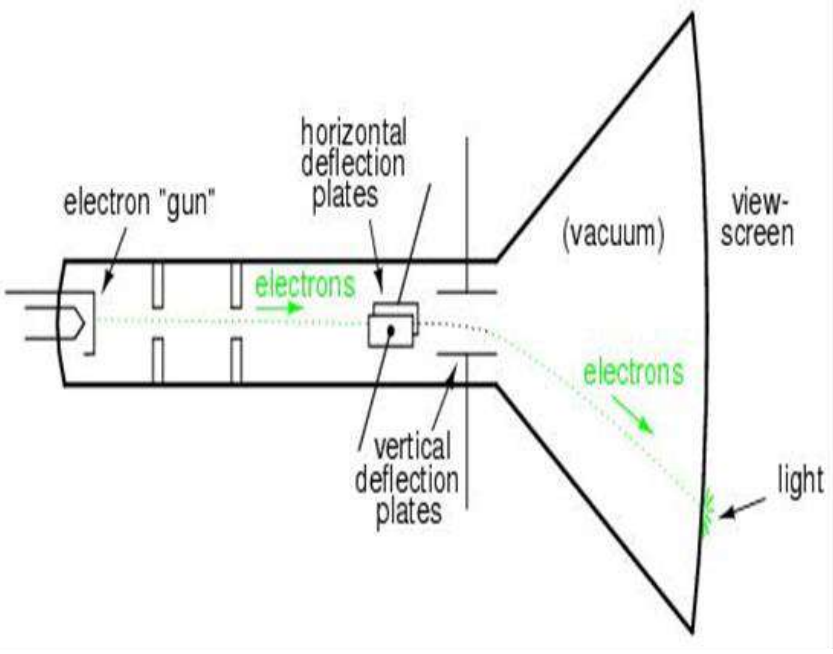
MCA-301

Computer Graphics



VARDHMAN MAHAVEER OPEN UNIVERSITY

KOTA

www.vmou.ac.in

MCA-301

Vardhman Mahaveer Open University, Kota

Computer Graphics

Course Development Committee

Chair Person

Prof. Ashok Sharma
Vice-Chancellor
Vardhman Mahaveer Open University, Kota

Prof. L.R. Gurjar
Director Academic
Vardhman Mahaveer Open University, Kota

Convener and Members

Convener

Neeraj Arora
Assistant Professor, Computer Science
School of Science and Technology,
Vardhman Mahaveer Open University, Kota.

Members

1. **Prof. (Dr.) Reena Dadich**
   Professor and Head (CS)
   University of Kota, Kota

2. **Prof. (Dr.) N.K. Joshi**
   Professor (CS) and Director, MIMT, Kota

3. **Dr. Harish Sharma**
   Associate Professor, CSE Deptt.
   Rajasthan Technical University, Kota

4. **Mr. Abhishek Nagar,**
   Programmer Officer, VMOU, Kota

5. **Dr. Anuradha Dubey**
   Deputy Director
   School of Science &Technology
   Vardhman Mahaveer Open University, Kota

Editor & Unit Writers **MCA-301: Computer Graphics**

Editor

Dr. Bharat Singh Deora

Assist. Prof., Dept. of CS & IT, JRN Rajasthan Vidyapeeth University, Udaipur

Unit Writers	*Units*
Mr. Gaurav Meena Assistant Professor, Department of CS, Central University of Rajasthan, Ajmer.	**1, 2**
Mr. Anil Kumar Sharma Assistant Professor, Department of CS, Govt.Women Engg.College, Ajmer.	**3**
Mr. Santosh Gupta Assistant Professor, Department of CS, M.I.M.T., Kota.	**4**
Mr. O.P. Suthar Asst. Prof. (SR. Scale), Department of CSE, J.I.E.T., Jodhpur	**5, 6, 7, 17**
Mr. Manish Gehlot Assistant Professor, Computer Science and Engineering, J.I.E.T., Jodhpur	**8, 11**
Mr. Pankaj Acharya Associate Professor and Head, Dept. of IT, J.I.E.T., Jodhpur	**9, 10, 12, 13**
Dr. Snehlata Kothari Pacific University, Udaipur	**14**
Mr. Neeraj Arora Assistant Professor of Computer Science, Vardhman Mahaveer Open University, Kota	**15**
Mr. Kamal Kulshreshtha Associate Professor & Head, Deptt. of Computer Sc. & Applications, MIMT, Kota	**16**

Academic and Administrative Management

Prof. Ashok Sharma
Vice-Chancellor, VMOU, Kota

Prof. L.R. Gurjar
Director (Academic), VMOU, Kota

Dr. Shiv Kumar Mishra
Director (MP&D), VMOU, Kota

Print: 2018 **ISBN: 978-81-8496-640-4**

Printed and Published on behalf of the Registrar, Vardhman Mahaveer Open University, Kota.

Printed by :

MCA-301

Vardhman Mahaveer Open University, Kota

Computer Graphics

Contents

Preface

The present book entitled “Computer Graphics” has been designed so as to cover the unit-wise syllabus of MCA-301 course for MCA 3rd Year students of Vardhman Mahaveer Open University, Kota.

The book is begin with some preliminaries of computer graphics and fundamentals necessary to understand computer graphics, and covers the topics like Color Filling Algorithm, Attributes of Output Primitives, Curves and Surfaces, Geometric 2D Transformation, Geometric 3D Transformation, Viewing Transformation, Clipping, Visible Surface Detection, Illumination model and Shading, Color Models and applications, Computer Animation, Graphical User Interfaces and Input Methods.

Each unit begins with objectives, introduction and principles together with illustrative and other descriptive material .The illustrative examples serve to illustrate and amplify the theory of computation. The units have been written by various experts in the field. We believe that this book is well suited to self-learning. The text is written in a logical sequence and is beneficial for students. The concise and sequential nature of the book makes it easier to learn. Although we have made all efforts to make the text error free, yet errors may remain in the text. We shall be thankful to the students and teachers alike if they point these out to us. Any further comments and suggestions for future improvement are welcome and will be most gratefully acknowledged.

UNIT-1

Introduction to Computer Graphics

Structure of the Unit

1.0 Objective

In this chapter we shall focus on the following topics:

- Computer Graphics
- Types of CG
- Types of Images
- Origin of Graphic System

- Applications

1.1 Introduction

A Computer is a machine that processes an input data into output information. It can store, process, manipulate or correlate data. Computer graphics is one of the most effective and widely used ways to communicate the process information to the user. It displays the information in the form of graphics objects such as pictures, images, graphics, charts, etc. The picture or graphics objects may be anything like architectural structure. Engineering Drawing or models, frame from an animated movie. Therefore, it is important to understand some point:

a) How are the images presented in computer graphics?
b) How are graphics objects prepared for presentations?
c) How a user interact with graphic objects?
d) What is the origin of the computer graphics?

In Computer graphics, pictures or images are presented as a collection of discrete picture elements called pixels. A pixel is the smallest unit of any image. Each pixel on the graphics display does not represent a mathematical point. Rather, it can contain a region which has infinite no. of points. The process of determining the appropriate pixels for representing pictures or graphic objects is known as rasterization. The process of representing continuous pictures as a collection of discrete pixels is called scan conversion. Computer graphics provide some features like rotation, translation, scaling & performing various projections. Users can apply these projections on the pictures before displaying it.

In the following chapter, we will study about the origin of computer graphics, tools for computer graphics and various applications of computer graphics.

1.1.1 Introduction to computer graphics

To display a picture of any size on a computer screen is a difficult process. Computer graphics are used to simplify this process. Computer graphics are visual representation of data (or info.) displayed on a monitor. Graphics can be a single image or series of images (i.e. Video). This computer graphics term first discovered by researchers Verne Hudson and William fetter in 1960. It is a vast area in computer science. Sometimes it is abbreviated as CG or CGI (computer-generated imagery). Development of computer graphics has a significant impact on many areas like media, animation, games, designs, etc. It also can be used in many disciplines like Presentation, Drawing, Painting, Design, Image Processing and Scientific Visualization.

1.2 Basic Concepts

I hope all of you are fond of video games and may be good at playing them. Have you seen the game of Mario?

It's a game played by one person with the keyboard, or sometimes we can use the game controller. In this game a small tiny animated person moves in a left or right direction.

So, when you start the game, you have to jump and hit some question marks and after hitting you'll get some power or coins that are a surprise. Different Kind of powers you can earn by hitting question marks like size changing, protected mode, can get bullets and jumping power, etc.

Now, how did you invent this video game? This has been done with the aid of computer graphics. To represent a series of pictures in a particular time or game graphics use the major use of CG. It helps to create and manipulate pictures with the computer, it concerns with the pictorial (pixel) representation of real & imaginary objects.

1.3 Origin of Computer Graphics

Many years of research and development were made to achieve the goals in computer graphics field. During the first half of the twentieth century, the developments were made the advances in electronics, electrical engineering, and Television. Screens could display art to create effects for the earliest films from 1895, but such displays were limited and not interactive.

The first cathode ray tube, the Braun Tube, was invented in 1897. In 1950, the first computer-driven display was used to generate only simple pictures. The display made use of a cathode-ray tube similar to the used in Television Sets. During 1950's, interactive computer graphics made little progress because computers of that period were unsuitable ~~to~~ for interactive use. These types of computers were used to perform lengthy calculations.

The phrase "Computer Graphics" itself was introduced in 1960 by William Fetter, a graphic designer from Boeing. In 1961, Steve Russell (MIT Student) created a video game named **Space War**. The single event that did the most to promote interactive CG as an important field was the publication in 1962 of thesis entitled "Sketchpad: A Man-machine Graphical Communication system" by Ivan E. Sutherland. It provides that interactive computer graphics was a feasible, useful and exciting field of research.

Sometimes in early 1960's, automobiles would provide a boost through the work of Pierre Bezier at Renault, who used Paul de castellan's curves (called Bezier Curves) to develop 3D modeling techniques for Renault car bodies. By the mid-1960's, large computer graphics research projects were undertaken at MIT and Bell Telephone labs. In 1966, Ivan E. Sutherland invented the first computer controlled Head Mounted Display (HMD). It displays two separate wireframe images, one for each eye. In 1968's, Arthur Apple described the first algorithm of ray casting, which is a basis point for almost all of modern 3D graphics or Photorealism in graphics. Thus the golden age of computer graphics began.

