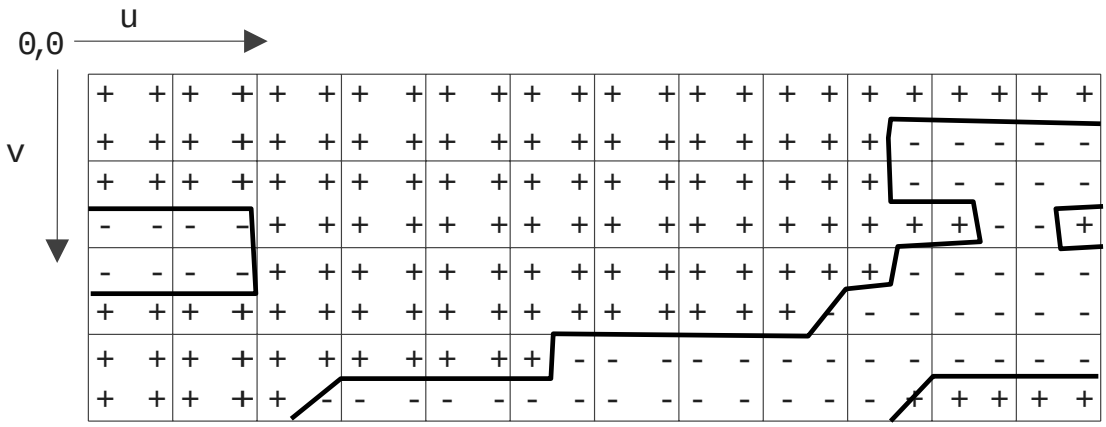


(a) Mesh Method

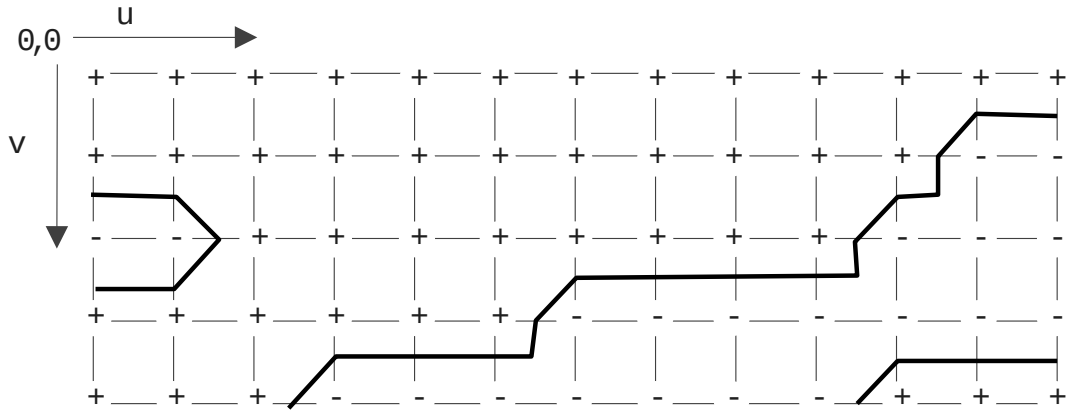


(b) Srf-Node Method

Figure 4.4. These images show the control mesh (in uv-space) for a surface where +, -, or 0 denotes the sign of the dot product $E(u,v) \cdot n(u,v)$. For the Mesh Method, there are up to four dot products that need to be calculated per mesh point, only one per mesh point for Srf-Node Method.

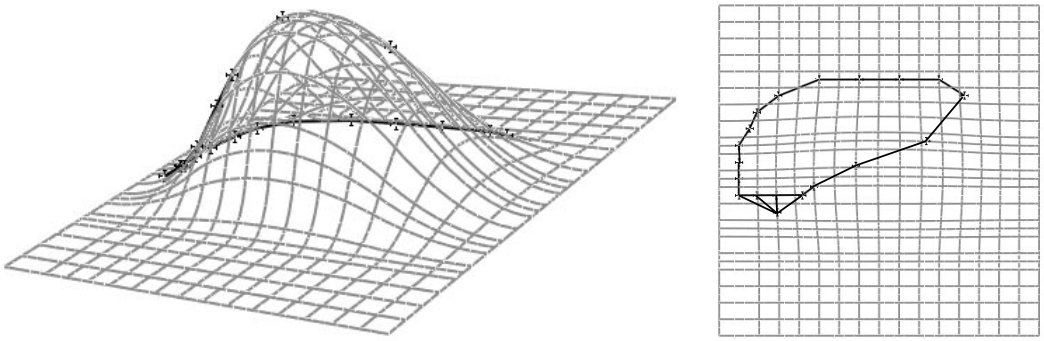


(a) Mesh Method



(b) Srf-Node Method

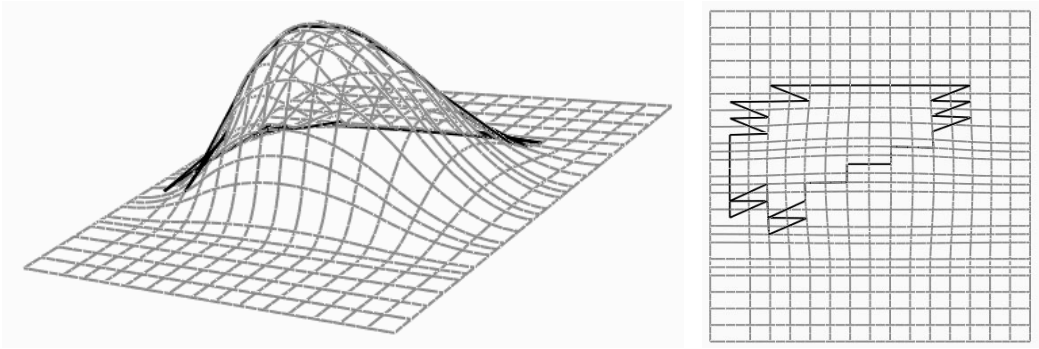
Figure 4.5. These images show the control mesh (in uv-space) for a surface, with approximations to the silhouettes. The sign of the dot product $E(u,v) \cdot n(u,v)$ are denoted by +, -, or 0.



(a) View of surface with silhouettes generated with mesh method.

(b) Looking down on surface with silhouettes.

Figure 4.6. Mesh Method.



(a) View of surface with approximate silhouettes generated with the tessellated mesh method.

(b) Looking down on surface with approximate silhouettes.

Figure 4.7. Tessellated Mesh Method.

4.1.2.4 Srf-Node Method

The Srf-Node Method is the most concise. Nodes correspond to parameter values which are the average of consecutive sets of $(order - 1)$ knots from the knot vector, ignoring the first and last ones. There are exactly the same number of nodes as there are control points. It is often convenient to use the nodes when a parameter value or point on the curve needs to be associated with a control point [19].

A normal is calculated for every node point on a surface, as shown in Figure 4.4(b). This calculation can be done as a preprocess and only has to be done once per surface.

Then, $E(u, v) \cdot n(u, v)$ is calculated for every view and every node point, where $n(u, v)$

is the surface normal at the node point and $E(u, v)$ is the vector from the eye to the point on the surface. The resulting signs of the dot products, $\delta_{i,j}$, are stored in a table, one per node point, as shown in Figure 4.8. If $\delta_{i,j}$ is zero then there is a silhouette at that node point on the surface. By searching the table in the u direction and then in the v direction, a silhouette can be found by comparing $\delta_{i,j}$ to $\delta_{i+1,j}$ and $\delta_{i,j}$ to $\delta_{i,j+1}$, respectively. If a sign changes from + to - or from - to +, then there is a silhouette between those two points on the surface, as shown in Figure 4.4(b).

When a region containing a silhouette point is found between two node points, it is linearly interpolated, as shown in Figure 4.1. The interpolation is based on two surface points and the respective angles formed by the normal and the eye vector, calculated as in Equation 4.1 and 4.1 and as discussed in Section 4.1.2.1.

In order for this method to work, the surface has to be sufficiently refined or it may miss silhouettes, as discussed in Figure 4.9. Surface refinement only needs to be done once and can be done as a preprocess over the whole surface. However, the refinement increases the number of control points and thus the number of checks necessary to locate the silhouette points. It may be better to refine the area where a silhouette may be, based on testing the control mesh.

Using a 2D marching-cube data structure makes it easy to connect the silhouette points to form linear silhouette curves. Figure 4.5(b) provides a visualization of the $\delta_{i,j}$ and the approximate silhouette lines. This method results in edge lines as displayed in Figure 4.10. Another exaple is shown in the top down view shown in Figure 4.11 and the view from the eye point in Figure 4.12.

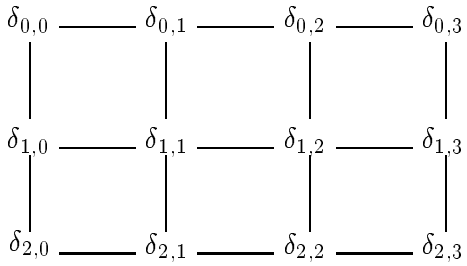


Figure 4.8. Visualize the Srf-Node Method as a table of signs ($\delta_{i,j}$), where $\delta i, j$ can be +, -, or 0.

