DOCUMENT RESUME

ED 281 497 IR 012 614

AUTHOR Sachter, Judy E.
TITLE The Basic Concepts of Three-Dimensional Computer Graphics for Artists.
PUB DATE 84
NOTE 102p.; Master's Thesis, Ohio State University.
PUB TYPE Dissertations/Theses - Master Theses (042)

EDRS PRICE MF01/PC05 Plus Postage.
DESCRIPTORS Aesthetic Values; Art Education; Color; *Computer Graphics; Light; *Three Dimensional Aids; *Visual Arts
IDENTIFIERS Cartesian Coordinates; Polygons

ABSTRACT

Arguing that an artist using any new medium must understand its techniques and limitations, this master's thesis is intended to demystify state-of-the-art 3-D computer graphics through a discussion of the technical literature from the artist's point of view and an analysis of the curricular needs of the artist. The first of six chapters discusses art, art education, and computer graphics; learning to use and make new tools; and aesthetic and artistic issues. Modeling the world in the computer is the focus of the second chapter, which provides an overview of 3-D computer graphics as well as discussions of the time and space and computational costs; visualizing the world; the Cartesian coordinate system; and the creation of data. The third chapter discusses describing a changing world, including object transformations, scaling, translation, rotation, object or scene descriptions, and computer animation. Viewing a model of the world is discussed in the fourth chapter, including the illusion of depth, view transformations, and hidden-surface removal. Focusing on 3-D realism, the fifth chapter describes light, shading, and color models, as well as surface and image qualities. Possibilities for the artist in 3-D computer graphics are summarized in the concluding chapter. Sixty-one figures illustrate the text, and both a glossary and a 59-item bibliography are included. (MES)

Abstract

THE OHIO STATE UNIVERSITY
GRADUATE SCHOOL

NAME: Judy E. Sachter QUARTER/YEAR: Spring 1984

DEPARTMENT: Art Education DEGREE: Master of Arts

TITLE: The Basic Concepts of Three-Dimensional Computer Graphics For Artists

The basic principles of 3-dimensional computer graphics are presented from the artist's point of view. Modeling the world in the computer with polygonal data in the context of the Cartesian coordinate system is explained and illustrated. The manipulation of objects, view parameters, color and light sources is described and illustrated.

Thomas E. Linehan
Advisor

THE BASIC CONCEPTS OF THREE-DIMENSIONAL COMPUTER GRAPHICS FOR ARTISTS

A Thesis

Presented in Partial Fulfillment of the Requirements
for the Degree Master of Arts

by

Judy E. Sachter, B.F.A.

The Ohio State University
1984

Approved by

Thomas E. Linehan
Advisor
Department of Art Education

ACKNOWLEDGEMENT

I want to thank Dr. Thomas Linehan, my advisor, as well as the other members of my committee, Professor Charles Csuri and Dr. Barbara Boyer for their assistance and support.

The illustrations were created at the Computer Graphics Research Group at Ohio State University on a VAX-11/780. The images were displayed on a 640 X 480 X 32 bit frame buffer designed and built by Frank Crow and Mark Howard. The software used was "scn_assmblr raster display software originally designed by Frank Crow. Some of the data was created by hand and others were created using "dg" a data generation system developed by Wayne Carlson. "Inspect" and "check", developed by by Paul MacDougal, were used to manipulate and edit data. "vw", developed by David Zeltzer, was used to transform and combine data. The script interpolation was generated with "front" developed by Jose Garabis and Judy Sachter.

My additional thanks is extended to all of the members of the Computer Graphics Research Group at The Ohio State University, especially Julian Gomez, Paul MacDougal, Marla Schweppe, Dr. Edwin Tripp III, Kathy Simpson, and Kris Conrad.

Most of all I want to thank my husband, David Zeltzer, for his support and patient explanations.

TABLE OF CONTENTS

ACKNOWLEDGEMENT ii
TABLE OF CONTENTS iii
TABLE OF FIGURES iv
CHAPTER 1: INTRODUCTION..... 1
1. ART, ART EDUCATION, AND COMPUTER GRAPHICS..... 1
1.1. LEARING TO USE THE NEW TOOLS..... 2
1.2. BRIDGING TWO DISCIPLINES..... 2
1.3. LEARING TO MAKE NEW TOOLS..... 4
1.4. AESTHETIC AND ARTISTIC ISSUES..... 4
1.5. PROBLEM..... 5
CHAPTER 2: MODELING THE WORLD..... 8
2. OVERVIEW: 3-D COMPUTER GRAPHICS..... 8
2.1. TIME AND SPACE: COMPUTATIONAL COST..... 9
2.2. VISUALIZING THE WORLD..... 11
2.3. THE CARTESIAN COORDINATE SYSTEM..... 11
2.4. DATA..... 12
CHAPTER 3: DESCRIBING A CHANGING WORLD..... 15
3. OBJECT TRANSFORMATIONS..... 30
3.1. SCALING..... 30
3.2. TRANSLATION..... 35
3.3. ROTATION..... 36
3.4. OBJECT OR SCENE DESCRIPTION..... 42
3.5. COMPUTER ANIMATION..... 47
CHAPTER 4: VIEWING THE MODEL..... 50
4. MODELING THE WORLD: THE ILLUSION OF DEPTH..... 50
4.1. VIEW TRANSFORMATIONS..... 51
4.2. HIDDEN-SURFACE REMOVAL..... 55
CHAPTER 5: 3-D REALISM..... 57
5. LIGHT, SHADING AND COLOR MODELS..... 57
5.1. LIGHTING MODELS..... 58
5.2. SHADING MODELS..... 63
5.3. COLOR..... 67
5.4. COLOR MODELS..... 71
5.5. SURFACE QUALITIES..... 76
5.6. IMAGE QUALITY..... 78
CHAPTER 6: SUMMARY..... 83
Appendix A: SELECTED GLOSSARY..... 87
Bibliography 91

TABLE OF FIGURES

Figure 1. Cartesian coordinate system: X axis..... 12
Figure 2. Cartesian coordinate system: X and Y axes. 13
Figure 3. Cartesian coordinate system: X Y Z axes... 13
Figure 4. Coordinate points for a 2-D square..... 14
Figure 5. Coordinate points for a 3-d cube..... 14
Figure 6. Orthographic projection..... 16
Figure 7. Warping points..... 17
Figure 8. Data generation: projection plans..... 18
Figure 9. Data generation: projected lion..... 18
Figure 10. Data generation: solid of revolution..... 19
Figure 11. Data generation: solid of revolution..... 20
Figure 12. Data generation: lofting plans..... 21
Figure 13. Data generation: lofted banana..... 21
Figure 14. Wireframe and coordinate points..... 23
Figure 15. Face description and data set..... 24
Figure 16. Frontfacing and backfacing polygons and and face normal..... 25
Figure 17. Convex and concave polygons..... 26
Figure 18. Clockwise arrows and face normals..... 28
Figure 19. Levels of detail..... 29
Figure 20. "Hut" scaled by 4 0 0..... 31
Figure 21. "Hut" scaled by 0 4 0..... 32
Figure 22. "Hut" scaled by 4 4 4..... 33
Figure 23. Off-origin "hut" scaled by 4 0 0..... 33
Figure 24. Off-origin "hut" scaled by 4 4 4..... 34
Figure 25. Off-origin "hut" scaled by 4 4 4..... 35
Figure 26. Translate "hut" 4 4 0..... 36
Figure 27. Translate "hut" -2 3 2..... 37
Figure 28. Direction of rotations..... 38
Figure 29. X rotations..... 38
Figure 30. Y rotations..... 39
Figure 31. Z rotations..... 39
Figure 32. Off-origin X rotations..... 40
Figure 33. Off-origin Y rotations..... 41
Figure 34. Off-origin Z rotations..... 41
Figure 35. Order of rotations..... 42
Figure 36. Orthographic plans for a train..... 43
Figure 37. XY projection of train..... 44
Figure 38. XZ projection of train..... 44
Figure 39. Scaled part of train..... 45
Figure 40. Rotated part of train..... 45
Figure 41. Translated part of train..... 46
Figure 42. Completed train created by combining primitives..... 47

Figure 43. View parameters..... 52
Figure 44. View pyramid and screen space..... 53
Figure 45. Diffuse light model..... 60
Figure 46. Specular reflection lighting model..... 61
Figure 47. Lighting models: diffuse, ambient, and specular reflection..... 61
Figure 48. Transmittance and thickness..... 63
Figure 49. Faceted shading model..... 64
Figure 50. Gouraud shading model..... 65
Figure 51. Gouraud shading model, points doubled.... 65
Figure 52. Ray tracing 66
Figure 53. Light primaries..... 70
Figure 54. RGB color space..... 71
Figure 55. Color cube..... 72
Figure 56. Scaled color cube: intensity lowered..... 73
Figure 57. Vertex colors..... 75
Figure 58. Texture mapping..... 77
Figure 59. Fractals..... 78
Figure 60. Resolution 640 by 480 blown up 4 times... 80
Figure 61. Resolution 160 by 120 blown up 4 times... 80

CHAPTER 1

INTRODUCTION

1. ART, ART EDUCATION, AND COMPUTER GRAPHICS

In recent years computer technology has advanced in the capability to synthesize visual images. This area of computer science, called computer graphics, is developing commercial applications at a rapid rate. Computer graphics is becoming more sophisticated and at the same time the availability of computer graphics systems is increasing. Researchers in computer graphics are constantly working towards programming techniques for creating, manipulating, and displaying more realistic, three-dimensional (3-D) computer generated images. Currently, computer graphics is used for flight simulation, business, science, computer-aided design, advertising, and entertainment. The commercial uses of computer graphics, (e.g., films such as "Tron", "Star Trek II", and "The Empire Strikes Back",) are making the public more aware of the possibilities of state-of-the-art computer graphics for the visual arts. It is time for more visual artists to become involved in computer graphics.

This new field provides the artistic community with a new expressive medium. As Beverly Jones [33], an art educator, has observed:

> "Artists are using the computer to emulate and facilitate conventional processes. A few are also interested in developing artistic forms which can only be created with the aid of a computer. Some conventional art processes have been radically altered by computers... Artists working in such fine arts areas such as sculpture, printmaking, drawing and painting are experimenting with the computer as an aid to design or execution of their work. The speed with which designs can be created, manipulated, and stored in a computer graphics system contributes to its desirability of use by

artists."

Michael Noll [47] a scientist and artist at Bell Labs, has developed systems to assist the the artist. He believes the computer can be thought of as a creative partner. However, he explains that the overall control and direction of the creative process is the task of the artist. Computer graphics is an active medium free from many of the physical limitations of previous media with which the artist can interact.

1.1. LEARNING TO USE THE NEW TOOLS

An artist using any new medium must understand its techniques and and limitations. The first problem in computer graphics for the artist is to gain an understanding of the basic principles and to develop skill in this medium. The majority of the literature and software developed with the artist in mind addresses two-dimensional (2-D) computer graphics. Two-dimensional computer graphics allows the artist to approach art production in a manner very similar to previous artistic experiences [2,27,51], (e.g., drawing and painting).

In state-of-the-art 3-D computer graphics the artist has to have a higher level of understanding of the computer than in 2-D computer graphics. In 3-D computer graphics the artist must be aware of the numerical representations of objects, color, space and time. Usually the artist has to understand the workings of the computer, software, and peripherals more directly than in 2-D computer graphics.

There is little written for the artist on the underlying concepts of 3-D computer graphics. Most of the instruction in computer graphics is in computer science, engineering, communications, and mechanical or industrial design departments. Presently there are only a few schools that are offering instruction in computer graphics for the purpose of creating art, but most of this instruction is in 2-D computer graphics.

1.2. BRIDGING TWO DISCIPLINES

The artist using computer graphics must be both functional and literate in this new medium. This poses the

second problem for the artist. The artist interested in 3-D computer graphics has to be able to straddle two fields of knowledge: computer science and art. Science and technology have an influence on society as on well as the artist. In embracing this new art form the artist must become familiar with the technology that makes it possible. L. Maholy-Nagy [45], an artist and educator who believed in the integration of art, technology, and science said,

> Sometimes a whole chain of successive influences can be traced from science - to technology - to art - and back again to science ...
>
> this means that the statement, "form follows function", has to be supplanted, that is, form also follows - or at least it should follow, existing scientific technical and artistic development.

It is not an easy task for the non-computer scientist to become literate in computer graphics. Much of the current technical literature is inacessible to the non-specialist [46,11,22].

The literature in the arts relating to computer graphics predominantly consists of interviews with various artists using the medium. It consists of discussions of aesthetic issues or speculation of artistic applications [55,50]. Some of the artist-written literature is out of date due to the speed at which the field is growing and maturing [26,23,38]. This literature is interesting in an historical or artistic sense, but gives little information about the basic concepts used in state-of-the-art, 3-D computer graphics.

There are individuals using computer graphics who have bridged these two disciplines. Some articles for science publications are written by artists, such as Charles Csuri [18,17]. There are some scientists who write for art publications [37]. A few scientists have produced art. Very few artists are trained in both computer science and art. Duane Palyka, at New York Institute of Technology, is trained in both computer graphics and art [52,50,51]. There are also several examples of collaborations between artists and scientists [35,10,12].

The literature in art education tends to deal with possible ways for art educators to use computers, such as computer aided instruction (CAI), statistics, data retrieval, classroom management, and research in aesthetics and perception. Beverly Jones [32,33,34] has examined the ways that computer graphics can assist the art educator in their research and how computers can be used in teaching the arts. Other art educators, Ettinger and Rayala [21], White [59], Hubbard [28], Linehan [41,39], Madeja [43], and DiBlasio [20] have explored ways in which computers and computer graphics can be integrated into art curricula. Currently, there are others, for example Linehan [40], Stiny and Gips [24], who are exploring the ways aesthetics might be applied to computer graphics, and the ways computer graphics might be applied to aesthetics and aesthetics research.

There is considerable discussion of how computer graphics might be applied to the fields of art and art education. There is little written, however, which provides practical instruction to the artist in the process of 3-D computer graphics.

1.3. LEARNING TO MAKE NEW TOOLS

The third problem in computer graphics is the creation of "tools" for the artist. As any artist working with computers can testify, computer graphics hardware and software development have been directed toward science, technology, and business applications. Because of this emphasis, it is important for the artist to become involved in the designing and planning of new computer graphic systems for artists. This requires that the artist be both skilled and knowledgeable enough to develop specifications for these tools. The artist must also be conversant enough in the field to express these ideas to an expert, and to work as a collaborator.

1.4. AESTHETIC AND ARTISTIC ISSUES

Once the artist understands the medium, the last problem is to explore the aesthetic and artistic potential unique to computer graphics and to create significant art.

Computer art is slowly becoming available in public exhibitions. Several world-wide animation festivals show computer-generated animation. The entertainment industry

is using computer generated animation. Several conferences on computer graphics have juried art exhibits, e.g., SIGGRAPH of the Association for Computing Machinery (ACM), Computer in Art and Design: Research and Education (CADRE) at San Jose State University, Center for the Study of Systems and Advance Technologies (CESTA) in France, and National Computer Graphics Association (NCGA). SIGGRAPH also collects these images and animations from their film and video shows and produces slide sets and video tapes of SIGGRAPH reviews. The Ohio Foundation on the Arts (OFA) is exhibiting 50 images of computer graphics in their Touring Exhibition program. The SIGGRAPH Art Show also tours the United States and other countries. More and more publishers are focusing on computer graphics images, such as "Computer Pictures" and "Computer Graphics World".

Currently, computer art is still judged on the aesthetic values of drawing, painting and photography. As more artists become involved and skilled in creating art with this medium, an aesthetic particular to computer graphics will evolve.

1.5. PROBLEM

Emphasis is placed on the first problem for the artist, mentioned above: gaining an understanding of the basic principles of 3-D computer graphics. I hope to demystify state-of-the-art, 3-D computer graphics for the artist through a discussion of the technical literature from the artist's point of view. I analyze the curricular needs of the artist, in this complex technical field by explaining and illustrating the key concepts of 3-D computer graphics. This information is system independent and the basic concepts are transferable to any system the artist uses.

1.5.1. METHODOLOGY

I search the scientific literature for information about computer graphics and search the art and humanities literature for articles written by artists about computer graphics and computer graphics utilization. After analyzing the literature from these two disciplines, I integrate and illustrate the concepts applicable to 3-D computer graphics for the artist. This will introduce the artist with a limited exposure to computers to the methods used to create state-of-the-art 3-D computer graphics.

1.5.2. LIMITATIONS

The research is limited to 3-D computer graphics for the visual arts. I do not deal with the history of computer graphics, 2-D computer graphics systems, or the aesthetics of computer art. The research is limited to literature in English. Moreover, mathematical concepts are not discussed in depth.

1.5.3. SIGNIFICANCE

There are many people in art, art education and computer science who believe computer graphics has not yet achieved its anticipated potential as an artistic medium. Michael Noll [48] believes that for this medium to be accepted and uniquely used, it will have to be used by the trained artist. He goes on to say that we will know when computer graphics has been accepted as an art medium, when computer graphics is discussed for what has been produced rather than how it was produced. Before this can occur, the artist must understand how computer imagery is created.

Stanley Madeja [43] states that a problem which concerns those in the visual arts is the aesthetic and design issues which accompany this visual technology. There is a growing popularity of home computers with color graphics capabilities. Frank Crow [14] has stated that the basic techniques for generating 3-D objects have been worked out and this software is currently available for use in microcomputers. Many schools now have microcomputers with some graphics capabilities. Art educators need to organize and teach this new visual language. Trained computer artists and art educators are needed to help shape the quality and direction of this field.

Concentrating on how 3-D computer graphics are created, I believe is a place to start in addressing the other problems. In the process of learning the basics, the artist will be introduced to the literature and will be better able to understand it. By learning the basics the artist will be in a better position to work with computer scientists and to start programming if desired. And lastly, learning the basics is essential to the artist controlling his art. After reading this, perhaps the artist will avoid getting lost in the technology by gaining a visualization of these concepts that underlie the

technology.

CHAPTER 2

MODELING THE WORLD

2. OVERVIEW: 3-D COMPUTER GRAPHICS

Computer graphics opens up a totally new medium for the artistic community. In order to fully utilize this new medium, artists need to understand its techniques and limitations. Although computer graphics is a new technology, it is eclectic and has borrowed concepts from many disciplines. The nature of computer graphics demands a variety of both technical, artistic, and aesthetic skills. Previous artistic skills and concepts may or may not transfer to this new medium.

A computer graphics environment has both hardware and software components. The hardware for the computing environment includes a computer or computers, terminals, printer, tape drive, and extra memory discs. System software for the computing environment includes an operating system, text editor, drivers for the hardware and various programming languages. The graphics hardware includes display devices, such as a CRT, vector display or picture-system, a color raster display device, frame buffer, and various input devices such as knobs, dials, joysticks, switches, buttons, bit pad and digitizing camera. It is also important to have a way of recording the images on film or video tape.

The graphics software necessary in a computer graphics environment can be categorized by the purpose of the software: creating data, scene description, display algorithms or rendering programs, post production (image processing) and a system for saving, filming, and/or recording frames.

Most environments are not made up of one large program. Instead systems are made up of several efficiently designed special purpose programs [16,19]. These programs carry out their designed task and send the results to the

next program for it to implement its special task. This is called a pipeline. A computer graphics pipeline looks something like:

---

data -> scene description -> rendering -> opticals -> camera

It is important to become comfortable with the computer environment, both the hardware and the software. Artists should familiarize themselves with the use of the computer, the operating system, the text editor, and the graphics software, as well as the input and output devices available on the particular system they are using. One important fact to keep in mind is that practically all problems solved using a computer involve translating the problem into arithmetic terms [8].

Computer graphics software will differ from system to system. There may be more options on some systems and not on others. Graphics or animation languages may have a different syntax from system to system. The basic concepts of 3-D computer graphics addressed in this thesis are those involving the first three steps of the pipeline: data generation, scene description, and rendering. These concepts are independent of a particular language or graphics device.

2.1. TIME AND SPACE: COMPUTATIONAL COST

Artists consider the level of realism of their work in relation to: the subject matter, the materials used, the time requirements, their skills at rendering realism, the amount of detail necessary, the expressive content, and the observer's perceptions. This is also true in computer graphics. Computer scientists must take into consideration; the level of realism needed for a particular application; the amount of detail recorded in the model; the processing time required of a computer to generate the image; the capabilities of the computer and of the hardware display; and the perceptual effects of the image on the observer. This has led to choices in how the artist and the computer scientist model the world.

Modeling is often seen as a process of simplification; yet portraying the world in a manner that is

believable to the viewer requires detail. Simplification verses complexity involves tradeoffs in computer graphics. The more complex and detailed the image is, the more expensive it is to produce. Conversely, the simpler an image is, the less believable it may be, but the easier it is to produce. In computer graphics an important concern is the processing time and memory space required to calculate an image. Computation time can be increased due to: the complexity of an object; the number of objects in the scene; or the complexity of the light and shading models used. Planning and calculating animation, which involves a sequence of images, makes it a necessity to take these types of problems into consideration in the design process.

The complexity of the graphics software used to generate images can vary. Programs that try to closely represent the real world may be very complex and require extended computation time. Similarly, mathematical models that try to represent highly detailed objects will take longer and require more computer memory to calculate images. The time it takes to produce one image can vary from a few seconds to a few days!

An artist considers cost and the size of his studio when designing a work of art. For example, a sculptor may decide to cast a large bronze statue. He considers the cost of the metal and whether he can do the casting himself or if he must use a foundry. These considerations may determine the final art object. This is also true in computer graphics. The computer artist has to consider the time required to compute an image(s) and the amount of memory necessary for a particular project. If the artist does not have a computer or his computer is not capable of calculating or displaying a particular image, then he might rent time on a system. Cost in computer graphics refers to two costs; time and space.

Realism in 3-D computer graphics requires a fairly powerful computer, considerable mass storage, and a color display device with its own memory. The price of computers and memory has decreased substantially in the last five years, but they are still very expensive. Image generation involves a great deal of computation time and capital. The more sophisticated image generation becomes, the more potential there is for image making for the artist. Due to the computing speed and memory size

required, most computer graphics computation is done on mid-sized computers. Currently, there is a new generation of microcomputers that is approaching the performance of the mid range computer. Some of these micros are now being tested and used in computer graphics labs. We can look forward to the day when the computing power necessary for computer graphics is more readily affordable.

2.2. VISUALIZING THE WORLD

As artists see the world (real or imaginary) they create an image of this world in their "mind's eye". This mental image is transformed by the artist and actualized in either a two or three dimensional medium.

