THE ALIASING PROBLEM IN COMPUTER-SYNTHESIZED SHADED IMAGES

by

Franklin C. Crow

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

Doctor of Philosophy

Department of Computer Science

University of Utah

March, 1976

UNIVERSITY OF UTAH GRADUATE SCHOOL

SUPERVISORY COMMITTEE APPROVAL

of a dissertation submitted by

Franklin Crockett Crow

I have read this dissertation and have fo or a doctoral degree.

1-12-76
Date

David C. Evans
Chairman, Supervisory Committee

I have read this dissertation and have fo doctoral degree.

1-8-76
Date

Thomas G. Stockham
Member, Supervisory Committee

I have read this dissertation and have fo doctoral degree.

1-13-76
Date

Robert A. Schumaker
Member, Supervisory Committee

UNIVERSITY OF UTAH GRADUATE SCHOOL

FINAL READING APPROVAL

To the Graduate Council of the University of Utah:

I have read the dissertation of Franklin Crockett Crow 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.

1-12-76
Date

David C. Evans
Member, Supervisory Committee

A

Robert E. Stephenson
Chairman/Dean

Sterling M. McMurrin
Dean of the Graduate School

ACKNOWLEDGEMENTS

I would like to express my continuing thanks to Dave Evans whose enthusiastic support and perceptive guidance will be a source of inspiration long after this work is finished. I would also like to thank Tom Stockham and Bob Schumaker, who served on my supervisory committee and provided many hours of helpful discussion and careful advice on earlier drafts of this manuscript and Ivan Sutherland, who originally stimulated my interest in this area.

The enormous energy of the computer graphics and image processing communities at the University of Utah has provided a unique environment for the research discussed in this paper. I owe many thanks to my colleagues whose ideas and frequent software support stimulated my work on innumerable occasions. In particular, I would like to express my gratitude to Bui Tuong Phong whose spirited example and good-natured heckling helped push this research to completion.

Thanks must also go to Mike Milochik who is responsible for processing the photographs and to Brad Morgan who provided system support and general assistance on the computing facilities. And of course, the utmost appreciation must go to Judy (editor and illustrator) and Giles whose diversionary tactics and daily attentions made my work far more pleasant and my leisure well spent.

TABLE OF CONTENTS

ABSTRACT

This paper describes work toward improving the quality of computer-synthesized shaded images. Current (practical) hidden-surface algorithms produce an image whose precision is strictly limited by the number of picture elements. Problems caused by this limitation are described and explained in the text. Other image production media such as television and screen printing processes exhibit the same problems, but to a far lesser extent. Therefore, current image production media are surveyed and compared to computer-synthesized images with a view toward establishing a criterion for acceptable image quality.

The problems to be seen in computer-synthesized images are predominantly caused by a phenomenon known as "aliasing". Aliasing is an artifact of the process of producing an image made of discrete dots. A well known cure for aliasing is motivated and explained in the text. The notions of sampling and convolutional filtering are introduced to assist the explanation. Based on these techniques, an algorithm for computing the intensity of a single dot of an image is developed. The algorithm is applied to a test pattern which serves as a "worst case" for the problems caused by aliasing.

The dot production algorithm is then applied to three extant classes of hidden surface algorithms: scanning algorithms, z-priority algorithms and z-buffer algorithms. The latter two classes make use of a procedure known as a "tiler" and the implementation of a tiler is discussed in detail.

Some results of that implementation are compared with conventional computer- synthesized images of both equivalent and doubled resolution.

Many approaches to the aliasing problem have been proposed. A brief survey of the more interesting of these approaches is offered. Each method is explained, evaluated and compared with the approach advocated in this thesis.

An appendix is devoted to the problem of compensating for nonlinearities in the image production hardware. Without such compensation, grey level distortions can destroy the effect of measures taken to avoid aliasing.

CHAPTER ONE

INTRODUCTION

THE PROBLEM

In recent years, it has become relatively common to see computer-synthesized images of opaque objects with hidden surfaces removed. Until very recently most of the research effort has been primarily concerned with hidden-surface algorithms, i.e. with determining which parts of an object are visible and should be displayed. However, it has become obvious that higher quality computer-synthesized shaded pictures are needed for applications which require a realistic image.

Computer graphics has been quite successful in solving visual problems involving images which can be abstracted. Problems in electronic circuit design, for example, can generally be expressed in a symbolic language using easily generated symbols. On the other hand, visual simulators for training airplane, oil tanker or spacecraft pilots require a highly realistic image. Reasonably realistic images have not been hard to obtain. However, convincingly realistic images are still very rare. The gap between a recognizable and a convincingly realistic image is much wider than generally realized. Three short, curved lines surrounded by a circle can be instantly recognized as a face and few would complain that it didn't

look right. On the other hand, elaborate shaded pictures of faces are not as readily accepted, because viewers become concerned with details.

To date, little effort has been put towards correctly depicting fine details in computer-synthesized, shaded pictures. One reason is that most computer-synthesized images have come from rather simple "scene descriptions". A scene description is the numerical data describing the scene to be represented in the computed image. Many thought that the deficiencies in computer-synthesized pictures lay in the fact that the scene descriptions were not rich enough. However, recent images with much richer scene descriptions have demonstrated that major problems exist in the rendition of fine detail. These problems characteristically occur in three specific situations: (1) along sharp edges such as found in the silhouette of an object or a crease in a surface; (2) in very small objects and (3) in areas of complicated detail.

All problems associated with these three situations can be traced to the same source: since shaded pictures must be displayed on a finite medium, only a finite resolution need be calculated. For example, the most common display, a standard television receiver, is usually designed to display an image of from 300 to 800 horizontal lines. Therefore, computer-synthesized images with 512 lines are convenient.

A 512 line image requires that the intensity be calculated for each of 262,144 (512 times 512) positions on the screen where each position is represented by a fixed-size dot of the proper intensity. While this would

appear to require a great deal of computation, most of the dots can be calculated in groups using incremental techniques. Therefore, the computation per point can be modest. Furthermore, since the number of screen positions is always the same, the computation remains partially independent of the scene complexity.

Dave Evans of the University of Utah realized the importance of this independence in the mid nineteen-sixties. Out of that insight grew the shaded computer graphics work at Utah, including a number of algorithms which capitalize on the limited resolution available for display of the computed image. However, this approach leads directly to the problems which have motivated this thesis.

These problems are most easily seen along the silhouettes of objects where there is a sharp change in the intensity from object to background. A straight edge separating a light area in the image from a dark one will look straight only under special circumstances. Generally, the edge will appear jagged. This phenomenon has been referred to as the "staircase effect" or the "jaggies". Because of the limited resolution, an edge can lie only along the dots which make up the image. Since the dots are arranged on a grid of horizontal and vertical rows, if an edge is nearly but not exactly horizontal, it cannot lie along one row of dots. At some points the edge must move abruptly from one row of dots to the next. These points may often be obvious in the resulting image [Fig. 1.1].

In a simple picture there will be only a few of these jagged edges.

However, the eye emphasizes visible edges in an image and attention is drawn to precisely those areas in which problems occur. Furthermore, since the eye is far more sensitive to high-contrast than to low-contrast edges, staircase effects on an edge separating a black area from a white one will be much more noticeable than those on an edge separating a grey area from a somewhat darker grey area.

If computer-synthesized images are to achieve a greater degree of realism it will be necessary to generate images of arbitrarily complicated scenes. In a complicated scene there will not only be many potentially jagged edges, but also many very small objects and details. Small objects pose a problem similar to that of edges in limited-resolution images because they can get lost between the dots. This phenomenon occurs because each dot in the image represents a sample point in the scene, an infinitely small spot on some surface being depicted. The process of taking sample points from a continuous scene is known as "sampling". If an object is small enough to fit between adjacent sample points then it will sometimes be missed. Therefore, a very small object may disappear entirely; a long, thin object may appear in some places and not in others, giving the appearance of a string of beads; and a highly detailed object such as a human face may lose some of its features.

In still pictures, these effects can often be ignored if the scene itself is sufficiently distracting. Moreover, it is often possible to fiddle with the description of the scene itself, adjusting things this way and that to minimize jagged edges and missing objects. However, in an animated

sequence of images these limited resolution problems become much more obvious. High-contrast edges appear to have small armies of ants running along them as the slope of the edge changes; small objects and details flash on and off distractingly; slightly larger objects appear to change size and shape without reason; even a simple, horizontal edge which looks fine in a still picture can be seen to jump from one row of dots to another as it moves up or down the display.

GENERAL APPROACHES

There are three basic approaches to correcting or preventing the above effects:

(1) The first approach is simply to increase the resolution. Increasing the resolution does inceed improve the quality of the image, although it also increases the expense of producing the image. Doubling the resolution multiplies the number of dots in the image by four, and the necessary computation by more than a factor of two.

Increasing the resolution means that sample points occur more frequently. Thus if the image remains the same size, finer details can be represented and the position of edges and other features in the image can be represented more precisely. As a result, the jaggedness along edges may become fine enough that it won't be noticed by the casual observer, the

size and shape changes in small objects will not be so great and a line jumping from one row of dots to the next will not have to move as far.

However, no matter how small the distance between sample points, a point source of light, an infinitely small, infinitely bright spot, will always be lost. The sample points are infinitely small and therefore cover no area in the space to be represented by an image. So an arbitrarily placed point source of light has an infinitely small chance of lying in exactly the same place as a sample point. Thus it appears there is no way to represent such point sources.

Although infinitely small objects cannot be represented in a finite medium such as the storage area of a digital computer, the point source problem points out the fact that as long as the scene description (which must be stored in a computer) defines objects to a precision greater than is allowed by the display medium, the likelihood of small objects getting lost between the dots is great.

(2) Under certain conditions, the staircase effects along edges may be lessened by blurring the final image. These conditions obtain when the size or shape of the dots in the display is a primary cause of the undesireable effects observed. Blurring causes the intensity change across an edge to be smoothed, making it less obvious to the eye. While there is virtually no additional cost involved, blurring cannot help with the problems caused by the process of evaluating the image only at sample points. Furthermore, the image will lose sharpness which may be better

retained by other methods.

Other post processing techniques have been suggested, some of which will be discussed in chapter four. It appears possible to eliminate jagged edges with such methods. However, lost objects cannot be retrieved and size and shape changes in small objects may sometimes be exaggerated.

(3) Therefore, a more attractive approach is to allow the intensity of a given dot to represent a blend of the intensities on both sides of an edge. Instead of forcing each dot to take on one intensity or the other, a smooth gradation of shades can be used to avoid abrupt changes [Fig. 1.1]. A small object may also benefit by being represented as a blend of its intensity with that of its background. This would allow an object to fade smoothly into the sunset rather than to shrink to the size of a dot and then suddenly blink out. The blending is best accomplished by causing the dots in an image to represent finite areas from the scene description, not just infinitely small sample points. The reasons behind this approach and the methods for implementing it form the bulk of chapters two and three.

The cost of blending shades within a dot is not easy to measure. The blending needs to be done only where there are high contrast edges or small details. Thus the expense will depend on what might be called the "visual complexity" of the picture. Visual complexity is a measure of the business of the image. A photograph of a crowd at a sports event would be highly complex; a photograph of a solid colored beach ball on a solid color background would be much less complex. Those high-contrast edges and small

details which attract the eye also contribute to the visual complexity of the image. Workers in image bandwidth compression often try to isolate these parts of an image since they contain the information which makes the image recognizable. They have found that relatively few of the dots which make up an image contribute to the edges and small details [Graham 1967]. Furthermore, the images generated by computer with present-day methods are visually much simpler than those which have been subjected to bandwidth compression experiments. Therefore several times the effort can be expended to produce the dots which need blending before the expense of producing the entire image doubles.

ESTABLISHING A REFERENCE FOR IMAGE QUALITY

Images reproduced by photography, television and printing processes are viewed every day by hundreds of millions of people. It is informative to consider such mass-produced images in order to establish some reference points for image quality. Most computer-synthesized, shaded images are produced on a digital display which, for every calculated position in the image, puts up an individual dot varying in either intensity (in the case of cathode ray tube displays) or size (in the case of hard copy displays). There are generally either 256, 512 or 1024 rows of dots vertically and horizontally. Each dot represents a sample point on some single object in the scene description.

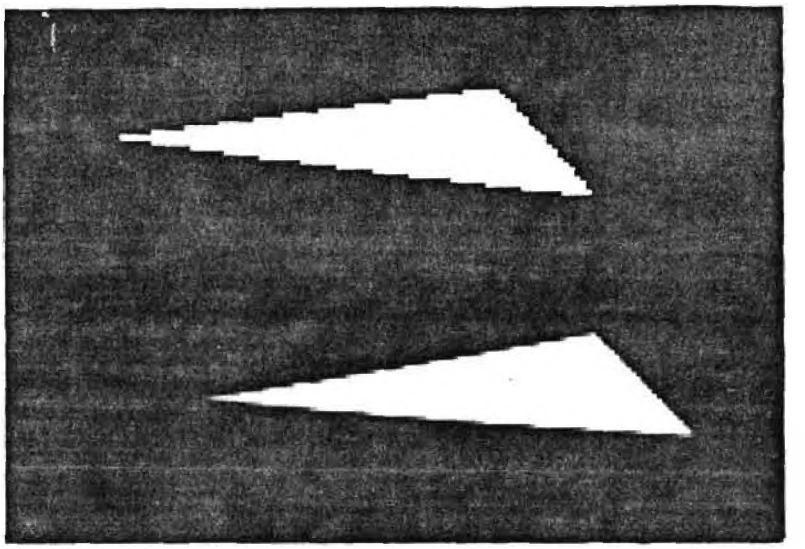


Figure 1.1 The "jaggies" cured by blending intensities

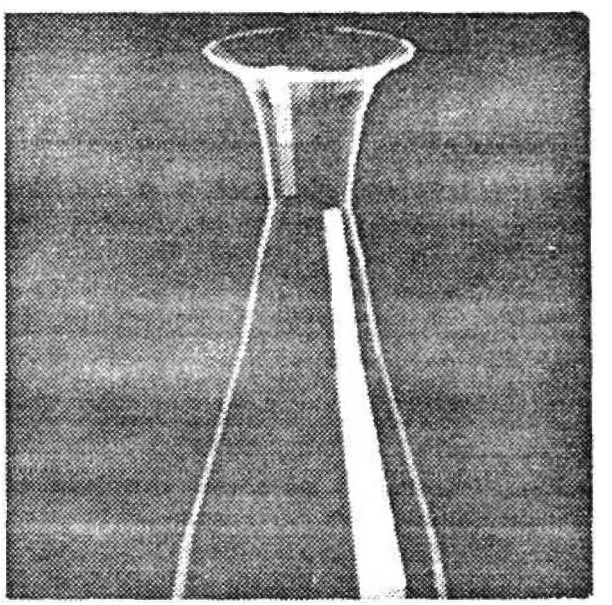


Figure 1.2 Slightly varied dot shapes enhance detail as seen along the top lip of the vase.

Audiences in the United States have become accustomed to viewing television broadcast at U.S. standard bandwidths. Under ideal circumstances the definition of American television (restricted to the standard four megaherz bandwidth) is approximately equal to the definition of a 512 by 512 dot image. However, there are important differences between a standard television display and a digital display.

In the television process, the image is broken up into continuous, horizontal strips. There are no such things as sample points and image dots along a horizontal strip. Therefore, the problems caused by sampling do not occur in the horizontal direction. However, the image is sampled in the vertical direction. In a sense, television takes line samples as opposed to point samples. The effects of sampling may sometimes be seen in television images, most frequently on the outlines of people whose images have been electronically superimposed on a different background and on clothing which has a pattern of just the right size, such as a tweed jacket or finely checked shirt.

The effects of sampling are not nearly as bad in television as in computer-synthesized images primarily because the television process samples areas of finite size rather than infinitesimal lines. If a television camera is finely focussed, so that a given horizontal scan represents a very fine line in the scene being pictured, the resulting image looks terrible. Chapter two provides a more thorough explanation of this phenomenon.

The structure of an image made on photographic film is much different from either television or a digital display. The image in a photograph is formed from small grains of random size, shape and distribution which tend to gather in clumps on the film.

The size of the grains and the distribution of the clumps determine the resolving power of the film. The resolution of the film is not so clearly defined as for television or digital displays. Because of the random distribution of the grain clumps there may be no grains in some microscopic areas of film and very dense packing of grains and thus high resolving power in other areas. As photographs are greatly enlarged, the grain clumps become quite evident. Although the film may be capable of resolving finer detail, the random clumps of grain give the image such a mottled appearance that finer detail cannot be isolated from the background.

As a rule of thumb, standard photographic film can be expected to resolve about 25 line pairs per millimeter before the graininess begins to obscure the image [Hunt 1975]. Of course the resolution varies among film types, but 25 line pairs per mm is a useful figure. This means that 8 mm film (with a 4.4 to 5.3 mm image width) can usefully depict about 110 to 135 line pairs. This is roughly equivalent to a 250 line digital display which can, of course, depict 125 black and white line pairs.

Extrapolating from these figures, it appears that a 512 line display is roughly equivalent in resolving power to 16 mm motion picture film; to achieve an image equivalent to the resolution of 35 mm theatre movie film,

a 1000 line display would be needed; and to match wide-screen 70 mm film a 1000 line display with at least 2000 dots per line would be needed.

These comparisons are somewhat misleading since motion picture film images are seldom viewed as still pictures. Because the position of the grains making up the image changes randomly from frame to frame, film has a distinct advantage over a digital display as a medium for animated images. While watching an animated sequence on motion picture film, the eye is able to sum the image over several frames. As several frames are summed, the random effects of grain clumps tend to cancel out. On the other hand, the image is reinforced. The graininess is a source of noise, a random disturbance, while the image is the signal or message to be conveyed. Therefore, summing several frames improves the signal-to-noise ratio. This allows the signal, the image, to be perceived at a higher resolution than in a still frame because the lessened noise allows finer details to be seen.

In a digital display, the limit of resolution lies in the grid of samples or points from the scene. Since the sampling grid is regular, noise cannot be introduced to interfere with the finer details and summing several images from a digital display cannot enhance the image.

A comparison between digital displays and film formats displayed as still pictures is perhaps more meaningful. Using the previously established rule of thumb, 35 mm slide film with a 24 by 36 mm image area should be roughly equivalent to an 1100 line display with 1650 dots per line. 4 by 5 film

such as that used by professional photographers could only be matched by a 5000 line display with 6300 dots per line. Of course there are some photographers who use 8 by 10 film, but the average person rarely sees an image of that quality.

As with the television process, it must be emphasized that each grain in the film responds not to an infinitesimal point but to a small, but finite, area in the scene. Thus even the smallest details, though they may not be distinguishable, have contributed to the image.