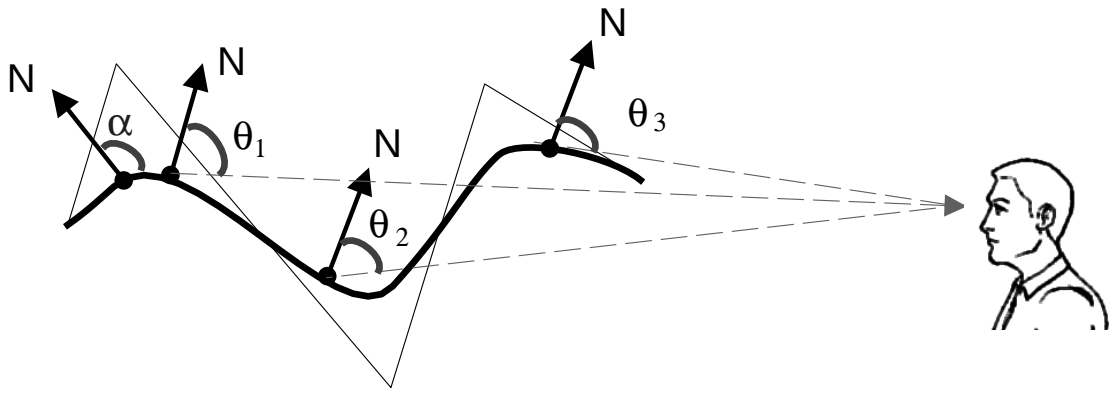
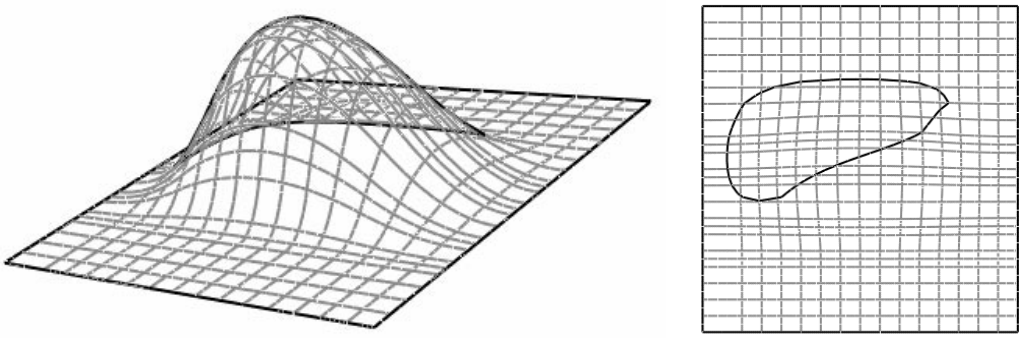


Figure 4.9. Srf-Node Method can result in missed silhouettes depending upon the node points. If for example, the node points were those that correspond to θ_1, θ_2,and θ_3, there would be three missed silhouette points because θ_1, θ_2,and θ_3, are all less than 90 degrees and there would be no sign change. However, if the nodes points were α, θ_2,and θ_3, then α is greater than 90 degrees and θ_2 is less than 90 degrees, so the silhouette between the two corresponding node points would not be missed and could be interpolated. The problem of missing these silhouettes can be remedied by refining the control mesh.



(a) View of surface with silhouettes and surface boundaries generated with Srf-Node Method.

(b) Looking down on surface with silhouettes and surface boundaries.

Figure 4.10. Srf-Node Method.

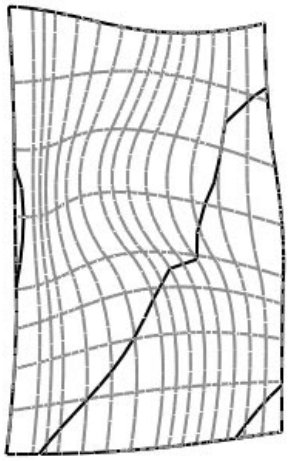


Figure 4.11. Looking down on surface with silhouette generated with Srf-Node method. Compare this image with the 2D projection and approximation of silhouettes shown in Figure 4.5 using the Mesh method and the Srf-Node method.

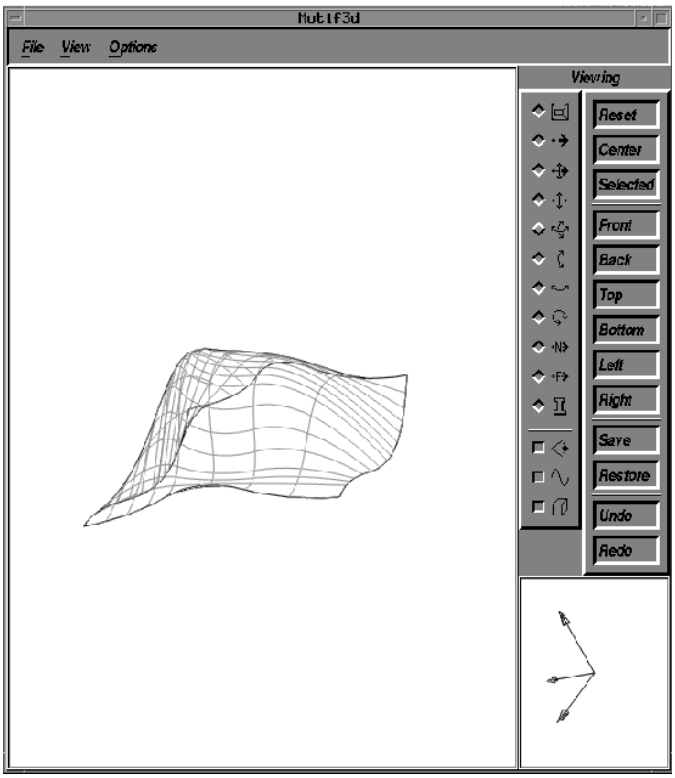


Figure 4.12. View of the same surface represented in Figure 4.5, 4.4, and 4.11 with silhouettes generated with the Srf-Node method.

4.1.2.5 Silhouette Finding Summary

The Mesh Method and the Tessellated-Mesh Method are counter-intuitive and do not have the elegant algorithmic nature that the Srf-Node method has. There appears to be far too many combinations of the signs of the dot product, $\text{sign}(E(u,v) \cdot n(u,v)) = \delta$'s, that could signal a possible silhouette in the Mesh Method and Tessellated-Mesh Method. Once it is determined that there is a silhouette on the control mesh and hence a possible silhouette on the surface, there would need to be a check to determine whether or not there is a silhouette on the surface.

The Srf-Node method is the most concise and does not require the maintenance of large data structures which may bog down the desired interactive rates. However, the Srf-Node method has the restriction that all of the surfaces must be sufficiently refined. Although this increases the number of dot products that need to be calculated, most of operations can be done as a preprocess.

In order to achieve real-time geometric silhouette calculations, one could take advantage of previously calculated silhouettes and the movement of the view to approximate the next set of silhouettes. As in Markosian's algorithm, there would have to be some way of determining where silhouettes may suddenly appear, i.e., a region that suddenly has silhouettes that were not in the previous frame. Normal or visibility cones [33, 20] could also be used to rule out where silhouettes could not be; then a test of the whole surface may not be necessary. If the model can be sufficiently tessellated, Markosian's algorithm may be preferred. However, for a highly trimmed NURBS model, tessellation may not be the best choice.

4.1.3 Other Methods for Calculating Edge Lines

Trying to find these silhouettes based on geometry may be too slow for very large models. The Srf-Node method requires sufficiently refined surfaces and bogs down on large models like the Bezier teapot model which has 23 surfaces.

At SIGGRAPH 1998, Cassidy Curtis presented a technical sketch entitled "Loose and Sketchy" [9]. Based on 3D geometry he calculated edge lines using a depth map. The depth map is converted into a "template image," which approximates the edge lines in the image. In the template image, each pixel represents the amount of ink needed in its immediate neighborhood. The template image is created by calculating the magnitude of the gradient of the depth map, thresholding it to give binary values, and then blurring the result. This technique may be faster than using the geometric information to get edge

lines. Curtis then uses this template image and another image called the "force field," also created from the depth map, along with a stochastic, physically-based particle system to create sketchy lines. This method is not interactive, and his algorithm takes about 10-60 seconds to generate "loose and sketchy" images, but may be interactive if used only to calculate the template image.

Other methods for implementing interactive/real-time edge line should be explored, especially using OpenGL, both in software and hardware. For example there is a sample program called "silhouette" which uses the OpenGL API's stencil buffer [26], but this method misses interior silhouettes.

4.2 Shading Algorithms

4.2.1 Traditional Shading of Matte Objects

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

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

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 4.13. This unsatisfactory image hides shape and material information in the dark regions. Both highlights and edge lines can provide additional information about the object. These are shown alone in Figure 4.14 with no shading. Edge lines and highlights could not be effectively added to Figure 4.13 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 4.1, 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 4.15. 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.