In 3-D computer graphics it is necessary to create a mathematical model of the world in the computer memory. Creating objects in this "digital" world is similar to creating sculpture or architecture. Therefore the artist needs a good understanding of space and form. Objects can be manipulated by changing their size, position, and orientation in space. Objects can be painted. Lights sources are placed to achieve desired effects. The artist then chooses both a window from which he will view this world and a direction of view. This building of models, the organization of space, color, light and view, is where the artist/user exercises control in the process. This is similar to a director setting up a stage with actors, props, and background for film or theater. The direction comes from the user, but in computer graphics, the actual changes are done mathematically by software. A discussion of the problems of modeling objects, transforming objects, viewing objects, modeling of light, shade, and color is presented in the following sections.

2.3. THE CARTESIAN COORDINATE SYSTEM

In a digital computer, the world must be modeled mathematically. Each 3-D model must have a complete definition in 3-space. The frame of reference for this space is the Cartesian coordinate system, which provides us a standard mathematical reference system. The center of the system is called the origin. The X axis runs horizontally with positive X (X) to the right and negative X (-X) to the left of the origin as shown in figure 1. Y is the vertical axis with Y going up and -Y going down from the origin. The X and Y axes create the XY plane, as seen

Figure 1. Cartesian coordinate system: X axis and origin

in figure 2. Extending the Z axis with Z advancing in space and -Z recede in space, creates depth or the third dimension. This creates two more planes, the XZ which extends horizontally and the YZ which extends vertically. Figure 3 illustrates the Cartesian coordinate system.

The artist must understand how to represent a 3-D object numerically by plotting points within the Cartesian coordinate system. Each coordinate point in this space can now be defined as an "X, Y, Z" triple. Points are always as an ordered triple. Figure 4 demonstrates how points are plotted in two dimensions to create a simple 2-D square. Extending this to 3-D, it is easy to see how to plot the points for a cube. An example of this can be seen in figure 5.

2.4. DATA

As noted above, the mathematical model is fundamentally a simplification of form. The more the mathematical model approximates reality, the more complex the data will be. Realistic objects may become very complex. As the amount of detail grows, so do the requirements for storage

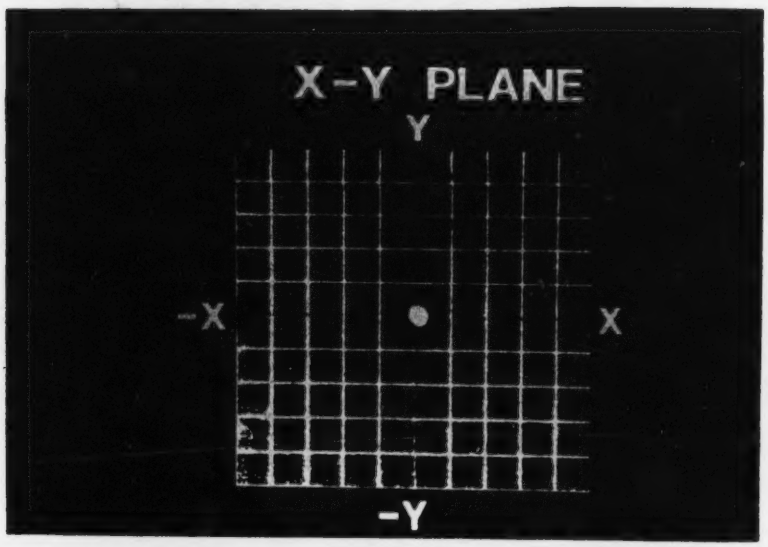

Figure 2. Cartesian coordinate system: X and Y axis

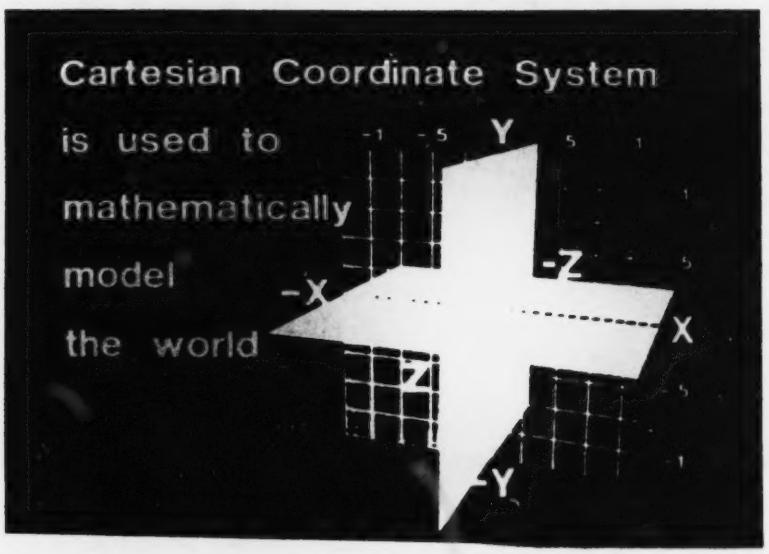

Figure 3. Cartesian coordinate system: X, Y, and Z axes

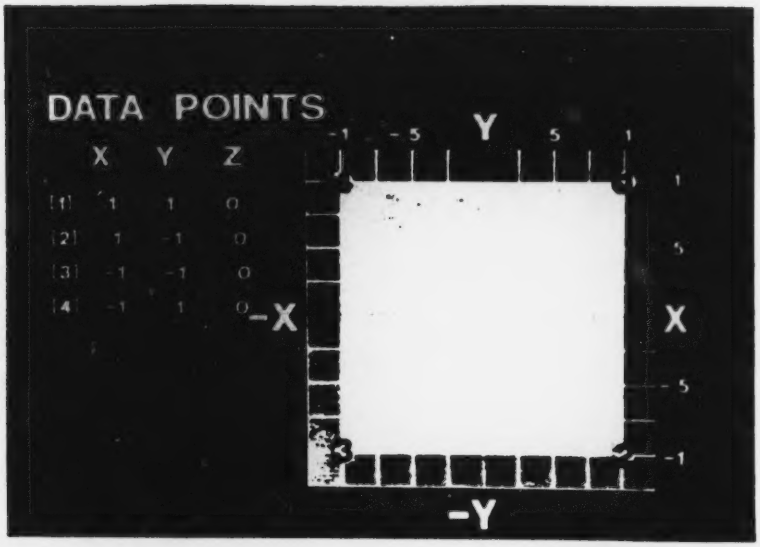

Figure 4. Coordinate points for a 2-D square

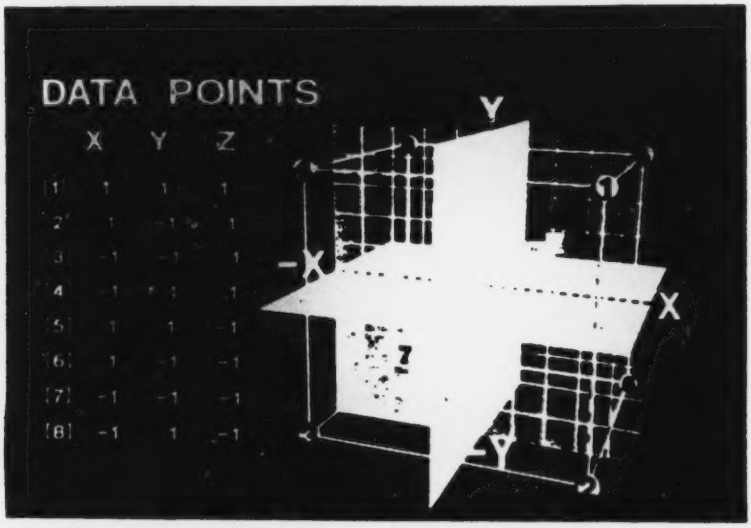

Figure 5. Coordinate points for a 3-D cube

space and processing time. Moreover, creating data can be very time-consuming and tedious.

In generating data we have to be concerned with how to simplify a form and still represent the desired objects. It is fairly simple to represent a cube in 3-space, but objects with curved lines and curved volumes becomes more difficult. The method used is similar to the connect the dot drawings found in children's books. A straight line is a line between two points. To approximate a curved line several points are plotted and straight lines are drawn to connect the points. For example, a simple circle could be made up of six points and would look like a hexagon. A detailed circle made up of 360 points would reduce the length of the straight edges joining the points, and as a result they would be difficult to detect. Where the simple circle would be made up of straight edges, the detailed circle would appear to be a continuously curved line.

2.4.1. METHODS OF CREATING DATA

The design of an object should take into consideration that it can be viewed from any point in space. As in sculpture, the data may need to be interesting from multiple viewpoints. There are various methods for planning an object and determining the coordinate points when making data by hand. One method that is very useful is the orthographic projection. This involves plotting 3-D points on a piece of 2-D graph paper using the Cartesian coordinate system. The front view of the object is projected onto the XY plane. A side view is projected onto the YZ plane. A top or bottom view is projected onto the XZ plane. These views are similar to an architect's blueprints. The plan view is the top view looking down and the front and side views are the elevations. Folding the orthographic projection into a box with the side and top folded back provides a model of this object in 3-space. Figure 6 is an example of a simple house or "hut". This figure displays the orthographic projections of the "hut" on each plane, and the numerical value for each point.

Orthographic projections make it easier to ascertain the coordinates of a point because the artist only has to

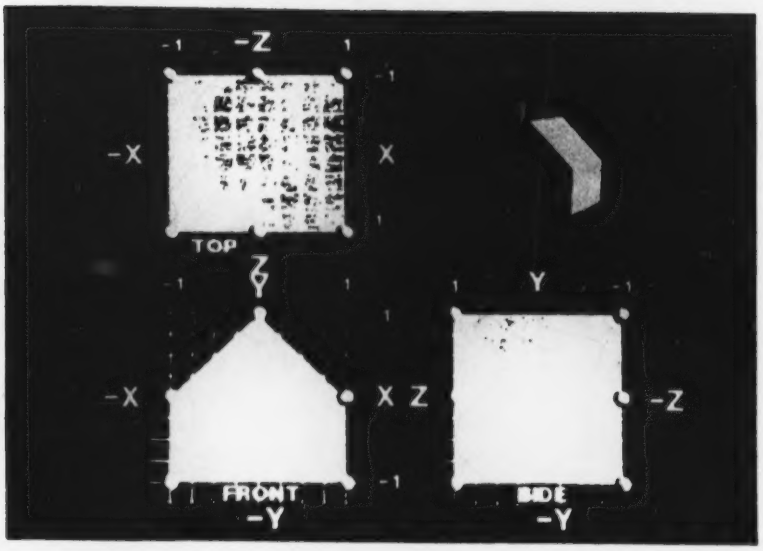

Figure 6. Orthographic projection

consider two axes at a time. By marking the origin on the graph paper and counting spaces along each axis, it is fairly easy to plot the coordinates of the data. One consideration is the location of the origin in relation to the object. Usually, the origin is at the center of the object. The reason for this will be discussed when scaling and rotating are presented in a later section.

Many graphic systems have a library of pre-defined geometric models, or primitives, (e.g., cubes, cones, spheres, pyramids, and cylinders, etc.). These primitives can be scaled, rotated, and placed in space to build new objects. This is the building block, or combinatoric, approach to data generation. Since a sphere can be flattened and then stretched to become an ellipsoid, and a cylinder can be stretched into a long thin rod or squashed into a flat coin, it is possible with just a few primitives to create very complex objects. In the same way an entire house could be built used only one primitive, a cube. The cube could be used for everything from bricks, to flooring, to windows, to roof tiles, to stairs, and doors. Many forms may be created by combining various primitives. An example of this method will be discussed in more depth later.

To make data generation easier some basic methods for creating new data have been developed, (e.g., warping, projection, solid of revolution, and lofting.) Warping is the moving of the points of a primitive or an existing data set to create a new object. For example, some of the points making up a sphere can have the value of their coordinates contours. In effect, we can "push" and "pull" points to create a new shape. Figure 7 is an example of warping the points of a sphere and a grid to create new forms.

Another simple way to create data is projection. In projection a shape or a contour is extended into space. This is similar to making a cookie with a cookie cutter or extruding clay through a template. For example, a square becomes a cube and a circle becomes a cylinder through projection techniques. The coordinate points for a shape are plotted in the XY plane and all points are given the same Z value of zero. All these X and Y coordinate points are then repeated with a changed value in Z to one. An example of this method can be seen in figure 8 which shows the plans for a lion. The upper left portion of figure 9

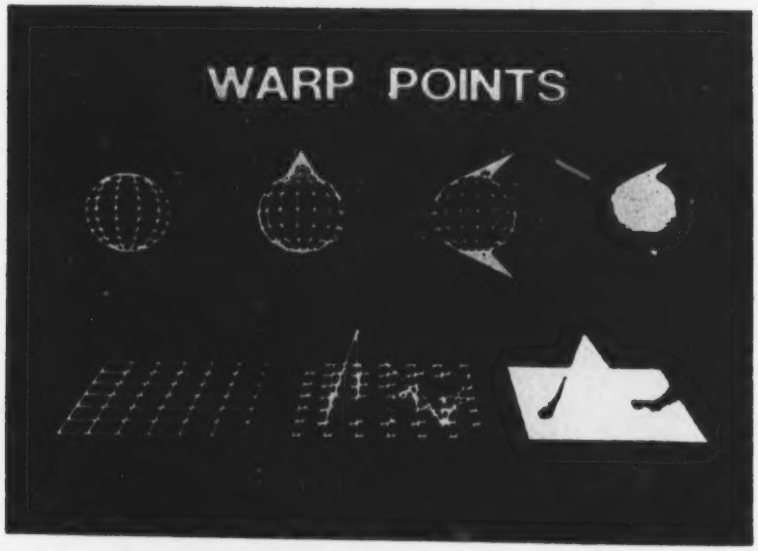

Figure 7. Warping points

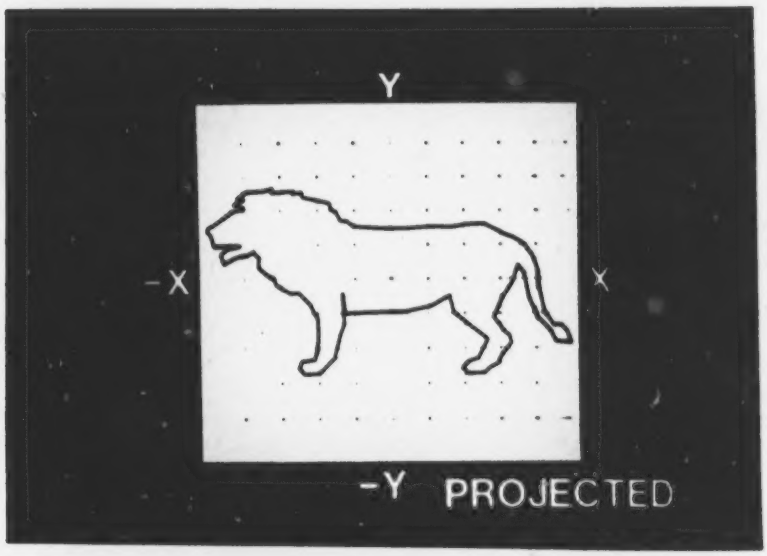

Figure 8. Data generation: projection plans

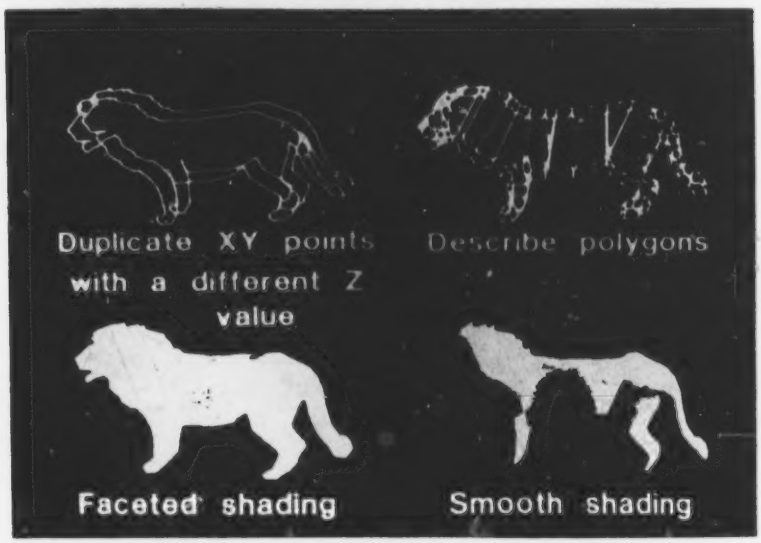

Figure 9. Data generation: projected lion

represents two outlines of the lion with different Z values. The upper right illustrates the polygon

24

description of the lion, which will be discussed below. At the bottom of this figure the same data set for the lion is displayed using two different display techniques: faceted shading and smooth shading. The difference between these two algorithms will be explained later, but it is easy to see the different effects.

Solid of revolution is very useful in plotting the points for a symmetrically curved surface, for example a goblet. This is done by designing the object and dividing it into equal size pieces as if we are cutting up a pie. Figure 10 shows the plans for a goblet with the top view in the XZ plane. Notice that the entire circle of this object is planned and then divided into pieces of equal size. The darker section is the piece we will create. The right represents the profile or side view of this "piece". Only this one "piece of pie" is plotted. The entire goblet can then be created by rotating the piece at the correct intervals to form the entire object. Figure 11 starting on the left illustrates the piece we plotted, then the entire object made up of rotated pieces, and the

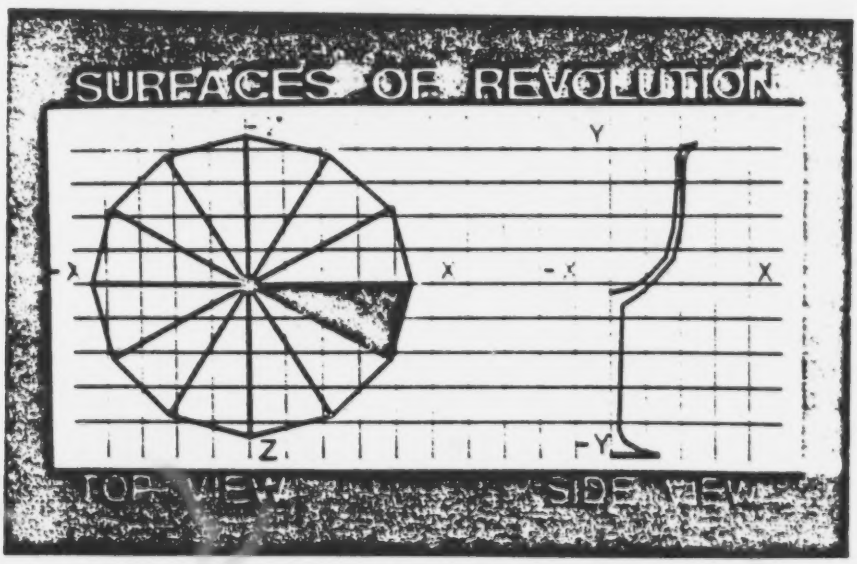

Figure 10. Data generation: solid of revolution plans

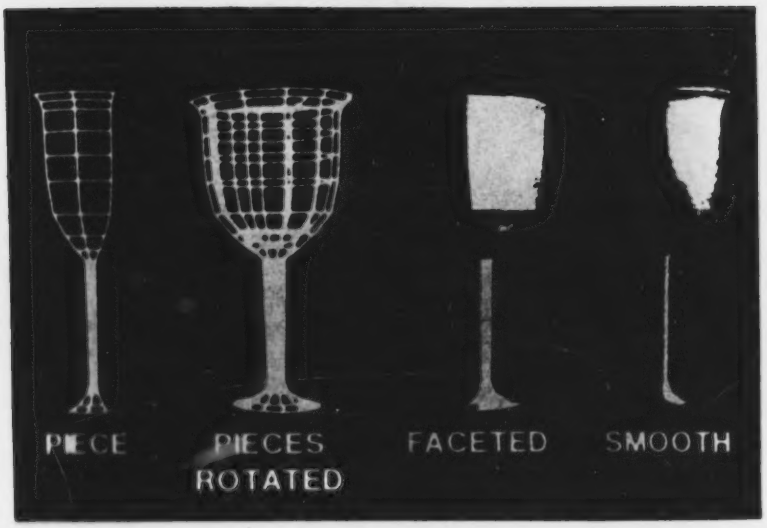

Figure 11. Data generation: solid of revolution goblet

resulting goblet both faceted and smooth shaded.

Lofting is very similar to defining the contours for a topographic map. A series of cross-sections of an object are defined and the altitude of each cross-section is specified. This method of creating data can be either very simple or complex depending on the thickness of the slice. If an artist wants realistic data, he could create a wax model of the object. This model is then cut into thin slices and the points for the contours plotted [25,56]. Figure 12 illustrates the plans for a simple banana. The left image on the top is a view of the contours in the XZ plane, and the image on the right shows the thickness of the slices. This view gives us the coordinates in the XY plane. This object is vertically symmetrical. The same contours in this example are each used twice, once with a positive Y value and once with a negative Y value. The resulting data can be seen in figure 13, starting on the left with the polygon description, then the faceted and smooth shaded banana. A more detailed representation of a banana could be created by making more accurate contours and by creating more slices.