It is also instructive to look at screen printing processes. Printed images appearing in everything from newspapers to glossy magazines are all produced by screen processes. In a screen process an image is reproduced by regularly spaced dots of ink which vary in size and, to some extent, in shape. The screen processes, therefore, yield an image much more like that of a digital display.

There are important differences however. In a cathode ray tube digital display, the intensity is varied while the spot size remains the same over the image. In a hard-copy digital display as produced by electrostatic printer-plotters, xerographic printers or other dot matrix printers, the dot group size is varied to achieve the same effect. An image printed using a contact screen, however, uses variation not only in the size of the dot but also small variations in the shape of the dot. The shape variations allow the dots to bleed together in such a way that thin lines and other small details are favored [Fig. 1.2]. Thus a well-printed screen

process image has advantages over a digital display of the same dot density. Nonetheless, the images are similar enough to make rough comparisons meaningful.

The coarseness of the screen used in printing is generally dependent on the quality of paper used. In newspaper work where the low quality paper doesn't allow very closely spaced dots, a screen with as few as 20 or as many as 35 dots per cm may be used. Thus a 2 column picture about 10 cm square might consist of as few as 200 rows of about 200 dots apiece. A full-page image on a 14 by 21 inch page would encompass about 1750 by 2650 dots at 20 dots per cm or about 3150 by 4650 dots at 35 dots per cm.

In magazine work, greater densities of dots can be used due to the higher quality paper. About 40 dots per cm is a common density for slick magazines, so a full-page ad in the standard 8 1/2 by 11 inch format would involve about 860 by 1120 dots. Screens as fine as 120 dots per cm can be made and used. Even with such fine screens however, an area of around 16 by 20 inches would be needed to achieve roughly the definition available in 4 by 5 photographic film.

The characteristics of the human eye must also be considered. Under normal conditions, psychophysical measurements have shown a resolving power of about 20 cycles per degree maximum [Cornsweet 1970, Hunt 1975]. Under conditions involving very high contrast, 50 or 60 cycles per degree can be detected. However, in shaded images which are printed or displayed on a television screen, 20 cycles per degree is a reasonable maximum.

Psychophysical measurements made on cathode ray tubes at maximum attainable contrast ratios have shown an ultimate resolving power of 25 cycles per degree for that equipment [Levinson 1972].

At close viewing distances of around 250mm (10 inches), no more than 5 cycles per mm. could normally be resolved. Thus at 250 mm the finest detail in a screen print of about 100 dots per cm (250 per inch) could be seen by only a few people. There is therefore little point in making screens finer than this for printing purposes. Additionally, by the earlier figures for 35 mm slide film, an enlargement to a 4 by 6 inch print would yield no perceptible graininess at a 10 inch viewing distance, although some details can emerge at greater enlargements. The preceding also indicates that a 512 line digital display on a 21 inch screen (measured diagonally) can be viewed from as close as 7 or 8 feet before the finest detail can be seen. Of course there is considerable variance in visual acuity among people, and these figures, which correspond closely with the author's experience, can only be considered approximations.

A further relevant fact about the eye is that it can detect a flicker rate of no greater than 50 cycles per second under normal conditions. Thus motion picture film displayed at 24 frames per second has to be flashed twice per frame to avoid flicker. Television pictures delivered at 30 frames a second have to be interlaced so that half of the picture is displayed in each of two passes to achieve an effective flicker rate of 60 cycles per second.

Furthermore, the eye retains an image for about two-tenths of a second [Rose 1973]. Thus in viewing motion pictures on film or television there is a constant overlap of 5 to 7 images. This overlap serves to strengthen the image at the expense of random deviations in the display medium such as graininess in film or noise in television signals.

IN SUMMARY

The figures derived in the previous section relating the resolving power of various media to that of digital displays don't provide the information needed to evaluate methods for getting rid of jagged edges and disappearing detail. The media described avoid these problems by allowing each dot, grain or small segment of scan line to represent the average intensity of a finite area in the scene depicted. Thus edges and small objects are blended with whatever lies adjacent to them in the image. If computer-synthesized images are to be compared with other media, the images must be similarly generated, i.e. edges and small details must be blended with surrounding parts of the scene.

By doubling or quadrupling the resolution and leaving the viewer in a proper position for viewing the base resolution, several dots can be made to merge. Thus, blending of a crude sort can be made to occur. As stated above, this sort of blending does not help the representation of very small objects. Doubling the resolution, which yields four dots where before

there was one, makes it possible to produce four different blends of the intensities of two surfaces along an edge. If the resolution is quadrupled, 16 different blends of two surfaces can be produced. By the same argument, an algorithm which blends at the base resolution should be able to produce superior pictures at the same viewing distance since it can be expected to produce many more possible blends of the two surfaces thus more accurately reflecting the actual position of the edge.

Because of the preceding arguments, the discussions in the following pages will focus on developing methods for blending surfaces within image dots which lie along object outlines, in small objects or in other areas of fine detail. An approach to generating images at higher resolutions with only a minimal increase in cost is explained in chapter four along with other less acceptable approaches to the problem. However, the principal effort of this work, which is discussed in chapters two and three, will concentrate on achieving better images at currently practical resolutions.

CHAPTER TWO

FILTERING

MOTIVATION

In the last chapter it was mentioned that television and screen printing systems operate in such a way that each small area or dot in the final image represents a particular small area in the original scene. By contrast, a dot in a computer-synthesized image usually represents an infinitessimal point on some surface in the scene description. Therefore a computer-generated dot can represent only one surface at a time where television and printing systems can represent a blend of several different surfaces. It is worthwhile taking a closer look at exactly what happens at the dot level in screen printing and television.

Consider what happens when a television camera is aimed at a very small white spot on a black background. An image of that spot will appear within the television camera as a charge distribution on a plate known as a target. An electron beam scans the target from top to bottom in a series of horizontal sweeps which allow the charge to be read off as a continuously varying signal. The camera is defocussed to diffuse the image of the dot on the target. The image is diffused over an area large enough to cause some charge to be read off by at least two adjacent sweeps of the electron beam [Fig. 2.1]

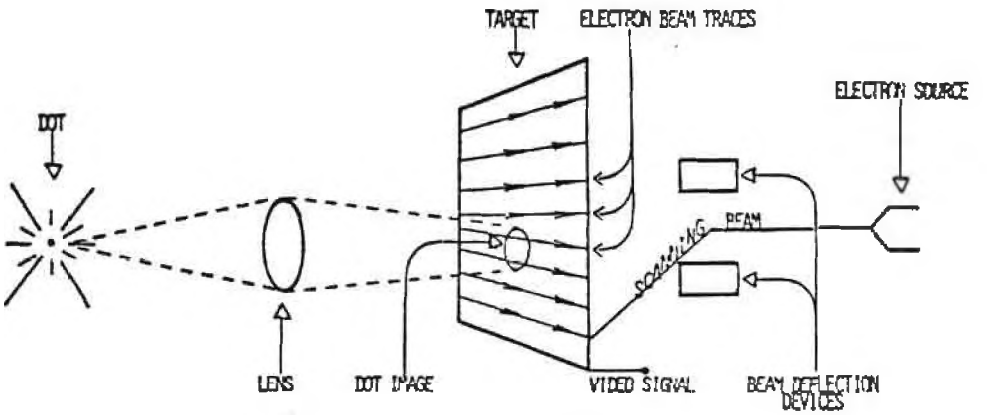


FIGURE 2.1 TELEVISION IMAGING SYSTEM

When the image is displayed, the corresponding electron beam in the television receiver is defocussed sufficiently to ensure that adjacent sweeps will overlap by an amount equivalent to the spread of the dot's image on the target. The defocussing ensures that no details of the scene will be lost between the lines and effectively limits the range of frequencies in the signal to those which can be transmitted in the broadcast bandwidth. Therefore, the picture can be said to have been filtered by the television process.

A less obvious filtering process occurs in screen process printing. Commonly, a printed image in a magazine or other publication is produced with a contact screen. A contact screen is a sheet containing a grid of small transparent windows. Each window is fully transparent at its center, but becomes increasingly opaque towards its edges. A printed image is

produced by placing the screen in contact with a high-contrast photo-sensitive material and exposing it to the desired light pattern. Since the photo-sensitive material has an all-or-none characteristic, it produces a printed image consisting of small dots of varying size.

Because the screen actually contacts the photo-sensitive material, small details may affect the shape of a dot. For example, a line crossing one of the transparent windows will produce a dot somewhat elongated in the direction of the line. This effect serves to enhance the definition of detail in the image (see Figure 1.2).

Clearly, if there is a very small, bright spot in the light pattern as it reaches the screen, this spot may fall on an opaque area of the screen resulting in a loss of detail. Therefore, the light pattern must be sufficiently defocussed to insure that no detail in the image is lost.

When a flat grey area is reproduced by the black and white television process, the resulting image appears solid grey even on close inspection, assuming high-quality, well adjusted equipment. The vertically adjacent strips in the image are designed to overlap such that they will sum to the same shade of grey at all positions. On the other hand, a color television image seen at close range appears to consist of many triads of red, green and blue dots or slits. When a grey area is reproduced by a screen printing process, the resulting image is a grid of black dots on a white background.

There are really two filtering stages in the processes just described. The first filter stage involves properly defocusing the light pattern as described for both television and printing processes. The second filter stage is markedly different for the two processes. In the black and white television process, the electron beam is properly defocussed to allow adjacent strips to overlap. In the printing and color television processes, the image is left in a relatively unfiltered state; in this case, the observer's eye serves as the second filter. Filtering is discussed more thoroughly in the following section.

SAMPLING

Whenever a continuous signal is to be represented digitally it must be sampled, i.e. the signal must be measured at discrete positions. The measurements provide a digital representation of the signal. If the signal and the positions at which it is measured satisfy certain conditions, the sampling theory has shown that the continuous signal can be faithfully reproduced from its digital replica [Oppenheim and Schafer 1975].

It is important that the signal be filtered before sampling to insure that these conditions are met. The consequence of failing to properly filter the signal before sampling is a phenomenon known as "aliasing". Aliasing occurs when a lower frequency signal appears as an "alias" of a high frequency signal after sampling. For example, a signal with a frequency of

12 cycles per mm., when sampled at a rate of 10 samples per mm, will appear identical to a signal with a frequency of 2 cycles per mm. [Fig. 2.2]

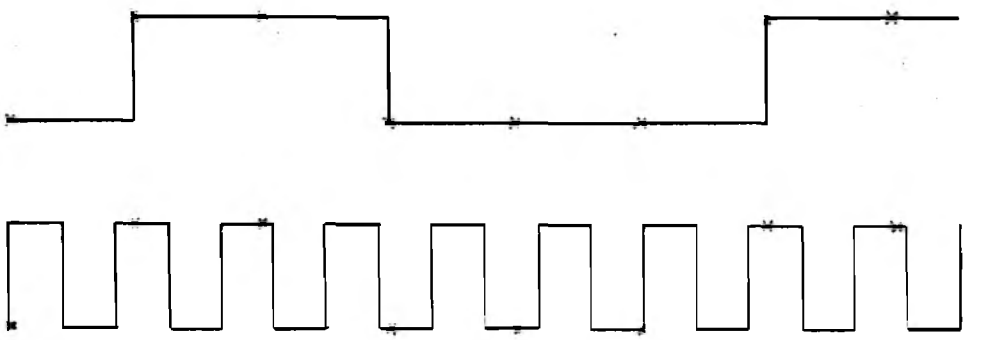


FIGURE 2.2 ALIASING

Therefore, highly periodic images of scenes involving, for example, picket fences or venetian blinds may appear, when sampled, to be made up of a few broad strips rather than many fine lines. When scenes including such objects are encountered, it is better that the fence or blind appear a uniform grey if its frequency is too high to be reproduced. This can be achieved by eliminating the higher frequencies through a process known as "low-pass filtering", i.e. the application of a filter which passes lower frequencies, but blocks higher frequencies.

Assuming that the proper filtering has occurred before sampling, the filtered signal can be faithfully reproduced. Reproducing the signal involves representing each sample in such a way that the reproduced signal

has no frequencies higher than the original filtered signal. This can be accomplished by representing each sample as a rectangular pulse and then low-pass filtering the resulting signal.

The result of failing to properly filter the signal during reconstruction, in the two-dimensional case, is known as "rastering". Rastering is an artifact of the actual structure of the final image. If the beam in a television monitor is incorrectly focussed, the resulting effects can be called rastering.

Rastering appears in many computer-synthesized images where a television monitor has been driven to produce a grid of 256 by 256 dots while the beam is focussed for a grid of 512 by 512 dots. The individual dots can be clearly seen, a condition known as "static rastering". Static rastering is also evident when a printed image is viewed too closely.

"Dynamic rastering" can be seen in Figure 1.1. In this case, elements of the picture are not visible as long as adjacent elements are not of noticeably different shades. In short, static rastering occurs where the dot structure of the image is visible and dynamic rastering where the boundaries between adjacent elements are noticeable, the latter a condition dependent on varying shades.

Figure 2.3 shows the entire process of low-pass filtering, sampling and then reproducing a signal. For faithful reproduction of frequencies up to 1/2 the sampling rate, both the pre-sampling and final filter must be ideal

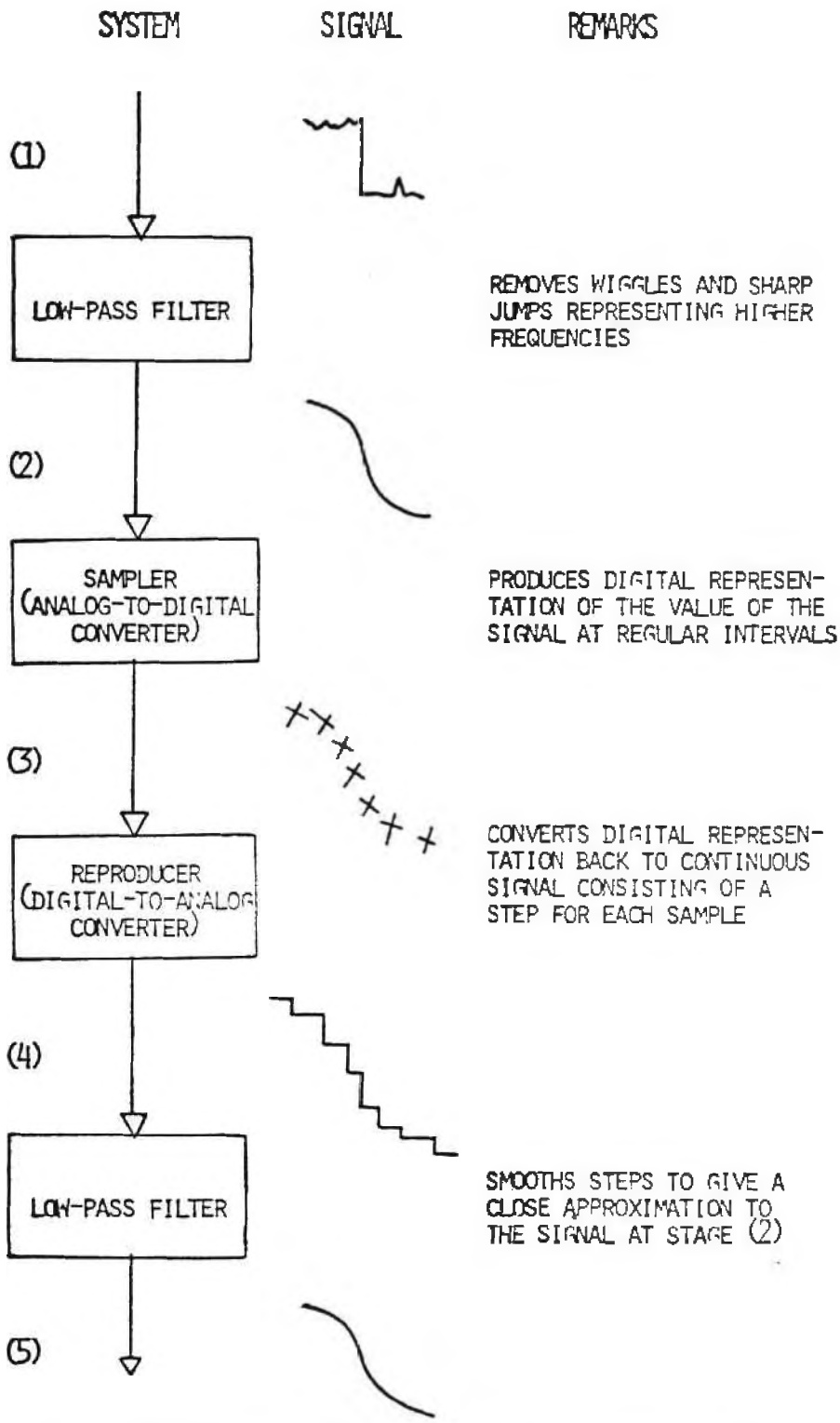


FIGURE 2.3 SYSTEM FOR SAMPLING AND REPRODUCING A SIGNAL

or perfect low-pass filters. Unfortunately, such perfection is not attainable when filtering images since negative numbers are required. There is no such thing as a negative intensity.

The scene description from which a computer-synthesized, shaded image is produced is an ideal representation of a two-dimensional continuous signal. The two-dimensional signal is usually a representation of a perspective transformation of data which is itself a model of some real-world object. A hidden-surface algorithm must generate a regular grid of samples from the scene description and send them to a display [Fig. 2.4].

It is clear from the above that the scene description must be low-pass filtered before sampling. Edges on object outlines in the scene description cause discontinuities in the corresponding two-dimensional signal. Such edges are equivalent to a step-function on the neighboring regions of the signal. A signal containing a step has infinitely high frequencies. By way of explanation, consider a step as part of a square wave.