4.2.2 Tone-based Shading of Matte Objects

In a colored medium such as air-brush and pen, artists often use both hue and luminance (grayscale 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 [21]. Another quality of color used by artists is the *temperature* of the color. The temperature

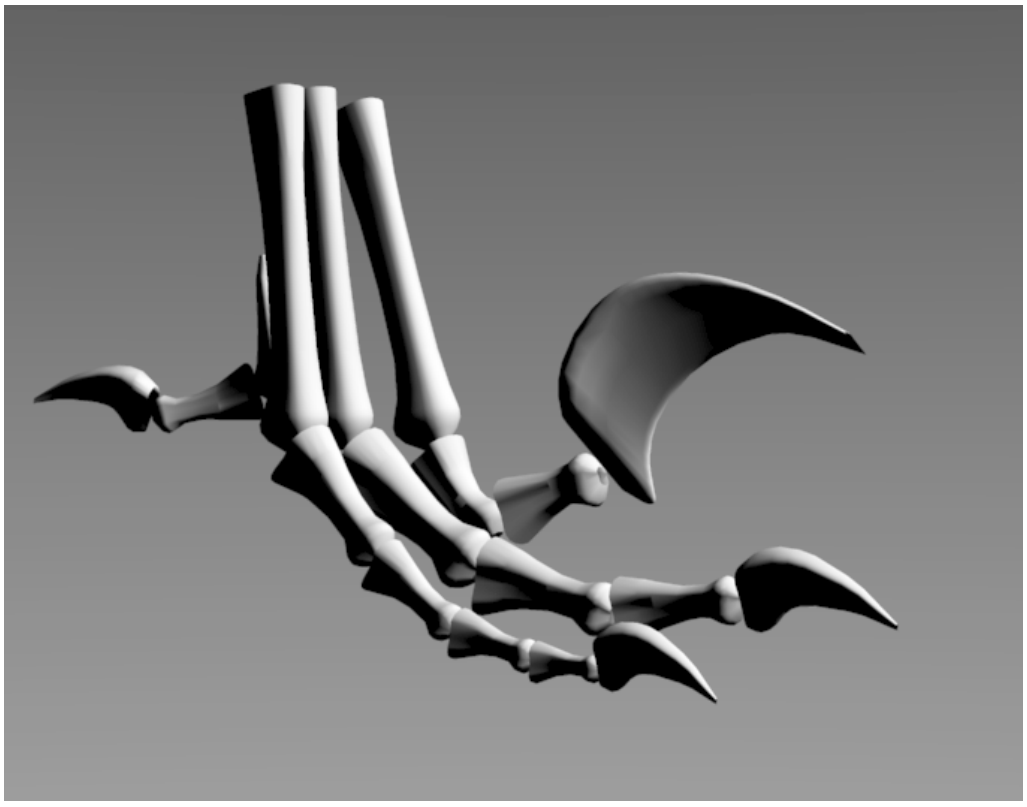


Figure 4.13. Diffuse shaded image using Equation 4.1 with $k_d = 1$ and $k_a = 0$. Black shaded regions hide details, especially in the small claws; edge lines could not be seen if added. Highlights and fine details are lost in the white shaded regions.

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 4.1 to blend between two RGB colors,

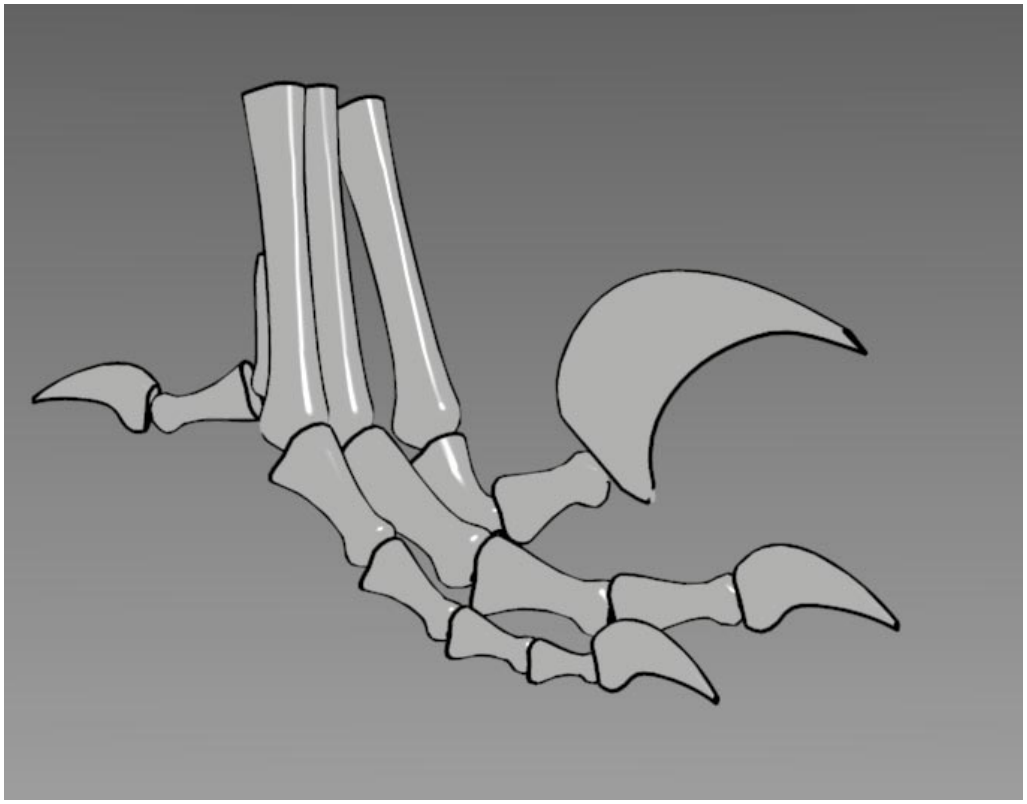


Figure 4.14. Image with only highlights and edges. The edge lines provide divisions between object pieces and the highlights convey the direction of the light. Some shape information is lost, especially in the regions of high curvature of the object pieces. However, these highlights and edges could not be added to Figure 4.13 because the highlights would be invisible in the light regions and the silhouettes would be invisible in the dark regions.

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 [14], it is best to position the light up and to the right and to keep this position constant.

An image that uses a color scale with little luminance variation is shown in Figure 4.16. 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

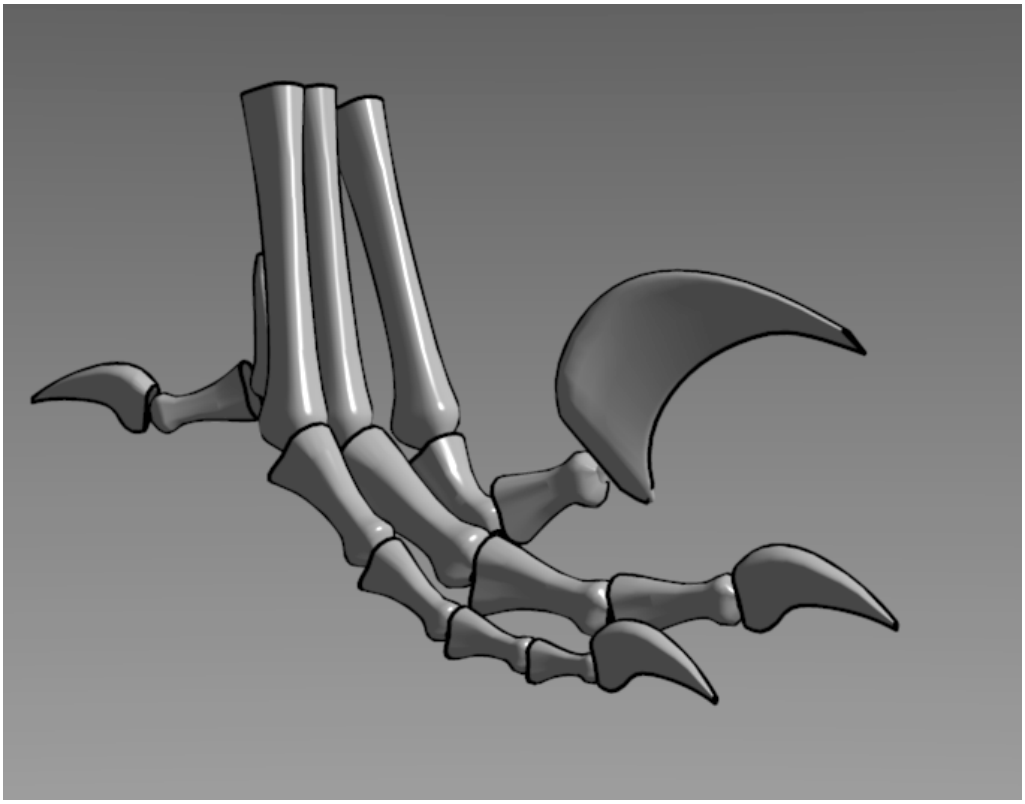


Figure 4.15. Phong-shaded image with edge lines and $k_d = 0.5$ and $k_a = 0.1$. Like Figure 4.13, details are lost in the dark gray regions, especially in the small claws, where they are colored the constant shade of $k_d k_a$ regardless of surface orientation. However, edge lines and highlights provide shape information that was gained in Figure 4.14, but could not be added to Figure 4.13.