In 1970, Edwin Catmull (Founder of Pixar) worked on graphics animation. Catmull loved animation, but he did not have the talent of drawing. So, He saw computers as the natural progression of animation. The first animation that Catmull saw was his own. He created an animation of his hand opening and closing. With Catmull, Fred Parke created an animation of his wife's face. Martin Newell in the University of Utah with Ivan Sutherland made a picture of Utah teapot and its static renders. This has become representative of CGI development during 1970's. Jim Binn innovated in 1978 by introducing Bump Mapping, which is a technique for simulating uneven surfaces. Lots of games were developed in the 1970's like modern video game arcade with the first arcade games using real-time 2D sprite graphics. In 1972, Pong was one of the first hit arcade cabinet games. In 1974, speed race featured sprites moving along a vertically scrolling road. In 1975, gun fight featured human-looking sprite character graphics. In 1978, space invaders featured large numbers of sprites on screen.

In 1980's, it was an era of modernization and commercialization of computer graphics. In this age home computers were increasing day by day and adopted by a much larger audience, with this no. of graphics developers were increasing

significantly In the early 1980's, the availability of bit-slice and 16-bit microprocessor started to revolutionize computer graphics terminals. Modern computers often use graphical user interface (GUI) to present data & information with symbols, icons and pictures rather than text. Graphics are one of the five key elements of multimedia technology. In 1982, Japan's Osaka University developed LINKS-1 computer graphic system (a super computer) to render realistic 3D computer graphics. Links-1 was able to render highly realistic images rapidly. It was used to create the world's first 3D planetarium- like videos of entire heavens that was made completely with CG. Links-1 was the world's most powerful computer as of 1984. In 1986, David Immel and James Kariya developed an important step towards implementing global illumination, which is necessary to continue Photorealism in computer graphics. In 1988, the first Shaders - a small program, designed specifically to do shading as a separate algorithm was developed by Pixar (A computer animation film studio). In late 1980, SGI (Silicon Graphics Inc.) were used to create some of the first fully computer generated short films at Pixar. Silicon graphics machine were considered a high watermark for the field. The 1980's is also called the golden era of video games.

In 1990's, the emergence of 3D modeling on a mass scale. 3D graphics become more popular in gaming and animation. At the end of the 1980s and beginning of the 1990s, in France Quarxs- the first HDTV computer graphics series by Maurice Benayoun and Francois Schuiten were created. In films, Pixar began its commercial rise in this era under Edwin Catmull, with its first major film release in 1995;" Toy Story". In video games (1992), "Virtual racing" running on the Sega Model-1 arcade system Board. It was a fully 3D racing game and popularized real-time 3 D polygonal graphics in the video-game industry. In 1993, Sega Model-2 and, in 1996, Sega Model-3 pushed the boundaries of commercial real-time 3D

graphics. Technology and algorithm for rendering & shadowing continued to improve greatly.

In 2000, CGI became ubiquitous in intense during this era. Video games and CGI cinema had spread all over the graphics in the late 1990's and continued to do so in 2000 also. Computer graphics used in films and video games gradually began to be realistic. With traditionally animated cartoon films like Ice Age and Madagascar as well as numerous Pixar offering like Finding Nemo dominating the box office in this field. In 2001, ''final fantasy: the spirits within" movie released was the first fully computer-generated feature film to use photorealistic CGI characters and be fully made with motion capture. In video games, the Sony playstation3, the Microsoft Xbox lines of consoles etc. Marquee CGI-heavy titles like the series of grand theft auto, Assassin's Creed, Final Fantasy, Bio shock, Kingdom Hearts, Mirror's Edge and other approaches to photorealism. Microsoft decided to expose to the independent developer with the XNA program. DirectX itself is a commercial success. OpenGL is also a tool for computer graphics. The second generation shader languages HLSL & GLSL began to be popular in this decade. In the scientific computing, GPGPU technique is used to pass large amount of data to and from the GPU.

In 2010s, pre-rendering graphics are nearly scientifically photorealistic and CGI is used in videos. Texture mapping has many stages with multiple layers, but it is common to implement texture mapping, bump mapping normal mapping, lighting maps including specular highlights and reflection techniques and shadow columns into one rendering engine using shaders. Shaders become a necessity for advanced work in the field. In cinemas, most animated movies are CGI now, a great many animated CGI films are made per year and most are 3D animated cartoons.

In video games, the Xbox One by Microsoft Sony PlayStation-4 & Nintendo Wii U currently dominate the home space and are all capable of highly advanced 3D graphics; the windows PC is still one of the most active gaming platforms as well.

1.4 Types of Computer Graphics

According to the type of Interaction, we can divide computer graphics into two parts:

A. **Non-interactive Computer Graphics**: It is also known as passive computer graphics. In this, the observer has no control over the image. For e.g.: this type of CG includes the title shown on TV and other forms of computer art.

B. **Interactive computer graphics**: It is also known as active computer graphics, it has two-way communications between the computer and the user. In this, the observer has some control over the images with some input devices. For e.g. video game controller etc. These input devices help to send a request to the computer.

The computer on receiving signals from the input device can modify the displayed picture appropriately. It shows to the user that picture is changing instantly in response to their commands. The user can give series of commands and each one generates a graphical response from the computer. So, in this, user maintains a conversation with the computer. Another example of interactive CG is flight simulator. It helps to train the pilots of our airplanes. Flight simulator creates containing all the usual controls and surrounded by screens on which we have the projected computer generated views. Advantages of simulator over real aircrafts are fuel savings, safety and can get better training.

1.4.1 Types of Images or Subsets of CG

In computer graphics, images can be divided into two dimensions, i.e. 2D & 3D. They are made differently and used differently. Let's explore the difference and similarities between them.

Two-Dimensional (2D): It is a computer-based generation of digital images; mostly from the 2D geometric models or texts. 2D images have two dimensions either x or y in a plane. 2D graphics started in the 1950s based on vector graphic devices or raster based devices. 2D computer graphics are mainly used in applications that were originally developed upon traditional printing and drawing technologies such as cartography, typography, advertising, technical drawing etc. 2D graphics can be split into two categories:

A. **Vector graphics**: It uses lines, shapes and text to create more complex images. If any vector graphic image is shown on a very big screen, then it will look fine or as good as the regular size. For e.g. a car, or a light bulb etc.
B. **Raster Graphics:** It uses pixels to make up a larger image. Raster programs often have tools like paint brush, paint buckets or eraser to make a picture. This is often used as a part of what the user sees when they use computer program (like Adobe Photoshop or paint). Sometimes people do use only pixels to make an image. This is called pixel art and it has a unique style. For e.g. Photographic of any object, pixel art of ''Rose".

Three-Dimensional (3D): This type of images or graphics exactly looks like real objects because they are 3-Dimensional. It has three things – height,

length, depth. 3D graphics are used in movies and video games and many animated shows etc. For e.g.: modeling of any device (car) in 3D studio max.

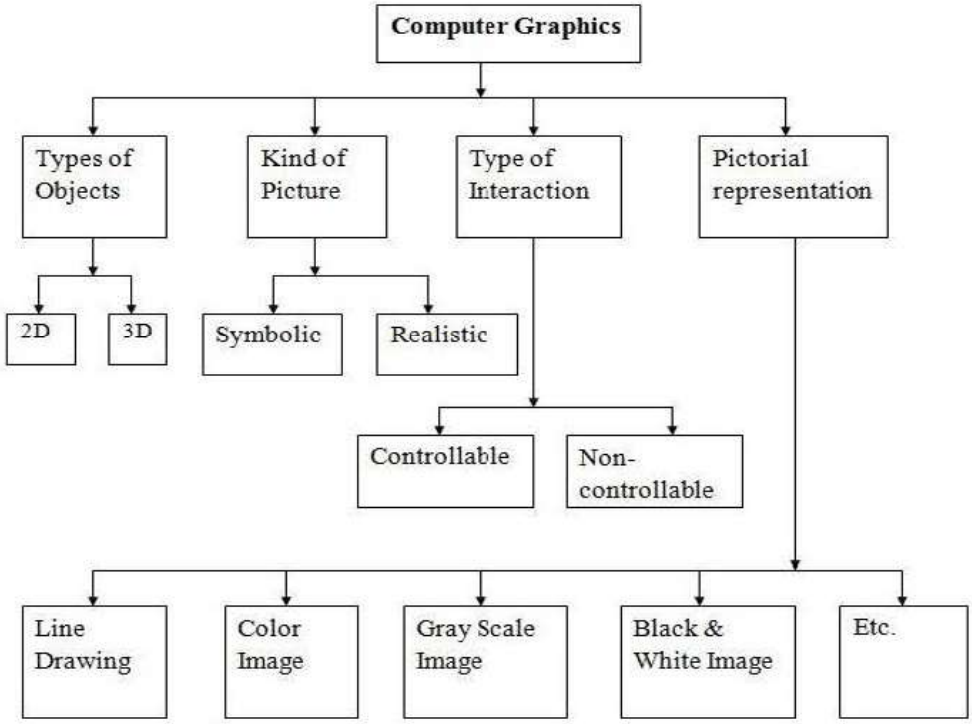


Figure 1.1 : Computer Graphics Hierarchy.

1.5 Graphics APIs

API (Application Program Interface) is a set of routines, protocols and tools for building software applications. An API specifies how software components should interact. APIs are used when programming graphical user interface (GUI) components.

The Graphics API is the software that renders the video or image you see on the screen. By learning Graphics APIs users can build their own 2D or 3D graphics. Given that learning a special API is a very large undertaking, it goes without saying that the learner or user would like to make the best choice possible so that

they aren't wasting their time. I'm writing small guidelines about all recent and previous graphics APIs so that you can make the best informed decision on what you might prefer to learn. Graphics APIs are following:

DirectX 9 or Below: DirectX is a collection of APIs for handling tasks related to multimedia, especially game programming and videos on Microsoft platform. The name DirectX was coined as a shorthand term for all of these APIs (X stands for the particular API name i.e. Direct 3D, DirectDraw, DirectMusic, DirectSound & so on). Version 9 of DirectX was first released for windows in 2002 and its subsequent updates in 2003 & 2004. DirectX 9 (Direct 3D 9) is a primary graphics interface on windows vista. It remains the ideal API to use for writing 3D games & applications that need to run on the broad range of existing hardware and windows release. However, it wasn't well organized and has some limitation in taking advantage of modern hardware.