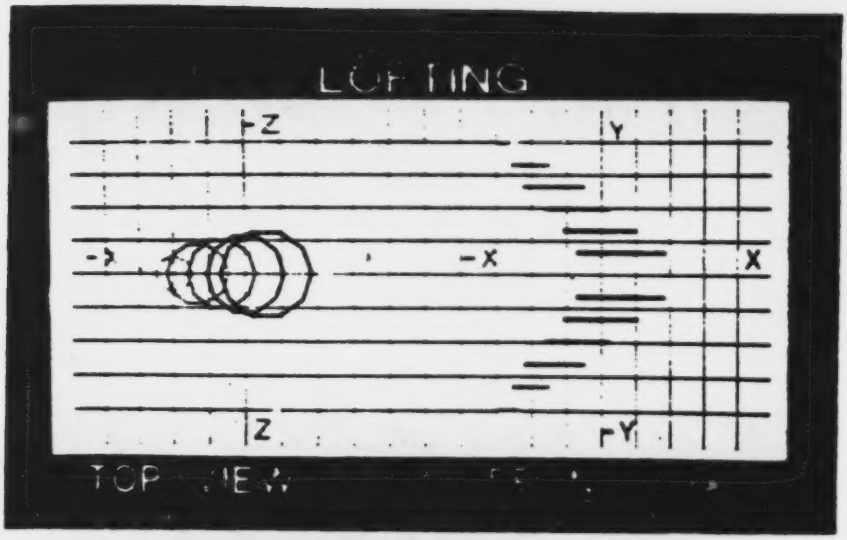

Figure 12. Data generation: lofting plans

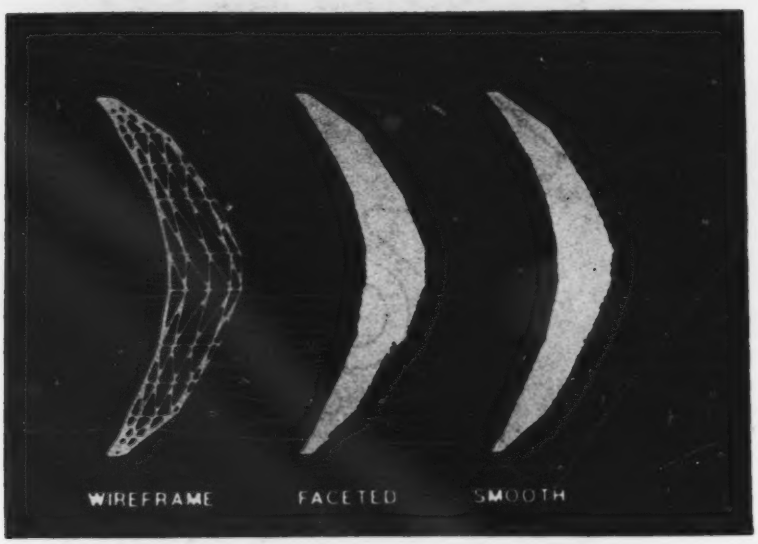

Figure 13. Data generation: lofted banana

Many of the above methods of creating data, as well as others, have been implemented in software [53,9]. Many

CAD/CAM and computer graphics systems provide the used with various methods of designing and manipulating objects. One basic concern in data-generation systems and in computer graphics in general is the user interface. The user interface refers to communication between the user and the computer. The ease of both learning and using a system is important. In an interactive system the artist can communicate with the computer and the computer responds via display devices or text. Entering information into the computer is done through various input devices, (e.g., a keyboard, a joy stick, a mouse, switches, dials, tracker ball, light pen, or digitizing tablet). The digitizing tablet, sometimes referred to as a bit pad, is the most frequently used device for entering graphical data into the computer. The choice of method can generally be chosen from a menu. Various sensing apparatus are used by the data generation program to record and save the points entered. There are 3-D input devices which register all three axes for each point, but as yet none have proven reliable. The method most commonly used is 2-D input from a digitizing tablet. The artist may enter points by "drawing" on the surface of the tablet and receive immediate visual feedback on a line drawing or calligraphic display device. In this way it is possible for the artist to create objects with an approach similar to drawing. The reader interested in data generation is referred to the systems developed by Rick Parent [53] and Wayne Carlson [9].

2.4.2. POLYGONAL MODELS

Until this point only models that represent objects as sets of points have been discussed. Connecting these points with lines or edges provides a line drawing of the 3-D object, called a wireframe view of the object. The "hut" we designed in figure 6 can be seen as a wireframe display in figure 14. In this illustration the points of the data are emphasized, though normally only the edges between the points are seen.

If an object is to be shaded, its surfaces, not just the edges, need to be defined. One way of approximating a surface is by modeling it as a set of polygons, i.e., as a polyhedron. A cube is a simple polyhedron with each side composed of a polygon made up of four connected points. Polyhedral or polygonal data is completely covered with flat, straight edged tiles. An example of this is the

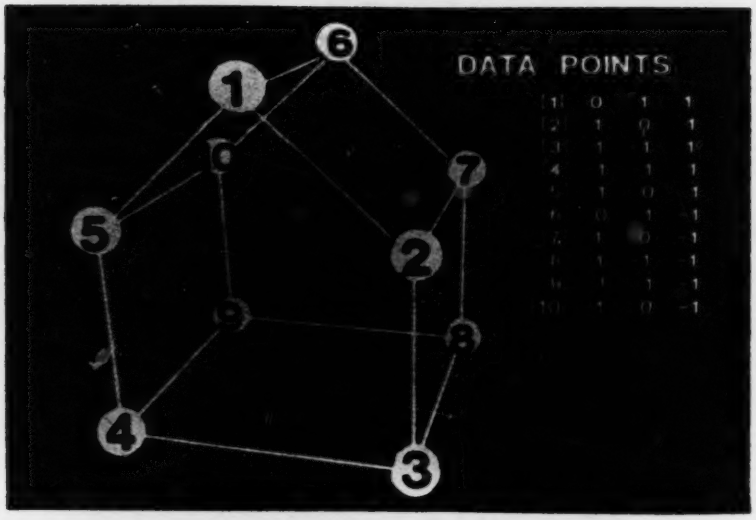

Figure 14. Wireframe "hut" and coordinate points

"mirrored ball" often seen hanging from the ceiling of a discotheque or ballroom. These flat mirrors are placed edge to edge to approximate a sphere.

There are several things to consider when defining polygons. One thing to keep in mind, is that a polygon has to be planar. A triangle, because it only has three points, will always be planar. Most surfaces are not planar, therefore polygons are broken down into triangular polygons. Several of the previous examples of data show polygons of various numbers of points. When defining a polygon the choice of points to be connect is determined by which points are in the same plane.

Having discussed the surface of an object, we can refer to its "inside" and "outside". Each polygon has two sides or faces that need to be distinguished from each other. A consistent order is used when describing the polygon by listing every point or vertex around the periphery of the surface. The order used when defining the polygon is important to determine if the polygon is facing towards or away from the viewer. There is a simple method for telling front from back. When defining a polygon, each vertex in the polygon is listed in a clockwise order

as seen from the observer position or the outside of the object. This clockwise order is used to define the front of the face or a frontfacing polygon. The arrows in figure 15 connect the five points of the front of the "hut" into one polygon in a clockwise order, (starting at point 1 then, point 2, 3, 4, and 5). The software usually assumes that the last and the first point are connected for closure. The reverse order as seen from the eyepoint would indicate that the polygon is facing away or backfacing [46].

Figure 16 is an illustration of frontfacing and backfacing polygons with a direction arrow embedded in them. Normally the backfacing polygon would not be seen, but here it is shown on the left. Keeping track of frontfaces and backfaces is a way of speeding up the display algorithm. If the face is clockwise it must be visible, if counterclockwise it is discarded or culled and not sent on to the display program [14].

Creating backfacing polygons may be done accidentally or intentionally as part of a design consideration.

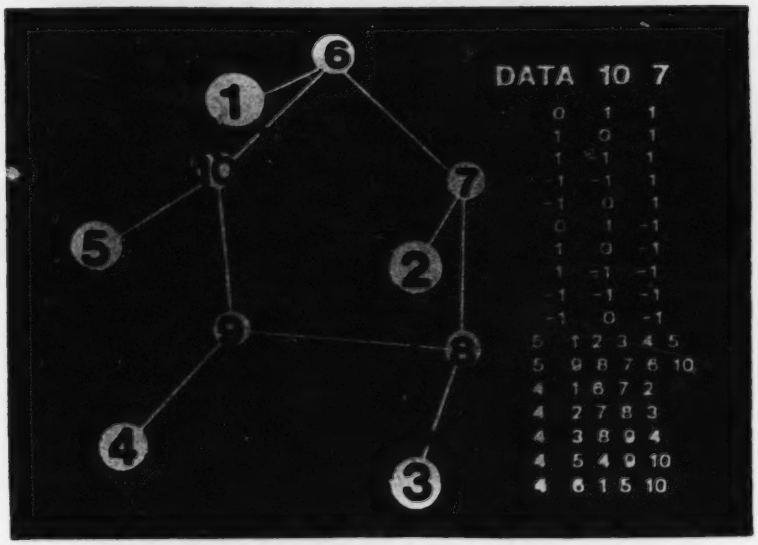

Figure 15. Face description and "hut" data set

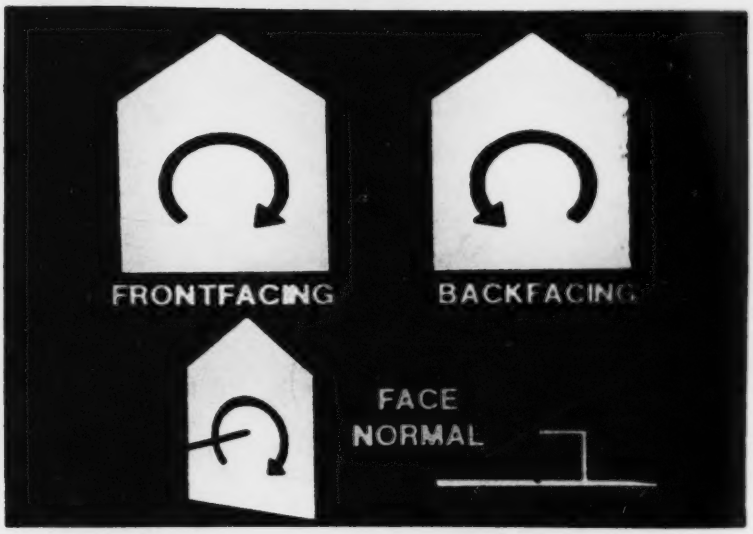

Figure 16. Frontfacing and backfacing polygons and face normal

Backfacing polygons can leave holes in an object because those polygons will not be displayed. If the eyepoint is placed inside a box, the box would not be visible because all the polygons would be backfacing. To eliminate this problem, all the polygons can be defined twice, once clockwise and once counter-clockwise. Another method is to turn off the culling of backfacing polygons for the object. Purposely designing in backfaces allows the artist to play with space in the object. A different object may be designed on the inside, or the backside of an object. An example of this can be seen in figures 28-31. This object is designed to be different from two sides.

By assuming all the polygons are planar it is easy to use the clockwise description and mathematically determine a line that is perpendicular and pointing outward from the frontface of the polygon. This line is called the face normal or normal vector. This face normal is not visible, but is mathematically calculated and used by the display algorithm. In the lower half of figure 16, the normal is represented as a straight arrow. These two views illustrate the normal's perpendicular relationship to the face.

On a frontfacing polygon the software would determine that the normal is pointing toward the viewer; in a backfacing polygon it would be pointing away from the viewer.

By mathematically creating a line, or vector, from the observer's eye to the normal we can determine the angle between these two vectors. If the angle between these two vectors is less than 90 degrees then the polygon is visible. If the polygon is equal to or greater than 90 degrees then the polygon is backfacing and not displayed [46]. The importance of the face normal will be better understood when lighting and shading models are explained in a later section.

Some hidden-surface algorithms only handle convex polygons. A convex polygon has no internal angles greater than 180 degrees. If it can be assumed that all polygons are convex, the calculations may be less complex. Consequently we can speed up the time for image generation [46]. Figure 17 shows convex polygons on the right and a procedure to break up a concave polygon into convex polygons on the left.

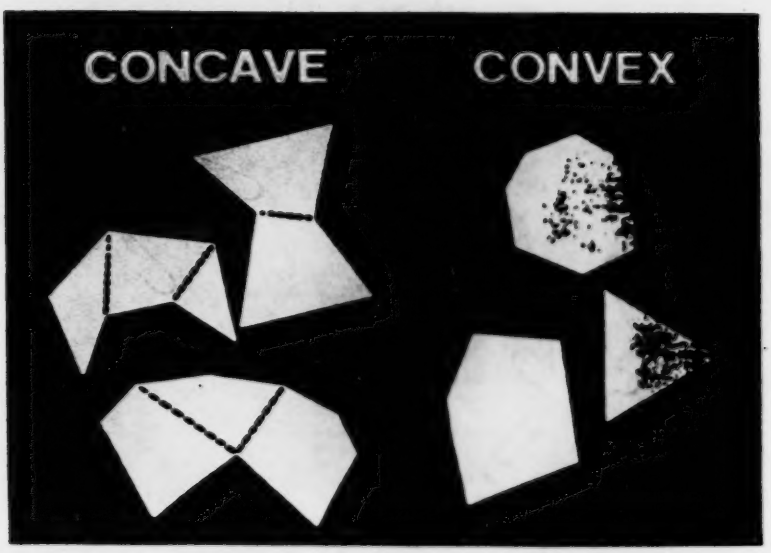

Figure 17. Convex and concave polygons

An object is a collection of adjoining polygons. It is important to store the data set for an object in memory in the most compact form. Since neighboring polygons share vertices along common edges, data can be easily and compactly defined by listing each vertex once. The data set for the "hut" is listed below and can be seen in figure 16. The format for this data is the data line with the first number representing the number of vertices in the data and the second number representing the number of polygons in the data. This means that the first 10 lines after the data line are coordinate points, each an XYZ triple. These are followed by the polygon descriptions, which are each made up of two parts. Part one is the first number which represents the number of vertices in the polygon. Part two is the list of vertex numbers. These vertex numbers are pointers to the coordinate point list [14]. For example the first polygon is made up of 5 vertices, (vertices 1, 2, 3, 4, and 5).

Data 10(points) 7(polygons)

	X	Y	Z
Vertices			
(1)	0	1	1
(2)	1	0	1
(3)	1	-1	1
(4)	-1	-1	1
(5)	-1	0	1
(6)	0	1	-1
(7)	1	0	-1
(8)	1	-1	-1
(9)	-1	-1	-1
(10)	-1	0	-1

Polygons						
(1)	5	1	2	3	4	5
(2)	5	9	8	7	6	10
(3)	4		1	6	7	2
(4)	4		2	7	8	3
(5)	4		3	8	9	4
(6)	4		5	4	9	10
(7)	4		6	1	5	10

Once the coordinate points and polygons for the "hut" are defined, it can be displayed as a solid, shaded object. Figure 18 illustrates the clockwise order of the

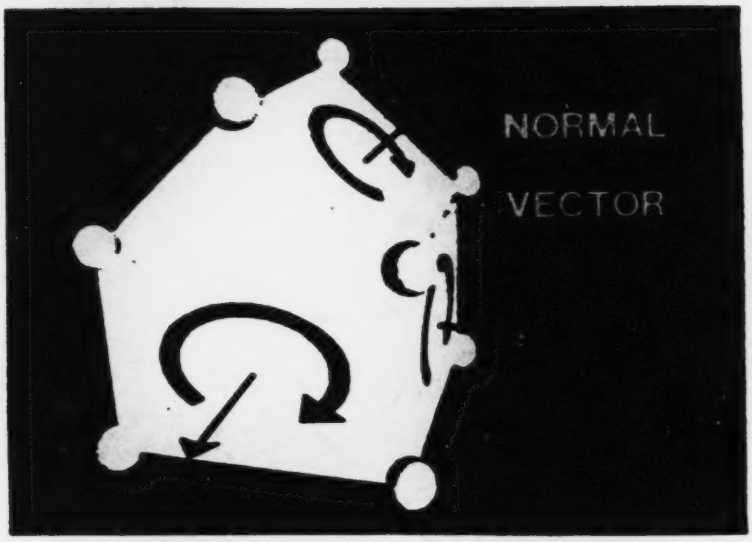

Figure 18. "Hut" with clockwise arrows and face normals

face definition and the normal to each face. Notice that when we look through the cut-out clockwise arrow into the inside of the "hut", there is nothing displayed. The interior of the hut consists of all backfacing polygons from this view, and is not displayed.

Because complex objects will increase the computation time needed for display, having the same object with different levels of detail can be helpful. An object displayed very small, or in the distance, does not need all the points and polygons. It may be as small as a dot on the screen, and still take almost the same amount of time to display as a detailed object with a large number of points and polygons in the foreground. Figure 19 shows how an object can have different levels of detail for various purposes and distances. On the top are wireframe images of a station wagon at three levels of detail. The number of points and polygons are listed for each object. At the bottom of the figure we can see that when an object is either scaled small or moved back in space, it is not necessary to display the object at its highest level of complexity.

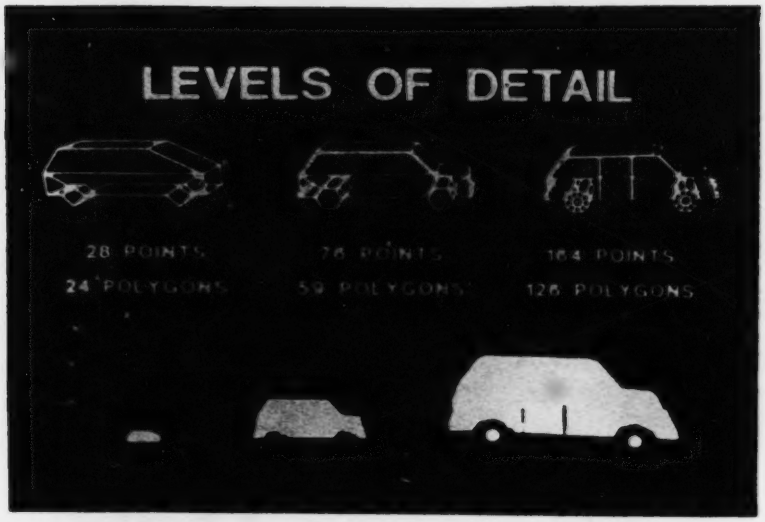

Figure 19. Levels of detail

Polygonal data is a planar approximation of a curved surface. Smooth shading, discussed below, can approximate the curved surfaces of the interior of a form close enough to fool the eye. However, the outer contours of the form are still made up of straight edges. There are other methods of creating data that more closely represent the curved surfaces of the interior and the outer contours of a form. Examples of these higher-order approximations are B-spline, Coons, and Bezier patches [25,46]. Patches are like a mathematical sheet of rubber that can be used to make up an object. The form of the object can be adjusted by moving control points. These control points define the surface of the form. When these points are moved, the form changes. Sometimes these control points are given a weighted value which controls the area of their influence over the form. Foley and Van Dam [22] is a good source for a more detailed description of these processes.

CHAPTER 3

DESCRIBING A CHANGING WORLD

3. OBJECT TRANSFORMATIONS

Once various objects are defined in the computer there are several things we can do with each object to create a scene. Objects can be made larger or smaller, or scaled. They can be moved around in space, or translated. They can be turned in space, or rotated. The manipulation of an object can be performed on a copy or instance of the original. In this way we can have several instances of the same original object and manipulate each individually. Each copy can be given a name to assist in keeping track of the many objects in a scene. These manipulations of the object are called transformations. They are based on standard mathematical techniques of coordinate geometry, trigonometry, and matrix methods [46]. These transformations can be expressed mathematically in matrix algebra by a single entity called the transformation matrix. Several transformations of an object can be combined or concatenated into one matrix [14].

Usually the software creates and manipulates the matrices, and the artist/user need not deal with them. A painter does not have to know chemistry and mix the ingredients for his own paints in order to paint. However, if the artist is interested in this part of the process, he can gain the knowledge and skill to mix his own paint. This is also true in computer graphics. The artist can remain at the user level and create his art; or if he desires he can write his own software to create and manipulate the matrices.

3.1. SCALING

Each coordinate is multiplied by a scale factor [14] in order to scale an object. One is free to scale independently in each axis and thus change the shape or proportions of an object. Objects can also be scaled

equally in each axis and maintain a larger or smaller version of the same object. In this way it is very easy to turn a cube into a board by scaling it up in X and scaling it down in Y. For an example we can use the coordinate points for the "hut" which were listed previously. In order to scale it by 4 in X, each point in the X column would be multiplied by 4. The Y and Z coordinates would remain the same. Thus the outer limit or bounding box of the "hut" has been changed. Notice that this object is designed with its origin at the center of the coordinate system, therefore scaling will be uniform.

Original bounding box	New bounding box
(-1 X to 1 X)	(-4 X to 4 X)
(-1 Y to 1 Y)	(-1 Y to 1 Y)
(-1 Z to 1 Z)	(-1 Z to 1 Z)

The resulting object can be seen in figure 1. The original "hut" is the small dark shape in the center of the grid and the new scaled "hut" is seen stretched in the X

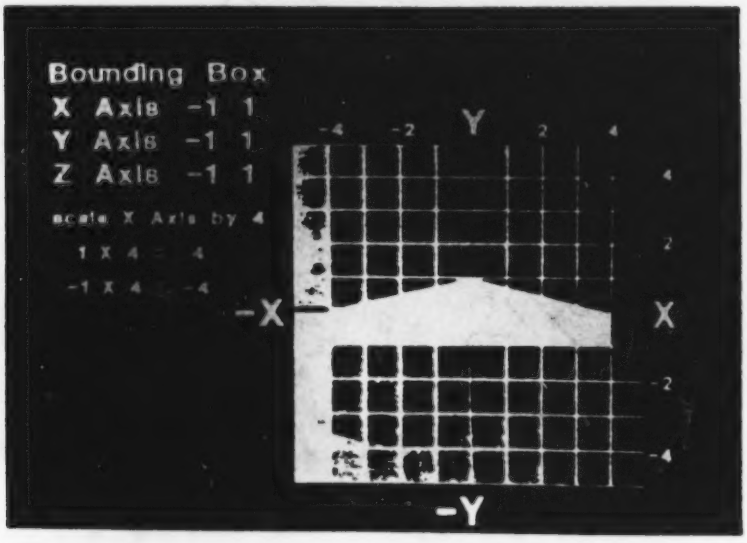

Figure 1. "Hut" scaled by 4 0 0

axis. Figure 21 shows the "hut" scaled only in Y and then scaled by 4 in X, Y, and Z in figure 22.

While discussing the design of data the location of the origin of the object was mentioned. Scaling of an object whose centroid or center of the object is not coincident with the origin of the coordinate system will not always be uniform. In Figures 23 and 24 the "hut" is designed with its origin in the lower left hand corner. Any points with a coordinate value of zero will, of course, remain zero when multiplied by the scale factor. The two bounding boxes below illustrate this concept. The result of scaling the hut by 4 in X can be seen in figure 23.