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

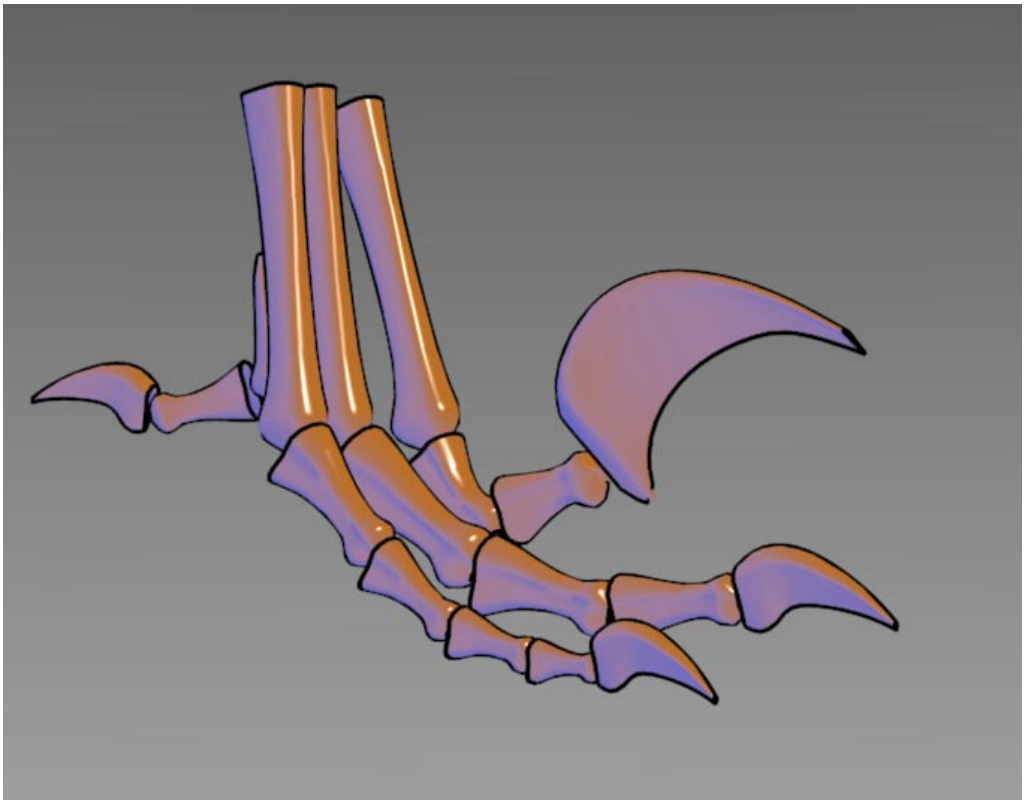


Figure 4.16. Approximately constant luminance tone rendering. Edge lines and highlights are clearly noticeable. Unlike Figures 4.13 and 4.15 some details in shaded regions, like the small claws, are visible. The lack of luminance shift makes these changes subtle.

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 $\hat{\mathbf{l}} \cdot \hat{\mathbf{n}} < 0$. 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:

$$\begin{aligned} k_{cool} &= k_{blue} + \alpha k_d, \\ k_{warm} &= k_{yellow} + \beta k_d. \end{aligned} \tag{4.1}$$

Plugging these values into Equation 4.1 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 4.17.

Substituting the values for k_{cool} and k_{warm} from Equation 4.1 into the tone Equation 4.1 results in shading with values within the middle luminance range as desired. Figure 4.18 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 4.19

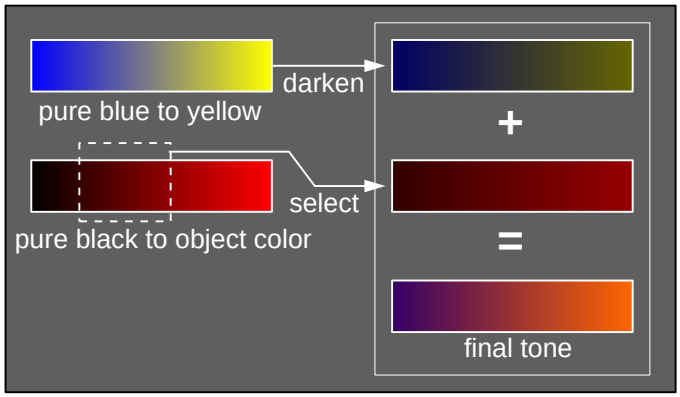


Figure 4.17. How the tone is created for a pure red object by summing a blue-to-yellow and a dark-red-to-red tone.

with $b = 0.55$, $y = 0.3$, $\alpha = 0.25$, and $\beta = 0.5$. Unlike Figure 4.15, subtleties of shape in the claws are visible in Figures 4.18 and 4.19.

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 4.20. 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. I hypothesize a few reasons why this may be so. The linear ramp of the shading may be too uniform and cause the spheres to flatten. The shading presented here is just a first pass approximation to the shading artists use and

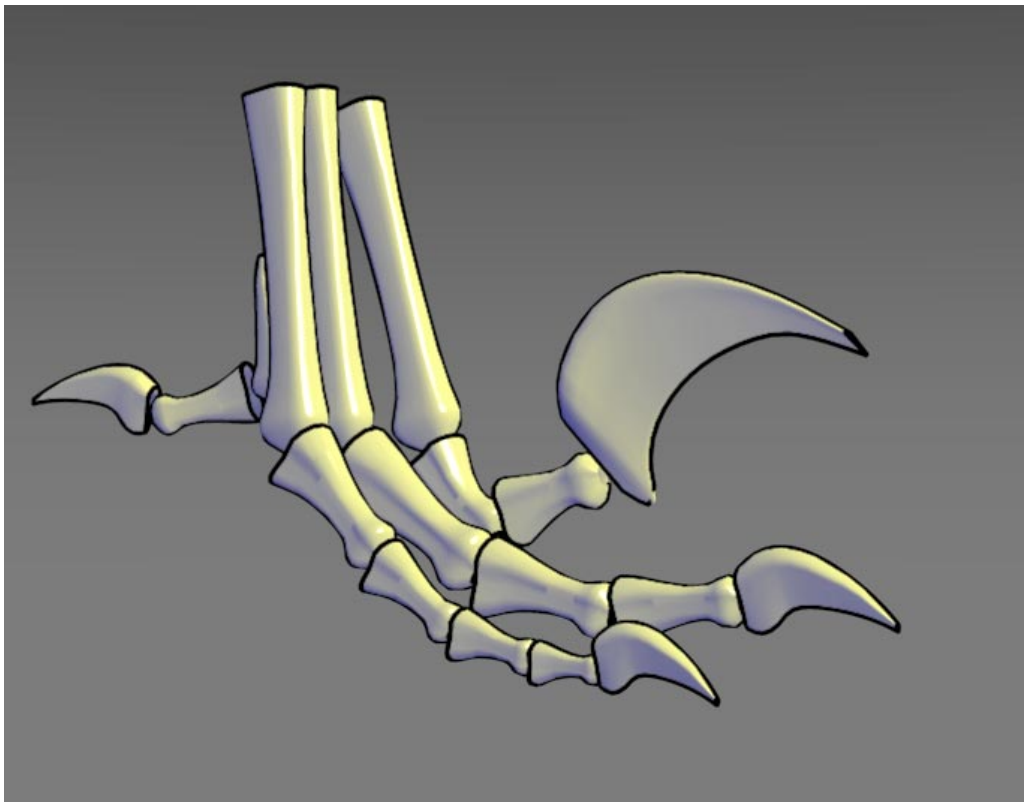


Figure 4.18. Luminance/hue tone rendering. This image combines the luminance shift of Figure 4.13 and the hue shift of Figure 4.16. Edge lines, highlights, fine details in the dark shaded regions such as the small claws, as well as details in the high luminance regions are all visible. In addition, shape details are apparent unlike Figure 4.14 where the object appears flat. In this figure, the variables of Equation 4.1 and Equation 4.1 are: $b = 0.4$, $y = 0.4$, $\alpha = 0.2$, $\beta = 0.6$.

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 4.21 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 my 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 4.22 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

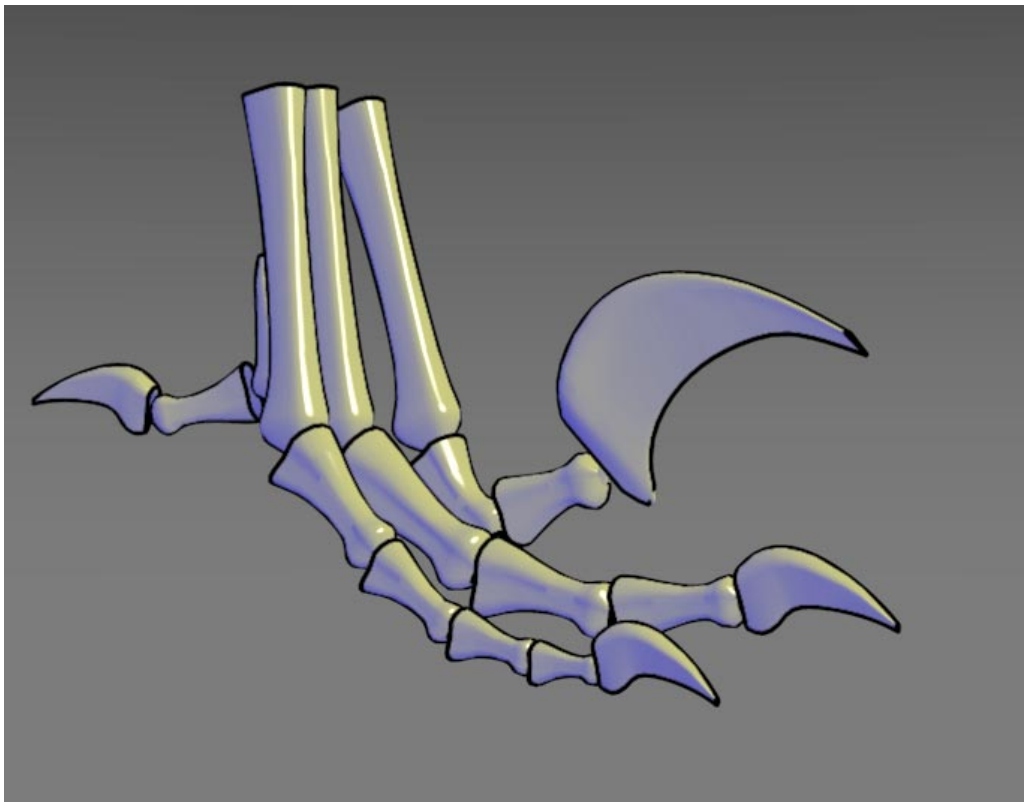


Figure 4.19. Luminance/hue tone rendering, similar to Figure 4.18 except $b = 0.55$, $y = 0.3$, $\alpha = 0.25$, $\beta = 0.5$. The different values of b and y determine the strength of the overall temperature shift, where as α and β determine the prominence of the object color, and the strength of the luminance shift.