DirectX 10 and DirectX 11: DirectX 10 (Direct 3D 10) was previously the best modern graphic API to learn 3D graphics programming. Its only disadvantage is that it doesn't support hardware tessellation (However, it is a minor problem). This API was designed from the ground up with a primary focus on taking advantage of new hardware with a strong secondary focus on organization within the APIs. Because of this focus on making the API well organized. This same organization was carried forward into DirectX 11 as well.

DirectX 11 is the leading industry standard graphics API. Even alongside DirectX 12 Microsoft released DirectX 11.3 at the scene time knowing that the vast majority of people writing graphics engines would not need such low-level control that DirectX 12 provides. DirectX11 has the same features list as the DirectX 12. Regarding support for coding from other programmers, the majority of websites and forums will concentrate on DirectX 11.

OpenGL 3 & OpenGL 4: OpenGL 3.3 was the DirectX 10 equivalent that could handle cross-platform. Although this is the widely used API (with lots of web resources)

OpenGL 4 is the equivalent to DirectX 11 regarding features and modern hardware utilization. The main advantage of this is being cross-platform. The changes in OpenGL 4 from its previous version didn't make it any more complicated to use or learn. Comparing the two APIs, the differences are minimal between them. So, if you are a primary developer and wants to develop any OS like Linux then OpenGL 4 would be your only choice for a high-level graphics API that utilized hardware acceleration.

Vulcan: Vulcan is the newest low overheated cross platform 3D graphics and computes API first Announced at GDC (Game Developers Conference) 2015 by the Kharnos group. The Vulcan API was initially referred to as the "next generation OpenGL initiative" or "OpenGL Next". Just like the DirectX 12 this is an expert's API. It Parallels the features offered by DirectX12 and is equally complicated in terms of programming. This is not an appropriate API for the beginner. OpenGL 4 is the best option for beginners. It is the low-overhead successor to OpenGL.

Mantle: Mantle is a low-overhead rendering API targeted at 3D video games. It is also an expert's API that was created by AMD (Advanced micro devices). Mantle was designed as an alternative to Direct3D and OpenGL. It was the first 3D graphics API released to give low-level control over the GPU and was the driving catalyst pushing both Microsoft and the Khronos group to develop their low level APIs. However, it only supports AMD graphics hardware which is incredibly limiting in a market that is dominated by NVIDIA.

GNM: Low-level API of the PlayStation 4.

GNMX: high-level API of the PlayStation 4.

Metal: low-level API for Apple IOS.

1.6 Applications of Computer Graphics

Computer Graphics is a study of technique to improve communication between human and machine through pictures, charts and Diagrams. Computer graphics used in diverse areas like advertising, Healthcare, Education, Engineering, Science, Entertainment, multimedia etc. Let's discuss the representative uses of computer graphics:

Plotting of Graphics and charts: To produce illustrations those summarize various kinds of Data; summarize financial, statistical, mathematical, economic data for research reports, scientific and other reports. It increases the understanding using visual tools like line graphs pie charts, bar charts, surface graphs etc. Except 2D, 3D graphics are good tools for reporting more complex data.

Entertainment: Computer graphics methods are commonly used in making motion pictures, music videos; television shows, video games and animated films. Graphic objects can be combined with live actions or can be used with image processing techniques to transform one object to another.

Computer Aided Design: Computer Graphic is a useful tool for generating or designing the architecture, drawing and structures. In engineering and architectural systems, the products are modeled using computer graphics tool such as CAD (Computer Aided Design). CAD applications are also used in computer animations. The motion of an object can be simulated using CAD.

Art and commerce: There is a lot of development in the tools provided by computer graphics. CG is used in both fine arts and commercial application. Fine artist uses other computer technologies to produce images. They use a combination of 3D modeling packages, texture mapping, drawing program and CAD Design software. In commercial art, uses for logos and other designs page layouts combining texts, graphics and advertising etc. This allows users to create artistic pictures which express messages.

Education & Training: Computer graphics can make us understand the functioning of a system in a better way. Computer-generated models like physical systems, financial system, physiological system, economic systems used as educational aids. Various educational pictures with animations are used to present better understanding for learning.

Visualization: For analyzing scientific, Engineering, Medical business data or behavior where we have to deal with large amount of information. It is a very ineffective process to determine trends. But if it is converted into a visual form, it becomes easier to understand. This process is called visualization.

Graphical User Interface: It is used to make a software package more interactive. A major component of a graphical interface is a window manager that allows a user to display multiple window area. Interfaces also display menus and icons for selection of processing options or parameter values.

- The icon is a graphical symbol i.e. designed to look like the processing option it represents.
- Menu contains lists of textual descriptions and icon

Image Processing: It provides techniques to modify or interpret existing images such as photographs etc. One can improve picture quality through image processing techniques. To digitize the shading and color, sharper, improve the contrast of the scanned image and to transfer them to monitor uses image processing techniques. In medical applications, image processing techniques can be applied for image enhancements and widely used for CT scan (Computer X-ray Tomography) and PET (Position Emission Tomography) images. In space applications, this technology can be used to analyze satellite photos of the earth and photos of galaxies.

Simulation and Animation: Uses of graphics in simulation makes mathematic models and mechanical systems more realistic and easy to study. The interactive graphics supported by animation software proved their use in the production of animated movies and cartoon films.

Process Control: With Computer Graphics it is possible to control various processes from a remote control room. In these cases, process systems and processing parameters are shown on the computer with graphic symbols. It makes easy for the operator to monitor and control various processing parameters at a time.

Cartography: CG is also used to represent geographic maps, weather maps, cantor maps, oceanographic charts, population density maps and so on.

1.7 Summary

The term *computer graphics* refers to something involved in the creation or manipulation of images on the computer, including animated images. It is a very

broad field and one in which changes and advances appear to come at a dizzying pace. It can be difficult for a beginner to know where to start. However, there is a core of fundamental ideas that are part of the foundation of most applications of computer graphics. This chapter attempts to cover those foundational ideas. While it is not possible to cover the entire field in a first chapter or even a large part of it so here we are considering basic knowledge of computer graphics.

This short chapter provides an overview and introduction to the Graphics and Images and tools used for creating games and series of images. Some points you have learned that are:

- Computer-Graphics is a new, rapidly evolving field of CS
- Hardware progress makes new techniques feasible
- Input/interaction, processing, and output technologies
- Frame buffer contains rasterized representation of the scene.

1.8 Self Learning Exercise

Q.1 which device(s) provide positional information to the graphics system?

a) Input Devices
b) Output Devices
c) Pointing Devices
d) Both a and c

Q.2 The primary output device in a graphics system is_____.

a) Scanner
b) Video Monitor
c) Neither a nor b
d) Printer

Q.3 _____ allows screen positions to be selected with the touch of a finger.

a) Touch Panels
b) Image Scanner
c) Light Pen
d) Mouse

Q.4 Which of the following device is not the input device?

a) Trackball and space ball
b) Data glove
c) Impact printer
d) Light pen

Q.5 The quality of a picture obtained from a device depends on

a) Dot size
b) Number of dots per inch
c) Number of lines per inch
d) All of the above

Q.6 Aspect Ratio means

a) Number of pixels
b) Ratio of vertical points to horizontal points
c) Ratio of horizontal points to vertical points
d) Both b and c

Q.7 Random-scan system mainly designed for

a) Realistic shaded screen
b) Fog effect
c) Line-drawing applications
d) Only b

Q.8 which displays devices allows us to walk around an object and view it from a different side.

a) Direct view storage tube
b) Three-dimensional devices
c) Flat panel display devices
d) Plasma panel display devices

Q.9 In which system, the Shadow mask methods are commonly used

a) Raster-scan system
b) Random-scan system
c) Neither a and b
d) Both a and b

Q.10 Virtual reality, CAD, and animations are the application of

a) Z mouse
b) Digitizer
c) Data tablets
d) Image scanners

1.9 Review Questions

Q.1 Define Computer Graphics.

Q.2 What are the applications of Computer Graphics?

Q.3 What is the hardware devices used for computer graphics?

Q.4 What do you mean by interactive computer graphics?

Q.5 What is pixel?

Q.6 List the advantages of interactive and non-interactive computer graphics.

Q.7 Explains the representative uses of Computer graphics.

Q.8 What do you mean by GUI?

Q.9 Write short notes on:

a) Trackball b) Joystick c) Touch Panel d) Scanner

e) Light pen f) Digitizer

Q.10 Write short note on Graphical APIs.

1.10 Answers to Self-Learning Exercise

Q.1 (d) Q.2 (b) Q.3 (a) Q.4 (c) Q.5 (d) Q.6 (d) Q.7 (c) Q.8 (b) Q.9 (a) Q.10 (a)

References and Suggest Reading

1. "Computer Graphics C version", Donald Hearn and M. Pauline Baker, Pearson Education.

2. "Computer Graphics Principles & Practice", second edition in C, Foley, VanDam, Feiner and Hughes, Pearson Education.

UNIT-2

Computer Graphic Systems

Structure of the Unit

2.0 Objectives

In this chapter we shall focus on the following:

- Understands which are the important display devices and input devices.
- Understands how the display devices are important for computer graphics work.
- Learning various advantages and disadvantages of various devices.
- Understands the working of different display systems in computer graphics.

2.1 Introduction

As we know that, computer graphic is used to represent images and series of images (i.e. Video). So, there are several display devices those are used for displaying results on screen. It displays images when processing gets completed. The image can be in different terms; first, a combination of objects, lights and a camera; second, in the form of pixels. The image has lots of pixels or we can say many tiny dots that make up the representation of an image. Display devices are output devices for presentation of information in the visual form. When the input information is given that has an electrical signals then the display is called an electronic display. Electronic visual displays are Television and Computer monitors.

In the rest of the chapter, you will learn about many graphical input and output devices.