Original bounding box
(0 X to 1 X)
(0 Y to 1 Y)
(0 Z to 1 Z)

New bounding box
(0 X to 4 X)
(0 Y to 4 Y)
(0 Z to 4 Z)

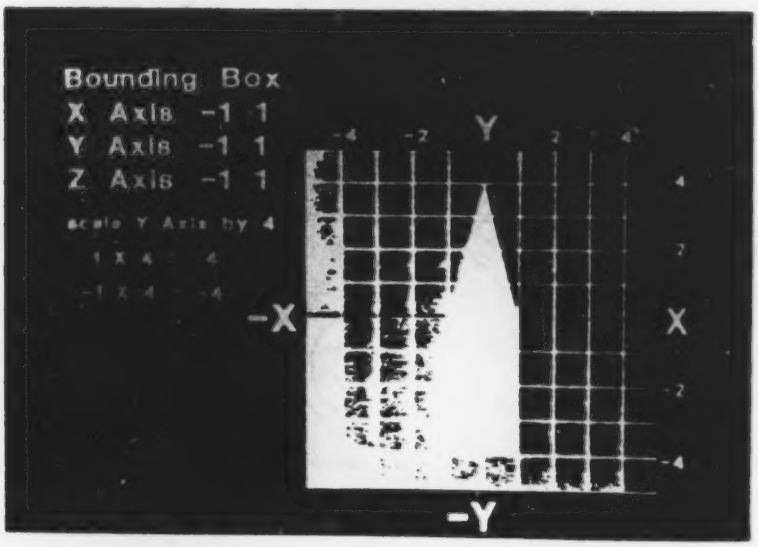

Figure 21. "Hut" scaled by 0 4 0

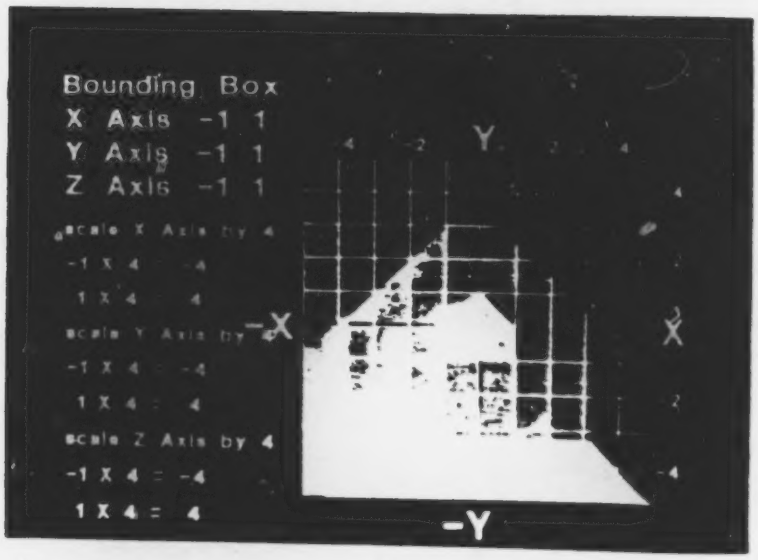

Figure 22. Scale "hut" by 4 4 4

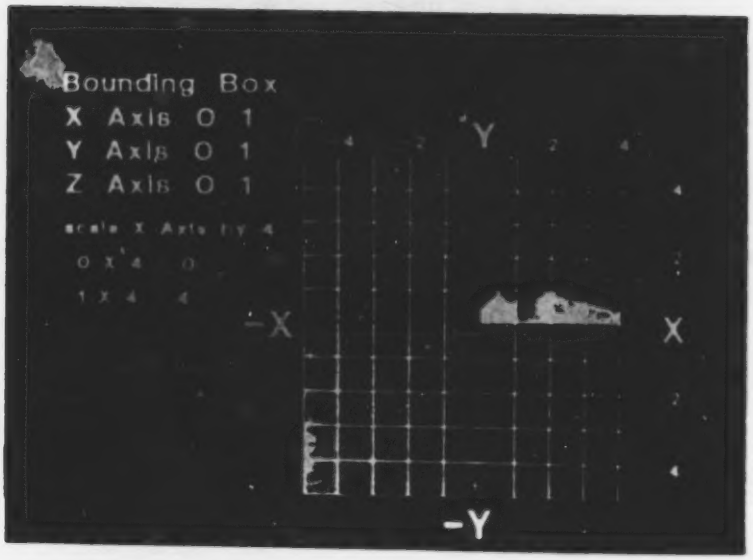

Figure 23. Off-origin "hut" scaled by 4 0 0

This same principle holds true when the hut is scales by 4 in X, Y, and Z as shown in figure 24. If there are no

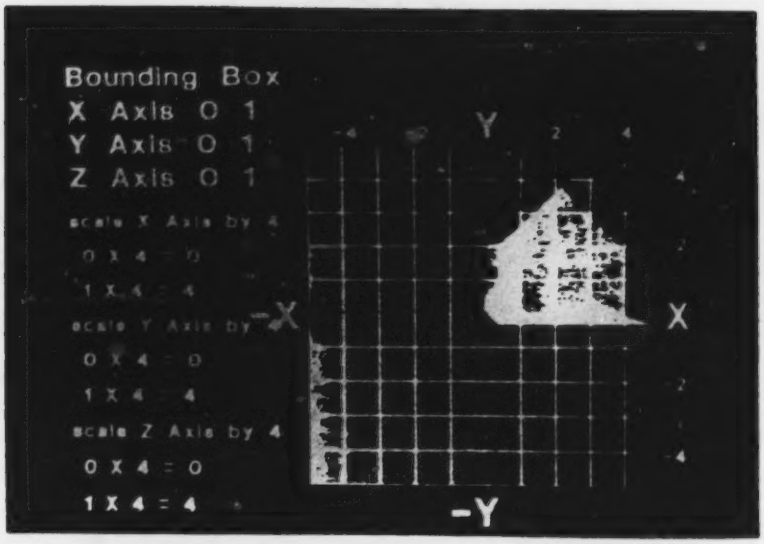

Figure 24. Off-origin "hut" scaled by 4 4 4

points on the origin of an axis, then the object will appear to move when it is scaled. The bounding box below is for a "hut" designed with its lower left hand corner at 1 1 0. Notice what happens to its bounding box when it is scaled by 4.

Original bounding box	New bounding box
(1 X to 2 X)	(4 X to 8 X)
(1 Y to 2 Y)	(4 Y to 8 Y)
(0 Z to 2 Z)	(0 Z to 4 Z)

Figure 25 demonstrates what happens when scaling an object designed like this. This type of object can be difficult to control because scaling the object can also appear to move it. Although, in effect, it is really the coordinate system being stretched. An object may be consciously designed with its origin not at the origin of the coordinate system. For example an object designed in several pieces may need to have all the fitting parts designed in the correct relationship to each other so that when they are scaled they maintain the original relationship.

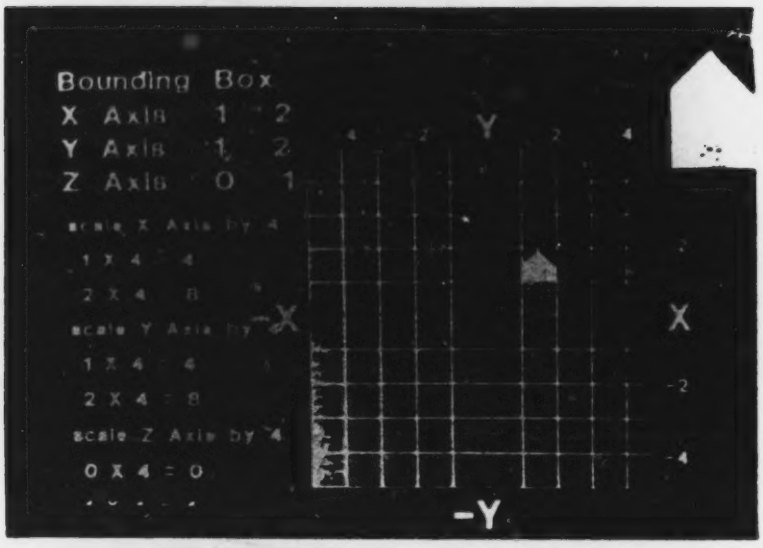

Figure 25. Off-origin "hut" scaled by 4 4 4

Scaling can be used for special effects. For example an object can be scaled by such a small number that it will disappear or such a large number that it will cover the entire screen. Scaling an object by a negative value will turn it inside out and in effect reverse the frontfacing and backfacing polygon descriptions. An object designed with another object inside it can provide an exciting use of this principle when scaling.

3.2. TRANSLATION

In translation a value is either added to or subtracted from each coordinate in order to move it around in space [14,46]. As with scaling this can be done independently along each axis or in more than one axis to place the object where desired in space. A positive number moves the object in the direction of the positive end of the axis. A negative number moves it towards the negative end. In the example of the "hut" listed before, the object has its origin at 0 0 0. In order to move the object a translation is given, for example 4 4 0 in figure 26. We can think of 4 being added to each X and Y value of all the points. If the translation is -2 3 2, these values will be added or subtracted respectively from each

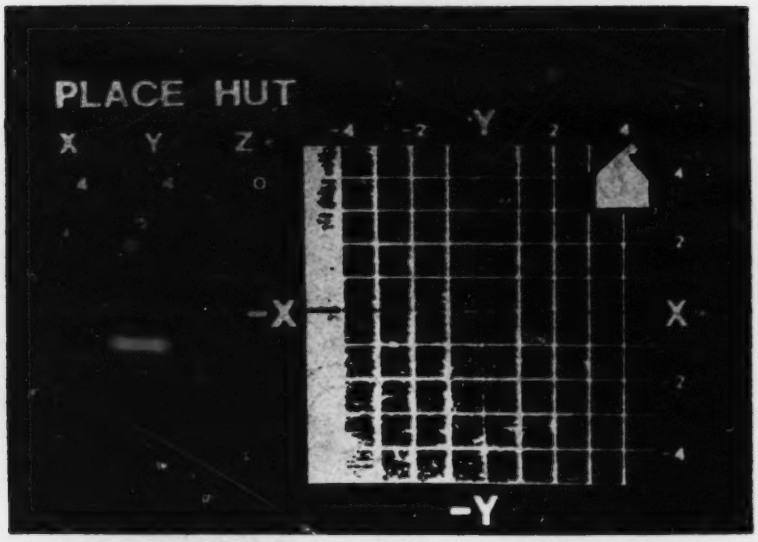

Figure 26. Translate "hut" 4 4 0

coordinate of the object as seen in figure 27. It is easier to think of translating the object origin to -2 3 2, for example, rather than keeping track of the addition and subtraction of these values.

Translation is used to organize and position objects in space. Instancing and placing the same object at various intervals can be used to build more complex images, for example a picket fence, bars for a crib, or stairs.

3.3. ROTATION

Rotations are mathematically more complex. They are done through matrix multiplication using sines and cosines [14,46]. As mentioned above the matrices are transparent to the artist/user. In scaling, large numbers expand the object and small numbers contract it. In translation, a positive number moves the object towards the positive end of an axis and a negative number moves it towards the negative end of the axis. In rotation we have to think of the direction of the rotation. The direction of a rotation is dependent on the particular space and whether the value is positive or negative. The space used here will be right-handed space. To visualize the

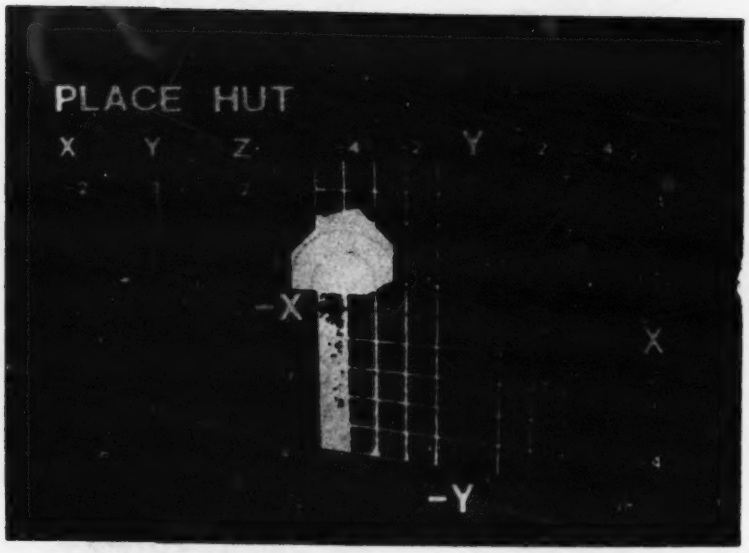

Figure 27. Translate "hut" -2 3 2

direction for a rotation, we grasp the axis with the thumb pointing towards the positive end of the axis. The fingers will curl in the direction of a positive rotation. When looking down an axis from the positive end towards the origin, a positive rotation will be counterclockwise [14]. Figure 28 uses arrows to indicate the positive direction of the rotations in right-handed space.

Rotations are usually specified by the axis of rotation, and degrees of rotation, either positive or negative for the direction. A simple rotation is a rotation around a single axis in the coordinate system [14]. The object in figures 29, 30, and 31 is designed with its origin coincident with the origin of the coordinate system. Therefore, when rotating this object on the X axis, it is rotating on the X axis of the coordinate system. This is called an on-origin rotation. Figures 29, 30, and 31 demonstrates on-origin rotations, rotated in increments of 60 degrees about each axis independently in a positive direction.

Rotations off-origin are different from on-origin rotations. This can be thought of in terms of the solar system. An on-origin rotation is like the earth spinning

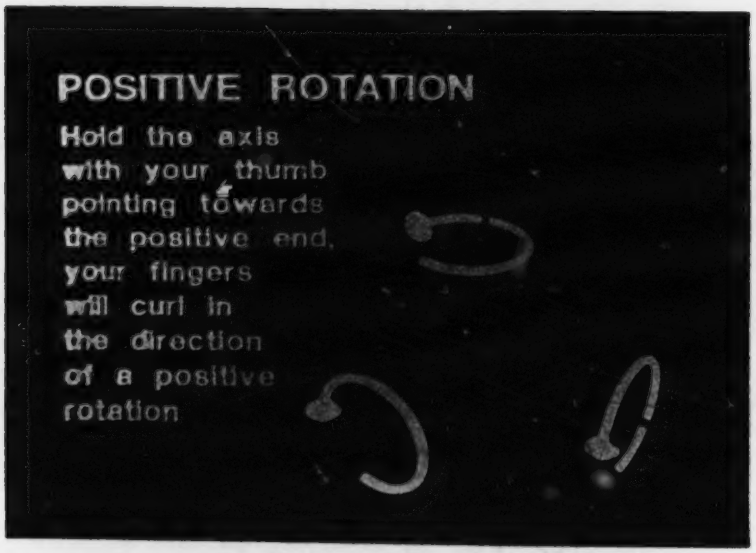

Figure 28. Direction of rotations

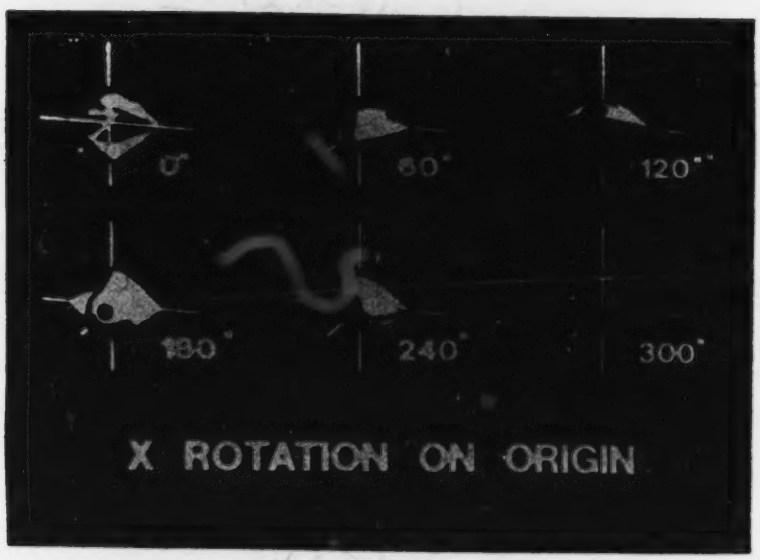

Figure 29. X rotations

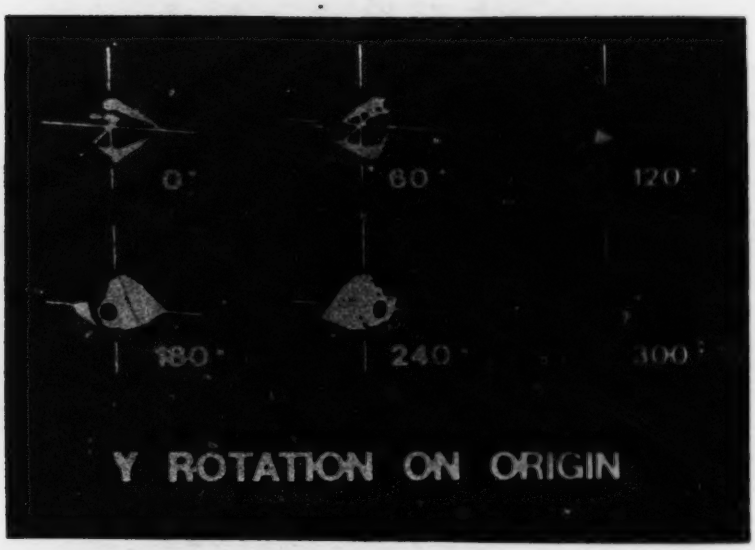

Figure 30. Y rotations

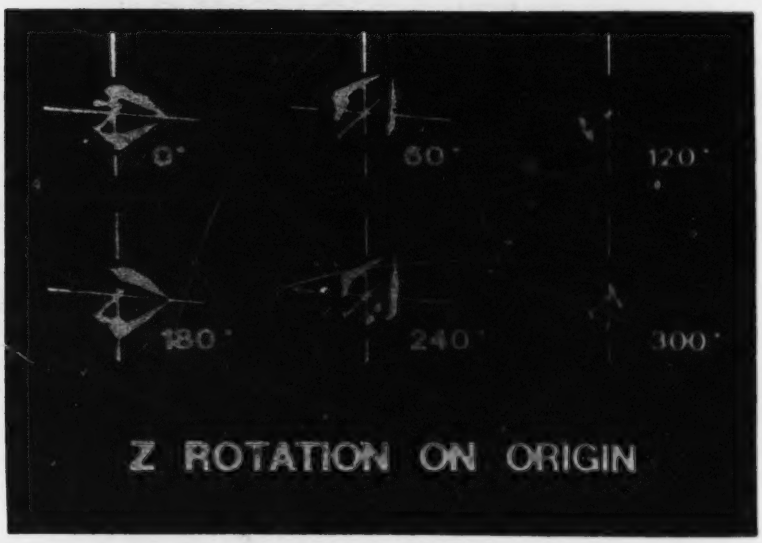

Figure 31. Z rotations

45

on its own axis, and an off-origin rotation corresponds to the earth orbiting around the sun. An object rotating around a point, not on its center, will rotate around the point as if in a circular path or orbit. Figures 32, 33, and 34 are examples of off-origin rotation in each axis.

It is important to realize that several rotations on-origin and off-origin can be going on at once increasing the complexity. Some systems even allow for rotations about an arbitrary axis.

A concatenated rotation is a rotation made up of more than one rotation occuring simultaneously. An object rotated in X and then rotated in Y involves two simple rotations one after the other. A concatenated rotation consists of a rotation in X and Y at the same time. Because the final orientation is the same, the order of the rotations is important even when rotations are concatenated. Keeping track of rotations can become very

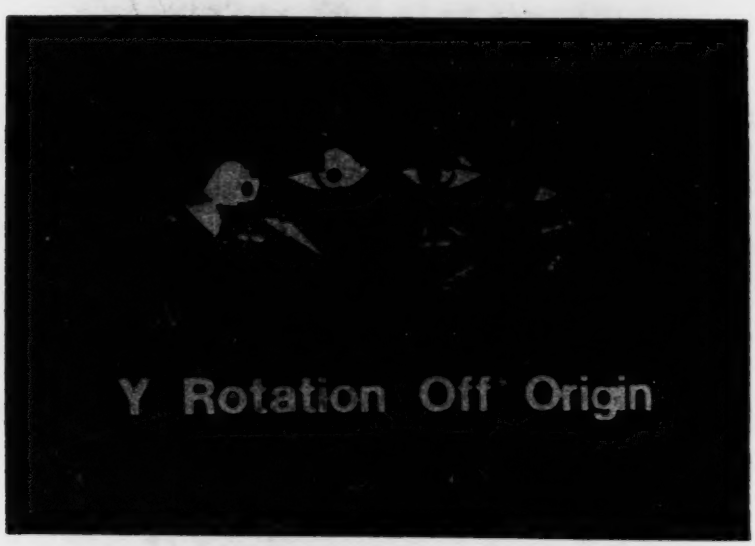

Figure 32. Off-origin X rotations

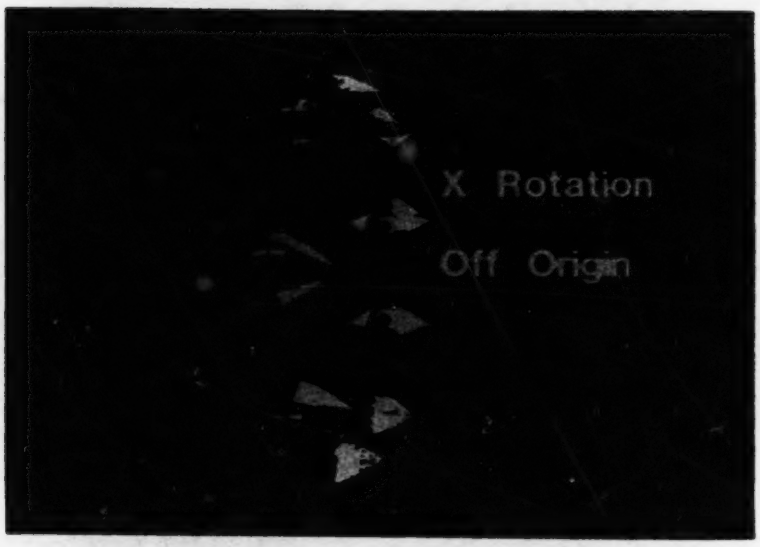

Figure 33. Off-origin Y rotations

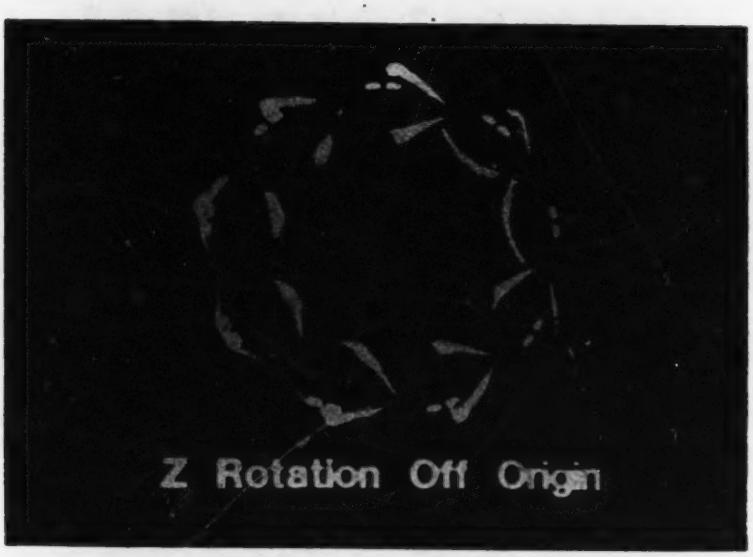

Figure 34. Off-origin Z rotations

complex and requires skill and a good sense of spatial orientation.