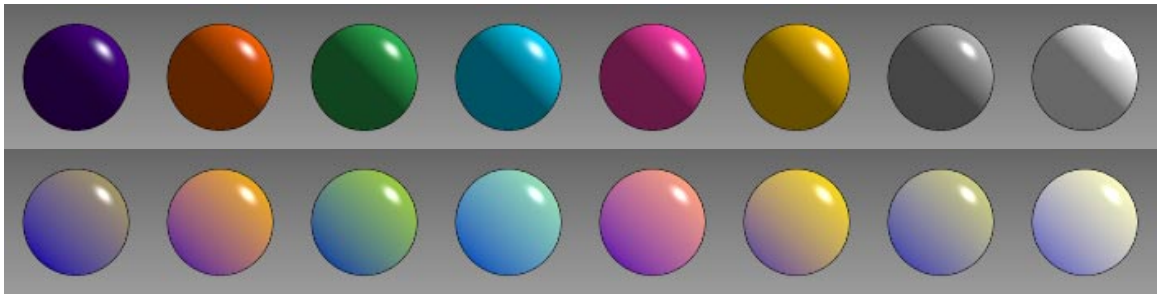


Figure 4.20. Comparing shaded, colored spheres. Top: Colored Phong-shaded spheres with edge lines and highlights. Bottom: Colored spheres shaded with hue and luminance shift, including edge lines and highlights. Note: In the first Phong-shaded sphere (violet), the edge lines disappear, but are visible 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."

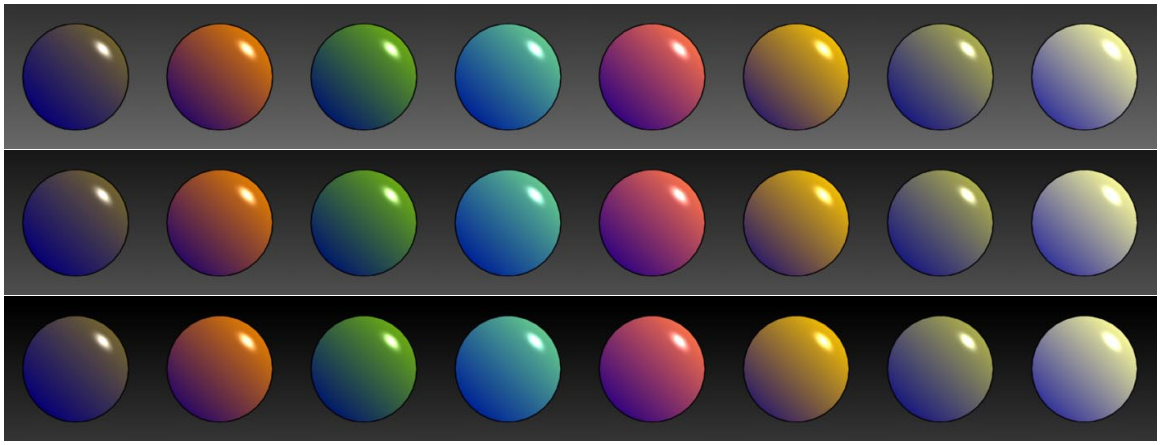


Figure 4.21. Tone and undertone shaded spheres with backgrounds getting darker.

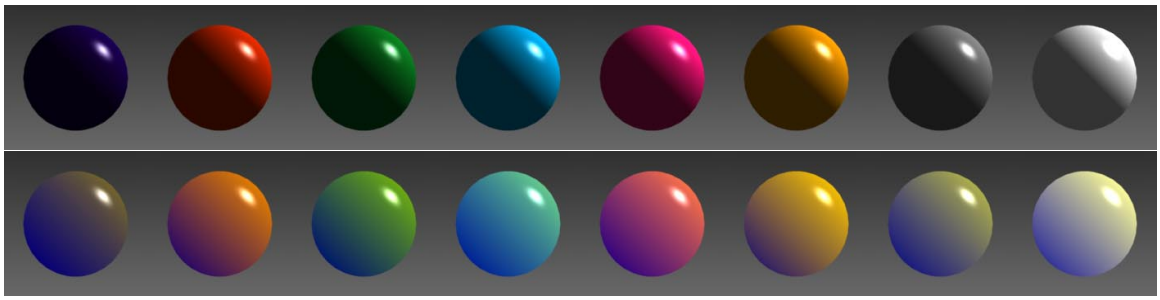


Figure 4.22. Shaded spheres without edgelines. Top: Colored Phong-shaded spheres without edge lines. Bottom: Colored spheres shaded with hue and luminance shift, without edge lines.

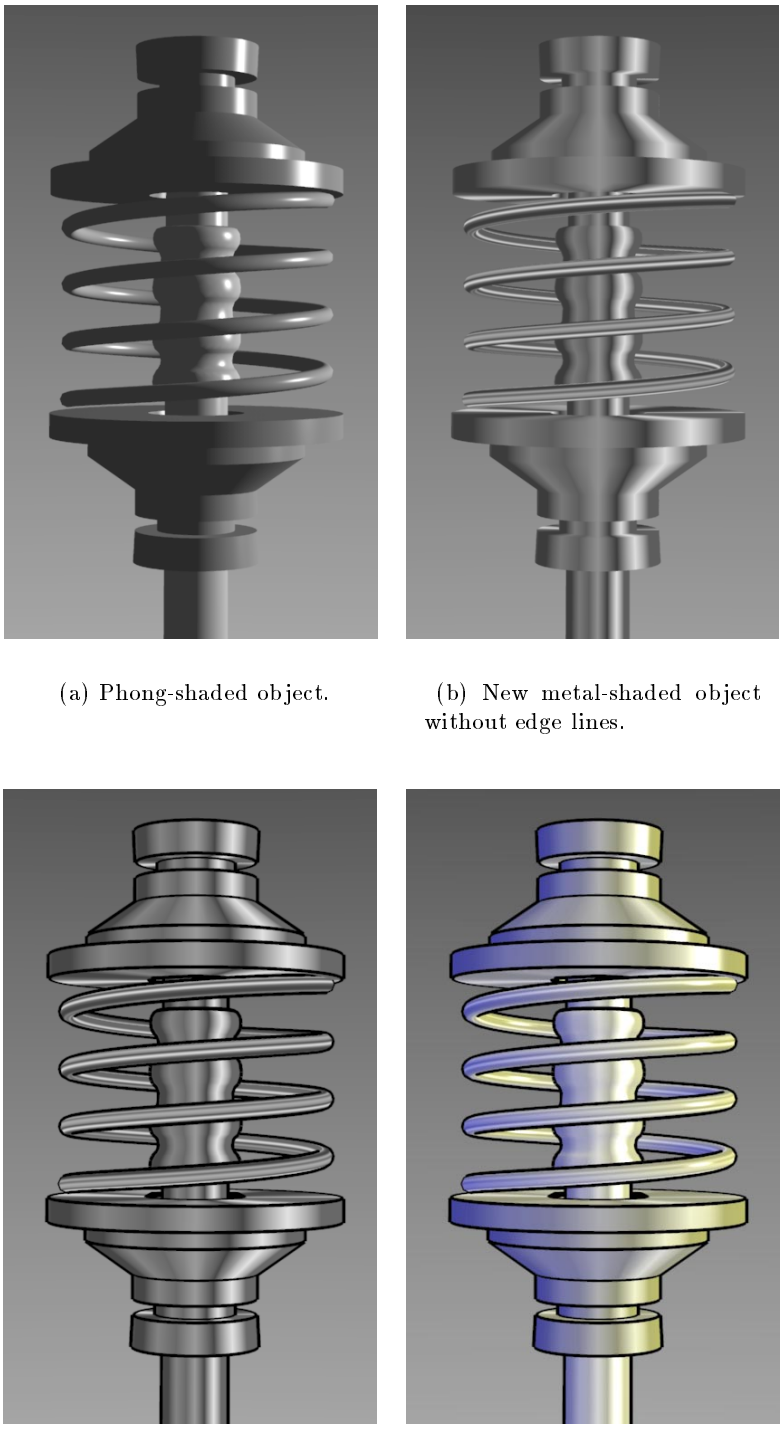
a sphere. However, it is a computer graphics tradition to test a shading model on the spheres.

4.2.3 Shading of Metal Objects

Illustrators use a different technique to communicate the surface properties of metallic objects, as shown in the photograph in Figure 4.23. 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 [25, 28]. This convention emphasizes that realism is not the primary goal of technical illustration.

To simulate a milled object, Gooch et al. [15] 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 4.24. 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 hue 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 4.2.2, so it is useful when both metal and

Figure 4.23. An anisotropic reflection can be seen in the metal objects in this photograph.