2.2 Graphic Devices

The Graphical System includes Processor, Memory, Frame buffer, Output devices, Input devices etc. In the graphical system, we will talk about graphical devices that are input and output devices. There are lots of computer graphical devices:

CRT (Cathode Ray Tube), EGA (Enhanced Graphic Adapter)/CGA/VGA/SVGA monitors, plotters, keyboard, joystick, mouse, data matrix, laser printers, films, flat panel devices, video digitizers, scanners, LCD Panels, touch screen, trackball etc.

2.2.1 Input Devices

There are lots of devices available for data input in the computer graphical system. These include: Digitizer/ Graphical tablets, keyboard, mouse, joystick, scanner, space ball, trackball and so on. We will discuss most of the input devices here.

a) ***Digitizer/ Graphical Tablet:*** A graphical tablet is known as a digitizer. It is a computer input device that enables a user to draw images, animation and graphics with a special pen (named *stylus*). It is somewhat similar to draw an image with a pencil and paper. It can also use to trace an image from a piece of paper which is secured to the tablet surface. This way of capturing by tracing or entering the lines or shapes is called Digitizing.

These devices contain a flat surface and stylus is pen-like drawing apparatus. Different graphic tablets use different techniques for measuring position. Most graphic tablets use an electrical sensing mechanism to determine the position of the stylus. Electromagnetic signals generated by electrical pulses applied in sequence to the wires. The strength of the signal induced by each pulse is used to determine the position of the stylus. The signal strength is also used to determine roughly how far the stylus or cursor is from the graphical tablet. Sometimes, the user may not want to enter stylus position into the computer. In this case, the user can take out the stylus or off the tablet by pressing button provided on the stylus. Figure 1 shows the graphic tablet and stylus.

Figure 2.1: Digitizer or Stylus

b) ***Keyboard:*** Computer keyboard is typewriter- type device which is primary input device of any computer system. It is used for entering alphabets and numbers. In graphics, it is used to enter data related to any picture such as labels x-y coordinates etc. The layout of the keyboard is like the traditional typewriter, although there are some additional keys provided for performing additional functional. Keyboards are available in various sizes and shapes. Figure 2 shows the standard keyboard.

Figure 2.2 : Keyboard

The keys on the keyboard are as follows:

Sr. No.	Keys	Description
1	Typing Keys	These keys include the letter keys (A-Z) and digit keys (0-9) which give the same layout as that of typewriters.
2	Numeric Keypad	It is used to enter numeric data or cursor movement. It consists of a set of 17 keys that are laid out in the same configuration used by most adding machines and calculators.
3	Function Keys	The twelve function keys are present on the keyboard which are arranged in a row at the top of the keyboard.

		Each function key has a unique meaning and is used for some specific purpose.
4	Control keys	These keys provide cursor and screen control. It includes four directional arrow keys. Control keys also include Home, End, Insert, Delete, Page Up, Page Down, Control(Ctrl), Alternate(Alt), Escape(Esc).
5	Special Purpose Keys	Keyboard also contains some special purpose keys such as Enter, Shift, Caps Lock, Num Lock, Spacebar, Tab, and Print Screen.

When we press a key on the keyboard, keyboard controller sends information to the keyboard buffer. This information is called scan code. The keyboard controller informs CPU about the pressed key with the help of an interrupt signal, and then CPU reads scan code from the keyboard buffer, shown in figure 3.

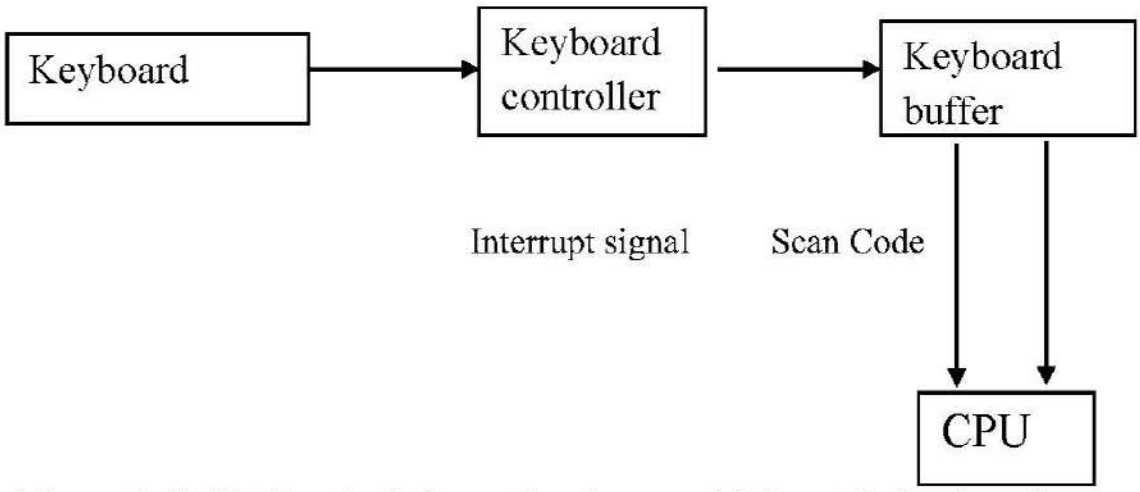


Figure 2. 3: Getting the information (scan code) from the keyboard.

c) ***Mouse:*** Mouse is a most popular pointing device. It is a very famous cursor-control device having a small palm size box with around ball at its base which senses the movement of the mouse and sends corresponding signals to CPU when the mouse buttons are pressed. It has two buttons called left and right button and a wheel is present between the buttons. The mouse can be used to control the position of the cursor on the screen, but it cannot be used to enter text into the computer. Figure 4 shows the mouse.

Figure 2. 4: Mouse with scrolling wheel.

d) ***Joystick:*** Joystick is also a pointing device which is used to move cursor position on a monitor screen. It is a stick having a spherical ball at its both lower and upper ends. The lower spherical ball moves in a socket. The joystick can be moved in all four directions. The left or right movement is indicated by one potentiometer and forward or backward movement is indicated by another potentiometer shown in figure 5. Thus with a joystick, both x & y-coordinate position can be simultaneously altered by the motion of a single stick. The function of the joystick is similar to that of a mouse. It is mainly used in Computer Aided Designing (CAD) and playing computer games.

Figure 2.5: Joystick

e) ***Scanner:*** Scanner is an input device which works more like a photocopy machine. It is used when some information is available on a paper and it is to be transferred to the hard disc of the computer for further manipulation. The scanner captures images from the source which are then converted into the digital form that can be stored on the disc. The scanner uses the optical scanning mechanism to scan the pictures. The scanner records the gradation of gray or color and stores them in the array. Finally, it stores the image information in a specific file format such as JPEG, GIF, TIFF, BMP etc. Image of scanner is shown in figure 6. These images can be edited when they are stored such as rotate, resize, crop, scale using image processing software like Photo-Shop, paint.

Figure 2.6 : Scanner

Scanners are available in a variety of capabilities: Flatbed scanner, Sheet-fed scanner, handheld scanner, Drum scanner etc.

f) ***Trackball:*** Trackball is an input device that is mostly used in notebook or laptop computer, instead of a mouse. This is a ball which is half inserted and by moving fingers on ball, pointer can be moved. Since the whole device is not moved, a trackball requires less space than a mouse. A trackball comes in various shapes like a ball, a button and a square.

Figure 2.7 : Trackball

g) ***Light Pen:*** Light pen is a pointing device which is similar to a pen. It is used to select a displayed menu item or draw pictures on the monitor screen. It consists of a photocell and an optical system placed in a small tube. When the tip of a light pen is moved over the monitor screen and pen button is pressed, its photocell sensing element detects the screen location and sends the corresponding signal to the CPU. Images of light-pens are shown in figure 8.

Figure 2.8 : Light Pen

2.2.2 Output Devices

The output devices can be classified as display devices and hardcopy devices. Display devices are monitors and hardcopy devices are graphic plotters and printers. Following are few of the important output devices which are used in a computer.

a) Monitors

Monitors, commonly called as Visual Display Unit (VDU), are the main output device of a computer. It forms images from tiny dots, called pixels that are

arranged in a rectangular form. The sharpness of the image depends upon the number of pixels. There are two kinds of viewing screen used for monitors.

Cathode-Ray Tube (CRT) Monitor

The CRT display is made up of small picture elements called pixels. The smaller the pixels, the better the image clarity, or resolution. It takes more than one illuminated pixel to form whole character, such as the letter 'e' in the word help.

A finite number of characters can be displayed on a screen at once. The screen can be divided into a series of character boxes - fixed location on the screen where a standard character can be placed. Most screens are capable of displaying 80 characters of data horizontally and 25 lines vertically. There are some disadvantages of CRT:

- Large in Size
- High power consumption

Figure 2.9 : Cathode Ray Tube Monitor

Flat-Panel Display Monitor

The flat-panel display refers to a class of video devices that have reduced volume, weight and power requirement in comparison to the CRT. You can hang them on walls or wear them on your wrists. Current uses of flat-panel displays include calculators, video games, monitors, laptop computer, graphics display.

The flat-panel display is divided into two categories:

- **Emissive Displays** - The emissive displays are devices that convert electrical energy into light. Example are plasma panel and LED (Light-Emitting Diodes).
- **Non-Emissive Displays** - The Non-emissive displays use optical effects to convert sunlight or light from some other source into graphics patterns. Example is LCD(Liquid-Crystal Device)

Figure 2.10 : Flat-panel displays monitor (LCD)

b) Printers

Printer is an output device, which is used to print information on paper. There are two types of printers:

- Impact Printers
- Non-Impact Printers

Impact Printers

The impact printers print the characters by striking them on the ribbon which is then pressed on the paper. Characteristics of Impact Printers are the following:

- Very low consumable costs
- Very noisy
- Useful for bulk printing due to low cost
- There is physical contact with the paper to produce an image

These printers are of two types

1. ***Character printers:*** Character printers are the printers which print one character at a time.

These are further divided into two types:

- ***Dot Matrix Printer (DMP):*** In the market one of the most popular printers is Dot Matrix Printer. These printers are popular because of their ease of printing and economical price. Each character printed is in the form of pattern of dots and head consists of a Matrix of Pins of size (5*7, 7*9, 9*7 or 9*9) which comes out to form a character that is why it is called Dot Matrix Printer.