The order of rotation is very important. This is because matrix multiplication is, in general, not commutative. That is, (X * Y) may not be equal to (Y * X) when X and Y are matrices. Figure 16 shows the same object rotated first in X then in Y (on the left), and rotated first in Y then X (on the right). The degrees and axes of the rotation are the same for both images. Notice, however, that the final orientation in space is different due to the order of the rotations. The reader can demonstrate this principle by holding an object and performing these operations.

1.4. OBJECT OR SCENE DESCRIPTION

The basic operations of scaling, rotating, and translating can be used to create a scene as well as an

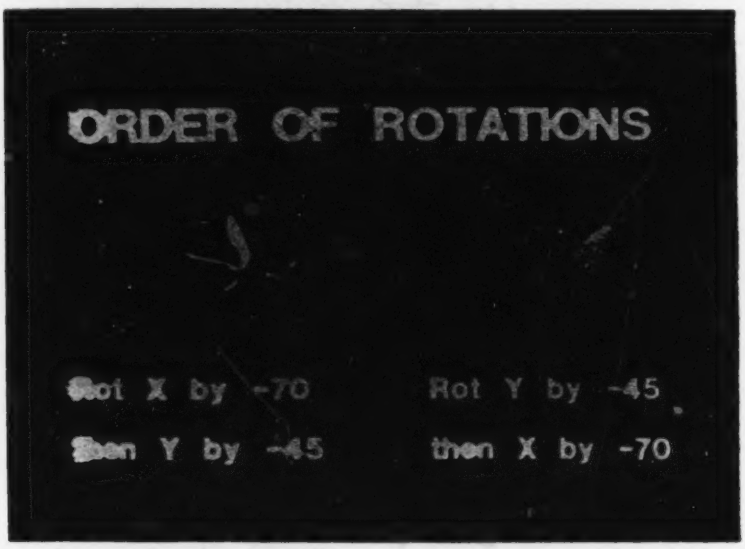

Figure 16. Order of rotations

object. One way to organize and plan a scene or an object is to create an orthographic projection of it. For example, to build a train out of geometric primitives, it is best to first create plans. Figure 36 shows the orthographic projection for a train made up exclusively of geometric primitives: a cone, a cube, a sphere, and a cylinder. Figure 37 shows the front view of the train and figure 38 is a top view. The numbers along the side provide a relative scale for the parts of the train. In order to scale the part correctly, it is merely a matter of knowing the bounding box and figuring the scale factor for each axis. The rotation can be figured by knowing the original orientation of the primitive and the new orientation of the part necessary. The translation necessary to correctly position the part of the train is easy to figure by taking the difference of where the original object is and where the new one has to be. Figures 39 through 41 demonstrate how this is done for one part of the train. In figure 39 the cylinder in the upper center is a scaled down instance of the original primitive in the upper right hand corner. Figure 40 shows the cylinder rotated to the

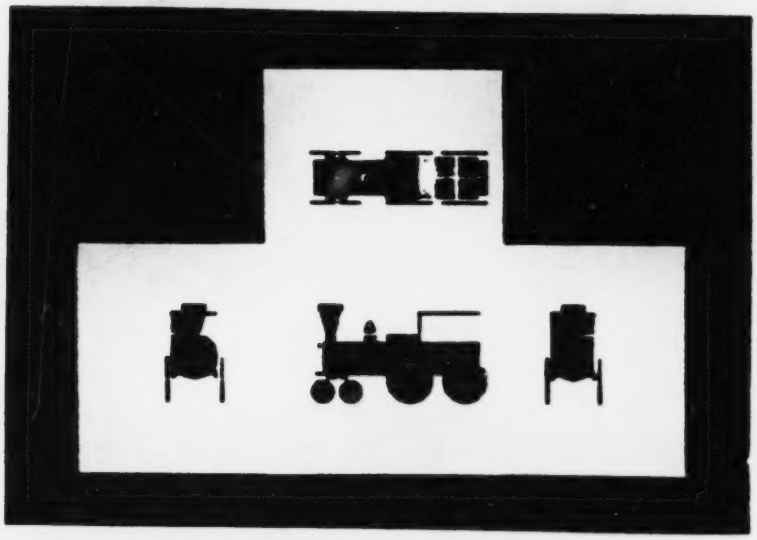

Figure 36. Orthographic projection plans for a train

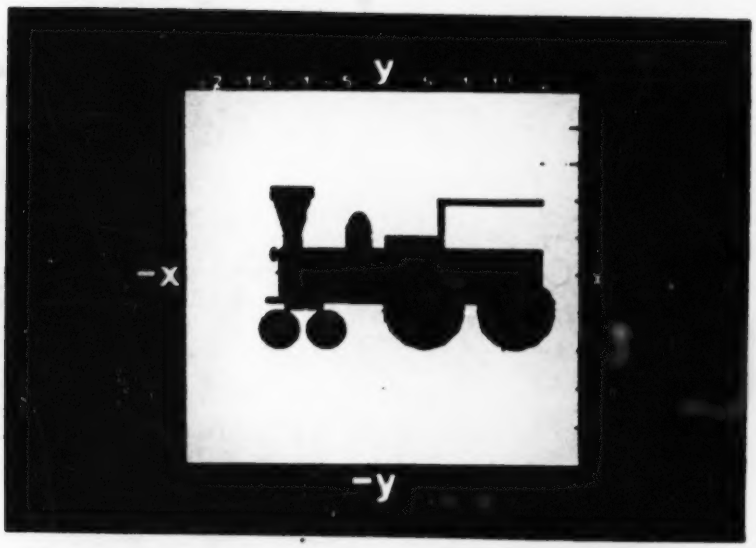

Figure 37. XY projection of train

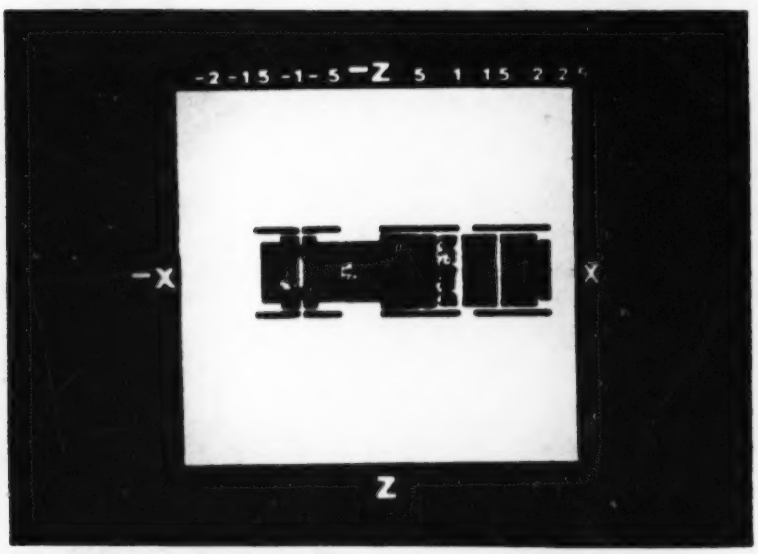

Figure 38. XZ projection of train

50

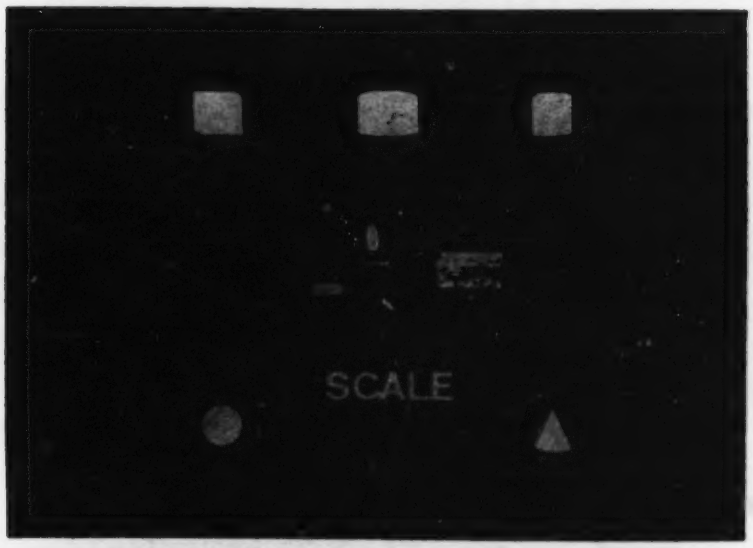

Figure 39. Scaled part of train

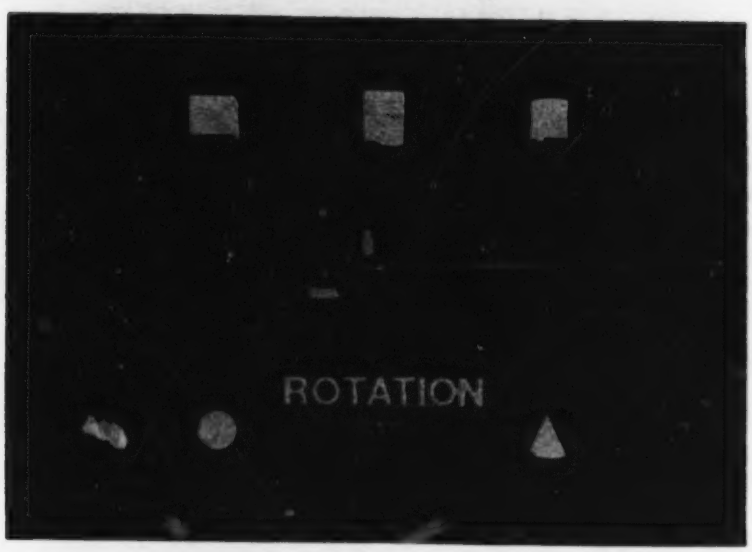

Figure 40. Rotated part of train

correct orientation in space and then translated to the correct position in figure 41. The completed train can be seen from several different views in figure 42.

Some graphics and animation systems allow objects to be attached to each other in hierarchical relationships. This option can be very powerful for setting up relationships among objects and maintaining these relationships. For example, we want to move the train that we just built. Figuring out the new position and orientation for each part of the train could be very time consuming, and of course leaves room for errors. If all of the parts of the train are attached to the cylinder we only need to move the cylinder to the new position or orientation. All the parts would maintain the same relationship to this cylinder and move as well. Attaching objects is not like permanently gluing them together. If this train is moved into a new position, of course we want the wheels to move with the rest of the train as well as rotate independently. By attaching the wheels to the cylinder and then rotating the wheels this effect can be achieved. Generally the order of attachments is very important and can

Figure 41. Translated part of train

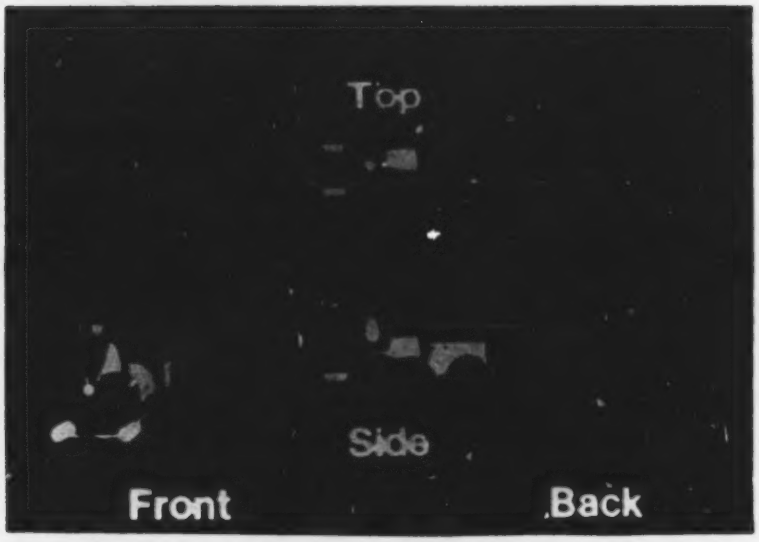

Figure 42. Completed train created by combining primitives

also become extremely complex.

3.5. COMPUTER ANIMATION

One advantage of the computer is its ability to deal with the complex relationships precisely and rapidly. The computer is able to repeat mathematical operations over and over again either identically or with small incremental changes. Besides specifying the parameters of the objects and the view of a scene, we can focus on the changes of these parameters over time for animation. Any parameter that can be controlled, can be changed for an animation sequence. In computer graphics we are able to manipulate and change the objects, the lights and the "camera".

The objects can be moved through the use of transformations, (scale, translate, and rotate). The principles of motion and tempo can be applied to these transformations. For example, scaling can be used to make an object grow larger, or shrink to nothing, or even turn inside out. Scale could be used rhythmically to simulate breathing. This process is done by incrementally changing the

scale value over a particular number of frames.

Translations can be used to organize and place objects in 3-D space. Incrementally placing the same object at various intervals can be used to build more complex objects, or to move an object to another position in space. By varying the distance between moves we can either speed up or slow down the motion. Depending on the system, the animator may be able to define a path for an object to follow. A path allows the artist to design complex motion through mathematical procedures. Computer graphics has some of the same problems as traditional animation. If the size of the move is too great it will create the effect of a visual stutter. Simulating motion blur is being explored in computer graphics as a possible solution to this problem.

Rotation, like scaling and translating, can be used effectively in animation. By incrementally rotating an object it can appear to spin, tumble, or roll. The speed of this motion again depends on the size of the increments. For example, if an object is rotated by .5 degrees per frame, there are 24 frames per second, it will turn 12 degrees a second, or revolve completely around twice in one minute of animation.

All of the above operations can also apply to objects that are attached hierarchically. We can build a bicycle out of separate parts. The wheels, the handle bars, pedals, seat, and rider can be attached to the frame of the bicycle. As we translate the bicycle frame down the street all of the parts come along. We can rotate the wheels and the pedals at the appropriate speed for the translation, and at the same time turn the handle bars. This motion of rigid articulated objects can become extremely complex, especially if we want to animate the rider.

Object attributes such as shininess, transparency, thickness, texture and color, (c.f. Chapter 5), can change from frame to frame. The background color can change slowly from day to night colors, or a dull opaque object may become shiny and transparent. Lights can also dynamically change. Their range of influence can be scaled incrementally, they can change colors, or even fly around influencing the lighting and color of other objects in the animation.

One salient feature unique to computer graphics is the ability to arbitrarily and weightlessly move the "camera" to views impossible to achieve in more traditional film media. We can simulate cinematic conventions or techniques, (e.g., a cut, pan, dolly, truck or crane shots, various camera lenses, and view angles).

The field of computer animation is still new and there are many problems to be addressed, but the potential is great. Once a model has been created it can be used and reused with many variations. If a scene is not correct, the artist does not need to redraw an entire sequence. Rather, some adjustments in the script can be made and the incorresct portions can be regenerated. There is a freedom of form, camera movement, and animation possibilities that no other medium provides.

CHAPTER 4

VIEWING THE MODEL

4. MODELING THE WORLD: THE ILLUSION OF DEPTH

The basic problem addressed by visualization techniques in 3-D computer graphics is depth cuing [46]. In Western civilization artists working in 2-D media, such as drawing or painting, have explored various techniques to create an illusion of 3-D space on a 2-D surface. Traditionally scale, position, overlapping, sharp and diminishing detail, converging parallels, and linear perspective have been used as visual cues for space organization [49]. For example, largeness of scale can be interpreted as nearness, and conversely, smaller scale is interpreted as spatial distance. The position of a form near the lower edge is perceived as near spatially and anything above the horizon line or the center is often interpreted as farther away [36].

Linear perspective is a geometric system which uses the spatial indication of size, position and converging parallels and converts size and distance into a unified spatial order as seen from one viewpoint. This visual logic of linear perspective can be programmed into the computer.

Overlapping planes or volumes are a powerful indication of space and take precedence over other depth cues. An object covering the visible surface of another object is assumed to be nearer [49]. In computer graphics, overlapping is handled by a hidden-line or hidden-surface algorithm. These algorithms eliminate surfaces or parts of surfaces that are occluded by another surface and not seen from the designated view. By eliminating these occluded surfaces and not seeing through an object to its back side, the image is less ambiguous.

Light is a significant indicator of the volume of an object in space. Light reveals the form and surface

qualities through a gradation from light to dark. Other surface qualities such as color, texture, and degree of reflection are the result of the play of light on the surface. Shadows indicate both the form of solid objects as well as position of the light source [3].

The qualities of light can be modeled in the computer in various ways. Diffuse lighting can be easily simulated in computer graphics. Some effects create more realism, such as cast shadows, transparencies, and reflections, but are often computationally expensive.

The construction of the human visual system does not allow the eye to see near and distant objects at the same time with equal clarity. This problem of "depth of focus" does not exist in computer graphics. It is possible to simulate this "depth of focus", in effect, but it usually is not done. Thus an object close up and one far away will both be equally sharp, but differ in size.

Painters have used a phenomenon called aerial or atmospheric perspective to give the illusion of depth to their work [36]. The effects of the atmosphere cause distant objects to appear flattened and to lose their sharp edges and surface details. Colors tend to be lighter, less brilliant or muted. The depth cue of atmospheric effects can be programmed with a haze factor or adjustment of colors in the distance. The techniques necessary to incorporate most of these depth cues into computer graphics algorithms are worked out [46]. However, when programming depth cues the scientist has to take into consideration the amount of computation time required.

4.1. VIEW TRANSFORMATIONS

Once all the objects are arranged as desired in the scene, a point in space, an eyepoint, must be selected to view the scene. The eyepoint is placed by locating it in the coordinate system through an X Y Z triple. The point in space at which we are looking must also be selected. This point is sometimes called the center of interest. It is placed by specifying its X Y Z position. These two points create a line of sight. The view angle, is the angle of view on either side of this line. Figure 43 uses the previous example of the train that we built and illustrates the location of the eyepoint, center of interest, and the defined view angle. The center of interest is

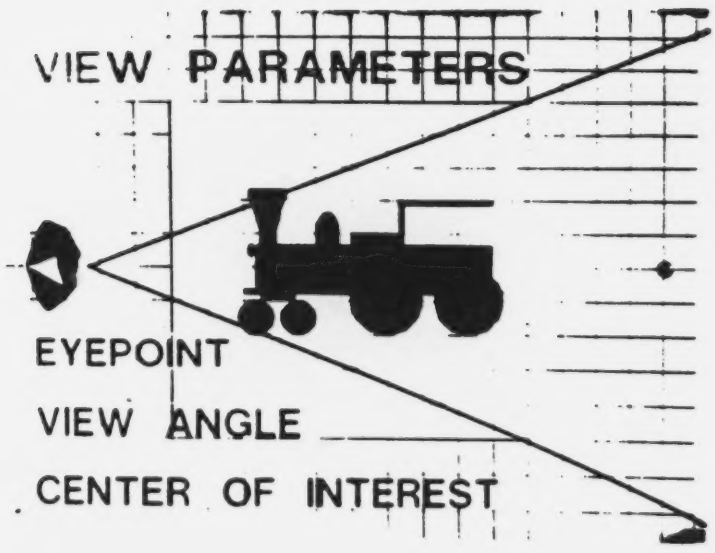

Figure 43. View parameters

illustrated as the small sphere to the right of the train. The eyepoint is at the apex of the view angle. The view angle in this case is 45 degrees. The line of sight runs down the center of the view angle. A view angle of 45 degrees indicates a 22.5 degree view on either side of this line of sight. Depending on the system, this view is either fixed or can be changed. It is important, however, to know what this angle degree is in planning a scene. The center of interest will always be at a point that would be the exact center of the display monitor.

Setting the view angle is similar to changing the focal length of a camera lens, in that the size of the angle controls how much of a scene will be seen in the display. A small angle is similar to a long focal length or a telephoto lens and yields a narrow field of view. Similarly, a large angle is like a short focal length or wide-angle lens, and it provides a very wide field of view. Anything outside the viewing field is not seen through the lens. In computer graphics any object or part of an object outside the view angle will be clipped and not displayed. The view angle actually defines a view pyramid that has a base with a four to three ratio the same as a display device. Figure 44 illustrates how the

top of the smoke stack and the bottom of the front wheels are outside the view angle. When the train is displayed from this view these parts will be clipped and not shown on the screen.

Transformations used in modeling and viewing are handled in two slightly different ways. When refering to objects and the transformations of scale, translation, and rotation we are working in world space. In this space we manipulate an object by changing its coordinates. View transformations are used to move a coordinate system that measures the position of objects [46]. View transformations can be thought of as global transformations, because they effect all of the objects in the scene.

Once the view parameters are defined, all the coordinates of the objects in the scene are transformed through matrix multiplication to line up with the eyepoint fixed at the origin. This new coordinate system is called eye space. Figure 44 illustrates how all of the points of the train are transformed to eye space with the line of sight down the center of the view pyramid on the Z axis. These

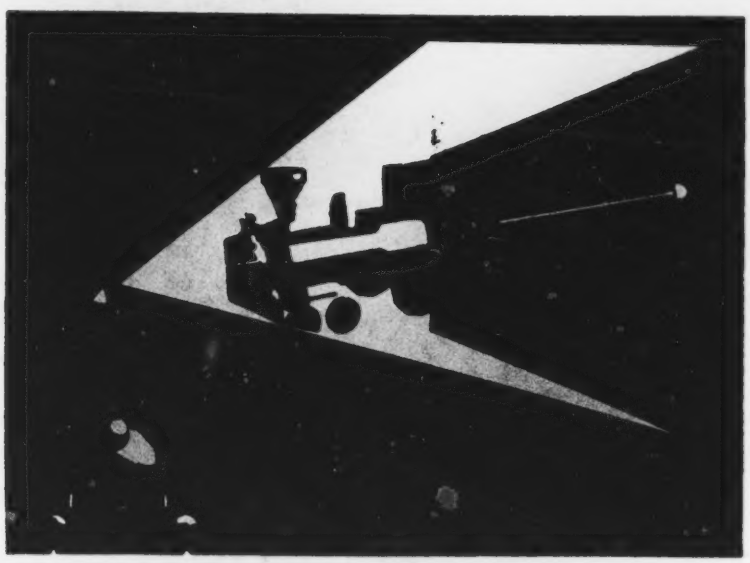

Figure 44. View pyramid and screen space

new X Y Z coordinates of the train are then projected onto a 2-D "picture plane" which is called screen space or image space. In screen space only the X and Y of the screen are dealt with, though sometimes the Z value is saved for use in determining which objects are in front of others for hidden-surface calculations.