(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 4.24. Representing metallic material properties.

matte objects are shown together.

4.3 3D Illustration

Imitating 2D technical illustrations is fairly straightforward. However, 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 shown in Figure 4.25.

The question remains, "how do the 2D illustration rules change for a 3D interactive technical illustration?" Adapting the shading and line conventions presented in Chapter 3 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 one 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 one looks at a small object in one's hand, one turns the object and does not move one's head, so the light stays in the same position relative to the eye. However when one moves an object in an modeling program or when one is looking at a large part, ones is actually moving the eye point, not the object. As shown in Figure 4.26, the shading model presented in Section 3.2 is used to its full advantage if the surface varies completely from cool to warm, as shown in comparing Figure 4.26(b) and Figure 4.26(c). This would mean moving the object, not

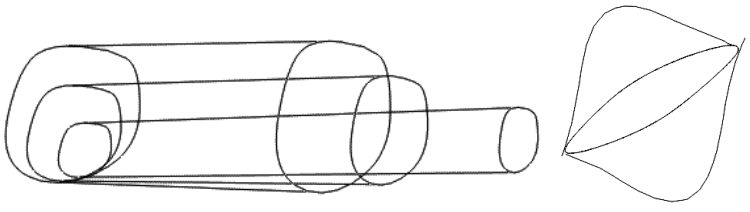
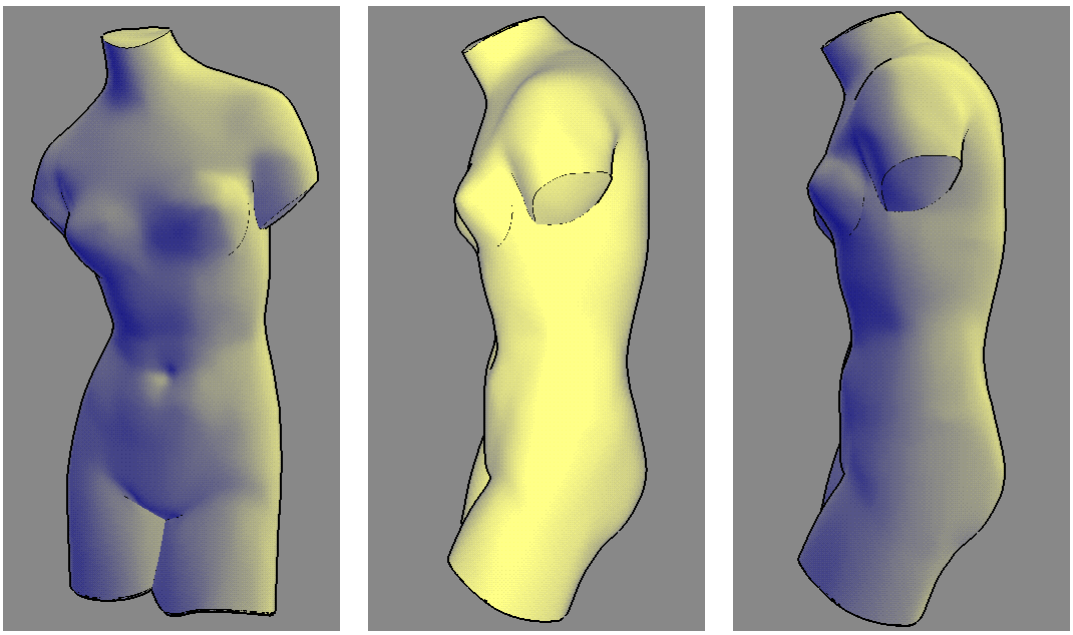


Figure 4.25. Comparing this figure to Figure 1.1, the edge lines displayed provide shape information without cluttering the screen.

the viewpoint.

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. [37]. 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 would be confusing.

Most material properties are semiconstant as the view direction or lighting changes. However the metal shading presented in Section 4.2.3 is the replication of the anisotropic reflection due to the surface of the object and the reflection of the environment. When a real metal part is rotated in one's hand, the banding does not stick to the object, but remains constant since the environment is not changing. However, in an non-photorealistic



(a) Frame of model with new shading in an interactive environment with lights positioned up and to the right.

(b) Frame after the camera position is moved to view the side of the model.

(c) Frame after moving the object instead of the camera, allowing the surface to vary completely from cool to warm.

Figure 4.26. Frames from the NPR JOT Program, to which I used Markosian et al.'s silhouette finding technique [24] and added the OpenGL approximation to the new shading model. This will be discussed further in Chapter 5.

interactive environment it may be too jarring to have the metal shading changing abruptly. Using a metal texture would 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.

CHAPTER 5

IMPLEMENTATION AND RESULTS

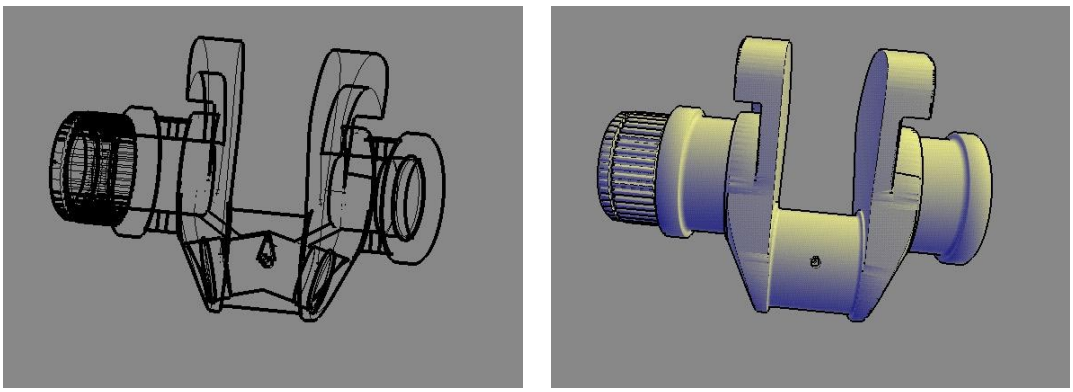
5.1 Edge Lines

The algorithms for the Srf-Node Method, the Mesh Method, and the Tessellated-Mesh Method were all integrated into the Alpha_1 system, using the existing NURBS surface evaluators and functions when possible. Alpha_1 is a B-spline research system which integrates computer graphics, modeling (geometric, simulation, and physically based), rendering (realistic and non-realistic), virtual prototyping, mechanical design, visualization, animation, teleconferencing, and human-computer interaction. In the viewer called "motif3d," one can load a NURBS model and toggle display options such as isolines, the control mesh, and Phong shading. I added the option to display silhouettes as well as the new shading, as explained in the next section.

Using the JOT program, a Utah-Brown collaboration, I was able to use the OpenGL shading model approximation presented in the next section along with the silhouette finding technique of Markosian et al. to produce interactive illustrations for polygonal models, as shown in Figure 5.1. The original Sketch [42] system was deeply intertwined with the Brown UGA [41] system, which prevented easy distribution of Sketch. To overcome this problem, Sketch was rewritten as part of the JOT framework. JOT allows Sketch to work with different underlying graphics kernels that provide basic services such as CSGs, intersection, creation of geometric primitives, etc. Currently JOT works with various graphics kernels including Alpha_1, the Amodeler package from Autodesk, ARCADE (from Fraunhofer IGD), and a partial implementation on top of Open Inventor. In addition, JOT can be used on various different UNIX platforms and Windows NT.

5.2 Approximation to New Shading Model

The new shading model presented in Section 4.2.2 cannot be implemented directly in high-level graphics packages that use Phong shading. However, the Phong lighting model



(a) Image of with edge lines only.

(b) Image with new shading and edge lines.

Figure 5.1. An Alpha_1 model that was tessellated and imported into the JOT NPR Program.

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. [37]. In most graphics systems (e.g., OpenGL) negative colors for the lights can be used. Then Equation 4.1 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, 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 [16].