Advantages

- Inexpensive
- Widely Used
- Other language characters can be printed

Disadvantages

- Slow Speed
- Poor Quality

Figure 2.11a : Dot Matrix Printer

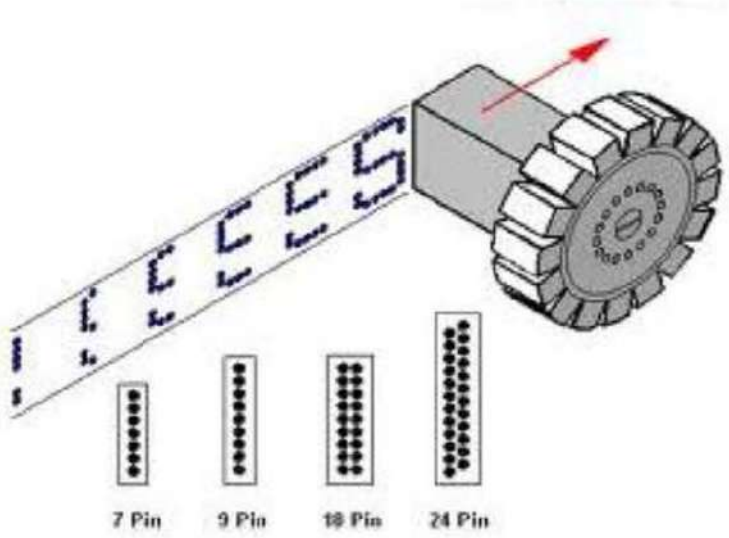


Figure 2.11b: Pins on head of a Dot Matrix Printer

- ***Daisy Wheel:*** Head is lying on a wheel and pins corresponding to characters are like petals of Daisy (flower name) that is why it is called Daisy Wheel Printer. These printers are used for word-processing in offices which require a few letters to be sent here and there with very nice quality.

Advantages

- More reliable than DMP
- Better quality
- The fonts of character can be easily changed

Disadvantages

- Slower than DMP
- Noisy
- More expensive than DMP

Figure 2.12a: Daisy Wheel Printer

Figure 2.12b: Daisy Wheel

2. ***Line printers***

Line printers are the printers which print one line at a time. These are of further two types

Drum Printer: This printer is like a drum in shape so it is called drum printer. The surface of drum is divided into number of tracks. Total tracks are equal to size of paper i.e. for a paper width of 132 characters, drum will have 132 tracks. A character set is embossed on track. The different character sets available in the market are 48 character set, 64 and 96 characters set. One rotation of drum prints one line. Drum printers are fast in speed and can print 300 to 2000 lines per minute. **Advantages**

- Very high speed

Disadvantages

- Very expensive
- Characters fonts cannot be changed

➢

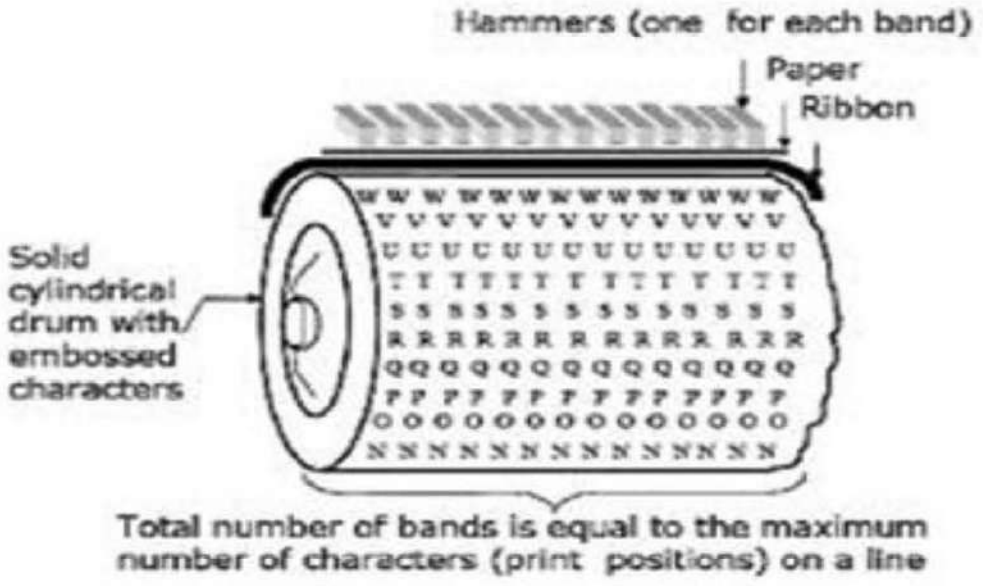


Figure 2.13: Drum Printer

➢ *Chain Printer*

In this printer, chains of character sets are used so it is called Chain Printer.

A standard character set may have 48, 64, or 96 characters.

Advantages

- Character fonts can easily be changed.
- Different languages can be used with the same printer.

Disadvantages

- Noisy

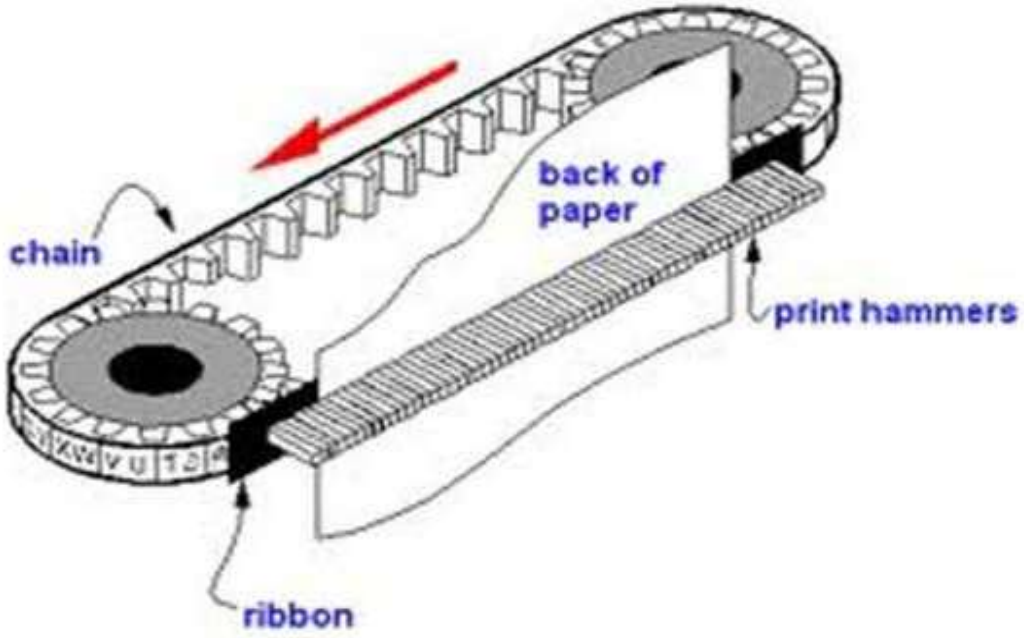


Figure 2.14: Chain Printer

Non-impact Printers

Non-impact printers print the characters without using ribbon. These printers print a complete page at a time so they are also called as Page Printers. These printers are of two types:

1. *Laser Printers:* These are non-impact page printers. They use laser lights to produce the dots needed to form the characters to be printed on a page.

   Advantages

   - Very high speed
   - Very high quality output

- Give good graphics quality
- Support many fonts and different character size

Disadvantages

- Expensive.
- Cannot be used to produce multiple copies of a document in a single printing.

Figure 2.15: Laser Printer

2. ***Inkjet Printers:*** Inkjet printers are non-impact character printers based on a relatively new technology. They print characters by spraying small drops of ink onto paper. Inkjet printers produce high quality output with presentable features. They make less noise because no hammering is done and these have many styles of printing modes available. Color printing is also possible. Some models of Inkjet printers can produce multiple copies of printing also.

Advantages

- High quality printing

- More reliable

Disadvantages

- Expensive as cost per page is high
- Slow as compared to laser printer

Figure 2.16: Inkjet Printer

3. ***Thermal Printers:*** A printer that uses heat to transfer an impression onto paper. It uses continues sheet of paper. Commercial applications of thermal printers include filling station pumps, point of sale systems, voucher printers in slot machines, printing labels for products, and for printing reading of ECG machine in hospitals.

Figure 2.17 (a): Thermal Printer

Figure 2.17 (b): Thermal Printer

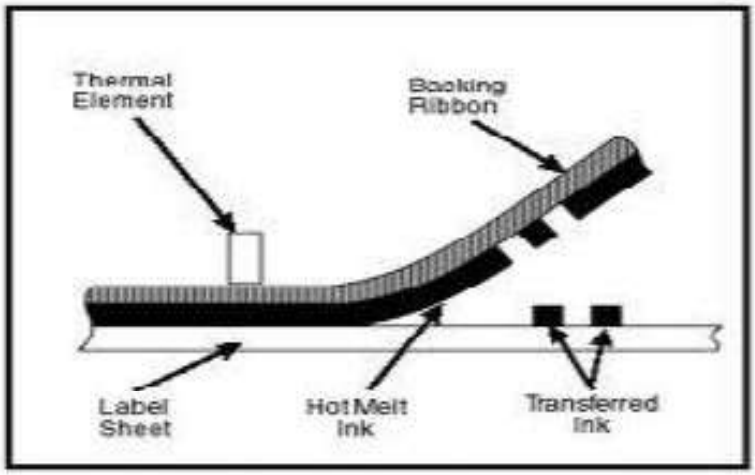


Figure 2.17c: Thermal Transfer Printing

Characteristics of Non-Impact Printers

- Faster than impact printers.
- They are not noisy.
- High quality.
- Support many fonts and different character size.

2.3 Cathode Ray Tube

The primary output device in a graphical system is the video monitor. The main element of a video monitor is the **Cathode Ray Tube (CRT),** shown in the figure 18.

The operation of CRT is very simple −

- The electron gun emits a beam of electrons (cathode rays).
- The electron beam passes through focusing and deflection systems that direct it towards specified positions on the phosphor-coated screen.
- When the beam hits the screen, the phosphor emits a small spot of light at each position contacted by the electron beam.
- It redraws the picture by directing the electron beam back over the same screen points quickly.