There are various techniques used to project eye space points to screen space, (e.g., perspective or orthographic projection). The perspective projection is the most familiar, because it is remarkably similar to the photographic image and is frequently used in the visual arts. Figure 45 shows a plane in front of the train onto which these new eye space coordinates are projected. This projection is done by a line from the eye through each visible vertex of an object. The intersection of this line and the 2-D plane is the new X and Y coordinate in screen space. The image in the lower left corner of figure 45 is the resulting image as seen on the screen. Thus the 3-D eye space is transformed to a 2-D screen space and completes the view transformations [14,25,46]. After the view transformations are complete the 2-D image is scan converted and displayed on a raster display device.

4.2. HIDDEN-SURFACE REMOVAL

In the physical world, of course, we cannot see through solid objects. An artist creating a rough drawing or a perspective drawing may draw in all the objects and then rework the drawing by erasing lines that make the position of objects in space less ambiguous. A painter using an opaque medium need not concern himself with which object or parts of objects obscure others in his painting. The paint can cover these obscured areas. If using a transparent medium, such as water colors, the artist must plan the space and may paint the front objects first or paint around where they will be.

In computer graphics this opacity of solid objects must be simulated by mathematically removing areas that are blocked from view by other objects. In 3-D line drawing displays this process is done by removing the hidden lines. In 3-D shaded graphics this is accomplished by removing the hidden surfaces. There are various methods for solving this hidden-surface problem. The solutions require a great deal of computing time. There is a concern for efficiency, speed, accuracy, and the impact of

computing time and memory space on the computer in designing this process. As usual in computer graphics, there may be a trade-off between speed and accuracy, or between speed and storage space. There may be additional trade-offs between the quality of the image desired and the computing power available.

Basically, these algorithms use some method of geometric sorting to determine which polygons are visible and which are hidden by other polygons as seen from the chosen view. They also make use of consistencies inherent in the image, called coherence, to help speed up the computation time. All these algorithms are about the same speed for a simple image, but when the complexity of the image grows so does the calculation time [57]. There are several approaches to the hidden-surface problem. Among them are the Z-buffer algorithm, the scan-line algorithm, and the subdivision algorithm.

The Z-buffer, sometimes called a depth-buffer, is the simplest of these algorithms to implement and is currently the method most commonly used. This "brute force" approach is fairly fast, but requires a large amount of memory space, called a buffer. A buffer consists of fast memory with memory space for every picture element or pixel in the display. It is used to refresh the raster display at video speed. This buffer usually only contains the intensity value for every pixel, but in the z-buffer it must also contain enough memory to store the depth value as well. A Z-buffer algorithm computes the shading for each pixel in the polygon, as well as the depth at each pixel. Every polygon is sequentially sent to the buffer, and its depth is compared to the current depth at the pixel. If the current depth is closer to the eyepoint than the stored value, it replaces the one already saved. This is done over and over until there are no more polygons sent to the buffer. Some of the disadvantages of the depth-buffer are that it calculates the shading for every polygon even if it ends up being overwritten or is outside the view pyramid. It is very difficult to eliminate jaggies with a z-buffer.

A scan-line algorithms generates the image one line at a time in the same order that the a TV raster image is scanned in. It sorts the objects vertically and then only processes the polygons that lie within the particular scan line. If there is more than one polygon in a pixel area,

their depths are compared and the appropriate intensity is stored in the refresh buffer for display.

The subdivision method tries to concentrate the work on the parts of the image that are complicated. The image is subdivided until the image is simple enough to easily determine which object is in front and which is obscured before it is displayed.

Both the scan-line algorithms and the subdivision algorithms are designed for generating more information about surfaces than a Z-buffer. As a result, these algorithms can compute antialiasing and transparency. An algorithm has been implemented that can generate images which cast shadows using the scan-line method of hidden surfaces.

The hidden-surface problem is still a current area of research. Usually, the artist need not become involved in this area. If there is interest, a great deal of literature exists on this subject. Both Newman and Sproull [46] and Foley and Van Dam [22] discuss various methods. An excellent article that compares these methods is written by Sutherland, Sproull, and Schumacker [57].

CHAPTER 5

3-D REALISM

5. LIGHT, SHADING, AND COLOR MODELS

3-D objects do not seem real or solid without the play of light on their surfaces. After computing which surfaces are visible, the next step is to compute the correct intensity value for each pixel in the shaded image. The brightness of objects that are observed in the real world depends in a complex manner on how the human eye perceives light and color, the distribution of light in the total environment, and the object's physical capacity to absorb and reflect the light it receives. This property is called luminance or reflectance. This brightness or luminosity of an object will tend to appear as a property of the object itself [3].

Direct, head on lighting will tend to flatten a form. Artists, such as Carivaggio and Rembrandt, found that a raking or lateral light will reveal more information about the 3-dimensionality of an object. Controlling the steepness of the gradient in shading reveals the underlying form regardless of the local color of the form. Each form has a basic highlight and shadow pattern. A subtle gradation of value will reveal a gently curved surface, and an abrupt change in light and dark appears angular. This shading may either be attached or cast. While attached shadows lie directly on the object and are created by the play of light on the surface of a form, the cast shadows are thrown from one object onto another or onto the ground [3].

Artists have approached light in two separate ways, the reflected light of an individual object and the light of a scene. Some artists paint with a technique which utilized this idea. They paint the illumination and the color of an object separately. The underpainting is painted achromatically in white, greys, and black, creating the shading gradients. Local colors are then applied

with colored transparent glazes over the top of the underpainting. Computer graphics uses a similar approach, in which the light reflection model calculates the luminance or the distribution of reflected light of an object and then applies this calculation to the color assigned to the object.

5.1. LIGHTING MODELS

Computing the shading gradient in 3-D computer graphics is the most time consuming and calculation intensive part of the entire process of displaying an image. It may use from 40 to 90 percent of the calculation time depending on the complexity of the lighting model. The model used for computing light and shading is an approximation of actual lighting conditions. There is a compromise between precision and computing cost.

The shading model is made up of three ingredients: the properties of the illumination falling on a surface, the brightness, and the degree to which the surface effects the light. The first area to be covered is the illumination model. This simulates what artists call attached shadows [46].

Three properties of light modeled in computer graphics are ambient light, diffuse light, and specular reflection [22,6]. Ambient light is the light bounced off of the environment and off of particles in the atmosphere. This seems to create lighting that comes equally from all directions. Ambient light is the easiest light source to model, because it produces a constant illumination on all surfaces. However, it appears very unrealistic in its effect. Ambient light is usually used as a value which is added in after the diffuse shading calculations are completed.

Modeling diffuse light depends on a light or multiple lights coming from a specific direction, which illuminate a matte surface in varying degrees. The illumination depends on the orientation of the surface to the light source(s). In the computer, diffuse light is modeled as a point light source. This point light source has a radius within which the light has an influence over objects. The light itself is not seen as is the sun, but the effects of the light on objects are modeled. This light source is positioned in space by specifying an X Y Z coordinate. An

object outside the realm of any light sources would appear to be a black shape. Usually the influence of the light source is scaled to a very large size and is placed a great distance from the scene. In effect the distance is that of the sun to the earth, and the light rays are almost parallel [13].

Diffuse illumination is light reflected off an object and is scattered equally in all directions. It is computed as an intensity value which is determined by the position of the light source and the polygon orientation. This is calculated using Lambert's Law for the reflection of light from a perfect diffuser. The shading gradient is calculated by computing a vector from the light source to the surface normal vector and then calculating the angle between these two vectors. A cosine function is then used to compute the gradation from light to dark for each surface or polygon [22]. This value is between one and zero. The closer the angle is to zero, the more closely these two vectors coincide, and the closer this cosine value is to one. Conversely, the closer the angle is to 90 degrees the closer the cosine value is to zero.

The resulting cosine value is used to scale the color value assigned to an object. The brightest face of an object would be the closest color to the original color assigned to that object, because the color of this face has been scaled down the least. Conversely, the more a value is scaled down, the closer to black it will become. Figure 45 illustrates the relationship of the light to the face normal and the resulting shading.

Specular reflection causes the bright highlights seen on a shiny surface. In looking at a ceramic coffee mug the highlights are caused by specular reflection, while the reflected light on the rest of the mug is caused by diffuse lighting [22].

In order to calculate specular reflection, the position of the eye is included in the intensity calculation. The incident ray from the light source to the surface is reflected off the surface at the same angle as it came in, but in the opposite direction. If this reflection is very close or identical to the vector from the eyepoint to the surface, then specular reflection is seen. To simulate this reflection, the angle between the vector from the eyepoint to the surface normal and the reflected light ray

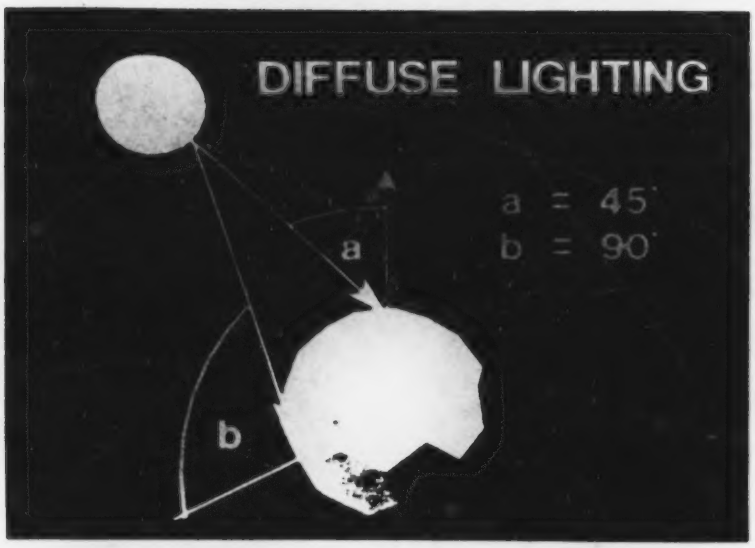

Figure 45. Diffuse light model

from the surface normal is computed. The cosine of this angle is used as a scalar to bump the intensity value to a value brighter than the base color of the object and thus model specular reflections. The light for specular reflection falls off very rapidly as the size of the angle gets larger. Specular reflection can be more easily understood by examining figure 46. It shows the relationships of these angles. In this example the eye vector and the reflected ray are identical. In some systems, the specular reflection can be controlled by scaling the influence of the cosine value on the size of the reflection. This is sometimes referred to as the shininess or gloss of an object.

All these lighting models have a different effect on the object's appearance when it is displayed. Figure 47 shows the effects of one or more of these lighting models on a sphere. The upper left demonstrates that an object will be completely black if it is not within the range of a light source. The upper right shows the effects of diffuse lighting. The lower left shows the combined effects of diffuse and ambient lighting. The lower right illustrates the effects of diffuse, ambient, and specular reflection.

Figure 46. Specular reflection lighting model

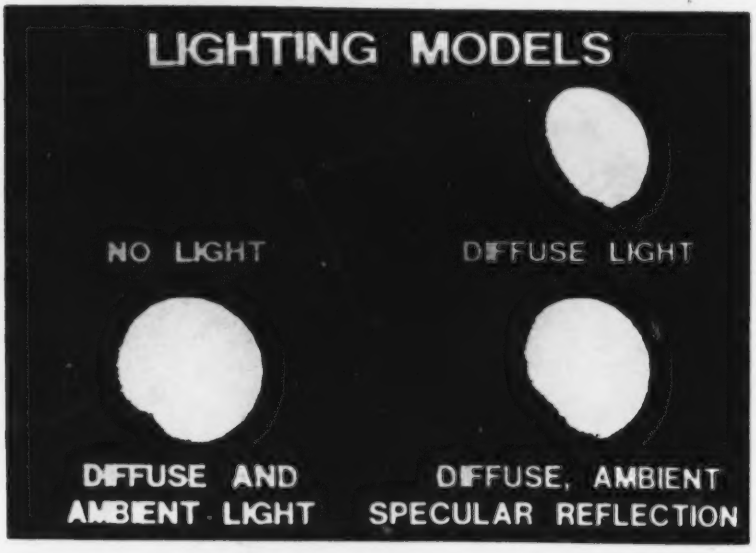

Figure 47. Lighting models; diffuse, ambient, and specular reflection

The discussion to this point has centered on the simulation of an attached shadow. The problem of computing cast shadows is much more complex. Currently cast shadows are seldom computed due to the length of time it takes to calculate them. There are, however, several methods used to accomplish this effect. An algorithm which computes the shadows cast by a point light source is analogous to a hidden-surface algorithm. The hidden-surface algorithm determines which surfaces can be seen from the eyepoint. The shadow algorithm determines those seen from the light source. Any surface that is visible from both the eyepoint and the light source is not in shadow. Those surfaces not visible from the light source and visible from the eyepoint are in shadow. The results are then combined and displayed [22].

Transparency can be simulated by modifying the hidden-surface algorithm. Instead of just considering which surfaces are visible, all the surfaces are considered. The percentage of incident light which passes through a surface is call transmittance. Transparency can be calculated if the transmittance along with the intensity and depth information is saved. This process is accomplished by starting with the front surface and working away from the eyepoint as each surface is blended with the one(s) in front of it [13]. The percentage of each surface color is determined by the product of the transmittances of all previous surfaces. The surfaces are considered until the transmittance falls to zero. In order to simplify the model, the refraction of light is generally not modeled.

The "thickness" of the transparent object must be modeled, in order to create a realistic model of transparency. The curved surface of a glass ball appears less transparent on the edges. This is because the properties of transparent materials reflect and refract light, which obscures what is behind them. To simulate this effect in computer graphics, the transmittance of the surface has to vary with the orientation of the surface relative to the eyepoint. A cosine function is used for the calculation as it is with luminance. Figure 48 show the effects of transparency and the relative "thickness" of the material. The sphere in the upper left is opaque, the sphere on the upper right is 20% transparent or has a transmittance of 20%. The two spheres on the bottom have a 95% transmittance, but the value used to indicate the thickness of the

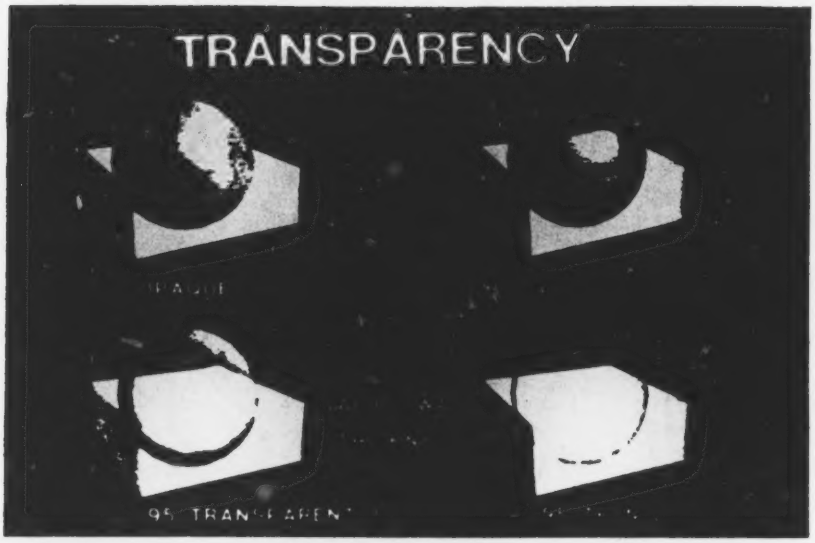

Figure 48. Transmittance and thickness

object has been varied. The reader can see the resulting difference of controls over the transparency and thickness of an object. Transparency often greatly increases the time it takes to calculate an image.

5.2. SHADING MODELS

The light models discussed need to be applied to polygonal objects. There are various methods used to determine how these lighting models effect the form of the object. The simplest way is to compute only one intensity value for an entire face. This method is called faceted shading. It gives an object a chiseled appearance as in cut gem stones. The effect of faceted shading accentuates the polygonal structure. With faceted shading the intensity is computed by using the face normal and its angular relationship to the light source. A simple cylinder is used to demonstrate the relationship of the normal to the face and the resulting shading model in figure 49.

Smooth shading is a method of approximating a curved surface, even though the data is polygonal [25,46]. Gouraud shading and Phong shading are two commonly used models for smooth shading. Gouraud shading is a fairly

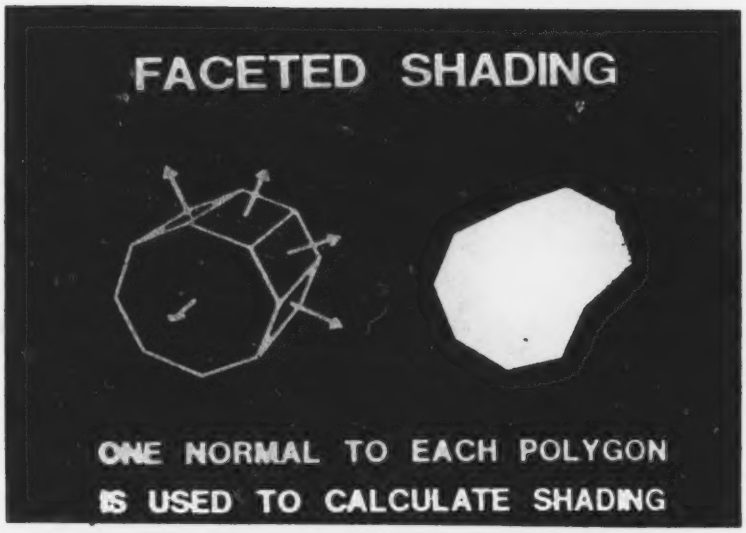

Figure 49. Faceted shading model

fast approximation of smooth shading. The normal is calculated for each vertex, rather than for each face. The intensity is then linearly interpolated along the sides and across the surface of the polygon [13]. Figure 50 illustrates Gouraud shading. Notice how this simple cylinder has changed from that in the previous figure. Smooth shading has a tendency to turn a cube into a marshmallow. This results because the vertex normals are an average of the normals of all the adjacent faces. When a crisp edge is desired, it may be necessary to duplicate the points in that polygon or along one edge of the polygon. These extra points cause the normal at a vertex not to be averaged, because adjacent polygons do not share the same vertex. Figure 51 shows the difference in the normals and the cylinder when the points are doubled on the end polygons. This is useful to know, because the desired effect may be a smoothly shaded curve around the sides of the cylinder, while the ends need to maintain a flat appearance.

Phong shading is a closer approximation to reality. This algorithm computes a normal at each pixel to obtain the intensity and includes the calculation for specular reflection. To visualize what the normals look like,

70

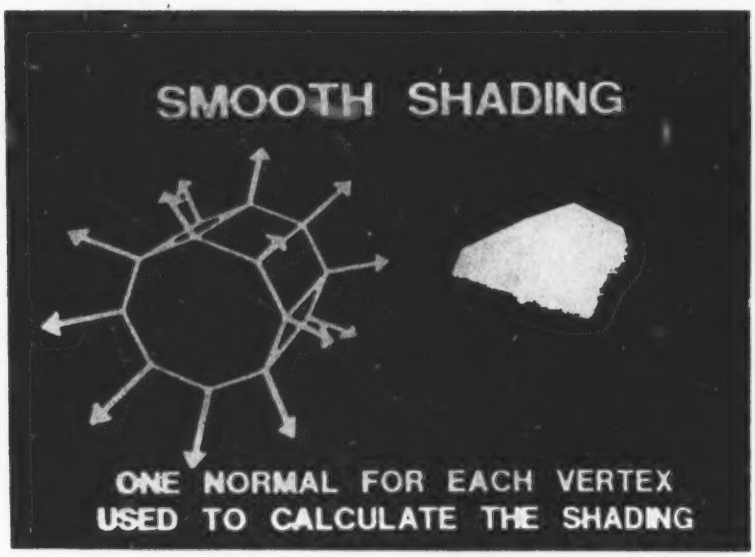

Figure 50. Gouraud shading model

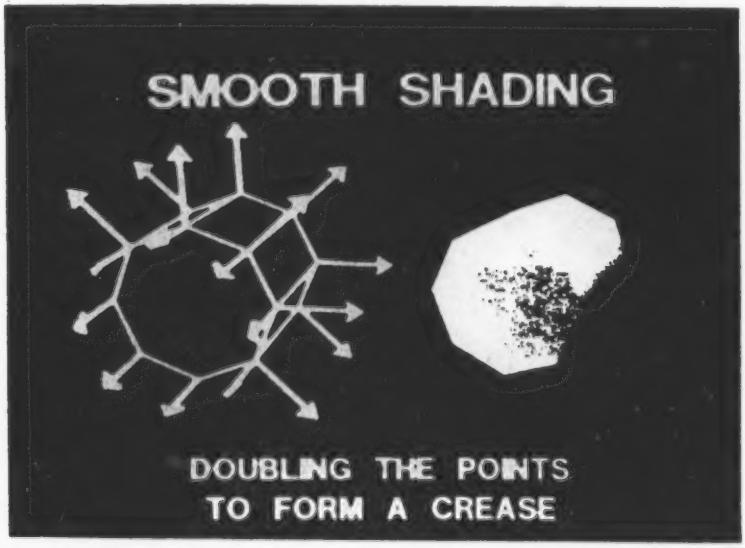

Figure 51. Gouraud shading model, points doubled

imagine a pin cushion with a pin at every pixel. It is actually too complicated to display for a figure. This

method is computationally more expensive, but produces a more realistic lighting model.

Ray tracing is a more realistic lighting model. The light is modeled by tracing the light ray from the light source to the surface of an object. To make the calculations more manageable the light is traced backwards from the eyepoint to the surface and then to the light source. This calculation is done for every pixel [25,22]. Ray tracing is extremely expensive to calculate. However, the resulting image is impressive for the quality of realism of reflections, transparencies, and refractions of light that can be obtained. Figure 52 is an example of ray tracing.