A square wave is actually a sum of sine waves of varying frequencies. A sine wave, of course, represents a single frequency. To be more precise, a square wave represents the following sum:

$$\sum_{n=1}^{\infty} \frac{\sin(2n-1)x}{2n-1} . \qquad (1)$$

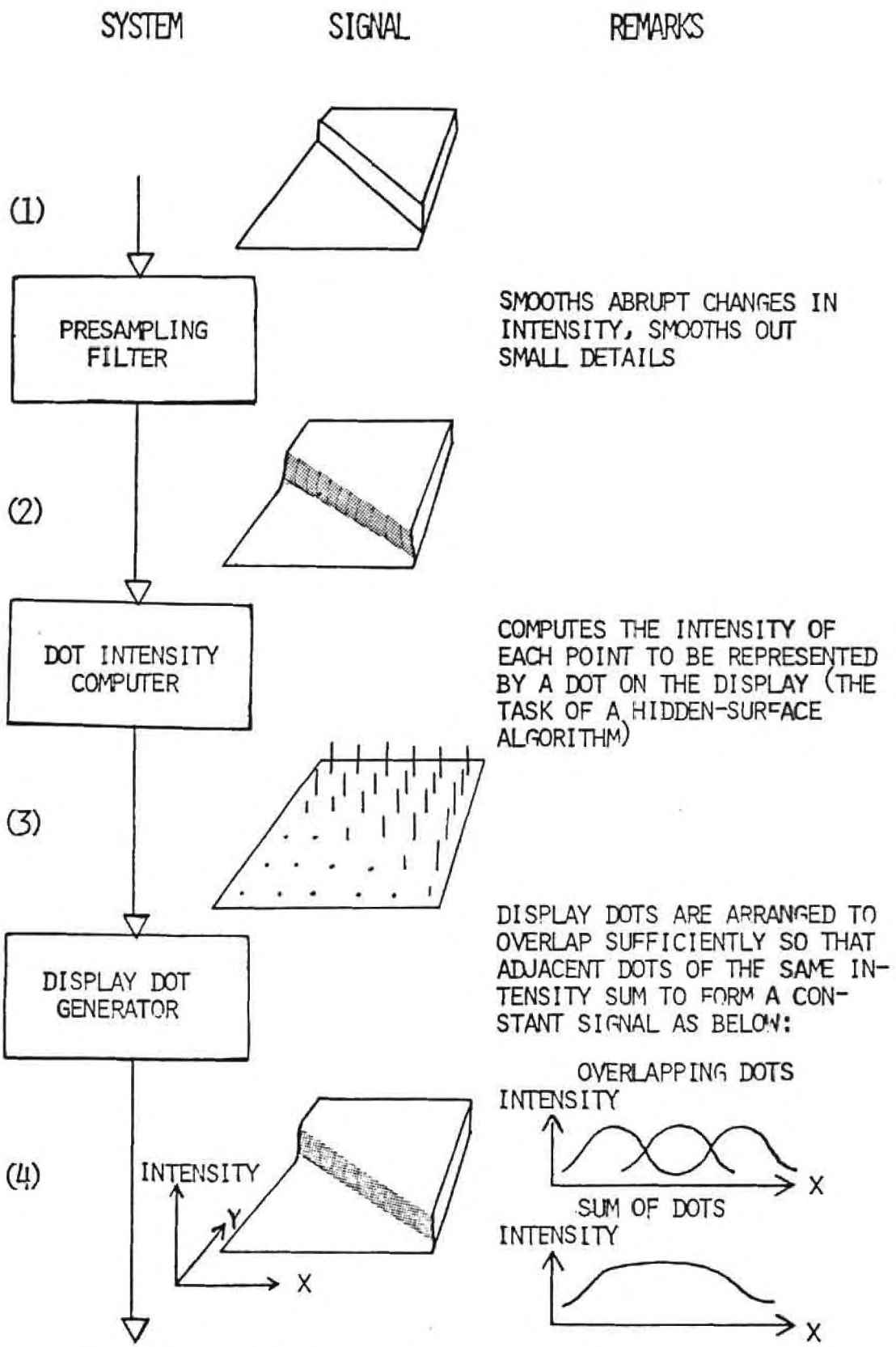


FIGURE 2.4 A TWO-DIMENSIONAL SAMPLING SYSTEM

Since the frequency of any component of the sum is simply n times a constant, a square wave contains infinitely high frequencies.

Clearly, any scene description containing edges will require filtering to minimize aliasing problems. The most obvious results of not filtering are the staircase effects described in the introductory chapter. An approach to implementing a filtering scheme for scene descriptions is described later in this chapter.

The problem of rastering or improper filtering of the final image presents a different situation. In the case of a cathode ray tube display, the spot size and overlap can be adjusted to present a flat field for any reasonable viewing distance. The intensity across a single spot generally represents a gaussian distribution of the energy of the beam. The gaussian shape of the spot acts as a low-pass filter on the image [Fig. 2.4].

In the case of a hard-copy display, the spot shape has no low-pass characteristics; the spots can be thought of as pulses of constant height and varying diameter. Therefore, the image can be considered a two-dimensional pulse-width modulated signal. This kind of signal clearly contains infinitely high frequencies. The intensity-modulated signal generated by a cathode ray tube display also contains high frequencies, but their strength is not as great.

Although properly frequency-limited signals cannot be produced on current displays, the situation can always be saved by removing the observer a

suitable distance from the display. As pointed out in the introductory chapter, the eye can detect only frequencies of less than approximately 20 cycles per degree on such displays. Therefore, the observer's eye will always serve as the necessary final low-pass filter, assuming that eye is sufficiently far removed from the display. An intensity-modulated signal and a pulse-width modulated signal from the same source will appear identical after adequate low-pass filtering.

CONVOLUTION

Convolution is a process by which a two-dimensional continuous signal may be filtered. Convolution is basically a process which combines two functions to produce a third function. Formally, convolution is expressed by the following integral:

$$G(x) = \int_{-\infty}^{\infty} F(s) \cdot H(x-s)\, ds \ . \qquad (2)$$

The function G is produced, in effect, by turning the function H around and sliding it past F. The value of G for a given value of its independent variable represents the integral of the product of the functions F and H for a particular relative position of the two. If either F or H is zero ouside some small region, then G becomes easy to evaluate for a given X since the integral is then identical to the integral over the finite bounds of the non-zero portion. For the purposes of this discussion, only those

convolutions in which one of the functions is non-zero over a limited range will be considered.

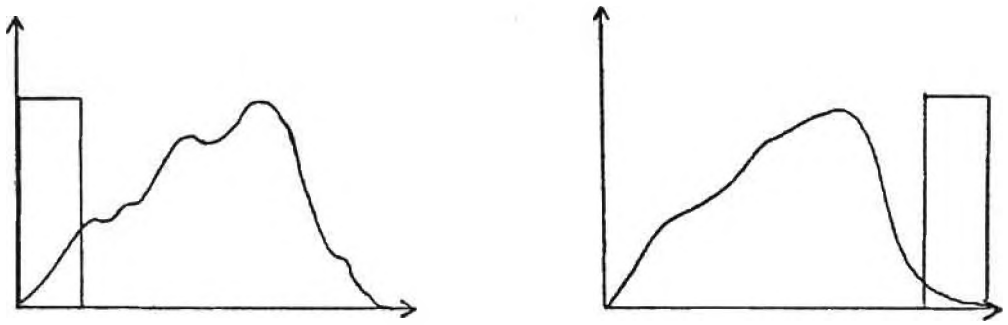


Figure 2.5 Convolution with a Box-shaped Function

As an example, consider what happens when one function is convolved with another box-shaped function [Fig. 2.5]. Notice that abrupt changes in the function are smoothed over and small wiggles ironed out. Now consider the result of convolving an impulse (a single non-zero point with a finite area under it) with the function [Fig. 2.6]. Notice that the result is simply the function itself. For any function chosen, convolution with an impulse will preserve that function. The magnitude of the impulse, i.e. the area under it, serves only to scale the function up or down.

Convolution with an impulse provides a convenient way of finding the function representing a physical system. For example, given that a television camera causes a function to be convolved with the scene being pictured, it would be nice to know what that function is. If a television

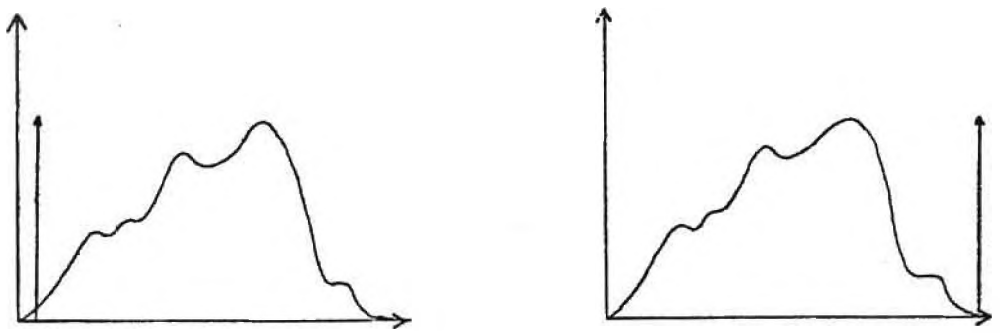


Figure 2.6 Convolution with an Impulse

camera is focussed on a scene closely resembling an impulse, the result should closely resemble the function representing the system. This function is known as the impulse response of the system since it represents the response of the system to an impulse. Therefore, the charge pattern stored on the target within the television camera represents the convolution of the scene with the impulse response of the imaging system of the camera.

In the case of images, of course, two-dimensional convolutions are being taken. Formally, a two-dimensional convolution is expressed as:

$$g(x,y) = \int\int_{-\infty}^{\infty} f(s,t) \cdot h(x-s,y-t)\, dsdt \qquad (3)$$

For a complete treatment of convolution consult [Guilleman 1963] or [Oppenheim and Schafer 1975]

As demonstrated earlier, convolution with some impulse responses tends to smooth abrupt changes and iron out wiggles. These abrupt changes and wiggles correspond exactly to sharp changes in intensity across edges and fine detail in an image. The result of such convolutions over a scene description for a computer-synthesized image is just what is desired; the effect of sharp changes in intensity and fine detail is spread over a larger area and thus appropriately blended in the final image.

In order to apply convolution to the production of a computer-synthesized image, it is necessary to move the notion of convolution from the continuous domain described above to the discrete realm of digital computers. Two-dimensional discrete convolution can be expressed as follows:

$$g(i,j) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f(m,n) \cdot h(i-m,j-m) \qquad (4)$$

Again, since this discussion is confined to functions of limited extent, the limits on the summations will be finite. Additionally, for a computer-synthesized image, the function G requires evaluation only at those values of i and j which represent dots in the final image.

APPLYING FILTERS TO COMPUTER-SYNTHESIZED IMAGES

In order to represent a scene description within a computer the scene must

be represented in a discrete space; numbers cannot be represented to arbitrary precision within a computer and thus a continuous space cannot be represented. However, there is enough precision generally available to locate the details of the scene to tolerances adequate for making a convincing image. Furthermore, there need be no worry of losing point sources of light since the continuous space of the scene description is not actually sampled, but mapped onto the discrete space. Thus each point in the discrete space actually represents a neighborhood in the continuous space. Since a discrete space is to be dealt with, discrete convolution must be used.

Clearly, a discrete convolution over an entire image requires a horrendous amount of computation. In this section, an algorithm is described for applying a filter which is separable in its independent variables to an approximation of the scene description; the algorithm is designed to limit the necessary computation. Experimental results have shown the algorithm to be quite effective in preventing the distracting effects mentioned in the introductory chapter.

Because of the particular implementation discussed here, it is important that the filter be separable in its independent variables. Thus, if the discrete function H(i,j) is being considered as a possible filter, H(i,j) must be expressible as a product of functions of i and j. Thus:

$$H(i,j) = H_i(i) \cdot H_j(j) . \qquad (5)$$

The function is applied in such a way that filtering is done separately for the vertical and horizontal directions and the results multiplied to yield the final values.

With this restriction, the discrete convolution can be expressed as:

$$g(i,j) = \sum_{l=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} f(l,m) \cdot h_i(i-l) \cdot h_j(j-m) , \qquad (6)$$

where G represents the filtered scene description produced by convolving the scene description, F, with the filter, H.

Furthermore, the function, F, can be approximated, as can any function, by rectangular blocks. If the function is broken into constant height rectangular blocks, then the function over any such block becomes a constant. Evaluating the filtered function can be made much simpler by using such an approximation. The summation can now be rearranged so that the contribution of each rectangular block is summed independently. Since the value for F is a constant within any such block, F can be replaced by a series of constants within the independent summations. The function G is now approximated as:

$$g(i,j) \approx \sum_{l=p_1}^{q_1} \sum_{m=r_1}^{s_1} c_1 \cdot h_i(i-l) \cdot h_j(j-m) + \ldots + \sum_{l=p_n}^{q_n} \sum_{m=r_n}^{s_n} c_n \cdot h_i(i-l) \cdot h_j(j-m) . \qquad (7)$$

Where p_n-q_n and r_n-s_n represent the bounds of a given rectangular block and there are n such blocks making up the approximation to the scene

description.

So that H_i and H_j may be considered separately, for a given rectangular block, the summation can be rearranged as follows:

$$\sum_{l=p_n}^{q_n} \sum_{m=r_n}^{s_n} c_n \cdot h_i(i-l) \cdot h_j(j-m) = c_n \cdot \sum_{l=p_n}^{q_n} h_i(i-l) \cdot \sum_{m=r_n}^{s_n} h_j(j-m) \quad . \tag{8}$$

This rearrangement allows relatively easy implementation of an algorithm for discrete convolution over an approximation of the ideal image. Notice that by making the rectangular blocks arbitrarily small, an arbitrarily good approximation to G can be obtained.

To implement the algorithm, lookup tables must built for summations over the functions H_i and H_j. Since H is always of limited non-zero extent, two finite tables can be built, one for H_i and one for H_j. Each table will consist of entries which represent a partial sum across the function from the lower non-zero bound to some point below the upper non-zero bound. To obtain the sum over the function between any two non-zero points, it is sufficient to find the difference between the table entries for those two points [Fig. 2.7]. With the help of lookup tables, any of the independent summations (as in eq. 8) making up the approximation to G for a given i and j can be found with four lookups, two subtracts and two multiplies.

For the application of this algorithm to a computer-synthesized image, it is necessary to calculate only those values of G which represent dots in

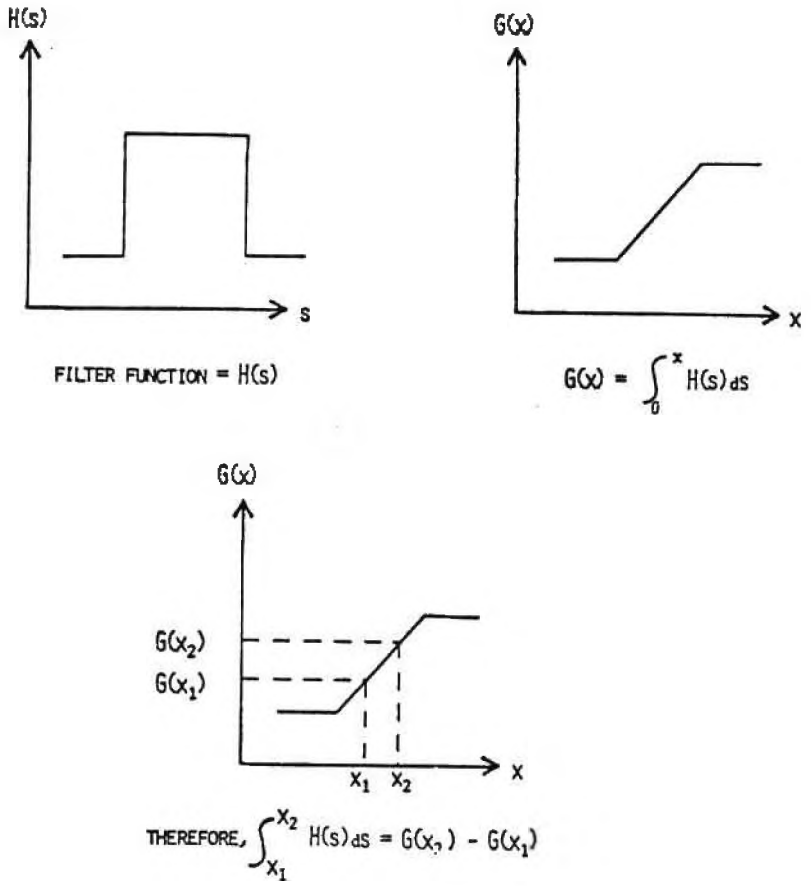


FIGURE 2.7 CONSTRUCTION AND USE OF A LOOK-UP TABLE

the final image. Each such value represents one position of the filter, H, over the image, F. To calculate the resulting function G, which will represent the intensity of the image for that dot, only that part of the image lying under the non-zero portion of the filter is considered. Generally, only a small part of the image, approximately the size of the minimum resolvable detail from the proper viewing position, will be treated for each position of H over F.

In experiments conducted to date, the non-zero portion of the filters considered has always been square. An implementation for a circularly

symmetric filter has been discussed. However, the computational load appears to be greater than with the square filter. Each such square can cover only a small segment of the ideal image; at most only a few edges in the image pass through it. So far it has been sufficient to approximate the image function by forming a single rectangular area for each of these edges [Fig. 2.8]. This results in only a few summations for each dot in the final image. In Chapter Three it will be shown that the summations need only be computed in the problem areas mentioned earlier. Therefore, the total computation remains within practical limits.

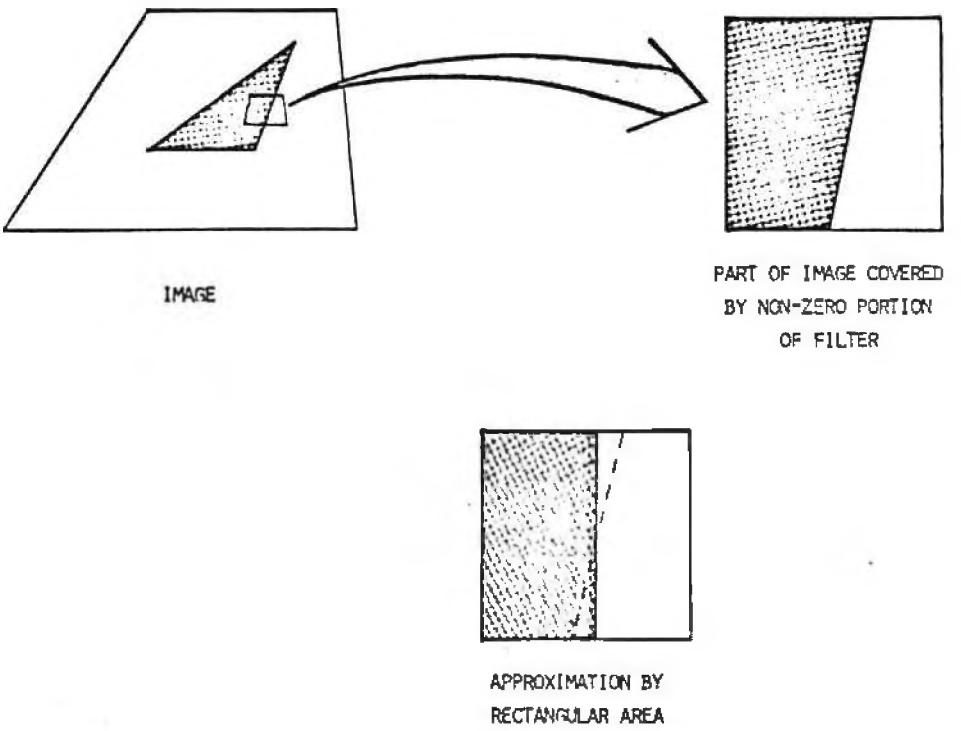


FIGURE 2.8 APPROXIMATION OF THE IDEAL IMAGE FUNCTION

EVALUATING CONVOLUTIONAL FILTERS

In order to evaluate the filter functions used with the preceding algorithm, it is necessary to develop a test pattern which emphasizes the difficulties to be corrected. Any useful test pattern must include both sharp changes in intensity across edges and areas of very fine detail. In order to enhance the cifficulties, a pattern has been chosen which generates moire patterns when edges and fine detail are not properly represented. Moire patterns have been touted for their ability to measure small variations [Oster and Nishijina 1963]. In this case they prove highly useful for detecting small inadequacies in the convolutional filter.