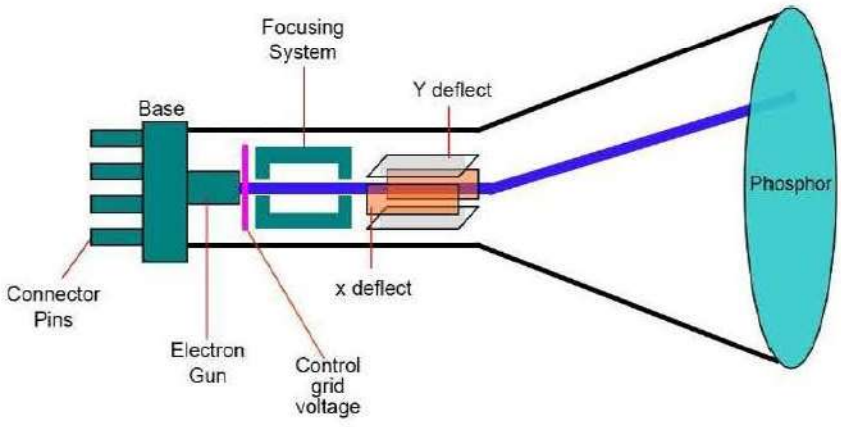


Figure 2.18: Cathode Ray Tube

The deflection system of the cathode-ray-tube consists of two pairs of parallel plates known as vertical and horizontal or y and x deflection plates respectively. The voltage applied to vertical plates controls the vertical deflection of the electron beam and voltage applied to horizontal plates controls the horizontal deflection of the electron beam. There are two techniques used for producing an image on the CRT screen: vector/ random scan and raster scan.

2.4 Direct View Storage Tube

An alternative method for maintaining a screen image is to store the picture information inside the CRT instead of refreshing the screen. A direct-view storage tube (DVST) stores the picture information as a charge distribution just behind the phosphor-coated screen. Two electron guns are used in a DVST. One, the primary gun, is used to store the picture pattern; the second, the flood gun, maintains the picture display as shown in the figure below.

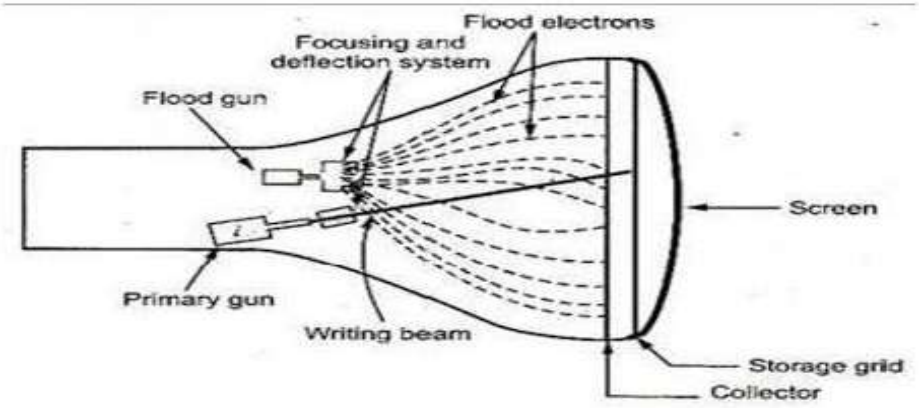


Figure 2.19: Working of Direct View Storage Tube

When high speed electrons hit the storage grid, it displaces the electrons creating a positive charge. The purpose of storage grid is to store image info in the form of

charge distribution. The displaced electrons are attracted towards the collector. A flood gun is used for picture display. Now, the continuous flowing slow speed electrons from flood electron gun are attracted to the positively charged regions of the storage grid. They penetrate the storage grid and hit the phosphor coating in CRT generating the output. Here, the collector is used to control the flow of flood electrons.

A DVST monitor has both disadvantages and advantages compared to the refresh CRT. Because no refreshing is needed, very complex pictures can be displayed at very high resolutions without flicker. Disadvantages of DVST systems are that they ordinarily do not display color and that selected parts of a picture cannot be erased. To eliminate a picture section, the entire screen must be erased and the modified picture is redrawn. The erasing and redrawing process can take several, seconds for a complex picture. For these reasons, storage displays have been largely replaced by raster systems.

2.5 Calligraphic or Random Scan Display System

In this technique, the electron beam is directed only to the part of the screen where the picture is to be drawn rather than scanning from left to right and top to bottom as in raster scan. It is also called **vector display, stroke-writing display,** or **calligraphic display**.

Picture definition is stored as a set of line-drawing commands in an area of memory referred to as the **refresh display file**. To display a specified picture, the system cycles through the set of commands in the display file, drawing each component line in turn. After all the line-drawing commands are processed, the system cycles back to the first line command in the list.

Random-scan displays are designed to draw all the component lines of a picture 30 to 60 times each second. Random scan displays are designed for **line-drawing applications** and cannot display realistic shaded scenes

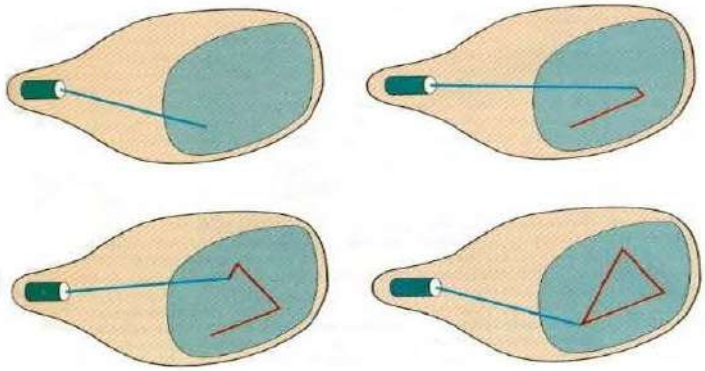


Figure 2. 20: Random Scan

Advantages

- Random scan displays have higher resolution than raster systems.
- Vector displays produce smooth line drawing.
- This minimal amount of information translates to a much smaller file size. (file size compared to large raster images)
- It remains smooth on zooming.

2.6 Raster Scan Display System

In a raster scan system, the electron beam is swept across the screen, one row at a time from top to bottom. As the electron beam moves across each row, the beam intensity is turned on and off to create a pattern of illuminated spots.

Picture definition is stored in memory area called the **Refresh Buffer** or **Frame Buffer**. This memory area holds the set of intensity values for all the screen points. Stored intensity values are then retrieved from the refresh buffer and "painted" on the screen one row (scan line) at a time as shown in the following illustration.

Each screen point is referred to as a **pixel (picture element)**. At the end of each scan line, the electron beam returns to the left side of the screen to begin displaying the next scan line.

The quality of a raster image is determined by the total number pixels (**resolution**), and the amount of information in each pixel (**color depth**). A black-and-white system: each screen point is either on or off, so only **one bit** per pixel is needed to control the intensity of screen positions. Such type of frame buffer is called Bitmap. High quality raster graphics system has **24** bits per pixel in the frame buffer (a **full color** system or a **true color** system). Refreshing on raster scan displays is carried out at the rate 60 to 80 frame per second.

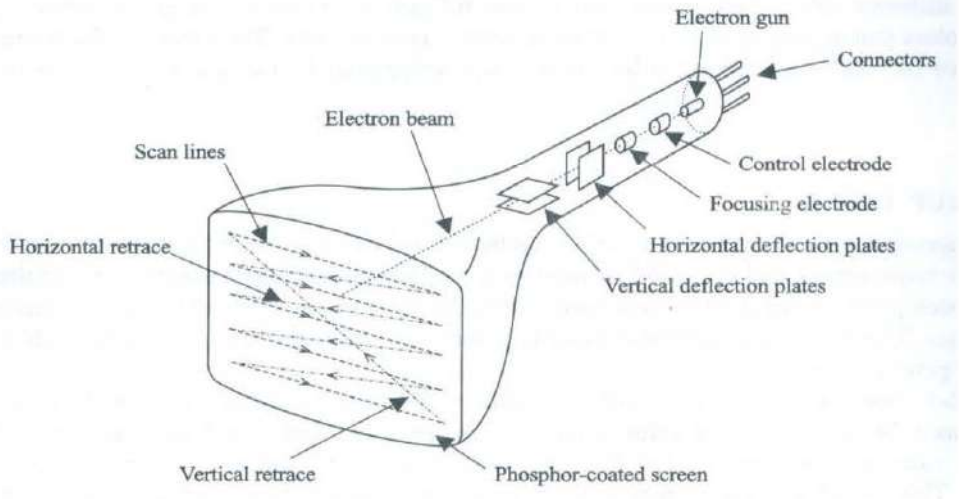


Figure 2. 21: Raster Scan Graphics System

Disadvantages

- To increase size of a raster image the pixels defining the image are be increased in either number or size Spreading the pixels over a larger area causes the image to lose detail and clarity.
- Produces jagged lines that are plotted as discrete points

Difference between Raster Scan System and Random Scan System

Base of Difference	Raster Scan System	Random Scan System
Electron Beam	The electron beam is swept across the screen, one row at a time, from top to bottom.	The electron beam is directed only to the parts of screen where a picture is to be drawn.
Resolution	Its resolution is poor because raster scan in contrast produces zig-zag lines that are plotted as discrete point sets.	Its resolution is good because this system produces smooth lines drawings because CRT beams directly follows the line paths.
Picture Definition	Picture definition is stored as a set of intensity values for all screen points, called pixels in a refresh buffer area.	Picture definition is stored as a set of line drawing instructions in a display file.
Realistic Display	The capability of this system to store intensity values for pixels makes it well suited for the realistic display of scenes contain shadow and color pattern.	These systems are designed for line-drawing and can't display realistic shaded scenes.
Draw an Image	Screen pixels are used to draw an image.	Mathematical functions are used to draw an image.

Cost	Cost is low.	Cost is more.

2.7 Summary

In this chapter, we have studied the major hardware and software features of computer graphics systems. Hardware components include video monitors, hard-copy devices, keyboards, and other devices for graphics input or output.