Depending on the system, the artist may have access to more than one light source. He may even be able to scale the influence and color the light source(s). He may

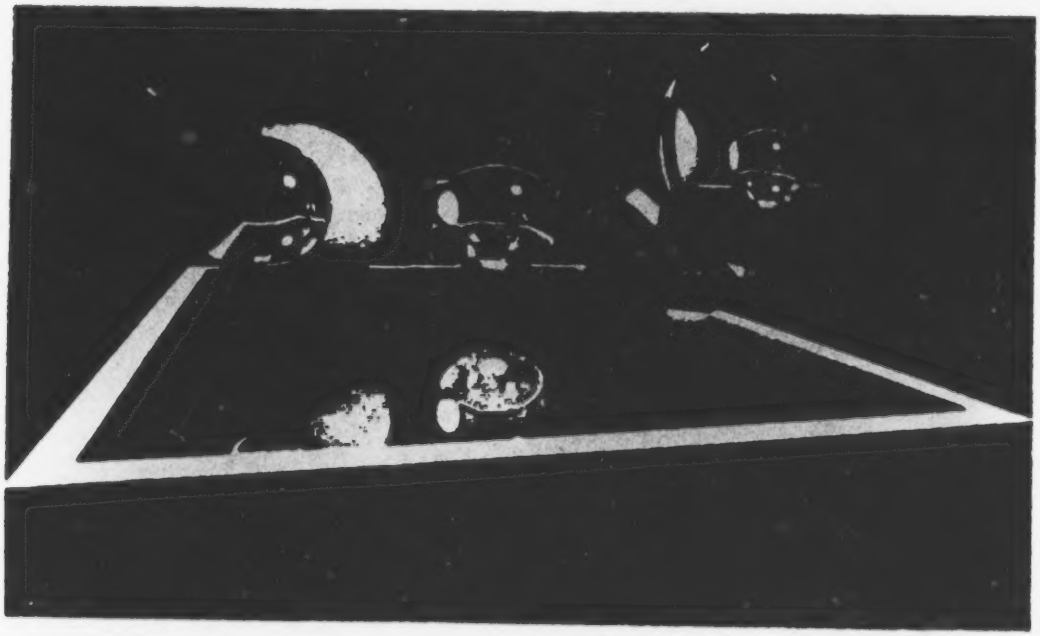

Figure 1. Ray tracing by Bob Conley, Ohio State University

be able to render transparent objects. There may be some qualities that are difficult to model realistically, depending on the light model (e.g., soft material, metallic surfaces, semi-gloss surfaces, or rough surfaces). There are some controls that are not modeled by the point light source, such as a controlled directed light or a visible light. A lighting model for directed light providing effects similar to stage lighting or lights set up by a photographer could prove exciting. David Warn [58] describes a model for lighting controls, such as spotlights or floodlights, and flaps or cones to restrict the path of light. New lighting models such as Warn's, that give the artist more control, are only in the developmental stages. However, when they come into wider use it will add a great deal to the potential of image making in computer graphics.

As with hidden-surface algorithms and view transformations, these light and shading algorithms are transparent to the user. An understanding of the principles used make it easier for the artist/user to achieve the desired effect when placing the light source and coloring the objects in a scene. If the reader is interested in understanding this in more depth there is a great deal of literature on the subject of lighting models.

5.3. COLOR

The output medium in computer graphics is most often a color television set. In this medium the artist does not mix colors as pigments but as light. Pigments used for painting are absorptive colors and their mixtures are governed by the rule of subtraction. In computer graphics color is transmitted as light. Color mixing is governed by the additive color system. Mixing color in computer graphics requires an understanding of light, the additive system, and the color model used in computer graphics.

Color is an extremely complex subject that deals with physics, psychology, physiology, perception, chemistry, and in art, a philosophy of color. Two basic approaches can be used to understand color phenomenon: the psychological and aesthetic approach and the technical approach [44]. In art we approach color psychologically and aesthetically. Color theory is based on perception and the visual properties and relationships of color in an image, as well as its emotional and psychological effects.

Color in art is subjective and relational which makes it difficult to measure. The skilled use of color involves aesthetic choices on the part of the artist. Aaron Marcus [44], a computer graphics artist and designer wrote:

> the most that can be said of much computer graphics is that it is colorful; however, it appears to reflect a lack of serious decision-making about color.

As Marcus points out, an expertise in color theory is important to the artist in computer graphics as in other forms of visual communication. If the reader is not familiar with color theory there is a great deal of literature in the arts relating to this subject, Albers [1], Itten [30], Ocvirk [49], Arnheim [3], and Parker [54].

A more technical approach is required to understand color as it is modeled in computer graphics. Color results from light waves which are electromagnetic energy. The human visual system can perceive wavelengths from about 400 to 700 nanometers. This range is referred to as the visible spectrum: red, orange, yellow, green, blue, indigo, and violet. There are three basic properties of color that are modeled in all systems: hue, saturation or intensity, and brightness or luminance. Hue is used to designate the common name of a color and to indicate its position in the spectrum, (e.g., red, blue, orange). Intensity refers to the purity of a color, or how little of the color is diluted by white light. Intensity distinguishes powder blue from ultramarine blue. Brightness refers to the achromatic notion of intensity which ranges from black to white. It is dealt with as an independent factor from hue and saturation [22].

The output medium in computer graphics can be an NTSC or an RGB (red, green, blue) color monitor. Usually, high resolution raster graphics use the RGB monitor, therefore this will be the one discussed. The RGB color raster display device has individual video inputs of red, green, and blue. It uses three separate input signals to control the three guns which excite the tiny red, green, and blue phosphor dots that make up each pixel. Each gun is individually controlled for intensity. The eye blends these dots into the visible colors of the spectrum when the

observer is a certain distance away from the screen [25]. This color phenomenon was used by the Impressionist and Pointillist painters who strived to depict not only the phenomenon of color but also light in their painting. They applied tiny dots of pure paint colors to their canvas, and let the eye of the observer blend the colors into a full and vibrant palette.

The choice of these three particular colors is based on the trichromatic color theory in photography [29,42]. Our color vision is made up of three light-sensitive pigments which are sensitive to red, green and blue light. Since the sensitivity curves of these overlap, we are able to see a very wide range of colors. The colors we perceive are a polychromatic phenomenon, not monochromatic or of a pure wavelength. Any three colors in the spectrum can be used to produce other colors, but red, green, and blue can produce the widest range of color. For this reason most color display systems are based on red, green, blue (RGB).

As a result of working directly with light in computer graphics, color mixing uses the <u>additive color</u> system, rather than the <u>subtractive color</u> system used when painting with pigments. With pigments the color we see is the color of light that is not absorbed by the paint but reflected off the surface after absorption. With additive we see colors that are mixed light or wavelengths.

The differences between these two systems presents some difficulty in the transfer of skills and understanding for the artist as he moves into computer graphics. The previously learned concepts of color relationships hold true in the computer graphics medium, but color mixing skills do not. If an artist has had experience with lighting for theater for instance, additive color does not seem as foreign.

The subtractive color system used in painting is based on red, yellow, and blue pigments as the primary colors. The secondary colors are orange, green, and violet. Most painters know that in theory these secondary colors can be mixed, but more often they are obtained and used to extend the range of the palette. White is added to "tint" the color; black is added to to create a "shade"; and both may be added to create a "tone". The absence of any color creates white and all the colors

mixed together produce black. The more color that is mixed together the darker the new color will be.

Red, green, and blue are the primaries in the additive color system used in computer graphics. Only these three colors and no others are available. All other colors are a mixture. We can think of these three colors as separate beams of light. Figure 53 illustrates these three beams of light overlapping. Notice that the secondary colors are yellow (red + green), cyan (green + blue), and magenta (blue + red). When the primaries are all mixed equally white light is produced. When none of these are present the result is black. The complimentary colors are any two colors that make white light when mixed together. Notice that mixtures of the primaries become lighter in value as more light is added until white is created by mixing all of them.

Color in computer graphics is used to "paint" objects and the background. As discussed above in lighting models, the color assigned to an object is really the brightest value for that object. To "paint" the

Figure 53. Light primaries

background a color is specified, and this color is used to fill the frame buffer. Later as other objects in the scene are displayed this value is covered over or replaced. This gives the artist a base color against which other colors will be seen, and can then be taken into consideration in design. Some systems allow the user to blend background colors for a more interesting effect. A previously created scene that has been saved may be used as a background as well.

5.4. COLOR MODELS

The RGB color model uses the Cartesian coordinate system as a unit cube. This color cube has black at 0 0 0 and white at 1 1 1. From figure 54 we can see this color cube. In this system we can specify a color by specifying an RGB coordinate triple. Below is a list of the coordinates of the primary and secondary colors which are at the corners of the cube. Notice that a diagonal from 0 0 0 (black) to 1 1 1 (white) is the achromatic intensity range.

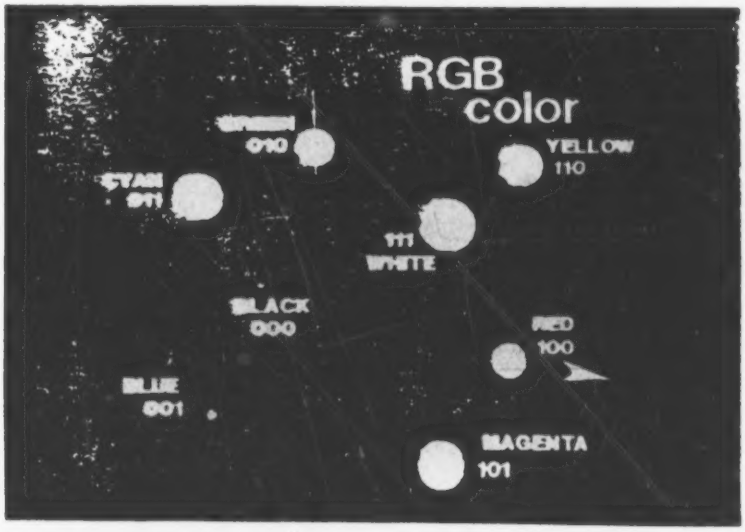

Figure 54. RGB color space

	R	G	B
red	1	0	0
yellow	1	1	0
green	0	1	0
cyan	0	1	1
blue	0	0	1
magenta	1	0	1
white	1	1	1
black	0	0	0

Colors are specified numerically with a value for each RGB component. The higher the number the brighter the color. The highest value is one. If we think of the RGB system as a coordinate system then it seems obvious that we can scale this system. A very bright saturated red would have the value of 1 0 0 and a darker saturated red could have the value of 0.2 0 0. Figure 55 illustrates a color cube with each side divided into a 10 X 10 grid. Each small square is the color value of its

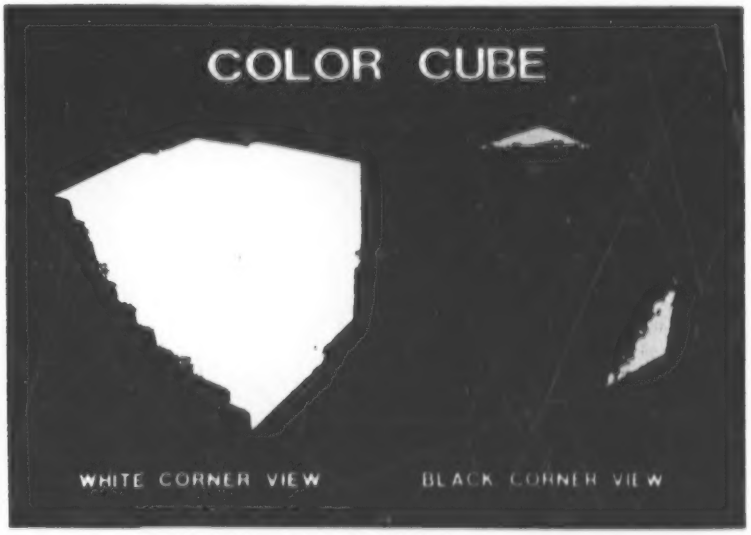

Figure 55. Color cube

coordinate in space. In figure 56 the brightness is lowered or in effect the RGB coordinate system is reduced in scale

Due to the problems users have adjusting to RGB color, other systems have been devised to make interaction with the computer easier. Joblove and Greenberg [31] wrote an interesting article discussing color spaces and Foley and Van Dam [22] have an excellent chapter discussing color in more depth. Foley and Van Dam discuss various color spaces. The CMY (cyan, magenta, yellow) model is a subtractive model that allows the artist to interact with color in a manner similar to previous experiences. The YIQ is used on normal broadcast television where the intensity, or Y axis of the CIE's X Y Z primaries, is a separate value from the hue. This separation of hue and value allows it to be shown on black and white televisions and still maintain appropriate value contrast. The HSV (hue, saturation, value) and HLS (hue, lightness, saturation) models are are based on the Munsell system of color.

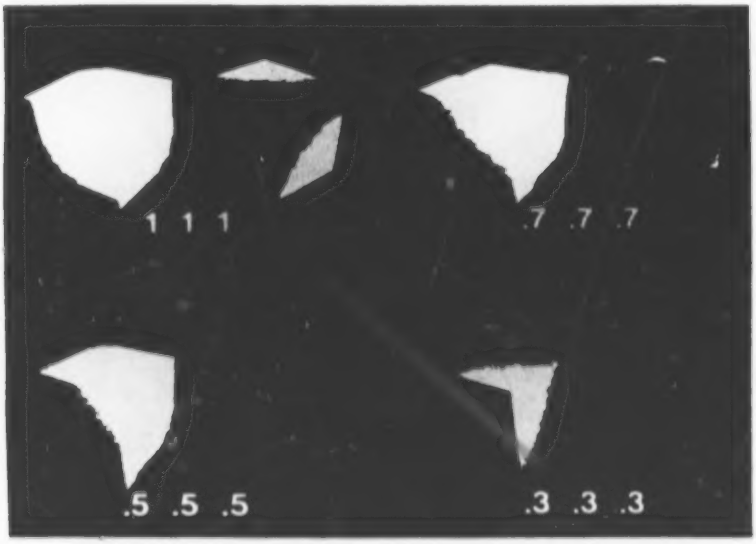

Figure 56. Scaled color cube: intensity lowered

The HSV is based on the artist's previous experience of tint, shade, and tone. This model is designed in a subspace of a single sixr-sided cone or hexcone. Black is at the point of the cone and red, green, blue, cyan, magenta, yellow are at the six corners and white is in the center of the base. It is likened to looking at the color cube from the white corner as in figure 54 and considering all the colors but black in the same plane. Adding white then corresponds to decreasing S and adding black corresponds to decreasing V. Hue is specified as an angle starting with red at zero and in a counterclockwise order yellow, green, cyan, blue, and magenta.

In the HLS model, color is modeled as a cylinder or a double hexcone. The hue is specified as a given angle of a circle around the vertical axis and lightness ranges from black at the bottom to white at the top. Saturation (S) is specified the same way on a vertical gradient. The hue is specified the same as in HSV except the purest hue is at 0.5 rather than at 1.

Both the CMY, HSV, and HLS systems can be easily converted back to RGB to accommodate hardware. They however, allow an alternative interface for the artist that can be easier to learn and use. CNS (color naming system) [4] has an interface, in which an "English" term can be used to refer to a color, such as red, blue-green or blueish green with 31 hue names possible. The achromatic colors or lightness can then be designated by black, very dark grey, dark grey, grey, light grey, very light grey, and white. The saturation can be specified by greyish, moderate, strong and vivid. A system like this has been found easier to learn and makes it easier to specify a color which was previsualized.

Sometimes the color model is completely transparent to the user. In this case a palette is displayed on the buffer and colors may be selected and/or created interactively. This approach is often used in a 2-D paint system where the artist interacts with the color system as they would in any painting medium. They choose and mix colors visually.

The color of an object is used in the calculation for illumination and shading, as mentioned earlier. Whichever color system is used, when designating a color for an object, one color is used for the entire object.

Sometimes this is not satisfactory. For example the "eye-mouth" object used to illustrate rotations would have to be made up of 4 objects to get the color effect desired. The lips and eyelids would be one color; the whites of the eye and the teeth yet another, the iris another, and the pupil another. An easier way would be to only have one data set and be able to color each polygon the color desired. This is sometimes referred to as polygon colors. Another example could be a chess board where the only piece of data needed is a grid. However the individual color of each polygon needs to be controlled. This gives us a very clear hard edged color area. Sometimes, as with faceted shading, this is not the desired effect. A subtle gradation from one color to the next may be desired. As the reader recalls, Gouraud shading uses a normal at each vertex instead of using one normal to a polygon. This same principle can be applied to color. Assigning a different color at each vertex achieves a smoothly shaded object with a changing gradation of colors. This is referred to as vertex colors. Some example of vertex colors can be seen in figure 57.

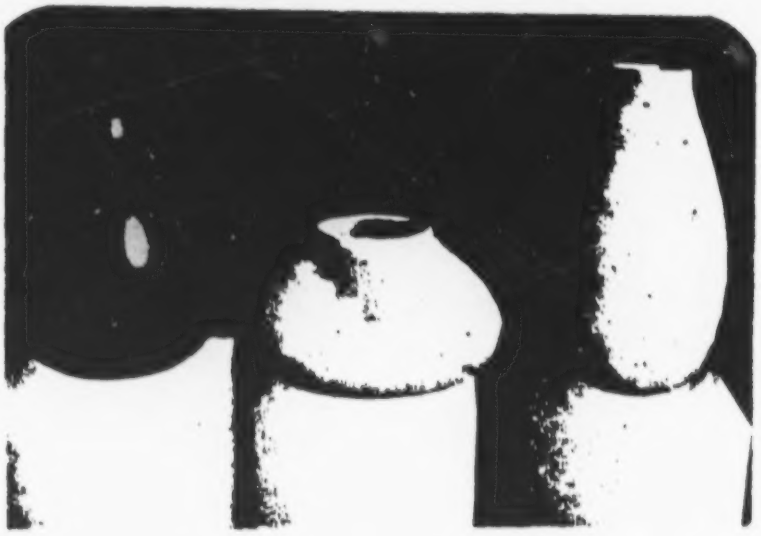

Figure 2. Vertex colors by Mike Collery, Ohio State University

The artist has several adjustments to make in using color on a computer. The color is specified with numbers and the color mixing is additive. Color in computer graphics can vary from system to system due the particular phosphors used, the software used, and the depth of the buffer. The artist has to practice using a particular system and a particular display device to be able to anticipate hue and saturation when mixing colors. The use of colored light or lights can drastically change the hue of a scene as well.

5.5. SURFACE QUALITIES

As discussed earlier, the shading model is made up of illumination falling on a surface and the surface qualities. Now that we have a general idea of illumination models and the problems of complex data, creating data complex enough to seem textured is clearly a problem. One solution to this problem is to create simple data and to only simulate the surface quality by creating a texture pattern. This is done by a process called texture mapping. The process is very similar to projecting a slide of a texture onto the surface of an object. In this way we can still see the basic form of the object, but also have the benefit of a textural quality. Texture mapping can be used to vary the intensity and color or the surface texture [25]. One consideration important in animation is that the projected texture move with the object as if they are one. Texture mapping can be broken down into two processes: first, the creation of the texture to be mapped, and second, the process of the mapping.

The texture to be mapped can be created by painting a texture with a 2-D paint program, by displaying an image in the frame buffer, by scanning in an existing object or image with a digitizing camera, or by generating the pattern mathematically. Depending on the system, this image or series of images used for textures may be stored as files or taken off another buffer. Basically this texture can be "attached" to an object at the vertices of polygons. Therefore, we have a choice to either spread the texture over the entire object or to repeat it a specified number of times. To help visualize this process we can see a sphere in figure 58 on the upper left onto which, the texture will be projected. This texture will be mapped one texture pattern to one polygon. First the sphere is "unwrapped" into a flat plane and the texture is

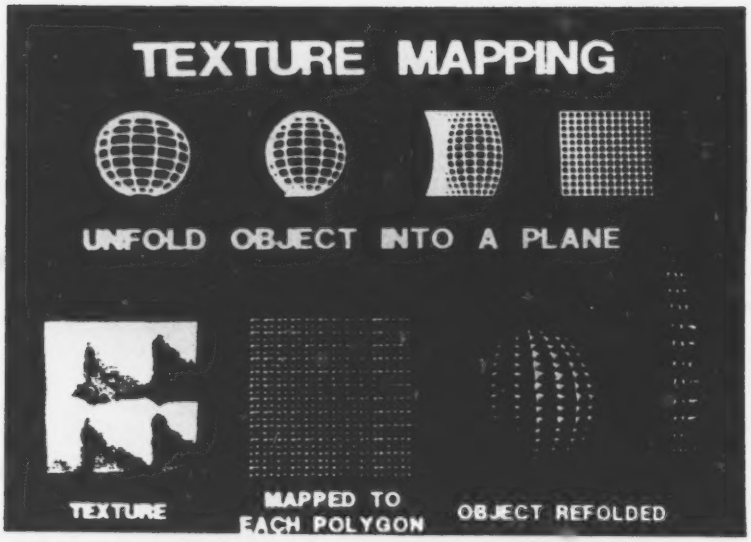

Figure 58. Texture mapping

projected onto each polygon. This process can be seen across the top of the figure. Then the sphere is wrapped back up into its original form with the texture mapped to each polygon. Below on the left is the original texture to be mapped. Then the sphere to the right has the texture mapped to each polygon. On the far right the sphere is scaled to simulate an ear of corn.

The use of texture mapping allows the artist to simplify the data and to map the surface details onto the form. For a more technical explanation of this process the reader is referred to the article by James Blinn [5], where he discusses mapping textures and reflections on the surface of objects.

Mapping a pattern of texture affects the surface intensity and color, but not the surface shape or geometry. Another method of achieving texture is to create fractal surfaces. Fractals can be used to model terrain, coastlines, tree branches and other natural and artificial systems of great complexity. One technique involves an object made of quadrilateral polygons (4 sides). Each polygon is subdivided into smaller quadrilaterals. The new points are randomly perturbed. This

method is applied recursively to create irregular surfaces from the original data set. If the reader has seen "Star Trek II", the mountainous terrain was created using fractals. Several examples of fractals can be seen in figure 59.

5.6. IMAGE QUALITY

The raster display device used in shaded raster graphics was briefly mentioned. The intention here is not to discuss the hardware involved in computer graphics, but the implications of that hardware on the image quality. In addition to the software used to display an image, the frame buffer can greatly influence the quality of an image. Several problems in image generation are due to the nature of the display device and its particular limitations.

The quality of an image is in part dependent on the memory size of the buffer. A buffer is usually discussed as the number of pixels horizontally and vertically and the number of memory bits for each pixel. Resolution is

Figure 3. Fractals by David Zeltzer, Ohio State University

determined by the number of pixels per scan line. This number can range from 64 to 1024. A buffer with at least 256 pixels per scan line is required to do high resolution images. The pixels are usually arranged in a ratio of 4/3 (4 pixels horizontally to 3 pixels vertically) to obtain square pixels. The figures used to illustrate concepts were created on a buffer with the resolution of 640 by 480.

One problem in a computer generated image is the "jaggies" or "staircasing" referred to as aliasing. The weaver has the same problem at the edges of color blocks where the threads create a stair-step effect. This is especially true in the case of a diagonal or circular form. This becomes less noticeable the finer the threads and becomes much more obvious the thicker the threads. This same principle holds true in computer graphics. The more pixels used to make up an image, the smaller the pixel, and the jagged effect is less noticeable. Higher resolution is a partial solution, to minimize the effect of "jaggies". This process of minimizing the effect of "jaggies" is called anti-aliasing. A better approach, but a more costly solution is to calculate an image at twice the resolution of the buffer and then average back to the intensity for one scan line. This saves the cost of having a buffer twice the size.