The pattern is produced by generating almost parallel sections of parabola using second order differences [Fig. 2.9]. The curvatures of the parabolas are maximal on the left and decreased to zero on the right. In addition, the distance between any two adjacent parabolas is decreased from left to right across the pattern. Thus, each curve is almost, but not quite, identical to the neighboring curves. Furthermore, the changes are linear across the pattern, causing any jaggedness along edges to be repeated with slight variation from curve to curve. The effects along groups of curves form elliptical patterns which are much easier to detect than jaggedness along a single edge. Furthermore, toward the right hand side of the pattern where the detail is too fine to be resolved by the display, similar patterns are caused by improper summing of the various details represented in a single image dot [Fig. 2.10].

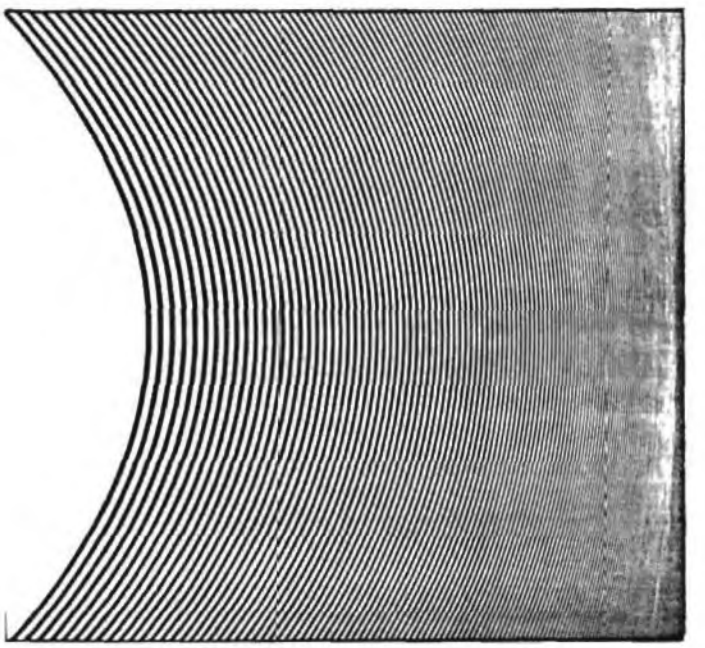


Figure 2.9 A test pattern consisting of closely spaced parabolic arcs.

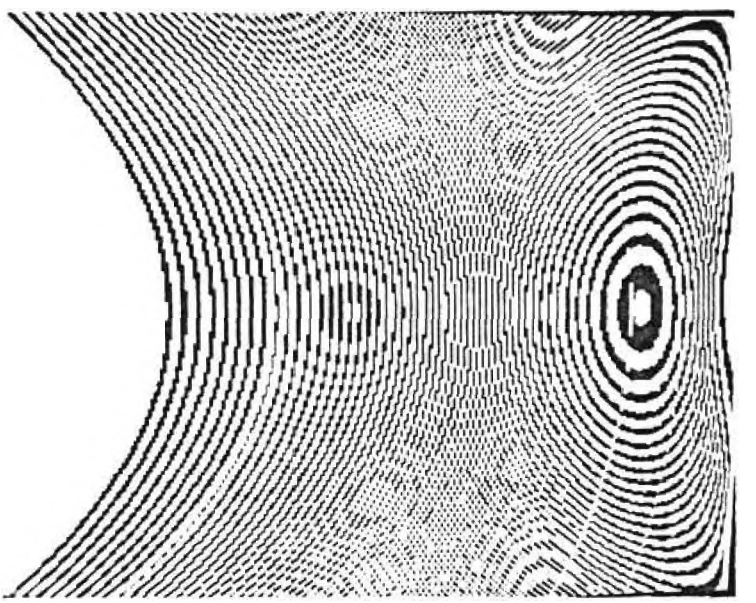


Figure 2.10 The test pattern synthesized on a 256 by 256 dot grid using techniques similar to those of conventional hidden-surface algorithms.

Programs exist to display the pattern convolved with various filters. An interactive filter design routine allows quick design and modification of a filter. Then, the pattern can be regenerated in a few minutes for visual evaluation. Equipment calibration routines are also included since the test pattern sensitivity is great enough to make consistent calibration an absolute necessity.

The figures which follow illustrate the effectiveness of various filters on computer synthesized images displayed on a standard color television monitor. In each figure, the curve at the lower left, labeled "filter" represents the presampling filter. The curve at the upper left, labeled "color map" represents the compensation curve which corrects for nonlinearity in the image production and viewing systems. See the appendix for a discussion of the compensation problem.

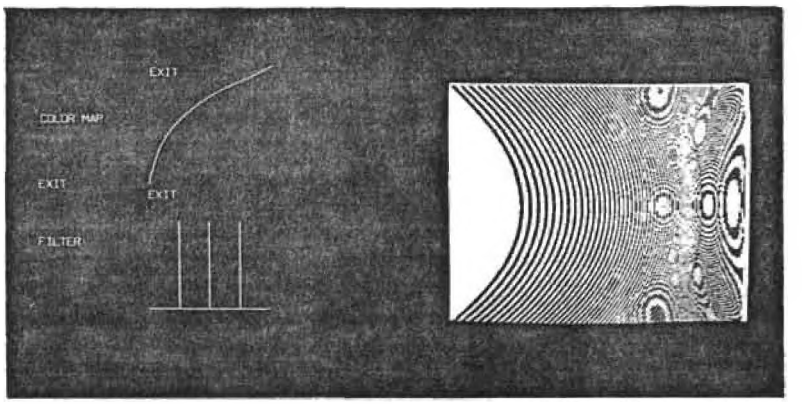


Figure 2.11 The resulting pattern when nine equally weighted samples are taken per image dot.

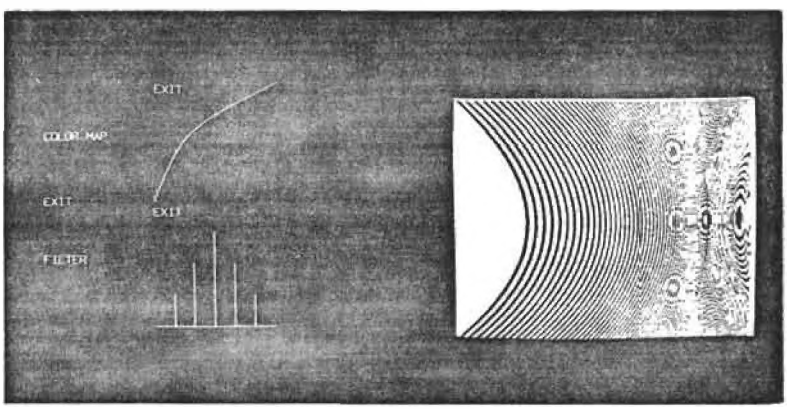


Figure 2.12 The resulting pattern when 25 unequally weighted samples are taken per image dot.

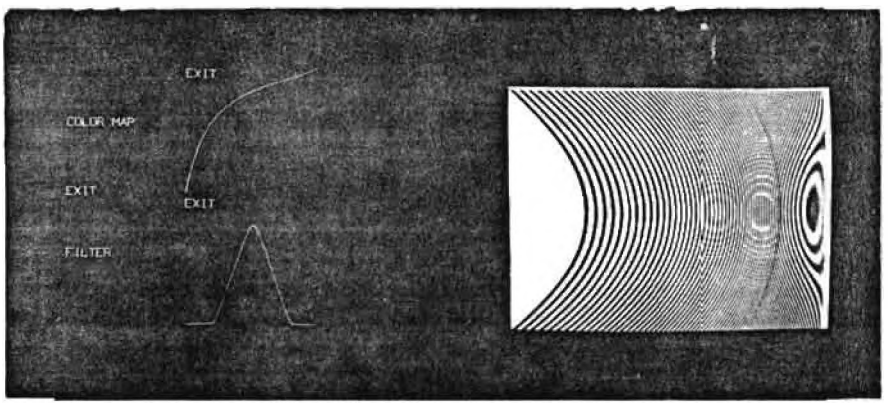


Figure 2.13 The resulting pattern when convolved with a roughly triangular filter with a base width roughly equal to the distance between sample points.

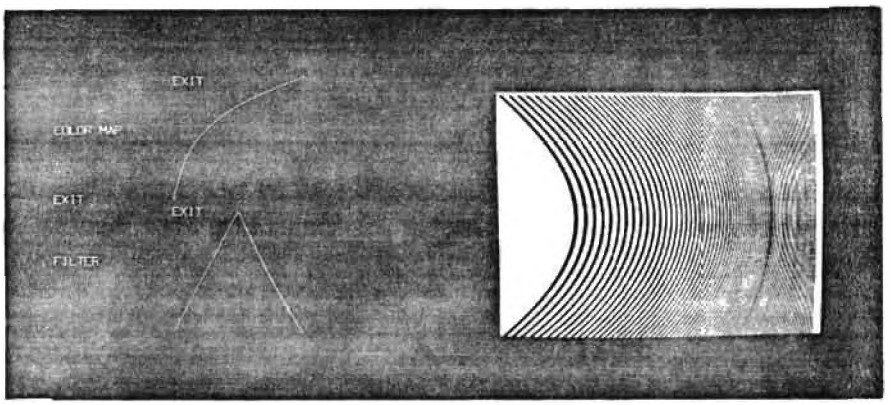


Figure 2.14 The resulting pattern when convolved with a roughly triangular filter with a base width of roughly twice the distance between sample points.

CHAPTER THREE

FILTERING IN HIDDEN-SURFACE ALGORITHMS

INTRODUCTION

In the introductory chapter, specific problems in computer-synthesized, shaded images were outlined. The second chapter (filtering) detailed a approach to these problems. In this chapter that approach is applied to three current solutions to the hidden-surface problem: (1) scanning algorithms in which the image is computed scan line by scan line; (2) priority algorithms in which the image is generated polygon by polygon in depth-sorted order and (3) the z-buffer algorithm in which the image is generated polygon by polygon without regard to order.

There has been much discussion in recent years over what to do about the problems of jagged edges and disappearing details in computer-synthesized, shaded images. Chapter Four covers some of the more interesting ideas which have arisen from this discussion. A few partial cures for the problem have been implemented. However, this paper marks the first serious attempt to probe the roots of the problem and offer improved methods based on the fundmental nature of computer-synthesized, shaded images. The filtering algorithm developed in Chapter Two has been applied to the design and implementation of a filtering "tiler" for rendering polygons. The tiler is an important part of two of the three classes of hidden-surface

algorithms discussed in this chapter and therefore its implementation is accorded an entire section toward the end of the chapter. The first parts of the present chapter are devoted to a more general discussion of the application of the filtering technique to the three abovementioned classes of hidden-surface algorithms.

As pointed out in the introductory chapter, current difficulties in computer-synthesized, shaded images occur under reasonably well-defined conditions. It is possible to isolate most of these conditions in the scene description data before the hidden-surface algorithm computation is begun; certain parts of the data can be tagged for special treatment. Tagging the data allows the hidden-surface routine to operate normally over most of the image, applying the more expensive convolution techniques described in Chapter Two only where necessary. The detection method used to decide which data is to be tagged is dependent only on the form of the data. Therefore, most polygonal scene descriptions can be handled by a single basic method. The first part of this chapter discusses methods for tagging the data and the latter part concentrates on the use of the tagged data in hidden-surface algorithms.

TAGGING THE DATA

It has been pointed out that nearly all the difficulties in shaded images appear where abrupt changes in intensity, and thus high spatial

frequencies, occur. If the elements of the scene description data which cause these occurrences can be tagged, then at least the difficulties can be localized. The abrupt intensity changes should occur only under the following conditions: along the silhouette or outline of an object, along creases, corners or other sharp changes in the direction of a surface and at the edges of colored patches on the surface.

If the objects represented are polyhedral in nature then every polygon edge is a potential source of aliasing problems. On the other hand, if curved surfaces are represented by a polygonal mesh, shading techniques may be used to conceal the polygon boundaries over smooth areas [Gouraud 1972, Phong 1973]. There have been a few implementations of hidden-surface algorithms which render curved surfaces directly [MAGI 1968, Mahl 1970, Catmull 1974]. This thesis, however, is primarily concerned with algorithms operating on polygons. The same low-level filtering techniques can be used for any algorithm once the exact image position of high frequency details is known, but finding those positions is a much different problem in algorithms applied directly to curved surfaces.

A curved surface approximated by polygons can be made to look smooth by calculating intensities based on the orientation of the surface at the vertices of the polygonal mesh and then using interpolation to find the intensities for the rest of the surface. The data structure for describing the polygonal mesh is usually arranged so that adjacent polygons can share data where they have common vertices. If there is to be a sharp change of surface orientation or color across a polygon border, there must be either

two sets of vertices defining the edge which joins the two polygons or two sets of attributes for those vertices. It is preferred that there be two sets of vertices since this results in a more consistent structure for the data and thus simpler access algorithms. Therefore, creases and color changes actually define two different edges over the same position. This characteristic can be used to isolate such edges. Since the most

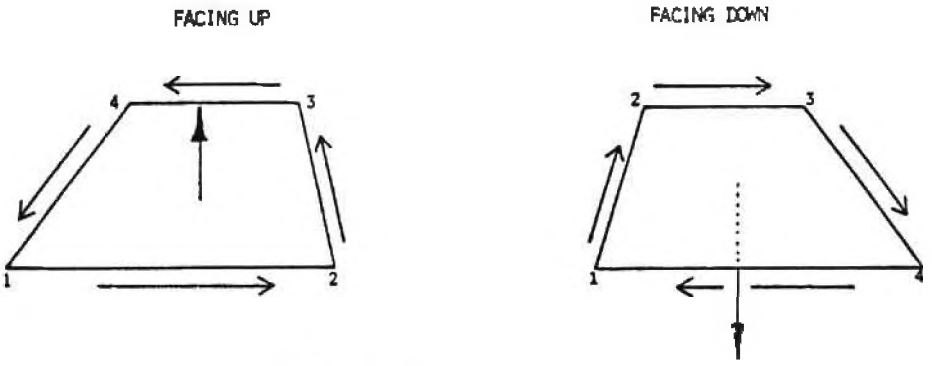


FIGURE 3.1 POLYGON ORIENTATION

noticeable staircase effects occur on the silhouettes of objects, it is clearly necessary to find those edges which lie on the silhouettes. Any edge which lies on a silhouette must join a polygon which faces the viewer to a polygon which faces away from the viewer (the vertex order, which must be consistent, determines which way a polygon faces [Fig. 3.1]). Of course an edge associated with only one polygon may also lie on the silhouette of an object. In this case the edge must lie on a surface edge as opposed to a polygon edge. A surface edge occurs wherever the surface halts, for example, at the edge of a sheet of paper, or a hole in the surface [Fig. 3.2].

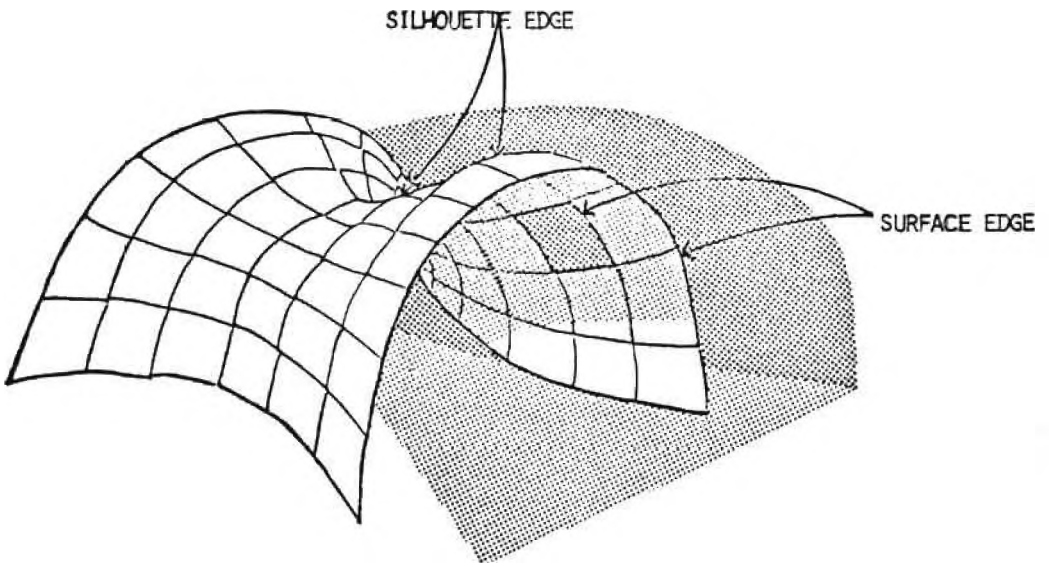


FIGURE 3.2 SILHOUETTE EDGES AND SURFACE EDGES

To save space, polygons facing away from the viewer are often discarded before the hidden-surface computations are done, in which case, silhouette edges become surface edges (and therefore belong to only one polygon). This step cannot be taken, however, until all the vertex coordinates have been transformed to the perspective space from which the image is derived. Thus pointers to these discarded polygons must be nullified immediately prior to the start of image generation. Creases, color changes and silhouette edges are therefore characteristically composed of edges belonging to only one polygon. This characteristic can be used to find and tag all such edges.