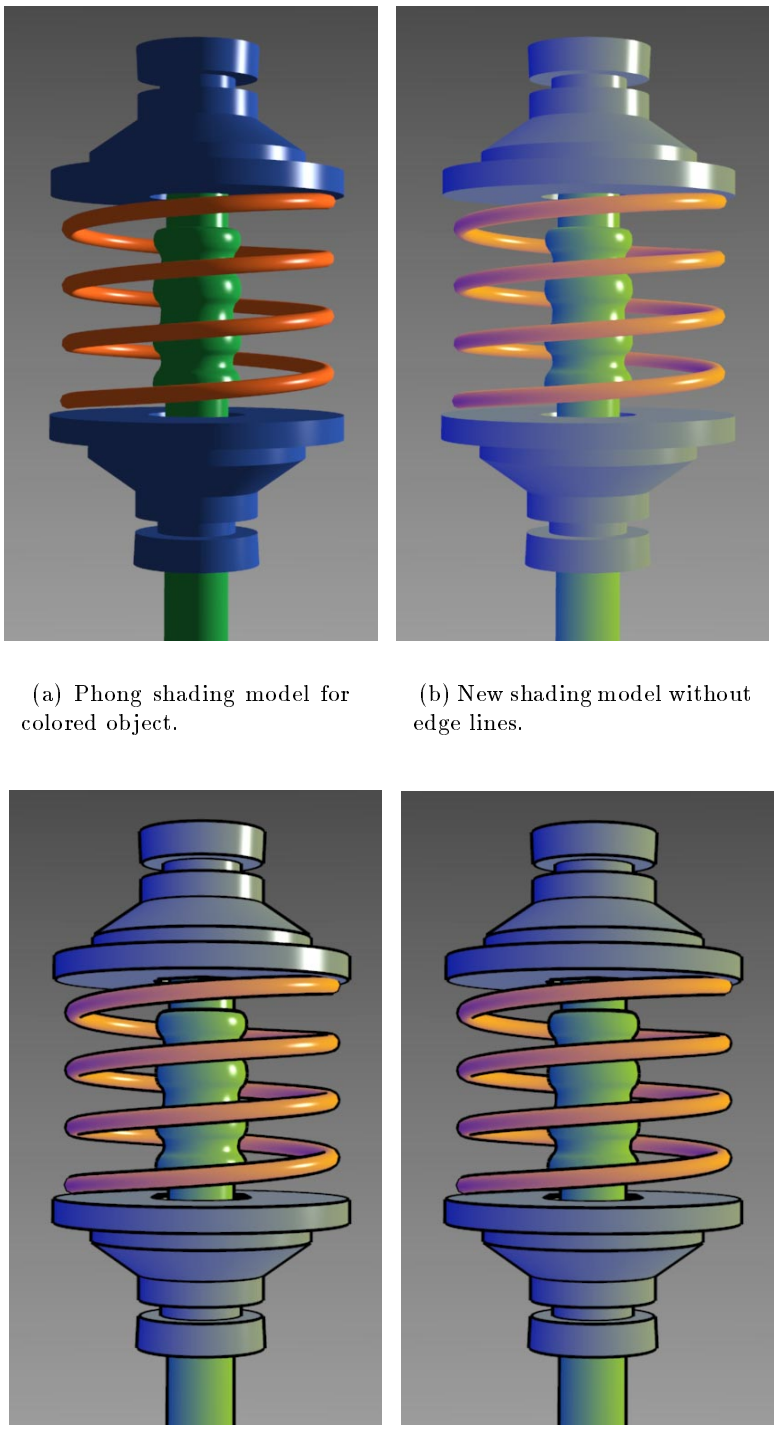
C++ Code fragment for generating the two lights, using the OpenGL API:

```
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 5.2. Using this approximation, I was able to add the new shading to the Alpha_1 system as well.

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



(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 5.2. Comparison of traditional computer graphics techniques and techniques for creating technical illustrations.

CHAPTER 6

CONCLUSION

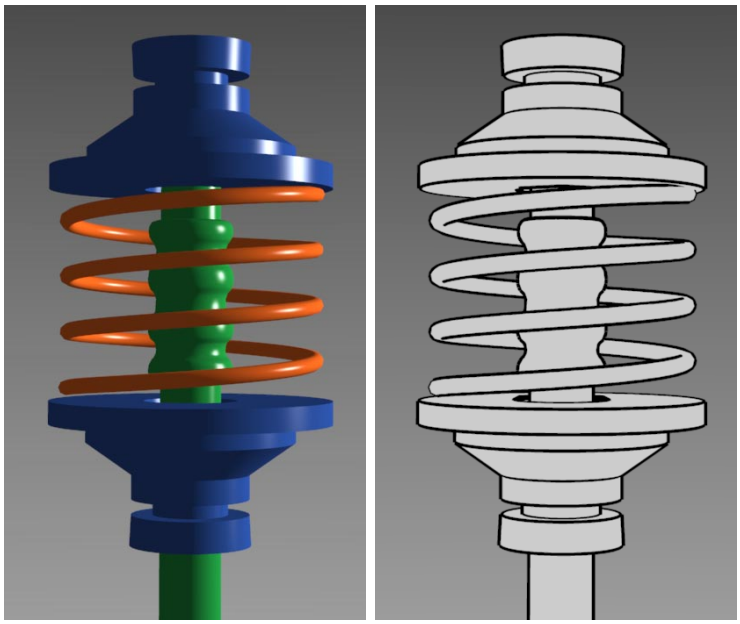
One of the largest goals in computer graphics has been realistic image synthesis. However, in a number of situations, an image which highlights particular information is valued above realism. For example, an automobile repair manual uses illustrations to remove unnecessary details and to draw attention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

Many computer-generated images have to be hand-tuned and they still convey shape poorly. The goal of my 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.

As seen in Figure 6.1(a), Phong-shaded 3D imagery does not provide geometric information of the same richness as human-drawn technical illustrations. In this thesis, I have reviewed some conventions to create computer-generated images and interaction which imitates colored technical illustrations. One of the most important characteristics of illustration is the use of lines to separate parts and to strengthen shape information, as seen in Figure 6.1(b).

The shading used in computer-generated illustrations should use a low dynamic range which does not interfere with black edge lines and white highlights. By combining the tone and undertone techniques of the shading presented in Section 4.2.2, the shading may strengthen the shape information, while maintaining the low dynamic range. It is easy to see how important the role is that edge lines play in distinguishing parts or surface discontinuities, as shown in Figure 6.2(a). This new shading is tailored not to interfere with edge lines and highlights. Shading without edge lines results in a subtle image, from which it is harder to distinguish key boundaries of the model.

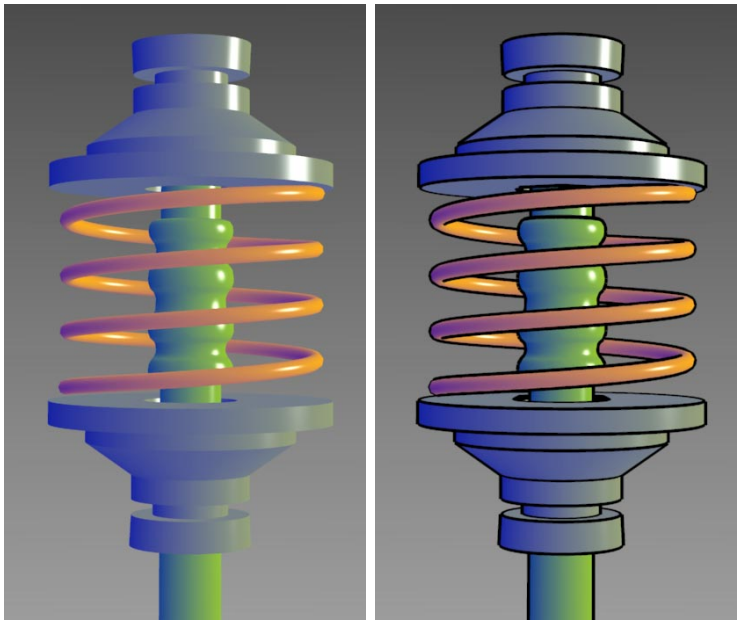
As shown in Figure 6.2(b), putting together a low dynamic range shading which uses tones and undertones with edge lines results in an image that may convey more shape information than the traditional Phong-shaded approach, Figure 6.1(a). Using the illustration model presented in this thesis, one is no longer required to guess the best



(a) Phong-shaded image.

(b) Image with edge lines only.

Figure 6.1. Phong shading versus edge lines.



(a) Image with new shading without edge lines.

(b) New Shading with Edge Lines.

Figure 6.2. Edge lines.

position for lights or to tweak the coefficients of the lighting model. In this thesis, I have also explored representing metal material properties, using a convention similar to that of most illustrators, as discussed in Section 4.2.3 and shown in Figure 6.3.

Since there are few examples of interactive 3D illustrations, there are not any conventions one has to follow. Adapting the shading and line conventions presented in Chapter 3 is straightforward as long as the line width conventions have frame-to-frame coherence. However, when interacting with 3D illustration one has to address the issue of whether to move the object or just change the viewer's position in model space. There are cases when one would want to move around a large object (changing the view), and conversely one may want to move a small object as if holding it (moving the object). This is directly related to whether the light moves or sticks to an object. It may be disorienting for the shading to change on a part as orientation of the part changes. However, the shading model presented in Section 3.2 is used to its full advantage if the surface varies completely from warm to cool. This would mean moving the object, not the viewpoint. However, when a manufacturer designs the hood of a car, they want to look at the way the appearance of the car changes as the light changes. Therefore, an interactive 3D illustration environment should consider the relative size of the object in order to control

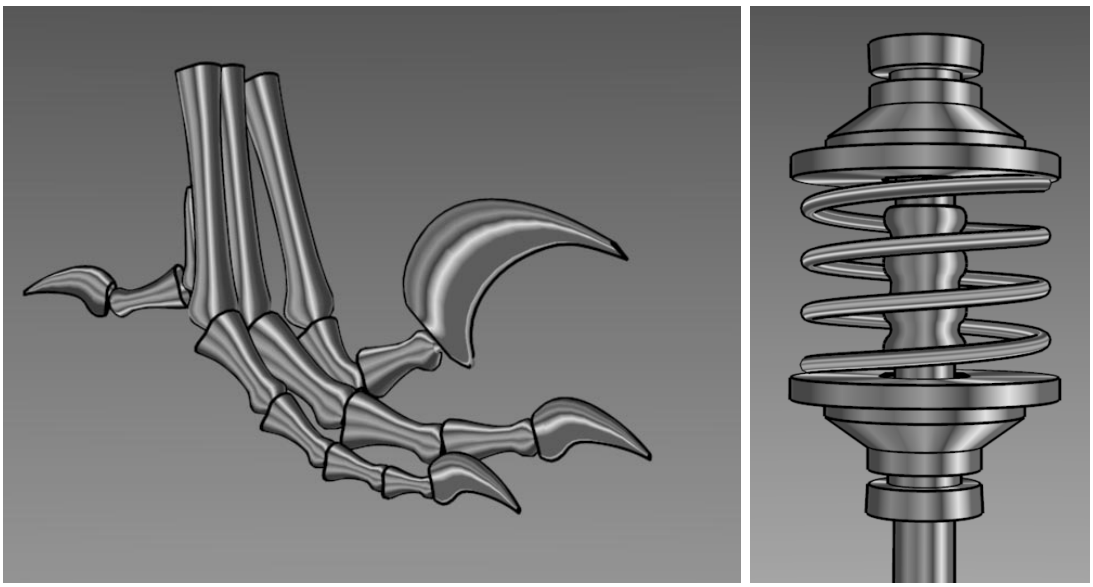


Figure 6.3. These computer generated images were produced by using the illustration convention of alternating dark and light bands, to convey the metallic material property of the two images. The convention is rather effective, even for an object that may not naturally be metal.

the motion of the viewer and the object, as well as the illumination source and allowing the user to control these options to maximize the amount of available shape information.

6.1 Future Work

The work presented here is exploratory and shows the advantages of non-photorealism in its ability to convey shape information. There are many possible improvements and additions to this work including incorporating other illustration techniques such as automatic layout, different line styles, cut-a-ways, exploded views, and object transparency. Exploring a nonlinear shading model or a perceptually uniform gradation from cool to warm may tap into more shape information.

I have proposed a new method of displaying and interacting with 3D models; however it has not been proven that these illustration methods are better than a photograph. It has been observed that in some images, the new shading may flatten an object, but I could only get opinions or make my own hypothesis. Scientifically analyzing whether or not the techniques presented here provide more shape information than traditional approaches in computer graphics may lead to more effective methods for conveying the important geometric properties of 3D models.

REFERENCES

[1] Appel, A. The notion of quantitative invisibility and the machine rendering of solids. *Proceedings of ACM National Conference* (1967), 387–393.

[2] Biederman, I., and Ju, G. Surface versus edge-based determinants of visual recognition. *Cognitive Psychology 20* (1988), 38–64.