Raster: A rectangular array of points or dot. A raster image is a collection of dots called pixels. An image is subdivided into a sequence of (usually horizontal) strips known as "scan lines" which can be further divided into discrete pixels for processing in a computer system.

Random: Random scan display is the use of geometrical primitives such as points, lines, curves, and polygons.

Many other video display devices are available. In particular, flat-panel display technology is developing at a rapid rate, and these devices may largely replace raster displays in the near future. Other display technologies include three-dimensional and stereoscopic viewing systems. Virtual reality systems can include either a stereoscopic headset or a standard video monitor.

The most popular "pointing" device is the mouse, but trackballs, space-balls, joysticks, cursor-control keys, and thumbwheels are also used to position the screen cursor. Other input devices include image scanners, digitizers, touch panels, light pens, and voice systems. Hard-copy devices for graphics workstations include standard printers and plotters, in addition to devices for producing slides, transparencies, and film output. Printing methods include dot matrix, laser, ink jet,

electrostatic, and electro-thermal. Plotter methods include pen plotting and combination printer-plotter devices.

2.8 Self Learning Exercise

Q.1 The quality of an image depends on

a) No. of pixel used by image
b) No. of line used by image
c) No. of resolution used by image
d) None

Q.2 Graphics and image processing technique used to produce a transformation of one object into another is called

a) Animation
b) Morphing
c) Half-toning
d) None of the above

Q.3 The amount of light emitted by the phosphor coating depends on the?

a) Speed of electrons striking the screen
b) Number of electrons striking the screen
c) Distance from the cathode to the screen
d) None of above

Q.4 Vector graphics is composed of

a) Pixels
b) Palette`
c) Paths
d) None of these

Q.5 Raster graphics are composed of

a) Pixels
b) Paths
c) Palette
d) None of these

Q.6 Random scan systems are designed for

a) Pixel drawing application
b) Color drawing application
c) Line drawing application
d) None of these

Q.7 A major disadvantage of DVST in interactive computer graphics is

a) Ability to selectively erase part of an image
b) Inability to selectively erase part of image from screen
c) Inability to produce bright picture
d) None

Q.8 The basic transformations include

a) Translation

b) Rotation

c) Scaling

d) All of the above

Q.9 Match the following

Part A	Part B
A. Plasma panel	i) Polarizer
B. DVST	ii) Zinc sulfide
C. LCD	iii) Dielectric mesh
D. Thin film electroluminescent	iv) Neon gas

a) A-ii, B-iv, C-i, D-iii

b) A-ii, B-iii, C-iv, D-i

c) A-iv, B-iii, C-i, D-ii

d) A-i ,B-iv, C-ii, D-iii

Q.10 The purpose of flood gun in DVST is.....

a) To store the picture pattern

b) To slow down the flood electrons

c) To enable color pixels

d) To focus the electron beam

2.9 Excercise

Q.1 What is refresh buffer?

Q.2 Write short notes on graphical input and 0output devices.

Q.3 Explain Direct view storage tube with its advantages.

Q.4 Write differences between Raster scan system and Random scan system.

Q.5 Give the classification of printers and explain various types of printers.

Q.6 Write a short note on Cathode ray tube.

2.10 Answers to Self-Learning Exercise

1- (a) 2-(a) 3-(b) 4-(c) 5-(a) 6-(c) 7-(b) 8-(d) 9-(c) 10-(b)

UNIT-3

Color CRT Monitor

Structure of the Unit

3.0 Objective

In this chapter, we shall focus on the following topics

- CRT monitor
- Color CRT Monitor
- Beam Penetration Method
- Shadow Mask Method

3.1 Introduction

Monitor is known as computer screen. "monitor" having the whole piece of equipment's, not just the screen part. Sometimes a monitor is called display.

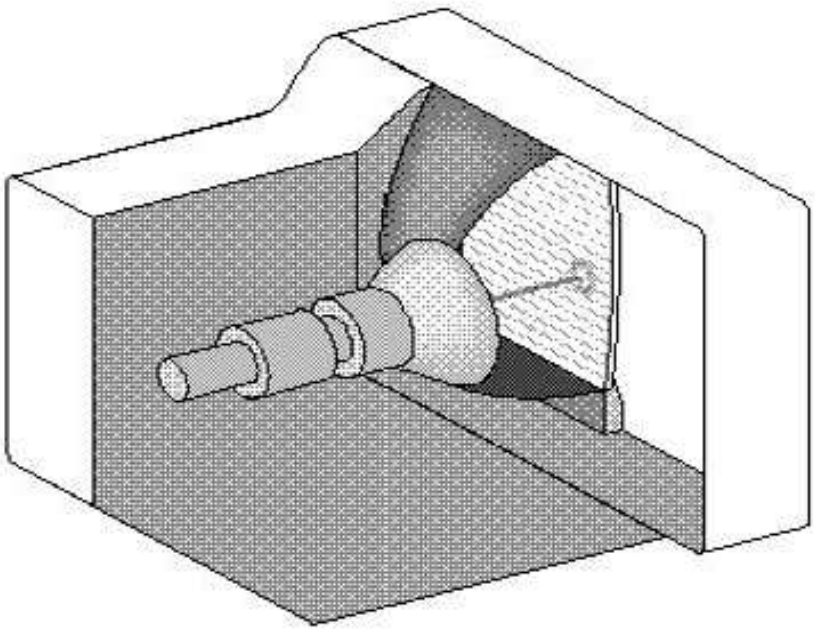


Figure 3.1: Computer Monitor

Since the mid-1900s, the Cathode-Ray Tube on the other hand CRT (in some cases called the Braun Tube) has had vital impact in showing pictures, films, and data. A patent was documented in 1938 for the CRT; be that as it may, this was an exceptionally straightforward usage. After some time, CRTs have progressed utilizing a wide range of systems to expand picture accuracy furthermore, quality. While the innovation has been around for a very long while and is entirely experienced, despite everything it has much opportunity to get better.

Advantage and Drawback of the CRT

Advantage

- High resolution technology
- Long life and reliability
- Bright display
- Excellent contrast/gray scale

- Inexpensive
- Broad application range

Drawback

- Consume large power
- weights
- Size of footprint

3.2 CRT monitors

The cathode ray tube (CRT) is a vacuum tube in which picture is delivered when an electron shaft strikes phosphorescent surface. Maximum desktop computers use of CRTs. A CRT in a PC presentation is like the "picture tube" in a TV input.

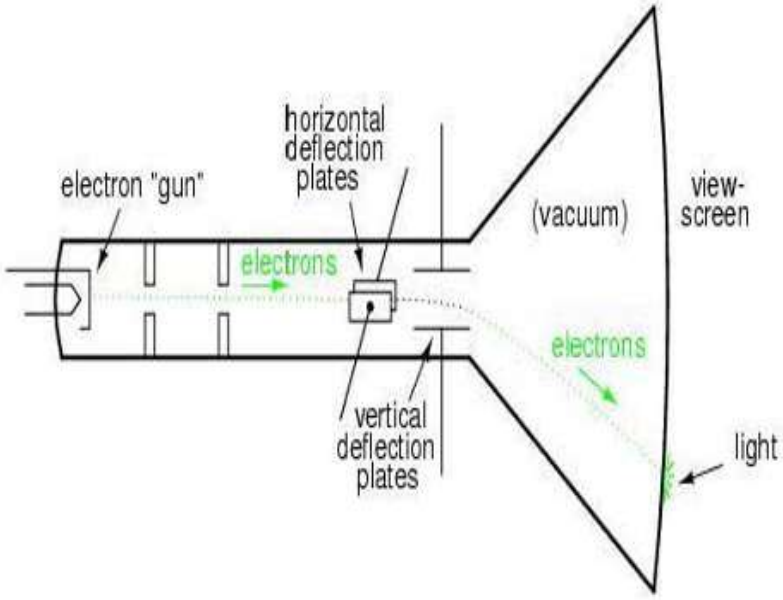


Figure 3.2: CRT Monitor

A cathode beam tube comprises of a few basic components, as outlined beneath. The electron firearm creates an arrow light emission. The anodes quicken the

electrons. Deflecting curls create a to a great degree low recurrence electromagnetic field that allows for and only arrangement of curls is indicated for simplicity.) The power of the pillar can be swing. Electron shaft delivers a modest,

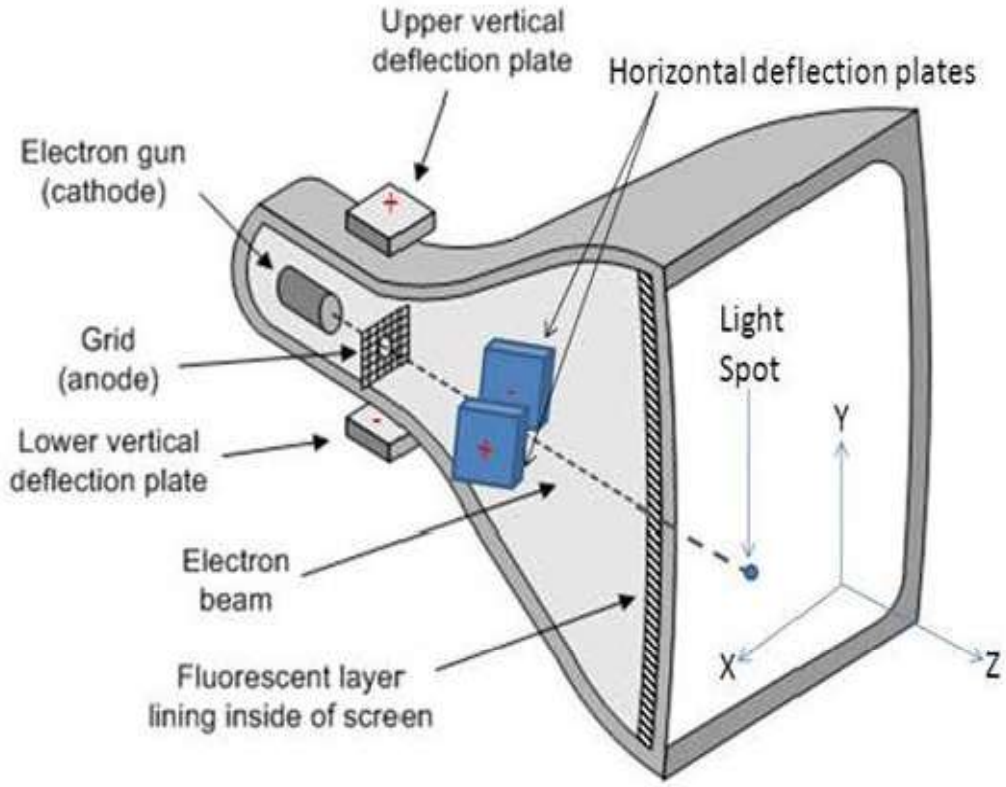


Figure 3.3: Cross sectional representation of cathode ray tube

brilliant observable spot when it strike phosphor-secured screen. The keeping away from loops, moreover to the contraption that controls the force of the electron pillar. Due to this the spot to race over the screen from right to left, and all the way, in a course of action of level lines called raster. As seen from front of CRT, spot move in pattern similar the way our eyes move when we read single-section page of content. In any case, the examining happens at such quick rate that our eyes sees a consistent picture over the full screen.