Another solution to aliasing is to blur the edges. This is similar to the artist trying to reduce the stark contrast of black and white at the edge of a charcoal drawing by smearing or blending the edge. A pixel is the smallest picture element, and can only have one color. If the edge of an object only covers part of a pixel, it ought to have two or more values. This problem can be solved by deciding what percentage of the pixel is covered by each value, then use that percentage to average the values together to come up with the one value for the pixel. This averaging in effect smears or blends the edges. Figure 60 and 61 illustrate both the resolution and this blending technique. Both figures are two arrows pointing in opposite directions. The top arrow of each image is antialiased and the bottom arrow is not. The reader can see the effect of the aliasing on the lower image. Figure 60 was computed at 640 by 480 resolution and blown up 4 times. If we squint, the top arrow of figure 60 shows almost no aliasing. Figure 61 is calculated at 160 by 120 and also is blown up 4 times. In this

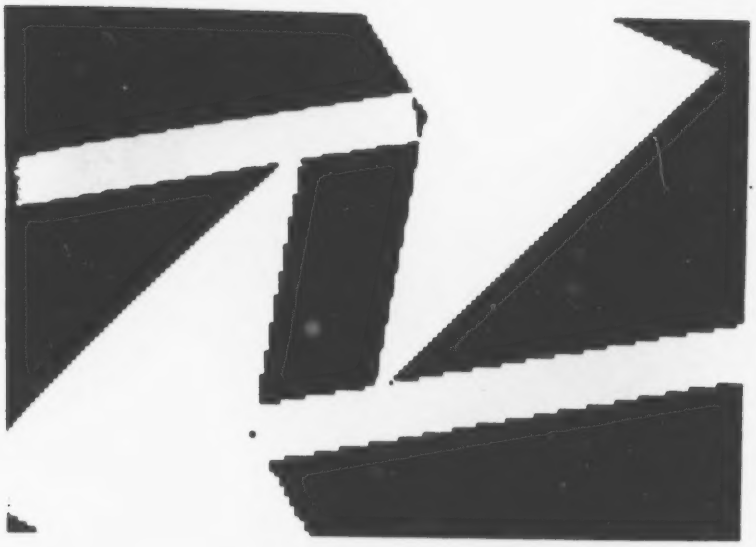

Figure 60. Resolution 640 by 480 blown up 4 times

Figure 61. Resolution 160 by 120 blown up 4 times

figure the aliasing is extremely noticeable and makes it very difficult to determining the shape of the object.

The top arrow here is also antialiased but now we can see each pixel and the effects of "blending" at the pixel boundaries. If the reader has an interest in this area, see the article by Frank Crow [15] comparing antialiasing techniques.

Color variety is largely determined by the amount of memory available for each pixel in the buffer. A black and white image only needs one bit of memory for each pixel. Each pixel is either on or off. If we want an image with gradation from black to white we need 8 bits. To produce color we need at least three times this for the red, green, and blue or 24 bits per pixel. An 8 bit buffer allows us to have 256, or 2 to the eighth, distinct colors. This will be barely enough colors for smooth shading, because these 256 colors include the shading gradations used for the hue. With an 8 bit buffer there are not enough colors available to utilize smooth shading on more than one object. We could shade one object with 256 colors, but we want more than one object. To shade two objects we could give each 128 colors which becomes barely acceptable to the human eye. The use of more than two objects makes this limitation more noticeable, unless all the objects are the same hue.

One method used to extend the possibilities of an 8 bit buffer is to use a color map or color lookup table (sometimes referred to as a palette). This lookup table can be used to extend the range of available colors. The map can be made unique for each scene, which makes it more versatile. Along with this, we could calculate color for 24 bits then average back to several close colors mapped to one color in the lookup table. This is done by using statistics on a scene to determine the "best" set of colors to be used and then by placing these in the color lookup table.

The ability to manipulate what is in the color lookup table and change its values is when we refer to the term palette. The user can create the value gradations of their own color palette. For example, a red object could have the light, middle, and shade values specified as different hues. We could assign 32 values for the range of this color. These values could be specified from yellow to orange for the brighter 10 values, the middle 18 values in the reds, and the 4 bottom values defined in the purples. This would give us an object with warm yellow-

orange highlights, base values of red, and purple shadows. We could control the range for a color by specifying the number of steps in that color. For example black could easily be only one value, zero, and red could be from 32 to 256 if desired. A similar method of changing the lookup table could be used to reverse the values for highlights and shadows, or even simulate color solarizations. The ability to read and write this color lookup table allows the artist to utilize it to the best advantage for creative effects. The ability to change the palette at refresh rates allows us to manipulate the palette for animation. This could be used, for example, to change the colors of a stream to give the illusion of moving water.

Dithering is a method of extending the range of available colors. Dithering is used to color an area with several distinct colors that will be mixed by the eye, similar to a color halftone. Dithering has a trade-off between spatial resolution and color resolution. Dithering groups a block of pixels together and computes the average color for this block of pixels. This in effect enlarges the size of the pixels, thus the spatial resolution is lowered. The block is filled with a color halftone in such a way that the average of the pixels will represent the average color for the block.

For example, if the color for the side of a box is calculated to be 0 0 0.25 (a dark blue) and I only have 0 0 0 or 0 0 1 in the color table. Then 75% of the time the pixel block would be filled with 0 0 0 and 25% of the time with 0 0 1. Dithering can be done in either a random or ordered pattern. The use of dither allows us to do smooth shading of objects with a limited palette, but may cause a problem when used in animation sequences.

Of course we could increase the size of our buffer to 24 bits, 8 bits of red, 8 bits of green, and 8 bits of blue. With 24 bits to 32 bits of color we have enough colors to calculate polygon colors, vertex colors, transparencies, and textures. The interested reader is referred to an article on raster graphics by James Blinn [7] for a more technical explanation.

CHAPTER 6

SUMMARY

The possibilities for the artist in 3-D computer graphics are endless. Currently, the artist needs a good understanding of the technology to create art. I hope the reader has a better understanding of these basic concepts of 3-D computer graphics, as a result of reading and seeing the illustrations presented. In order to create an image the artist/user creates a mathematical model and specifies the object transformations, the object attributes, the view parameters, the lights, and background color or image. The rendering of this scene is handled by the rendering program and displayed on a display device.

In 3-D computer graphics we try to model the world and its natural phenomenon using the tools of 3-D mathematics, a form the computer understands. The Cartesian coordinate system is used as a measure of 3-D space in this world. Polygonal data is only an approximation of a form. These objects are represented mathematically defining the X Y Z coordinate points within this Cartesian coordinate system. These points are connected to define the surface of a planar polygon if the object is to be a solid shaded object. An object is made up of many of these adjoining polygons. The final object is represented in a compact mathematical form and stored in memory to be used at will. Once an object is defined it can be used many times. Some basic methods for defining these objects are warping points, projection, solid of revolution, lofting, and combinatoric.

Once we have the mathematical representation, there are some basic operations we can perform on these objects. Through object transformations, an instance of an object can be scaled larger or smaller, translated or moved around in 3-D space, and turned in space or rotated. These operations can be performed independently for each axis or in more than one axis at a time. Scaling involves multiplication of the coordinate points. Large numbers expand the object and small numbers contract the object. In translation a positive value added to a coordinate, moves

an object towards the positive end of the axes. If a negative number subtracted from a coordinate it moves towards the negative end of the axes. Rotations are specified by the axes and degree of the rotation. A positive or negative number will determine the direction of the rotation. The order of more than one rotation is important in determining the final orientation of the object in space. The lights are modeled as a point light source. They can be placed anywhere, colored, and their influence scaled.

The attributes of an object, such as color, texture, transparency, or light reflectance are usually carried out in the display algorithm, yet the artist has to specify these parameters. In many systems these parameters are designated in the scene description, but in other systems are part of the object description. The appearance of these surface attributes is determined by the shading and lighting models, and the relationship of the object's surface to the lights and eyepoint.

Color in computer graphics is comprised completely of three primaries; red, green, and blue (RGB). All other colors are a combination of these primaries. The color mixing system is additive. The artist is mixing light primaries, rather than pigments in the subtractive system. All the primaries added together produce white light, and their absence results in black. The secondary colors are yellow, cyan, and magenta. Color is modeled as a unit cube in the Cartesian coordinate system. Colors are specified by assigning them an R G B value. The quality and quantity of color depends on the resolution of the screen and the amount of memory in the frame buffer. Various interfaces to the RGB system have been developed, (e.g., HLS, CMY, HSV, and CNS).

This assigned color is the base color for the object. If the artist wants to control more than one color for an object colors can be assigned to polygons or vertices. For the simulations of textures or complex colors a texture pattern can be created and mapped to the surface of the object.

We can view these objects from any desired position by specifying the view parameters, the eyepoint, the center of interest, and the view angle. The transformation of objects effects only the particular object.

However, the view transformation is global and effects all of the objects in a scene. The eyepoint and center of interest are placed at an X Y Z coordinate. These two points create a line of sight. The view angle, specified in degrees, determines angle of the view with the eyepoint at the apex of the view pyramid.

The ability to arbitrarily choose any view is one of the most salient features of 3-D computer graphics. Cinematic conventions, (e.g., pan, truck, crane shot, and dolly), can be simulated, by moving the eyepoint and center of interest. The view angle can be adjusted to simulate the focal length or view angle of a camera lens.

The rendering program can be implemented in software or hardware. Currently, there are many methods of displaying an image. Various methods have been developed to compute hidden-surfaces, view transformations, light and shading models, color, and scan-conversion. Hidden-surface and anti-aliasing have been the major concern in research on display algorithms. The display algorithm is often transparent to the user, they need only send a script to the renderer and see the results on the display device. A general knowledge of the process, of course, aids the artist in controlling his art.

A basic understanding and visualization of how 3-D computer images are created and rendered is the first step to competence in this new artistic and technological medium. By understanding the basic concepts of computer graphics and how the world and natural phenomenon are modeled, the artist has been introduced to the literature and will find it more accessible. Through an introduction to these basic concepts, the artist begins to gain an understanding of the terminology specific to this field.

Duane Palyka has asked the crucial question, for the artist interested in 3-D computer animation:

> What is the overhead in learning the computer art medium so an artist can start thinking in terms of forms, shapes, and colors through numbers and programs? [50]

Currently, the artist involved in 3-D computer graphics must have a very good understanding of the underlying

processes. This involves a knowledge of mathematics and programming. The approach to creating art is still indirect and must be specified numerically. Presently, the design of hardware and software is geared to science and technology, but this is changing. After more and more artist have training in computer graphics, they will be capable of communicating their interest and desires to computer scientists. Once there is a common language for communication, artist will be able to aid in directing the research and development of software for producing art. To do this the artist must be in on the design of the capabilities of a system as well as the interface with the user. Through the collaborations of these two fields, computer graphics can become a more natural, dynamic and artistic medium.

APPENDIX A

SELECTED GLOSSARY

Algorithm - a step-by-step procedure for solving a problem.

Anti-alias - a technique used to reduce undesirable visual effects in raster images, such as the stair-step effect or "jaggies" along the edges of objects.

CAD/CAM - Computer Aided Design / Computer Aided Manufacturing.

Center of Interest - the point in space at which an observer is looking.

Clipping - a process that discards all parts of the scene outside the field of view.

Color Look-up Table - a table storing the red, green, and blue values to be displayed on the color raster device.

Color space - a 3-dimensional coordinate system used to mathematically model color properties, (e.g. hue, saturation, value).

Coordinates - a set of numbers used to locate a point relative to a system of axes or surfaces called a coordinate system. The most frequently used system is the Cartesian coordinate system.

Data - any type of information input into the computer. In 3-D computer graphics it refers to an ordered set of coordinates and surface descriptions used to define a 3-dimensional object in the computer memory.

Digitizing Tablet - an electronic surface that senses input from a stylus or mouse along the X and Y axes; used to input graphical data to the computer.

Display Program - software that transforms 3-D data to an image on a vector or raster display device.

Eyepoint - a point in the Cartesian coordinate system which indicates the position of the observer's eye.

Frame - a single image or scene from a sequence of images for animation.

Frame buffer - a memory device used to store the contents of an image, pixel by pixel, to be displayed on a raster refresh device.

Hidden-surface algorithm - determines which surfaces are closer to the viewer and eliminates those surfaces obscured from view.

Light pen - a device that senses the location of a point of light on the screen; it can be used to input positional information into the computer.

Linear interpolation - evenly dividing the changes between two or more values, the first value changes to the second value in discrete increments, (e.g. 0, 0.2, 0.4, 0.6, 0.8, 1).

Matrix - an array of values used to represent relational information, to transform objects according to those relationships.

Mouse - a graphic input device that senses its location when rolled across a surface; it can be used to input positional information into the computer.

Orthographic projection - a method of representing solid objects on a 2-dimensional surface, as they would appear from the observer's eye when viewed from a given point. It is a parallel projection, where a point in 3-D space is identified with a point on the screen by a perpendicular line.

Perspective projection - a method of representing solid objects on a 2-dimensional surface, as they would appear from the observer's eye when viewed from a given point. Perspective projection conveys depth by making distant objects appear smaller than close ones through the use of vanishing points.

Pixel - a contraction of "picture element". A pixel is the smallest unit of a raster display that can be stored, addressed or displayed.

Polygon - is a shape defined by vertices and edges.

Polyhedron - a solid 3-dimensional object made up of polygons.

Raster - a raster display device within which an electron beam scans in the data as horizontal rows of pixels.

Refresh - rewriting an image to the raster display to excite the phosphors on the screen to maintain a constant image. Normally the refresh rate is 30 to 60 cycles per second.

Renderer - software that transforms 3-D objects to an image on the raster display device, usually refers to solid shaded objects.

solid shaded objects.

Resolution - the number of addressable pixels in a scan line and the number of vertical scan lines. The greater the number of pixels the clearer the image.

RGB - red, green, and blue. Used to refer to the color space, the color mixing system, or monitor in color computer graphics.

Transformation - a change of variables in an algebraic expression, in computer graphics usually done using a transformation matrix.

Vector - a connecting line between two points; the lines seen on a line drawing device. A line representing physical quantity that has a magnitude and direction in space.

Vertex - the intersection of two or more of edges in polygons or polyhedra.

Wireframe - the representation of the edges of a 3-dimensional object as seen on a line drawing device.

BIBLIOGRAPHY

1. Albers, Josef, Interaction of Color, Yale Press, London (1963).

2. Andree, Hans-Joachim, "Graphics and Animation by Personal Computer". Leonardo, 15, (1) pp. 34-36.

3. Arnheim, R., Art and Visual Perception a Psychology of the Creative Eye, University of Californa Press (1974).

4. Berk, T., Brownston, L., and Kaufman, A., "A New Color-Naming System for Graphics Languages". IEEE Computer Graphics and Applications, 2, (3) pp. 37-44.

5. Blinn, James, "Texture and Reflection in Computer Generated Images". Communications of the ACM, 19, (10) (1976).

6. Blinn, James, "Models of Light Reflection for Computer Synthesized Pictures". Computer Graphics, 11, (2) (1977).

7. Blinn, James, "Raster Graphics," in Tutorial: Computer Graphics, ed. Kellog Booth, IEEE (1979).

8. Blinn, James F., "Introduction to 3D Graphics". Course Notes, Seminar on The Artist/Designer and Computer Graphics, (July 1983). ACM SIGGRAPH 83.

9. Carlson, Wayne, "Techniques for the Generation of Three Dimensional Data For Use in Complex Image synthesis," Ph.D. Thesis, The Ohio State University (September 1982).

10. Coffey, Rebecca, "In the Mind of Lillian Schwartz". Computer Pictures, 2, (1) pp. 52-56.

11. Conrac-Division,, Raster Graphics Handbook, Conrac Corporation (1980).

12. Cornock, Stroud and Edmonds, Ernest, "The Creative Process Where the Artist is Amplified or Superseded by the Computer". Leonardo, 6, pp. 11-16.

13. Crow, F. C., "Shaded Computer Graphics in the Entertainment Industry". Computer, 11, (3) pp. 11-22.

14. Crow, F. C., "Three-Dimensional Computer Graphics, Part I". Byte, 6, (3) p. 54.

15. Crow, F. C., "A Comparison of Antialiasing Techniques". IEEE Computer Graphics and Applications, 1, (1) pp. 40-49.

16. Crow, F. C., "A More Flexible Image Generation Environment". Computer Graphics, 16, (3) pp. 9-18. Proc. ACM SIGGRAPH 82.

17. Csuri, Charles, "Computer Graphics and Art". Proceedings of the IEEE, 62, (4) pp. 503-515.

18. Csuri, Charles, "3-D Computer Animation". Advances in Computers, 16, pp. 1-57. Academic Press.

19. Csuri, Charles, Gomez, Julian, MacDougal, Paul, and Zeltzer, David, Beyond the Storyboard: A Tool Set for 3-D Computer Graphics, Computer Graphics Research Group, in preparation. January 1984.

20. DiBlasio, Margaret, "If and where to plug in the computer: A conceptual framework for computer assisted art instruction". Studies in Art Education, 1, (25) (1983).

21. Ettinger, Linda and Rayala, Martin, "Computers in Art Education". The Computing Teacher, pp. 24-29.

22. Foley, J. D. and Dam, A. Van, Fundamentals of Interactive Computer Graphics, Addison-Wesley (1982).

23. Franke, Herbert, Computer Graphics Computer Art, Phaidon Press (1971).

24. Gips, James and Stiny, George, "An Investigation of algorithmic aesthetics," in Visual Art, Mathematics, and Computers, ed. Franke Malina, pergamon Press, New York (1979).

25. Greenberg, D. P., "An Overview of Computer Graphics," in The Computer Image, ed. D. P. Greenberg, Addison-Wesley (1982).

26. Halas, John, Computer Animation, Visual Communications Book, New York (1974).

27. Harries, John, "Personal Computers and Notated Visual Art". Leonard, 14, (4) pp. 229-301.

28. Hubbard, Guy and Linehan, Thomas, "Arcade Games, Mindstorms, and Art Education". Art Education, 36, (3) pp. 18-20.

29. Hunt, R.W.G., The Reproduction of Colour in Photography, Printing and Television, John Wiley & Sons, Halsted Press, New York (1975).

30. Itten, Johannes, The Elements of Color, Van Nostrand Reinhold Co, New York (1970).

31. Joblove, George and Greenberg, Donald, "Color Spaces for Computer Graphics". Computer Graphics, 12, (3) p. 20. Proc. ACM SIGGRAPH 78.

32. Jones, Beverly J., "Computer Applications in Art Education Research," Unpublished Doctoral Dissertation, The University of Oregon (1977).

33. Jones, Beverly J., "Computers in the Arts and Humanities". The Computing Teacher, pp. 22-30.

34. Jones, Beverly J., "Microcomputer-Controlled Generation of Artifacts as an Interdisciplinary Teaching Aid". The Computing Teacher, pp. 42-45.

35. Jung,, Beck,, and Huebner,, "Computer Art: colored Pictures Based on a 'Chonocube'". Leonardo, 15, (2) pp. 118-119.

36. Knobler, N., The Visual Dialogue, Holt, Rinehart, and Winston, Inc. (1966).

37. Kugel, Peter, "Artificial Intelligence and Visual Art". Leonardo, 14, (2) pp. 137-139.

38. Levitan, Eli, Electronic Image Techniques, Van Nostrand Reinhold, New York (1977).

39. Linehan, Thomas, "Computer Graphics: Opportunity for Artistic Vision". Art Education, 36, (3) pp. 11-14.

40. Linehan, Thomas, "An Investigation of Criteria for Evaluating Computer Art". Computer Graphics and Art, 1, pp. 6-9.

41. Linehan, Thomas, "A Computer Mediated Model for Visual Preference Research and Implications for Instruction in Art Criticism," Unpublished, Doctoral Dissertation, Ohio State University (1980).

42. MacNichol, Edward, "Three-pigment Color Vision". Scientific American, 211, pp. 48-56.

43. Madeja, Stanley, "Computer Graphics: The New Subject Matter for the Art Curriculum". Art Education, 36, (3) pp. 15-17.

44. Marcus, Aaron, "Color: A Tool for Computer Graphics Communication," in The Computer Image, ed. D. P. Greenberg, Addison-Wesley (1982).

45. Moholy-Nagy, L., Vision in Motion, Paul Theobald Publisher, Chicago (1947).

46. Newman, W. and Sproull, R., Principles of Interactive Computer Graphics, 2nd Edition, McGraw-Hill, New York (1979).

47. Noll, Michael, "The Digital Computer as a Creative Medium". IEEE Spectrum, 4, (10) pp. 89-95.

48. Noll, Michael, "Computers and the Visual Arts: A Retrospective View". Catalogue, Exhibition of Computer Art, (1982).

49. Ocvirk, O., Bone, R., Stinson, R., and Wigg, P., Art Fundamentals Theory and Practice, Wm. C. Brown Co. Publishers (1975).

50. Palyka, Duane, "Computer Painting," pp. 61-64 in Artist and Computer, ed. Ruth Leavitt, Creative Computing, Harmony books, New York, N.Y (1976).

51. Palyka, Duane, "TWEEN (Artist's View)". Course Notes, Seminar on Two-Dimensional Computer Animation, pp. 82-88. ACM SIGGRAPH 82.

52. Palyka, Duane, "COMPUTER ART -- REFLECTIONS OF THE MIND," The Third Hemisphere Colloquium, Humanities Institute, Florida State University, Tallahassee, Florida (March 1983).

53. Parent, R., "A System for Generating Three-Dimensional Data for Computer Graphics," Ph.D. Thesis, The Ohio State University, Columbus, Ohio (1977).

54. Parker, W. Oren and Smith, Harvey K, Scene Design and Stage Lighting, Holt, Rinehardt & Winston (1979).

55. Reichardt, Jasia, The Computer in Art, Studio Vista, London (1971).

56. Stredney, Donald, "The Representation of Anatomical Structures Through Computer Animation for Scientific, Educational and Artistic Applications.," Masters Thesis, The Ohio State University (March 1982).

57. Sutherland, Ivan, Sproull, Robert, and Schumacker, Robert, "A Characterization of Ten Hidden-Surface Algorithms," pp. 387-441 in Tutorial: Computer Graphics, second edition, ed. Kellogg Booth, IEEE Computer Society (1982).

58. Warn, David R., "Lighting Controls for Synthetic Images". Computer Graphics, 17, (3) pp. 13-21. Proc. ACM SIGGRAPH 83.

59. White, Dennis, "Advanced Technology, Art, and Art Education: Researching Toward the Third Millennium". Art Education, 36, (3) pp. 8-10.