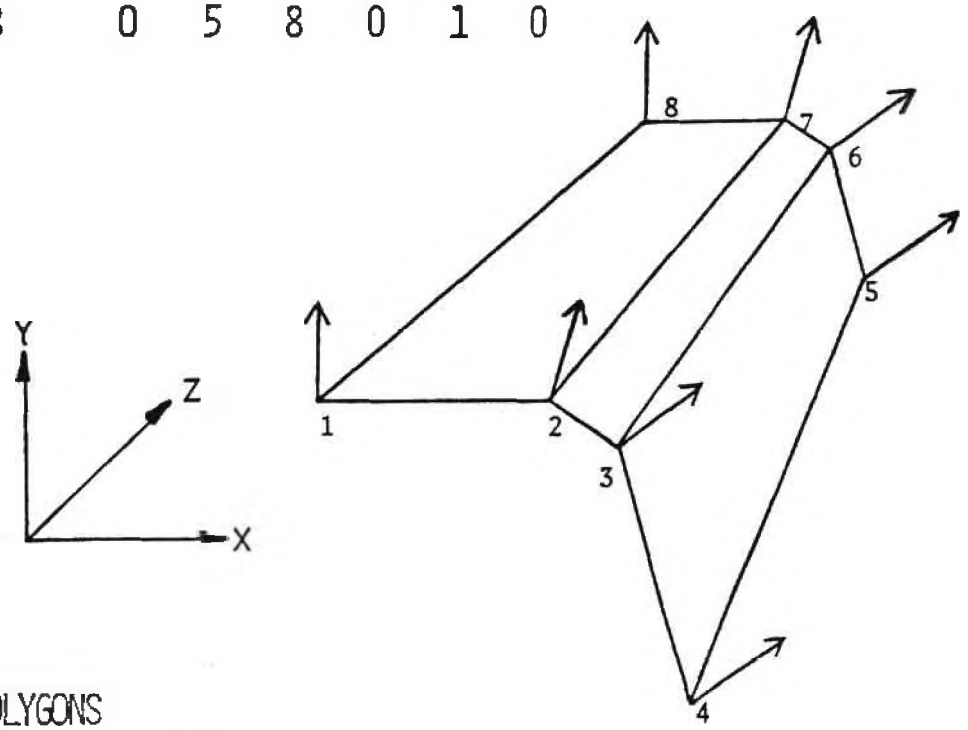
Although an exhaustive search could be used to find all the edges associated with only one polygon, a far more attractive alternative is to add an adjacent polygons list to the data for each object, providing a pointer to the adjacent polygon for each polygon edge. All such

neighboring polygons are then readily accessible. With this arrangement, a null pointer immediately indicates an edge associated with only one polygon. Without the adjacent polygons list, tagging edges adjoining polygons facing away from the viewer becomes a difficult task. With the list, a graph in which all adjacent nodes are bidirectionally linked is provided and thus tagging silhouette edges by nullifying appropriate adjacent polygon pointers is straightforward.

The following three-part polygonal data structure is used throughout the remainder of this thesis: (1) A list of vertices, generally with six numbers per vertex. The six numbers serve to identify the position of the vertex and the direction of the surface normal at that position. (2) A list describing polygons made from the listed vertices. Thus each polygon description consists of a short list of pointers to the vertices which make up that polygon. The vertices are arranged in the polygon list so that, viewed from the outside of the object, they come in counterclockwise order. (3) A list of the adjoining polygons across each edge of each polygon described in the second list. For an example of the application of this data structure to a simple object, see Figure 3.3.

Having tagged all the sharp edges in the data, it is then necessary to deal with the problem of small objects. It is quite possible to encounter a sharp change in intensity in a situation which is not caught by the above technique. Consider the case of a cube with rounded corners defined by three or four polygons running the length of each edge [Fig. 3.4]. When the image of such a cube is large enough so that each edge polygon spans

NO.	VERTICES X	Y	Z	SURFACE NORMALS X	Y	Z
1	0	5	0	0	1	0
2	5	5	0	2	3	0
3	6	4	0	3	2	0
4	8	0	6	2	1	0
5	8	0	8	2	1	0
6	6	4	8	3	2	0
7	5	5	8	2	3	0
8	0	5	8	0	1	0



POLYGONS

NO.	VERTEX LIST			
1	1	2	7	8
2	2	3	6	7
3	3	4	5	6

ADJOINING POLYGONS

NO.	POLYGON LIST			
1	0	2	0	0
2	0	3	0	1
3	0	0	0	2

FIGURE 3.3 THE DATA STRUCTURE APPLIED TO A SIMPLE OBJECT

several dots in the final image, no problems occur, the rounded edges appear rounded. However, when the cube is viewed from a considerable distance, the total span of the polygons may be considerably less than a single dot. In this case, the edge will appear as jagged as it would if the cube were made from the usual six square polygons. In additon to tagging unattached edges, then, it would be wise to tag edges which span only a few dots when measured in the image space.

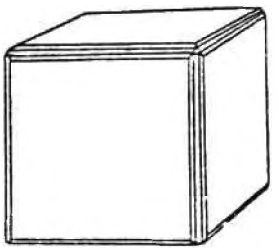


FIGURE 3.4 A CUBE WITH ROUNDED CORNERS

COMPARISON OF THREE BASIC HIDDEN SURFACE TECHNIQUES

In the following discussion, three basic approaches to the solution of the hidden-surface problem are compared in terms of their potential for dealing with the problems outlined above. The three basic approaches are the scanning approach, the z-priority approach and the z-buffer approach.

The scanning approach involves breaking the image into scan lines and then

calculating the intensity for each dot in the line. A major argument for this approach is that the mage is generated in an order suitable for driving a raster-scan display device such as a television monitor and therefore no intermediate storage for the image is needed.

In order to present the image in the proper order, the data must be sorted by their vertical and horizontal positions in the image. The usual sorting order is first vertical, ordering the polygon edges by their uppermost appearance in the image, and then horizontal, ordering those edges which intersect a given scan line by their appearance along the line [Sutherland, Sproul and Schumaker 1974]. The image is then generated, scan line by scan line, treating in turn each segment of a scan line as delimited by successive edge crossings. Thus as each image dot is calculated, all surfaces contributing to that dot may be taken into account.

Since the calculations for a given scan line are done in a single pass, all information necessary for the calculations must be present during that time. Therefore, the disadvantage incurred with this approach is that the size of available primary storage limits the complexity of the scene description. On the other hand, the approach has an advantage in that all the data describing the image along a given scan line is available. Thus the algorithm can benefit from a certain omniscience. All situations can be deciphered to the limits of the resolution given by the number of significant digits in the data. The only limitation lies in the time available to compute the image. In the other two approaches to the hidden-surface problem, only part of the data is available at any given

time.

The z-priority approach paints the image area by area where each area represents some specific piece of the scene description such as a convex area of surface, a polygon or a piece of a polygon. Painting the image area by area requires that some form of intermediate storage be available to hold these areas as the image is being generated. Therefore, z-priority algorithms generally assume access to special purpose hardware for holding the picture. This special purpose hardware often consists of a memory large enough to hold the entire image. In some implementations, the image has been encoded so that it can be made to fit in primary storage [Newell, Newell and Sancha 1972]. However, greater flexibility can be attained with a memory capable of individually storing the intensity of every dot in the image.

As the cost of memory comes down, more and more of these memories, known as frame buffers, are being built. The advantage of the frame buffer is that the data can be destructively overwritten so that no trace of earlier data shows through. By way of contrast, analog picture storage media such as film or cathode ray storage tubes generally sum the effects of successive writes up to the point of saturation so overwritten surfaces cannot hide what is underneath without saturating the medium.

In a z-priority algorithm the image is generated without regard for lateral order, i.e. position in the image. However, in order to effectively use the frame buffer, strict attention must be paid to the depth ordering. All

algorithms in this category establish a priority which determines the order in which parts of the scene description can be written to the frame buffer. Thus in a z-priority algorithm, when a tagged part of the scene description must be blended, the tagged data will always lie in front of the data already written to that part of the frame buffer. This allows blending of the current with any preceding surface to proceed on a well-understood basis.

Certain problems can arise when nothing is known about the underlying surfaces, i.e. those already written to the buffer. For example, consider the problem of representing an edge of a cube [Fig. 3.5]. By the priority rule, the rearmost face must be written first. As that face is written, its four edges must be blended with the background to prevent staircases. When the face on the other side of one of those edges is written, its edge will also have to be blended with the background. However, the background has already been modified by the first edge. In this situation the proper solution is to arrive at a blend confined to the shades of the two faces of the cube. However, because the faces must be written to the frame buffer individually (in the general case), each face in turn must be blended with the background. Therefore the background will show through along the edge of the cube. The resulting image can appear as though the edges have been accented with a pencil line. If the background is dark and a very bright object is lying behind the cube edge, the bright object shows through along a slit defined by the edge.

In a priority algorithm , each part of the scene is written to the frame

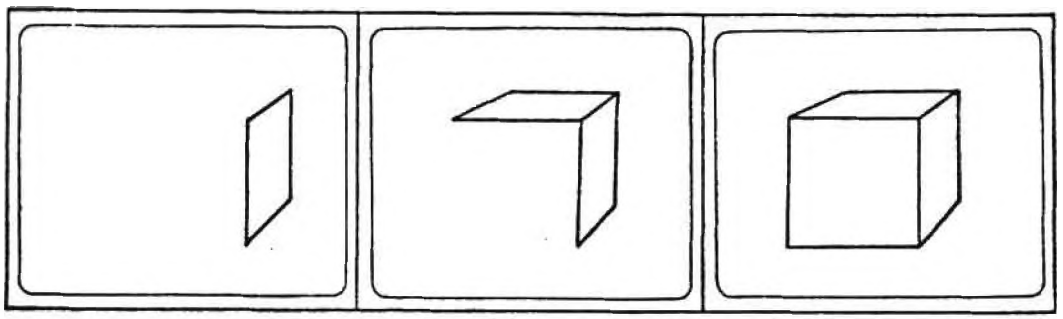


FIGURE 3.5 PROCESSING A CUBE IN A Z-PRIORITY ALGORITHM

buffer without knowledge of the neighboring parts of the scene or underlying surfaces. In addition to the problem of sharp edges on continuous surfaces, such as the cube mentioned above, there are ambiguities wherever two surfaces have tagged edges lying in the same dot in the image. The underlying surface may be completely obscured by the top surface or the two surfaces may not actually overlap [Fig. 3.6]. Obviously, no one blending rule can handle all possible situations; at best, a given blending rule will give incorrect results only a fraction of the time. For further discussion of this problem refer to the section devoted to the implementation of z-priority algorithms.

It is clear then that the lack of information at image-production time, which is inherent in the z-priority approach, can cause problems. The z-buffer approach entails a different set of problems since not even the priority of areas to be written to the buffer is controlled.

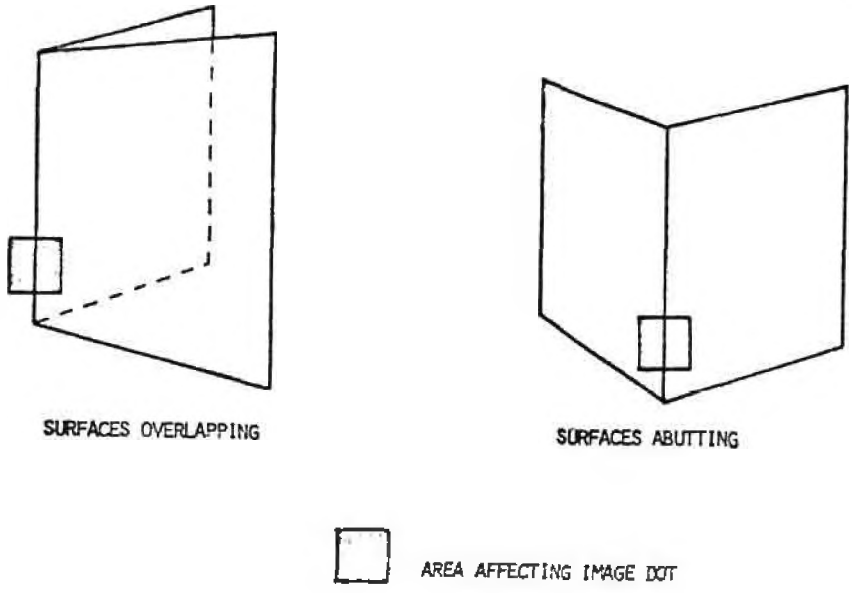


FIGURE 3.6 AMBIGUITIES IN BLENDING

A z-buffer is an even larger piece of memory than a frame buffer. Not only is the intensity stored at every point, but also some measure of the depth to the first surface. Therefore, surfaces can be written to the buffer in any order as follows: A point-by-point comparison is made to determine whether the surface being written lies in front of or behind previously stored data in the same position. If it lies in front, it is written over the current information; if it lies behind, no new information is stored and the information in the buffer is unchanged.

This approach causes many more problems than need be dealt with in the z-priority algorithms. If a surface lies in front of information already in the buffer, then the problems of the priority algorithms can be anticipated. Furthermore, a host of new problems arise when the new

surface lies partially behind what has already been stored [Fig. 3.7]. In this case the blending information for the first surface has been lost and there is thus no way to smooth the resulting edge. Although possible strategies exist for estimating the lost information from the intensities of surrounding dots, these strategies work in only the simplest of situations. Assuming available memory, a preferable solution would be to keep the blending information for the first surface at every picture element, in addition to the intensity and depth. If not enough space is available to store the blending information for every dot, a hashing scheme based on the image dot coordinates can be used to store the information. As long as the image is not highly complex, the space needed to store only the blending information should not be impractically large.

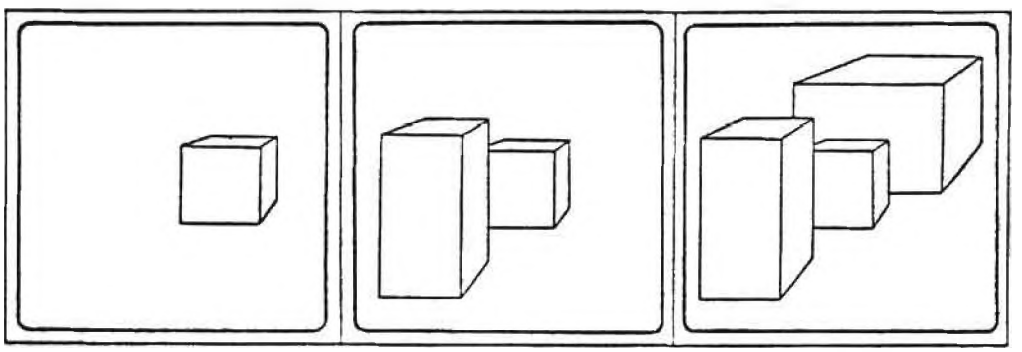


FIGURE 3.7 IMAGE BUILD-UP IN A Z-BUFFER

The three approaches to the hidden-surface problem discussed above exhibit a wide range of information available during calculation of the intensity

of a given dot. In the scanning algorithms, each scan line is calculated only once and therefore all information contributing to each dot is available at the time the dot intensity is calculated. The z-priority algorithms can always access the blending information needed so that this information may be taken into account when edges are written in. In the worst case, the z-buffer loses the necessary blending information and thus techniques must be developed to preserve or recover the information in order to make adequate images.

IMPLEMENTATION OF A SCANNING ALGORITHM

In the previous section on tagging data, a data structure and methods for tagging problem spots in the data were explained. That data structure is assumed in the discussion which follows. It is further assumed that the data have been clipped to the confines of the displayable space and transformed to the image space. Thus all data are ready for the hidden-surface calculations.

The first step in a scanning algorithm is to sort the data into the order it appears as the scan moves from the top to the bottom of the screen. However, in the algorithm described below, an initial sort orders the polygons by depth in order to establish a global order essential for later manipulations.. This algorithm allows only convex polygons. Although this restriction may seem unnecessary in view of the fact that other scanning

algorithms do not make such restrictions, there are cases in which the nonrestrictive algorithms cannot properly handle smooth shading [Fig. 3.8]. Furthermore, non convex polygons generally appear only in hand-generated data, whereas in the future most data will almost certainly be generated by machine-dominated methods. To complete the argument, any nonconvex polygon can easily be converted to a number of convex ones.

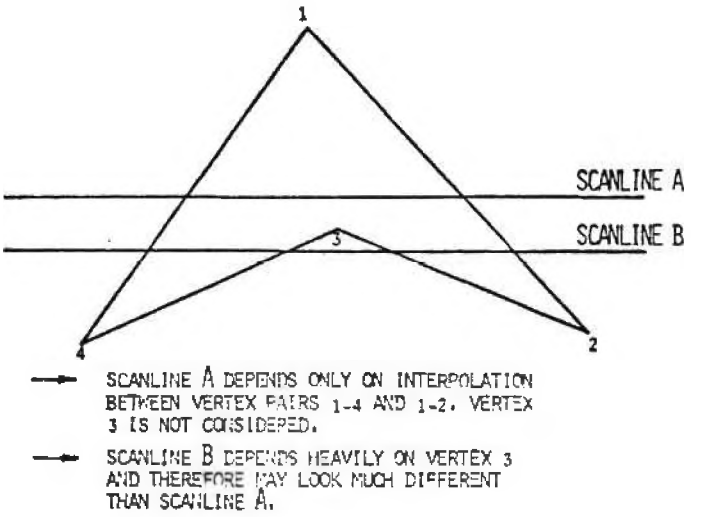


FIGURE 3.8 A SMOOTH SHADING ANOMALY

Convex polygons can intersect a given scan line in only one continuous segment. Therefore polygons are treated as entities throughout the algorithm. All polygons intersecting a given scan line are represented by a block consisting of six or eight numbers representing the x-position, depth and intensity for both the left and right ends of the segment. If transparent surfaces are to be handled, the transmittance of the surface at either end of the segment must also be included (the transmittance is the percentage of incident light which passes through the surface).

For the calculation of a given scan line, the polygon blocks for not only the current line, but also the preceding and succeeding lines are made available. As long as no tagged edges are encountered, the scan segments are written in order to a line buffer which keeps intensity, depth, transmittance and segment number for each dot. If both ends of a segment lie behind opaque parts of the same polygon, the segment is ignored.

When a tagged edge is encountered, affected picture elements are blended by the filtering algorithm described in Chapter Two. To achieve the proper blend, the non-zero part of the filter applied must be superimposed over each picture element in turn [Fig. 2.7]. The filters used in this work always cover an area two picture elements square. That is, if the filter function is centered over a sample point, the bounds of the non-zero portion are defined by the adjacent sample points such that the diagonally adjacent sample points lie at the corners of the square, non-zero area. Thus the polygon block list for the previous scan line determines where a tagged edge enters the top of the square area while the corresponding list for the succeeding scan line determines where the edge exits at the bottom. From this information, the area within the square lying to the right of the edge can be approximated and a blending constant can be computed by the method of Chapter Two. The blending constant is then stored in the line buffer as a transmittance.

In the case of an edge which does not intersect the previous and succeeding scan lines because it is either too small or too close to horizontal, the area to the right is computed based on the endpoints of the edge. The

for the previous and succeeding scan lines are then used to find the position of the implied edge. The intersection of the two segments involved can be computed to high resolution for both the previous and next scan lines. These intersections can then be passed to the filtering procedure as the implied edge's endpoints.

The preceding description illustrates that a scanning algorithm provides a number of important advartages. Since only one scan line (plus two polygon block lists) is needed at any one time, all information concerning a scan line can be kept reacily available without exceeding the limits of available storage. Therefore, since all information is available, all situations can be resolved. The solution presented here is only one approach. A different, more complicated approach can be developed around conventional scanning algorithms which keep an x-sorted list of edges and do all calculations for a given dot at one time. The simpler approach is chosen here for obvious reasons. Where it is necessary to provide more speed than can be obtained in this line-buffer algorithm, the x-sorted algorithms are suggested.