[3] Birren, F. *Color perception in art.* Van Nostrand Reinhold Company, New York, 1976.

[4] Braje, W. L., Tjan, B. S., and Legge, G. E. Human efficiency for recognizing and detecting low-pass filtered objects. *Vision Research 35*, 21 (1995), 2955–2966.

[5] Browning, T. *Timeless techniques for better oil paintings.* North Light Books, New York, 1994.

[6] Chin, N., and Feiner, S. Near real-time shadow generation using bsp trees. *SIGGRAPH 89 Conference Proceedings 23*, 3 (July 1989), 99–106.

[7] Christou, C., Koenderink, J. J., and van Doorn, A. J. Surface gradients, contours and the perception of surface attitude in images of complex scenes. *Perception 25* (1996), 701–713.

[8] Christou, C., Koenderink, J. J., and van Doorn, A. J. Surface gradients, contours and the perception of surface attitude in images of complex scenes. *Perception 25* (1996), 701–713.

[9] Curtis, C. SIGGRAPH 1998 technical sketch: Loose and sketchy. *http://www.cs.washington.edu/homes/cassidy/loose/sketch.html* (July 1998).

[10] Curtis, C. J., Anderson, S. E., Fleischer, K. W., and Salesin, D. H. Computer-generated watercolor. In *SIGGRAPH 97 Conference Proceedings* (Aug. 1997).

[11] Dooley, D., and Cohen, M. F. Automatic illustration of 3D geometric models: Surfaces. *IEEE Computer Graphics and Applications 13*, 2 (1990), 307–314.

[12] Driskill, E. *Towards the design, analysis, and illustration of assemblies.* PhD thesis, University of Utah, Department of Computer Science, Salt Lake City, Utah, September 1996.

[13] Elber, G., and Cohen, E. Hidden curve removal for free-form surfaces. In *SIGGRAPH 90 Conference Proceedings* (Aug. 1990).

[14] Goldstein, E. B. *Sensation and perception.* Wadsworth Publishing Co., Belmont,

California, 1980.

[15] Gooch, A., Gooch, B., Shirley, P., and Cohen, E. A Non-photorealistic lighting model for automatic technical illustration. In *Computer Graphics* (July 1998).

[16] Haeberli, P. The accumulation buffer: Hardware support for high-quality rendering. *SIGGRAPH 90 Conference Proceedings 24*, 3 (Aug. 1990).

[17] Haeberli, P. Paint by numbers: Abstract image representation. In *SIGGRAPH 90 Conference Proceedings* (Aug. 1990).

[18] Heflin, G., and Elber, G. Shadow volume generation from free form surfaces. In *Communicating with virtual worlds, Proceedings of CGI'93 (Lausanne, Switzerland)* (June 1993), Springer-Verlag, pp. 115–126.

[19] Integrated Graphics Modeling Design, and Manufacturing Research Group. *Alpha1 geometric modeling system, user's manual.* Department of Computer Science, University of Utah.

[20] Kim, D.-S., Papalambros, P. Y., and Woo, T. C. Tangent, normal, and visibility cones on bezier surfaces. *Computer Aided Geometric Design 12* (May 1995), 305–320.

[21] Lambert, P. *Controlling color: a practical introduction for designers and artists*, vol. 1. Everbest Printing Company Ltd., 1991.

[22] Land, B., and Alferness, J. Curvature-based drawings form 3-d polygonal objects. *http://www.tc.cornell.edu/ãlfernes/linedraw.ps* (October 1996).

[23] Litwinowicz, P. Processing images and video for an impressionistic effect. In *SIGGRAPH 97 Conference Proceedings* (Aug. 1997).

[24] Markosian, L., Kowalski, M., Trychin, S., and Hughes, J. Real-time nonphotorealistic rendering. In *SIGGRAPH 97 Conference Proceedings* (Aug. 1997).

[25] Martin, J. *Technical illustration: materials, methods, and techniques*, vol. 1. Macdonald and Co Publishers, 1989.

[26] McReynolds, T., and Blythe, D. silhouette.c from programming with OpenGL: Advanced rendering demo programs. A course presented at SIGGRAPH 96. *http://www.sgi.com/Technology/OpenGL/advanced/tomcat/silhouette.c* (1996).

[27] Meier, B. J. Painterly rendering for animation. In *SIGGRAPH 96 Conference Proceedings* (Aug. 1996).

[28] Ruppel, T., Ed. *The way science works*, vol. 1. MacMillan, 1995.

[29] Saito, T., and Takahashi, T. Comprehensible rendering of 3D shapes. In *SIGGRAPH 90 Conference Proceedings* (Aug. 1990).

[30] Salisbury, M., Anderson, C., Lischinski, D., and Salesin, D. H. Scale-

dependent reproduction of pen-and-ink illustration. In *SIGGRAPH 96 Conference Proceedings* (Aug. 1996).

[31] Salisbury, M., Wong, M. T., Hughes, J. F., and Salesin, D. H. Orientable textures for image-based pen-and-ink illustration. In *SIGGRAPH 97 Conference Proceedings* (Aug. 1997).

[32] Salisbury, M. P., Anderson, S. E., Barzel, R., and Salesin, D. H. Interactive pen-and-ink illustration. In *SIGGRAPH 94 Conference Proceedings* (Aug. 1994).

[33] Sederberg, T. W., and Meyers, R. J. Loop detection in surface patch intersections. *Computer Aided Geometric Design 5* (Feb 1988), 161–171.

[34] Seligmann, D. D., and Feiner, S. Automated generation of intent-based 3D illustrations. In *SIGGRAPH 91 Conference Proceedings* (July 1991).

[35] Tjan, B. S., Braje, W. L., Legge, G. E., and Kersten, D. Human efficiency for recognizing 3-D objects in luminance noise. *Vision Research 35*, 21 (1995), 3053–3069.

[36] Tufte, E. *Visual explanations.* Graphics Press, 1997.

[37] Walter, B., Alppay, G., Lafortune, E. P. F., Fernandez, S., and Greenberg, D. P. Fitting virtual lights for non-diffuse walkthroughs. In *SIGGRAPH 97 Conference Proceedings* (Aug. 1997), pp. 45–48.

[38] Winkenbach, G., and Salesin, D. H. Computer generated pen-and-ink illustration. In *SIGGRAPH 94 Conference Proceedings* (Aug. 1994).

[39] Winkenbach, G., and Salesin, D. H. Rendering parametric surfaces in pen and ink. In *SIGGRAPH '96 Conference Proceedings* (Aug. 1996).

[40] Wurm, L. H., Legge, G. E., Isenberg, L. M., and Luebker, A. Color improves object recognition in normal and low vision. *Journal of Experimental Psychology: Human Perception and Performance 19*, 4 (1993), 899–911.

[41] Zeleznik, R. C., Conner, D. B., Wloka, M., Aliaga, D., Huang, N., Hubbard, P. M., Knep, B., Kaufman, H., Hughes, J. F., and van Dam, A. An object oriented framework for the integration of interactive animation techniques. *SIGGRAPH 91 Conference Proceedings 25*, 4 (July 1991), 105–112.

[42] Zeleznik, R. C., Herndon, K. P., and Hughes, J. F. SKETCH: An interface for sketching 3D scenes. In *SIGGRAPH 96 Conference Proceedings* (Aug. 1996).