IMPLEMENTATION OF A Z-PRIORITY ALGORITHM

In a z-priority algorithm all polygons must be ordered by depth. This allows the polygons to be written in order to a frame buffer with the assurance that overwriting will always occur in the proper sequence. In

order to establish the priority, problems such as polygons which cyclically obscure each other and polygons which intersect must be solved. Newell, Newell and Sancha [Newell, Newell and Sancha 1972] and Sutherland [Sutherland 1973] offer solutions which involve splitting polygons until all priorities are resolved. Obviously, new edges created by the splitting of intersecting polygons must be tagged for filtering.

Since there will be less information available at the time an edge is to be filtered than in a scanning algorithm, certain precautions must be taken, particularly with regard to abutting edges on the same surface. Using only a frame buffer, no information remains to indicate surfaces previously written to a given dot; the only information available is the intensity. Therefore, other clues must be used to solve the problem of abutting edges. One technique is to modify the function of the adjacent polygons list described under the section on tagging data. Instead of using null pointers to indicate sharp edges and color changes, tagged pointers could be used. Then, the information about adjacent polygons, as well as the tagging, would be preserved.

As each polygon is written to the buffer, a bit can be set indicating that the polygon has been written. Thus when a tagged edge is to be written, it can first be checked to see if the adjacent polygon has been written; if not, all those picture elements which would otherwise be blended with the current polygon intensity are simply overwritten. Thus when the adjacent edge is written, the proper blending of only those two intensities occurs.

Non-abutting edges, such as found on silhouettes cause no problems except when they fall on the same dots as underlying tagged edges. In the case where two tagged edges overlap, there is no correct solution because the frame buffer holds only an intensity which represents a blend of surfaces and there is no way to tell whether the underlying surface is hidden by the overlying surface. Therefore, the best compromise is to treat the edges as though they partially overlap by simply blending the overlying intensity with the intensity already stored. If the two edges cross, the dot representing their intersection may have a slightly incorrect shade but no staircase effects can occur.

Small objects which occupy the same dot are represented somewhat incorrectly. Here again, there is no way to tell whether or not the objects overlap. Therefore, blending is always done as though small objects in the same dot partially overlap. As a result, small sudden changes in intensty can occur when two non-overlapping small objects move across the image together [Fig. 3.9].

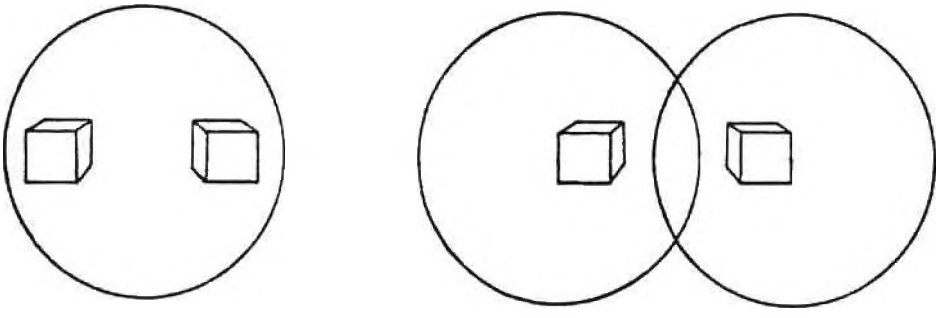


FIGURE 3.9 A DIFFICULTY INVOLVING SMALL OBJECTS

It is clear that lack of information reduces the algorithm's ability to properly handle adjacent small objects and unrelated tagged edges. However, quite acceptable images can be made with z-priority techniques. Only under the most pathological circumstances will errors be noticeable and then only by experienced observers. Since the techniques for individually rendering each polygon are similar in the z-priority and z-buffer methods, they are discussed in the section on tilers which appears later in this chapter.

IMPLEMENTATION OF A Z-BUFFER ALGORITHM

The attraction of a z-buffer lies in the ability to produce the polygons or surfaces in a picture totally independently of each other. It can be relatively easy or difficult to properly render detail in a z-buffer depending upon the organization of the data. Obviously if an entire object is rendered at once the object can be written in the same manner as in the scanning algorithm. Alternatively, the polygons making up the object could be sorted and the object written out as in a z-priority algorithm. In this section, the general case is assumed, i.e. the polygons can be expected to be written in any order. Thus the exercise here is to see how well detail can be handled when no information is available concerning other data in the scene description.

There is no problem blending a tagged edge if that edge is being written

over an underlying surface. In that case, all necessary information is available. However, when an underlying surface must be written under an already written tagged edge, problems arise. In this case, the blending information has been lost. One solution is to store the blending information for all previously written tagged edges. A z-buffer generally provides a large number of bits per image dot. Therefore, the easiest solution is to use some of these bits to store the first-surface blending information. In the event that extra bits are not available, a hashing function can be applied to access the blending information which can be kept in any available main storage. The use of a hashing function assumes that the number of picture elements for which blending information needs to be stored is small relative to the number of dots in the image.

When the visibility of a surface changes while it is being written to the buffer, the first or last dot to obscure the surface can be checked for blending information and then the two surfaces blended. Usually the correct intensity for the front surface can be obtained from an adjacent image dot. Thus the correct blend of the two surfaces is guaranteed. Obviously if there are more than two surfaces being blended in one image dot, the end result cannot be guaranteed correct unless they happen to occur in the proper order. The problem lies in the need to blend the underlying tagged edge with the background. Here again, the shade of the background surface may be determined in many cases from adjacent picture elements. However, there is no deterministic solution, and occasional anomalies can be expected.

In some cases the intensity of the first surface will have been lost. This will occur where the first surface represents a very small or highly detailed object and all involved image dots have already been blended with some underlying surface. One solution is to return to the hash-encoded store and store not only the blending factor but also the intensity of the front surface. This notion can be extended to handle an arbitrary number of surfaces using a list for each blended picture element. Such a method does provide a deterministic approach to blended dots but can cause a severe bottleneck in processing.

Keeping a list for all blended dots offers an expandable capability. By enlarging the list an arbitrarily lengthy set of attributes for the surfaces involved can be kept. Problems involving abutting versus overlapping edges can be resolved by keeping some local relative position information available. It is easy to add to the list of attributes to be kept in such a list. However, both memory space and available processor time can be quickly exhausted by overuse of the concept.

The z-buffer shows no hope of producing properly blended image dots in all situations without considerable help of the kind just outlined. However, the algorithmic simplification and the ability to handle arbitrarily large amounts of data for a single image (since sorting is unnecessary), may make the z-buffer approach attractive enough to justify the effort to construct the necessary extensions.

IMPLEMENTATION OF A TILER

Both z-buffer and z-priority algorithms require a procedure known as a tiler. A tiler translates a part of the scene description, in this case a polygon, into intensities for each sample point it covers. The tiler described here is a simple one, restricted to convex polygons and designed to produce one scan line at a time.

Since polygons are restricted to be convex, each scan line intersects the polygon in a single segment. Therefore, to produce each segment it is sufficient to establish the position and intensity of its end points and pass them to a shader-interpolator routine which generates the intensity For each sample point. Assuming the tiler works top to bottom, the polygon is first searched for its highest vertex. The vertices of the polygon are known to be stored in a particular order (usually clockwise or counter-clockwise). Therefore, if the vertices are stored counterclockwise the vertex previous to the topmost vertex defines an edge which lies to the right of an edge defined by the topmost vertex and the succeeding vertex [Fig. 3.10].

Thus a left edge block can be formed, giving the attributes of the left edge, and similarly for the right edge. The attributes for each block would include present position, increments yielding the position at the next scan line, shading attributes and their associated increments. Each block also includes a count of the number of scan lines remaining until the next vertex is needed. The increments are used to update the edge blocks

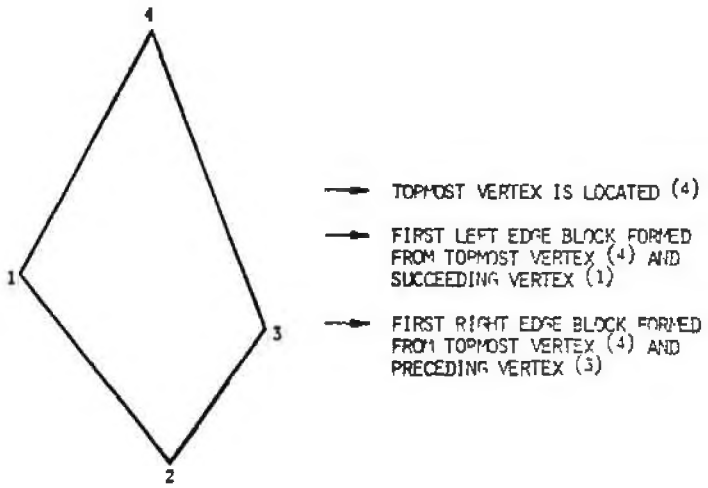


FIGURE 3.10 TILING A CONVEX POLYGON

after each invocation of the shader-interpolator routine. When a vertex is reached, the appropriate edge block must be recalculated to reflect the attributes of the next edge. The algorithm terminates when the next vertex for a block lies above the current one, when both edge blocks reach the same vertex or when the lowest extent of the polygon, determined by the bottommost vertex, is reached.

Figure 3.11 shows a flowchart for the tiler just described and figure 3.12 extends this tiler to include a presampling filter. One important difference between the nonfiltering (Fig. 3.11) and the filtering (Fig. 3.12) tiler is that the filtering tiler does not ignore edge blocks with a vertical range of less than one line. In particular, an edge block which lies along the top or bottom of a polygon and thus may have considerable horizontal extent, must be properly filtered.

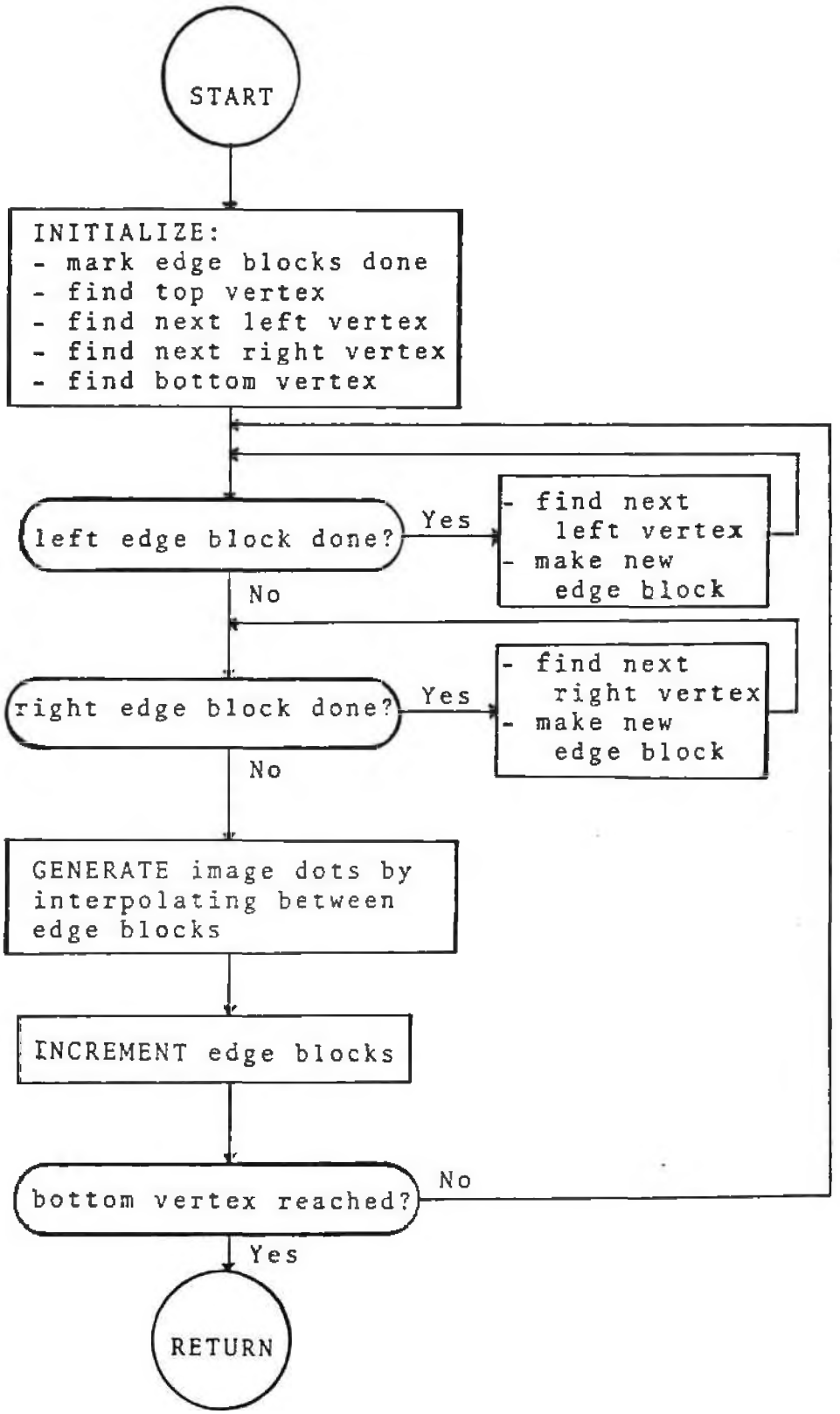


FIGURE 3.11 DIAGRAM FOR A NON-FILTERING TILER FOR CONVEX POLYGONS

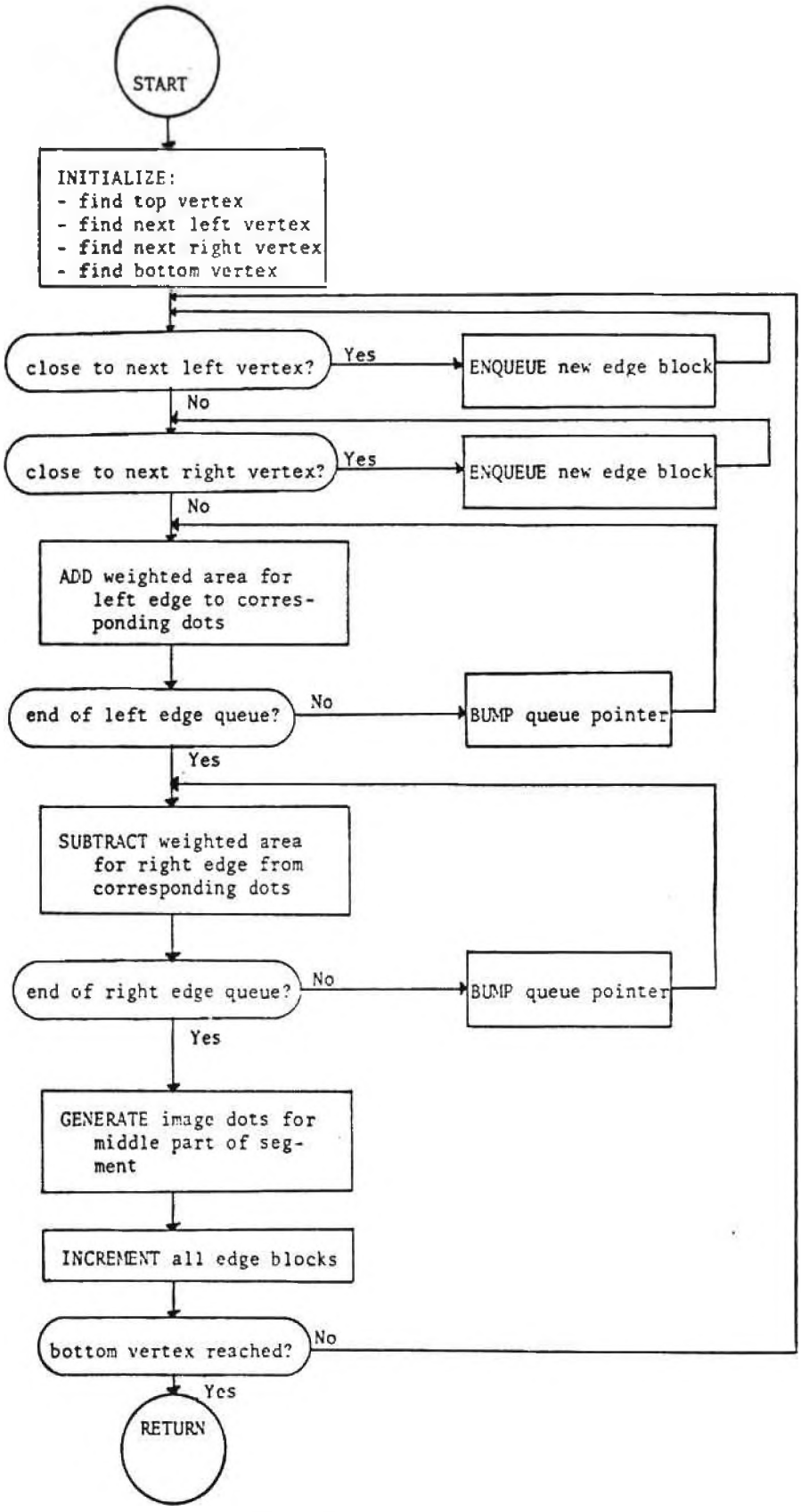


FIGURE 3.12 DIAGRAM FOR A FILTERING TILER FOR CONVEX POLYGONS

A further difference lies in the way the filtering tiler handles edge blocks. In the non-filtering tiler, ignorable edge blocks are immediately marked "done" and a new edge block made. The filtering tiler, on the other hand, must keep track of as many edge blocks as may affect intensities on a given line. Therefore, a queue of edge blocks must be provided for both the left and right sides. In practice, the length of these queues rarely exceeds two edge blocks and images can be made with confidence using queues restricted to that length.

The process of filtering the left and right edges proceeds as described in Chapter Two. The approximate area lying to the right of each left edge in each intersecting filtered area is weighted and then added to the intensity for the corresponding image dot. The contributions of right edges, on the other hand, are subtracted from the intensity for the affected image dots. Therefore, since the final sum is the weighted area covered by the polygon, very small polygons are properly handled (Fig. 3.13). Figures 3.14 through 3.17 compare the filtering tiler with conventional tilers.

Properly evaluating the differing, weighted areas resulting from possible relationships among edges and the area covered by a single position of the filter involves handling a myriad of special cases. Considerable care must be taken to implement the algorithm such that all these cases are handled as naturally as possible or the program quickly grows unmanageable in size and complexity. The fact that such extreme care is required to implement the above method for calculating the weighted area suggests that a simpler algorithm must be developed, if possible.

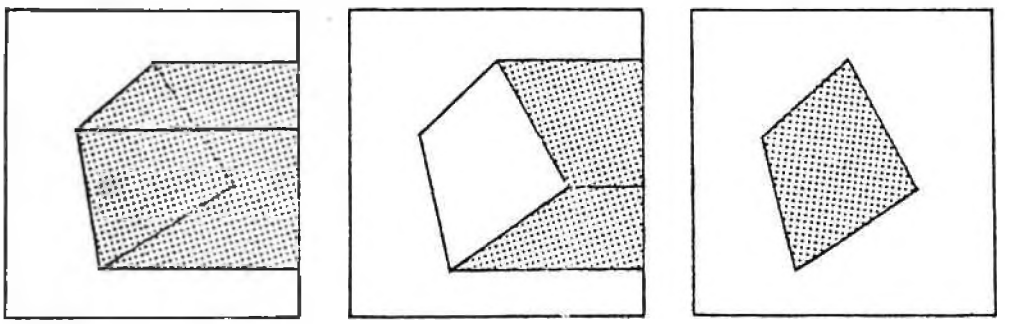


Figure 3.13 Left Areas - Right Areas = Area

Similarly, since the proliferation of frame buffers will tend to increase the importance of hidden-surface algorithms which employ tilers, further research into simpler and more efficient tilers is clearly necessary.

SUMMARY

The purpose of this chapter has been to describe methods for prefiltering the scene description applied to currently important classes of hidden-surface algorithms. A description containing the basic ideas involved in deciding which parts of the data to filter and how the filtered data can be blended in the final image has been provided.

The ideas presented in this chapter should serve as a guide for those

already immersed in their own implementation of a hidden-surface algorithm and of course for those with only an academic interest in the subject. Much remains to be done in this area; computer-generated images have only recently attained a level of realism which makes such attention to detail worthwhile. However, if the computer-generated image is to reach its potential in visual simulation, computer-aided design and even entertainment, the problems of fine detail must be resolved.

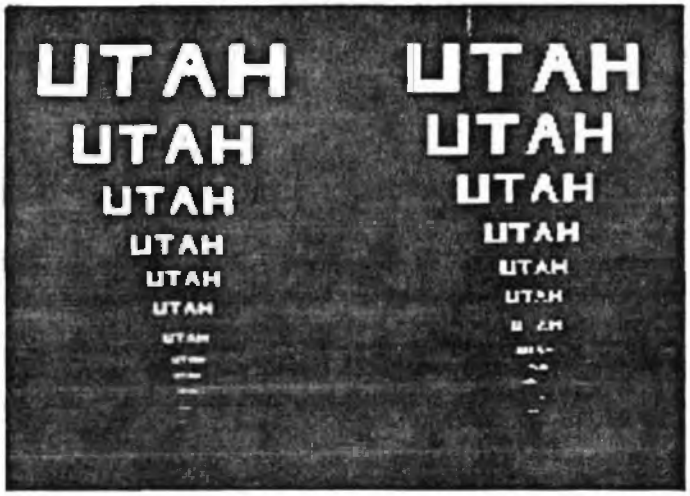


Figure 3.14 A 256 by 256 dot image comparing the filtering tiler with a conventional tiler.

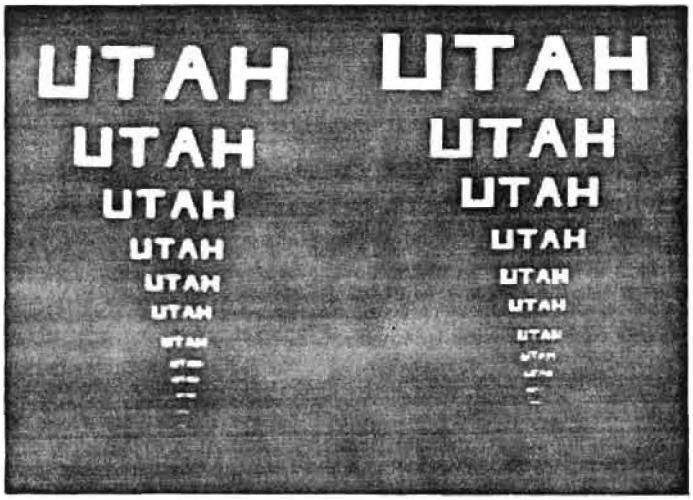


Figure 3.15 The filtering tiler compared with a conventional tiler working at doubled resolution. Four dots have been averaged to give a single dot.

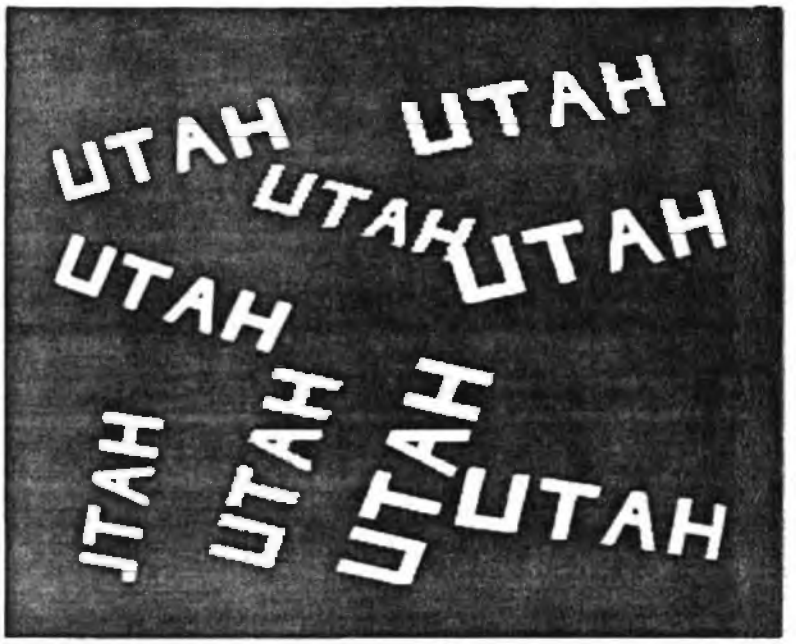


Figure 3.16 Images from the filtering tiler, the conventional tiler and the doubled-resolution tiler displayed together.

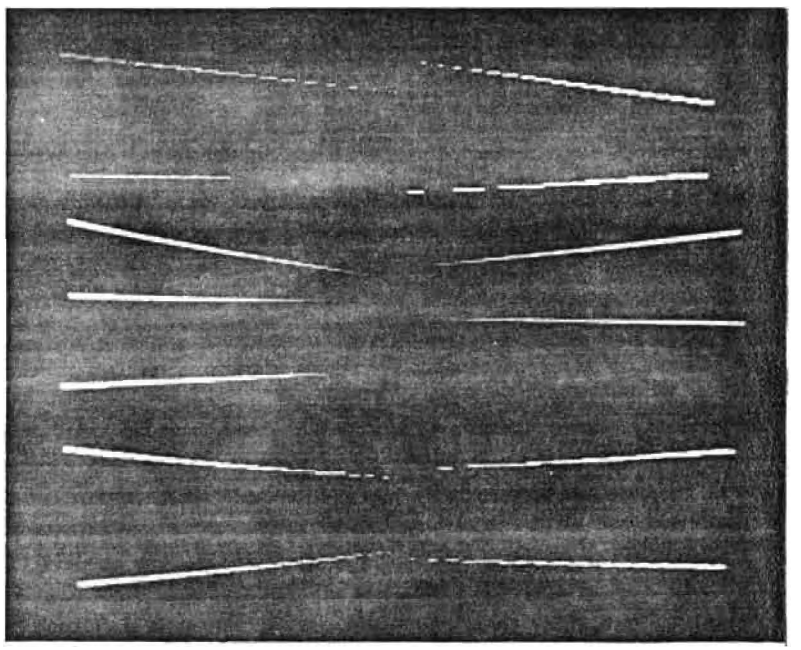


Figure 3.17 The three tilers compared on a particularly difficult type of object, very slender, nearly horizontal triangles.

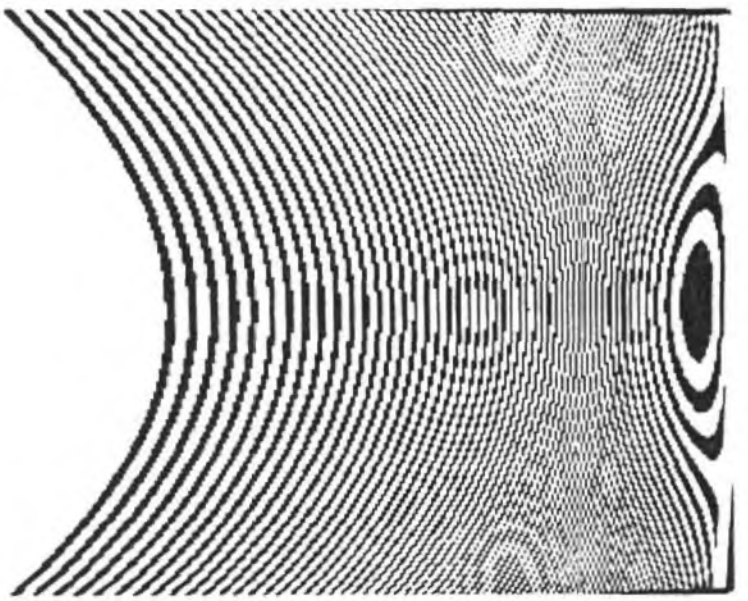


Figure 3.18 The non-filtered test pattern.

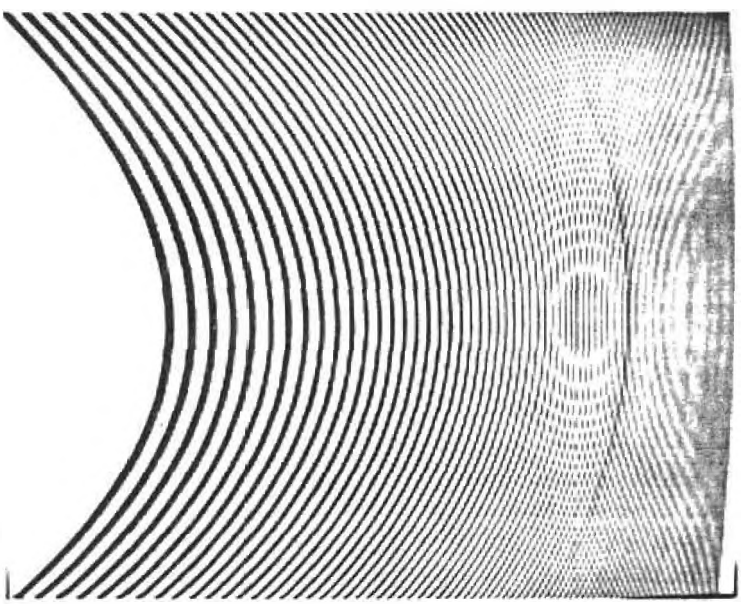


Figure 3.19 The filtered test pattern.

CHAPTER FOUR

OTHER APPROACHES

INTRODUCTION

This chapter explores several ideas all of which promise significant improvement in picture quality, but none of which offers a solution as comprehensive as that already presented.

The following ideas are covered: a temporal scheme which exploits the two-tenths of a second image retention time in the eye, a variable resolution algorithm for making images on a high-resolution medium, several schemes for processing the image after it has been generated, a pair of techniques dependent on analog equipment and finally, a slope-dependent scheme for blending near-horizontal and near-vertical edges. Only the last scheme has actually been implemented.

A TEMPORAL SCHEME

People are able to enjoy watching 8 mm home movies in spite of the fact that the film has only minimal resolution. This is partly because the eye

is able to integrate the image over several frames. Thus the random behavior of the film grain tends to be attenuated while the image, which is relatively constant, is reinforced. This can be considered an improvement in the signal-to-noise ratio clarifying the image and reducing the effect of random noise due to the grain in the film.

Computer-generated images have been restricted to display on a regular grid of dots. Therefore, in an animated sequence, the background noise is not random, but constant and staircases on edges are as constant as the image. The staircases are caused by the dot positions interacting with the edge position; when the dot positions are constant and the edge position slowly varying, the staircases are quite apparent.

It seems clear that a better looking animated image could be obtained by varying the position of the dots from frame to frame. Since the eye integrates over approximately six frames, a cyclic pattern following a random path about the points of a hexagon is appropriate. The hexagon would have to be about as large as the distance between two dots [Fig. 4.1]. As each node of the hexagon is visited, the data in the scene description is shifted to compensate. Therefore, the image always stays in the same place on the screen while the dot positions change.

When the dots change position relative to the data in each frame, the staircases along edges must also change position with each frame. The eye will integrate over several positions of the staircases significantly reducing the staircase effect. On the other hand, the eye is fully capable

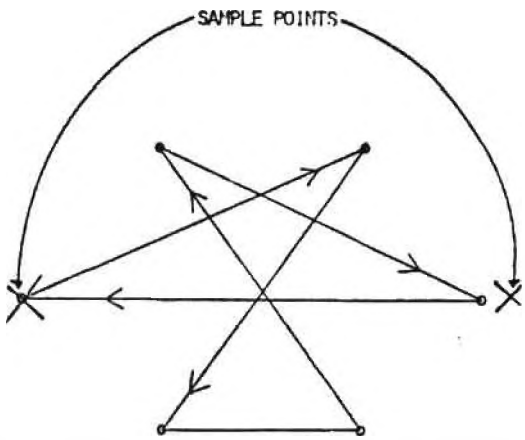


FIGURE 4.1 POSSIBLE PATTERN FOR AN IMAGE DOT IN SUCCESSIVE FRAMES

of detecting a flicker rate which would result from the 24-30 frame a second speed of the media. Thus, if a small object appears and disappears as the dots complete a cycle of positions, flickering in the object will still be noticeable.

In order to implement such a scheme, it is necessary to have control over the dot position. If the display device is a cathode ray tube driven by digital-to-analog converters (DACs) which take more bits than needed for the resolution expected (the usual case), then it is sufficient to set the least significant bits sent to the DACs for each frame. However, if the less significant bits of the DACs cannot be set or the display device is truly digital, for example a plasma panel or diode array, moving the dot position can only be achieved by moving the entire display. Assuming that the images are being photographed from the display onto motion picture film, it would certainly be possible to devise a mechanical system to

re-aim the camera between exposures. However, this would only be practical in a production environment where the high cost of custom-built equipment can be amortized over a long time. Under the proper circumstances, this temporal scheme for improving image quality in animated sequences clearly has merit. The chief advantage of this scheme is that virtually no additional computation is required. The chief disadvantage is that small objects will still disappear between dots and may sometimes flicker noticeably.

A VARIABLE RESOLUTION ALGORITHM

The major argument against increasing resolution to better represent fine details is the increased computation involved. If a high resolution display is available, it is worth trying a variable resolution hidden-surface algorithm. Instead of trying to calculate an intensity for each dot in the high-resolution image, an intensity is calculated only for every 4, 9 or 16 dots. Therefore, only a small fraction of the computation is needed to produce most of the image. For areas of fine detail, the dot groups can be treated at full resolution. A brief description of the application of the variable resolution notion to a scanning algorithm such as that described in Chapter Three follows.

For the purposes of this example, a display with a 2048 by 2048 dot grid and an algorithm running at a 512 dot-per-line resolution are assumed.

Thus, 11 bits are necessary to fully address the display. This means that all scene description data, once transformed to the screen coordinate system, should be kept to a precision of 11 bits. Normally, only the most significant 9 bits will be used. However, whenever an edge marked for special attention is to be processed or the frequency of edge crossings exceeds a certain level, indicating a detailed area, the full 11 bit precision will be used. Four scan lines will be produced simultaneously. Depending on the characteristics of the display, the four lines may be stored until completed or output in multiples of 16 dots.

There are two possible approaches for handling the areas which are to be computed to the full 11-bit resolution. Since the algorithm for producing an area of the picture at the lower resolution can be identical to that for a small, high-resolution section, a recursive routine for generating the image would be the simplest and most elegant approach. The other approach would be to write a special routine which handles only 4-dot by 4-dot images. This routine, which would be called whenever the higher resolution is needed, can be simpler and therefore potentially faster.

It is interesting to note that if no curved surfaces are involved, i.e. only truly polyhedral objects are to be shown, an extension of the first approach would be similar to Warnock's algorithm. Thus, if the picture contains edges, the algorithm does a recursive call for each of four subpictures. If any subpicture contains an edge, its subpictures are processed and the chain of calls may ultimately wind down to a single high-resolution dot. The variable resolution algorithm can be thought of

as a combination of the scanning approach and Warnock's image-space division approach. A two-evel implementation has been suggested above, but there is no reason to exclude a 3 or 4 level implementation. The number of useful levels depends on the average complexity of the image. Obviously, if no part of the image is simple enough to be generated directly at the lowest-resolution level, the algorithm must start at a higher resolution and fewer levels will be necessary.

Due to the relative scarcity of high-resolution displays, the fact that there are apparently no implementations of this or any similar scheme is not surprising. However, when high-resolution displays become relatively common, this technique shows promise for reducing the computation required for such displays, especially for images of limited visual complexity.

PROCESSING THE COMPLETED IMAGE

Several suggestions have been offered for improving the image after it has been generated by a hidden-surface algorithm. These schemes can reduce the staircase effect along edges and thus produce improved still pictures. However, they cannot restore details which have been lost or distorted by the hidden-surface algorithm.

The simplest of these schemes was suggested by T. L. Sancha of the Computer-Aided Design Centre at Cambridge, England and is based on the fact

that staircase effects are most noticeable on high-contrast edges. Scanning the image, neighboring image dots are compared; neighboring dots are those which are either vertically or horizontally adjacent. Whenever the intensity of neighboring dots differs by more than a preset amount, one of them is changed to a compromise intensity. Of course spatial consistency must be preserved. It must always be the upper-left dot rather than the brighter or dimmer dot that is changed. This scheme provides a way of selectively blurring the image which may yield more pleasing results than optical blurring.

A similar scheme involves using a very simple recursive filter. The intensity at each dot is computed as a weighted average of the intensities of the dot itself, the dot to the left and the dot above. A brief experiment with this scheme proved unencouraging. Although the results can not be significantly improved over those obtained by blurring, the filtering scheme might provide more precise control than optical blurring.

More complicated schemes for processing the final image have been suggested and investigated by Lance Williams at the University of Utah. These schemes grow out of work by H. Freeman [Freeman 1974] and others in image brightness contouring. The general idea is to follow intensity discontinuities around the image, blending dots along those discontinuities using a method which smooths the path of the edge causing the discontinuity.

One such scheme would smooth an edge by searching for places where the edge

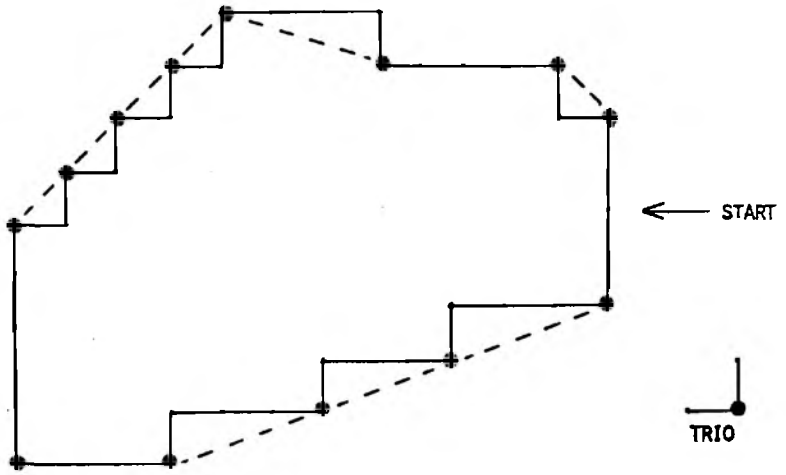


FIGURE 4.2 EDGE SMOOTHING BY TRIO DETECTION

moves from one dot row to the next. Wherever this occurs three edge dots taken in order form a right angled path. Tracing along the edge, the middle dot of each trio is noted. Subsequent trios are formed by starting at the last dot in the current trio [Fig. 4.2].

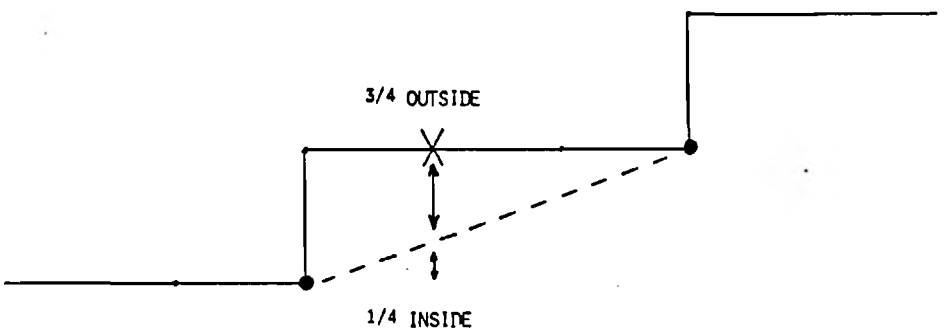


FIGURE 4.3 THE IMAGE DOT AT X RECEIVES AN INTENSITY 3/4 THAT OF THE OUTSIDE AND 1/4 THAT OF THE INTENSITY INSIDE THE POLYGON

When two trios have been found, a line is constructed between their mid dots. The position of this line as it passes between two dots is used to determine the shade of the dot which lies on the intensity discontinuity [Fig. 4.3]. This scheme may be somewhat more expensive but promises to completely eliminate staircase effects.

ANALOG TECHNIQUES

Two further approaches exploit analog hardware. The first, suggested by Yoshio Suzuki of the Japan Broadcasting Corp. (Nippon Hoso Kyokai), is a scan-oriented approach; the second, suggested by Martin Newell of the University of Utah, is a post-processing idea which originated at the Computer-Aided Design Centre in Cambridge, England.

Suzuki's suggestion depends on a low-pass filter acting on parts of a scan line. A low pass filter tends to smooth abrupt changes in a signal i.e. a sharp change in intensity along a scan line will be smoothed; the lower the cut-off frequency of the filter, the wider the smoothed area. Thus, if a variable low-pass filter is controlled by the slope of an edge, the proper spread can theoretically be achieved so that the edge will appear smooth when the scan lines are completed. Suzuki apparently intended to apply the idea to a real-time display. Running such a scheme at video rates sounds difficult; if there are several edges on a scan line, the cutoff frequency has to be changed several times during a single scan line, i.e. the cutoff

frequency must be changed in nanoseconds. Although this scheme stands little chance of being realized and even less chance of working well, it is an interesting one.

Newell's suggestion involves using the analog vector generation hardware in a calligraphic display. If a calligraphic display is being used to scan out a raster format image, it could also be used to draw calligraphically those edges responsible for staircase effects. Analog vector generators can make smooth lines of any desired slope and since the information necessary to draw the offending edges is available in the scene description data, the edges could be redrawn using the vector generation hardware. Thus the image is first written to the display in a standard raster format leaving jagged edges in the usual places. Then, polygonal edges along silhouettes and other features which are likely to appear jagged are redrawn by standard analog vector generation techniques. That is, a line is drawn directly between the two ends of the polygon edge with no regard for the raster.

By this method, staircase edges would be at least partially obscured by an overdrawn line. It would probably be difficult to implement this technique without significant problems involving object outlines. If all outlines are traced twice by the electron beam they will appear brighter than the rest of the image. However, it may be possible to compensate by leaving such edges darker during the raster pass. There is also the possibility of using a display which overwrites instead of summing the signals, or of using a signal mixing device such as the chroma-key used in television

work. These solutions, however, involve use of expensive equipment.

THE EVANS AND SUTHERLAND SHADER

The Evans and Sutherland Computer Corp. has produced a small number of real-time shaded picture systems [Evans and Sutherland Corp. 1974] which incorporate a clever scheme for preventing jagged edges. The scheme involves the use of two shaders, a shader being a device which produces individual dots for display. The information passed to the shader is sufficient for displaying one scan segment; an intensity and position for each end of the segment.

Nearly horizontal and nearly vertical edges are produced by two shaders working simultaneously; one shader for each of the two possible intensities. For example, an edge which moves ten dots to the right for every scan line will be treated as follows: Assume the edge separates a dark grey area at the left from a light grey area at the right. Each shader is responsible for producing a ten-dot segment. While the left shader increments from dark grey to black, the right shader increments from black to light grey. Summing the output, dot by dot, from the two shaders produces the correct blend of the two shades for each dot. The implementation of this scheme has produced pictures with admirably smooth edges. However, there remains considerable room for improvement.

SUMMARY

This chapter has surveyed a few of the many ideas for "clever" solutions to the problem of detail in shaded, computer-generated images. The difficulty with all of these schemes is that they address symptoms rather than the cause of the problem. Without a technique for guaranteeing that details are neither overlooked nor grossly misrepresented, computer-generated, shaded images will not achieve their potential for realism.

APPENDIX

GREY SCALE COMPENSATION

All photographs in this paper were taken directly from the face of a standard television monitor driven by an Evans and Sutherland Video Frame Buffer. The Frame Buffer incorporates sufficient memory to store 512 by 512 8-bit bytes and circuitry to scan the memory 30 times a second, representing each byte as a dot on the display (in this case a 19" Conrac television monitor). A section of hardware, known as the color map, converts each byte to a 12 bit value which drives the digital-to-analog converters supplying voltage to the electron beam and thereby determining the intensity of the corresponding dot on the television monitor. The 12 bit values are kept in a table which can be updated as necessary. Thus the color map can be used to control the grey level response to the range of values expressible in the frame buffer.

An interactive system has been designed and implemented on an Evans and Sutherland Picture System used in tandem with the Video Frame Buffer. The display as seen by the user is reproduced in figures 2.11 through 2.14. The curve at the upper left of these figures represents the color map. The horizontal dimension is the input byte's value and the vertical dimension is the value of the 12 bit output. Using this system, the color map can be manipulated in real time. The curve displayed on the screen is generated using a spline approximation to six control points. The control points may be moved using a data tablet attached to the Picture System.

To calibrate the system, the test pattern was displayed using a pyramid-shaped presampling filter (as seen in figure 2.14). The color map was then adjusted until the strength of the moire fringes was minimized (figures A.1 through A.3), that is, the system is calibrated by subjective feedback.

To obtain an objective measure for the compensation process, a linear grey scale (linear input to the color map) was displayed next to the test pattern and intensity readings were taken using a Tektronix J-16 digital photometer. When the test pattern showed least evidence of moire fringes, the intensity readings lay near a parabolic curve, i.e. the function given by the square roots of the intensity readings was close to linear. This result held true under a variety of conditions. At display intensities appropriate for viewing at normal indoor light levels, the color map curve was almost linear, indicating that the display was obeying the expected square law at normal viewing intensities. At very low intensities such as used for figures 2.11 through 2.14 the curve became convex upward (The television monitor intensity had to be lowered in order to photograph it alongside the vector display of the Picture System). Intensity readings taken at this low level indicated the same parabolic curve. Distorting the color map to compress the dynamic range of the image yielded similar results.

Figures A.1 through A.3 show the appearance of the test pattern and grey scale at normal viewing intensities with varying color map curves. The corresponding grey scale intensity readings are shown in figure A.4 along

with the square roots of those readings. The dynamic range of the photographs is not as great as that of the television monitor and the film adds an additional non-linear process between the ideal image and the viewer. Therefore, the printed images are somewhat degraded.

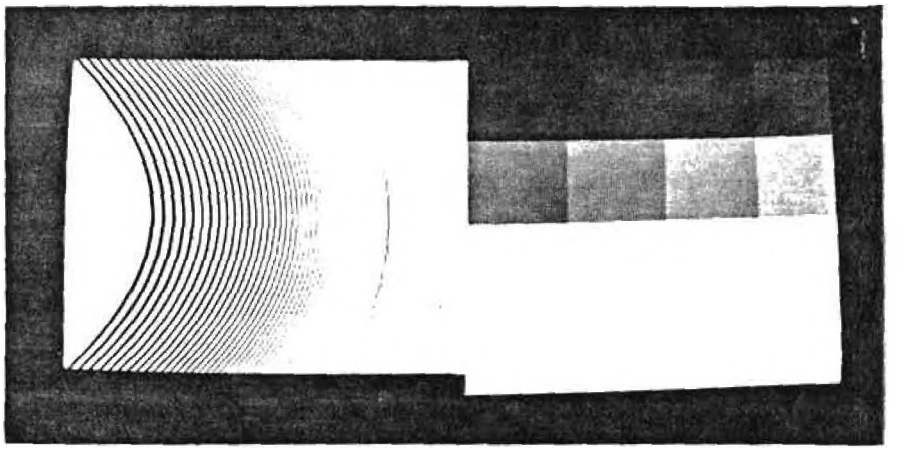


Figure A.1 Test pattern and grey scale with linear color map.

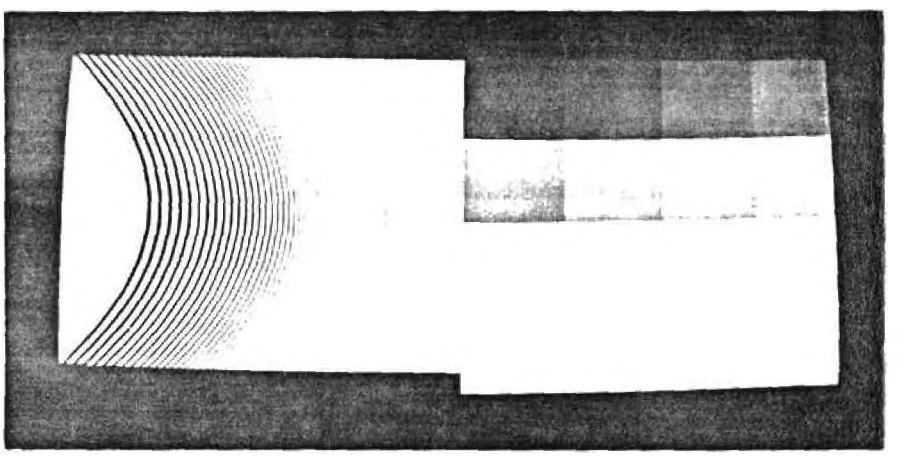


Figure A.2 Test pattern and grey scale with convex upward color map.

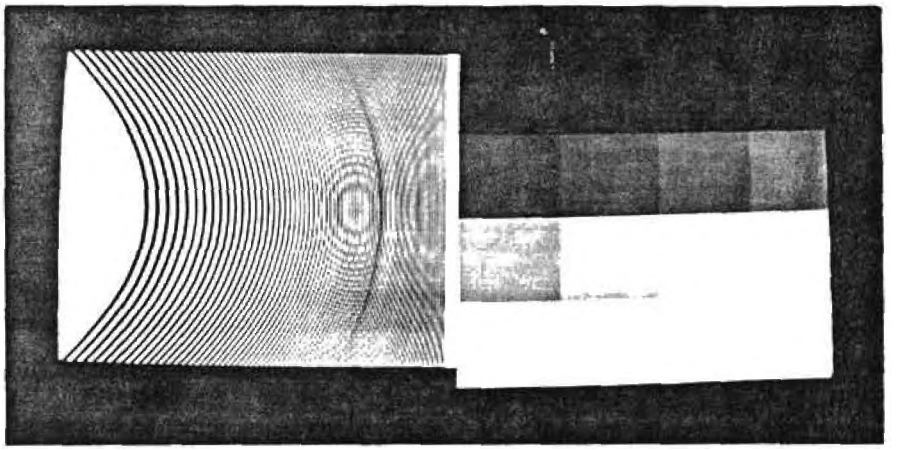


Figure A.3 Test pattern and grey scale with concave upward color map.

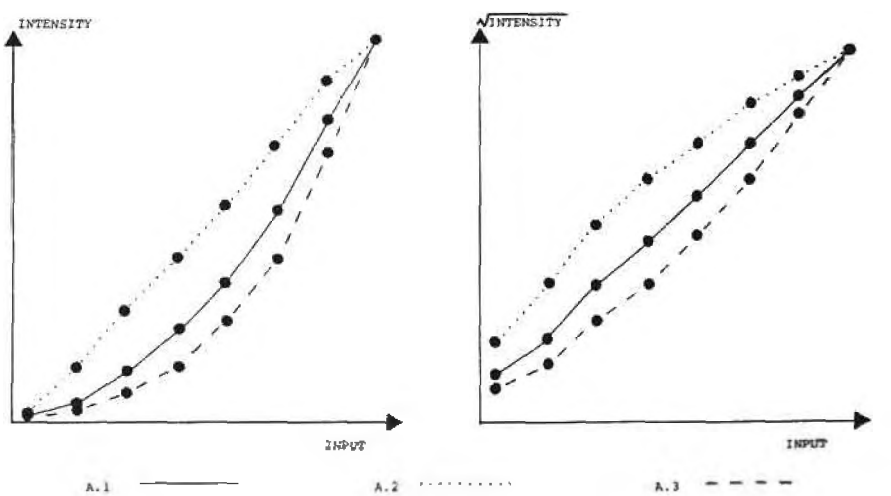


Figure A.4 Intensity readings for figures A.1 through A.3.

REFERENCES

Bui Tuong Phong, "Illumination for Computer Generated Images", Department of Computer Science, University of Utah, UTEC-CSc-73-129, July 1973. Abridged in Communications of the ACM, Volume 18, No. 6, June 1975.

Catmull, E., "A Subdivision Algorithm for Computer Display of Curved Surfaces", Department of Computer Science, University of Utah, UTEC-CSc-74-133, December 1974.

Cornsweet, T.N., Visual Perception, Academic Press, 1970.

Evans and Sutherland Computer Corporation, "The Shaded Picture System", Technical Brochure, Evans and Sutherland Computer Corporation, 1974.

Freeman, H., "Computer Processing of Line Drawing Images", ACM Computing Surveys, Volume 6, No. 1, March 1974.

Gouraud, H., "Computer Display of Curved Surfaces", Department of Computer Science, University of Utah, UTEC-CSc-71-113, June 1971. Abridged in IEEE Transactions on Computers, Volume TC-20, June 1971.

Graham, D.N., "Image Transmission by Two-Dimensional Contour Coding", Proceedings of the IEEE, Volume 55, No. 3, March 1967.

Guilleman, E.A., Theory of Linear Physical Systems, John Wiley and Sons, 1963.

Hunt, R.W.G., The Reproduction of Colour in Photography, Printing and Television, Fountain Press, 1975, (3rd Edition).

Levinson, J.Z., "Psychophysics and TV", in Picture Bandwidth Compression, Huang and Tretiak, Editors, Gordon and Breach Science Publishers Inc., 1972.

MAGI, Mathematical Applications Group Inc., "3-D Simulated Graphics", Datamation, February 14, 1968.

Mahl, R., "Visible Surface Algorithms for Quadric Patches", Department of Computer Science, University of Utah, UTEC-CSc-70-111, 1970.

Newell, M.G., R.G. Newell and T.L. Sancha, "A Solution to the Hidden-Surface Problem", Proceedings of the ACM, 1972 National Conference.

Oppenheim, A.V., and R.W. Schafer, Digital Signal Processing, Prentice-Hall, Inc., 1975.

Oster, G. and Y. Nishijina, "Moire Patterns", Scientific American, Volume 208, No. 5, May 1963.

Rose, A., Vision: Human and Electronic, Plenum Press, 1973.

Shoup, R.G., "Some Quantization Effects in Digitally-Generated Pictures", Society for Information Display, Proceedings of the 1973 Convention.

Sutherland, I.E., "Polygon Sorting by Subdivision: A Solution to the Hidden-Surface Problem", Unpublished Manuscript, October 1973.

Sutherland, I.E., R.F. Sproul and R.G. Schumaker, "A Characterization of Ten Hidden-Surface Algorithms", Computing Surveys, Volume 6, No. 1, March 1974.

Zworykin, V.K. and G.A. Morton, Television, John Wiley and Sons, 1954, (2nd Edition).

VITA

Franklin C. Crow

Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712

BIRTH

January 5, 1943 in Hanover, New Hampshire.

SCHOOLING

- Public schools in Madison, Wisconsin.
- BA 1965 University of Wisconsin (Mathematics).
- PhD 1976 University of Utah (Computer Science)
  Thesis: "The Aliasing Problem in Computer-Synthesized Shaded Images".

EXPERIENCE

- 1965-69 University of Wisconsin,
  Project Specialist (Computer Cartography, Information Retrieval).
- 1971-72 Southern Illinois University,
  Research Assistant (Computer Animation),
  Teaching Assistant (FORTRAN).
- 1972-75 University of Utah,
  Teaching Fellow (Operating Systems, Programming Languages, Algorithms and Data Structures),
  Research Assistant (Computer Graphics, Hidden-surface Algorithms, Computer-Aided Design).
- 1975- University of Texas at Austin
  Instructor (Computer Architecture, Computer Graphics)

CURRENT RESEARCH AREAS

- High-Quality Computer-Synthesized Images.
- Hidden-Surface Algorithms.
- Three-Dimensional Computer-Aided Design.
- Reconfigurable Multiprocessor Arrays.

PUBLICATIONS

- Wyllys and Crow; "Computer Support for a Small, Specialized Document Processing System, The ERIC Clearinghouse on Educational Facilities", American Society for Information Science (1969).
- Various computer-generated pictures used as illustrations in journal articles. Most notably, the cover of IEEE Computer, May 1974.
- Bui and Crow; "Improved Rendition of Polygonal Models of Curved Surfaces", Proceedings of the 2nd USA-Japan Computer Conference, August 1975.

PROFESSIONAL SOCIETIES

Association for Computing Machinery, SIGGRAPH, SIGARCH.
IEEE Computer Society.