The representation indicates one and only electron gun. It is ordinary of a monochrome, or single-shading, CRT. However, virtually all CRTs today render shading pictures. These gadgets have three electron weapons, one for the essential shading red, one for the primary shading green, and one for the essential shading blue. The CRT thus produces three covering pictures: one in red (R), one in green (G), and one in blue (B). This is the purported RGB color model.

In PC frameworks, there are a few presentation modes, or sets of determinations as per which the CRT works. The most common particular for CRT showcases is known as SVGA (Super Video Graphics Array). Journal PCs regularly utilize fluid precious stone display. The innovation for these showcases is vastly different than that for CRTs.

Advantages:

a) They work at any determination, geometry and angle proportion without the requirement for rescaling the picture.

b) CRTs keep running at the most elevated pixel resolutions for the most part accessible.

c) Produce an exceptionally dull dark and the most astounding complexity levels ordinarily accessible. Appropriate for use even in faintly lit or dim situations.

d) CRTs create the absolute best shading and dim scale and are the reference standard for every single proficient alignment. They have a superbly smooth dark scale with a vast number of force levels. Other presentation advances are relied upon to replicate the characteristic force law Gamma bend of a CRT, yet can just do as such roughly.

e) CRTs have quick reaction times and no movement ancient rarities. Best for quickly moving or evolving pictures.

f) CRTs are less costly than practically identical showcases utilizing other presentation advances.

Drawbacks:

a) The CRT's Gaussian shaft profile produces pictures with gentler edges that are not as sharp as a LCD at its local determination. Defective center and shading enrolment additionally diminish sharpness. For the most part more keen than LCDs at other than local resolutions.

b) All shading CRTs produce irritating Moiré designs. Numerous screens incorporate Moiré decrease, which typically doesn't kill the Moiré obstruction designs altogether.

c) Subject to geometric bending and screen control issues. Likewise influenced by attractive fields from other hardware including different CRTs.

d) Relatively splendid yet not as brilliant as LCDs. Not reasonable for splendidly lit situations.

e) Some CRTs have an adjusted circular or round and hollow shape screen. More current CRTs are level.

f) CRTs radiate electric, attractive and electromagnetic fields. There is significant debate in the matter of whether any of these represent a wellbeing risk, especially attractive fields. The most definitive logical studies presume that they are not unsafe but rather a few people stay unconvinced.

g) They are extensive, overwhelming, and cumbersome. They devour a great deal of power and create a ton of warmth.

3.3 Color CRT Monitors

Color Monitor was one the earliest CRTs to generate color display. Coated phosphors of various compounds can generate various colored picture. But main problem of graphics is not to generate a picture of a predetermined colors, but to generate color pictures, with the color characteristics opt at run time.

The main principle of colored displays is that combination of the three basic colors –Red, Blue and Green, can generate each and every color. By opt different ratio of three colors you can generate different colors. The basic phosphors can produce these three colors. So, one should have a technique to merge them in various combinations.

- Color CRT's are designed as RGB monitors also called full color system or true color system.
- Frame buffer contains 24 bits per pixel, for 256 voltage settings to adjust the intensity of each electron beam, thus producing a choice of up to 17 million colors for each pixel.

3.4 Various CRT Monitors

A color CRT screen shows shading picture by utilizing a mix of phosphors that emanate diverse hued light. By consolidating the radiated light a scope of hues can be produced. Two essential strategies for delivering shading showcases are:

- Beam Penetration
- Shadow Mask

1. ***Beam Penetration:*** This CRT is like the straightforward CRT, however it makes utilization of multi shaded phosphorus of number of layers. Every phosphorus layer is in charge of one shading. Every other game plan are like straightforward CRT. It can create a most extreme of 4 to 5 hues. The association is something like this - The red, green and blue phosphorus are covered in layers - one behind the other. On the off chance that a low speed shaft strikes the CRT, just the red hued phosphorus is initiated, a marginally quickened bar would actuate both red and green (since it can infiltrate further) and a significantly more enacted one would include the blue part too. Yet, the essential issue is a dependable innovation to quicken the electronic shaft to exact levels to get the definite hues - it is simpler said than done. In any case, a restricted scope of hues can be advantageously created utilizing the idea.

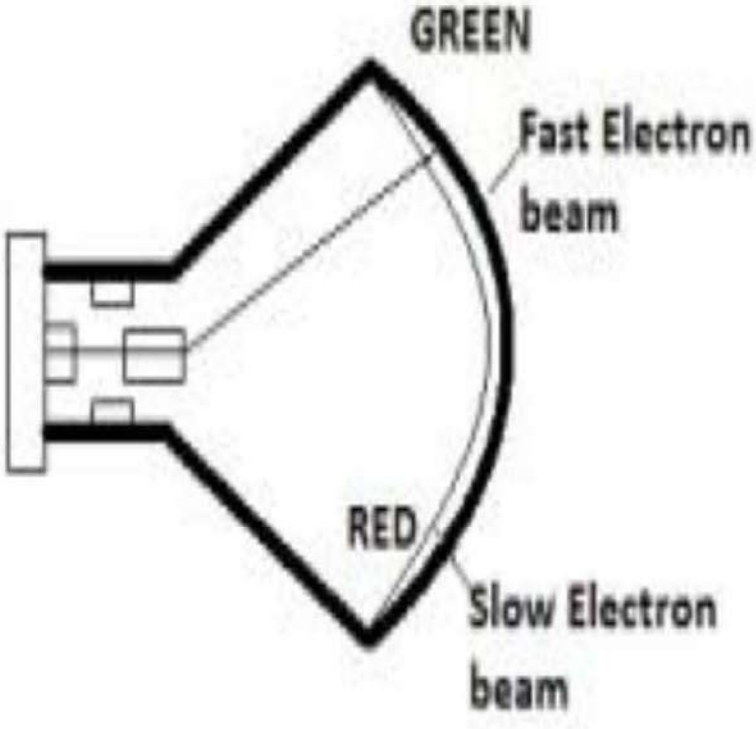


Figure 3.4: Beam Penetration

- Irregular output screens utilize the pillar infiltration strategy for showing shading picture.
- In this, within CRT screen is covered two layers of phosphor to be specific red and green.
- A light emission electrons energizes just the external red layer, while a light emission electrons enters red layer and energizes the internal green layer.
- At middle of the road shaft speeds, mixes of red and green light are discharged to show two extra hues orange and yellow.

Advantages

- Less expensive

Disadvantages

- Quality of images are not good as compared with other methods
- Four colors are allowed only

2. ***Shadow Mask Method:*** In 1938 German creator Werner Flechsig initially licensed (got 1941, France) the apparently basic idea of setting a sheet of metal simply behind the front of the tube, and punching little openings in it. The openings would be utilized to center the pillar just before it hit the screen. Autonomously, Al Schroeder at RCA chipped away at a comparable game plan, yet utilizing three electron firearms too.

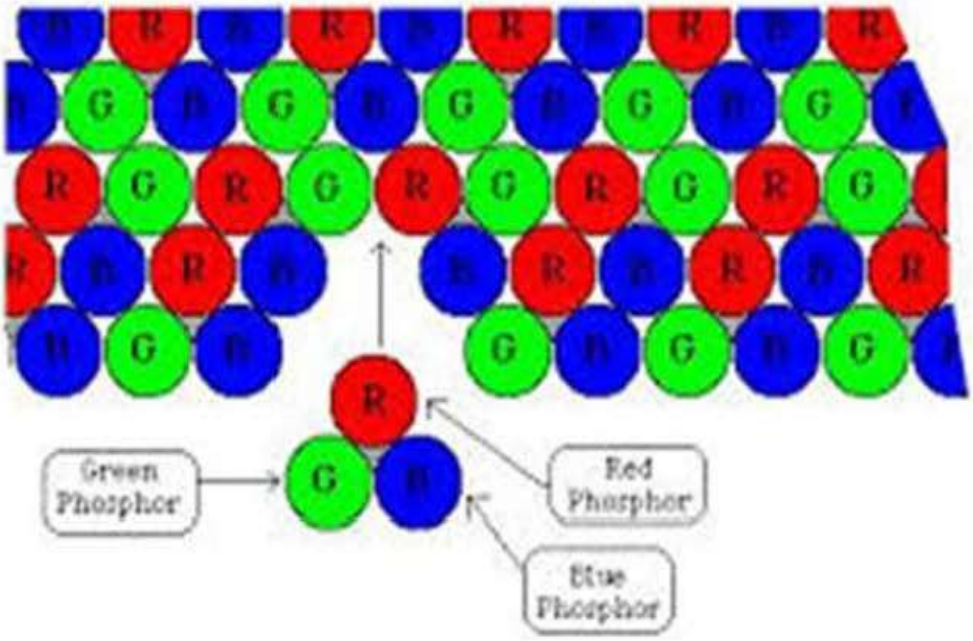


Figure. 3.5: Shadow Mask

At the point when the lab pioneer clarified the potential outcomes of the configuration to his bosses, he was guaranteed boundless labor and subsidizes to get it working. Over a time of just a couple of months, a few model shading TVs utilizing the framework was delivered.

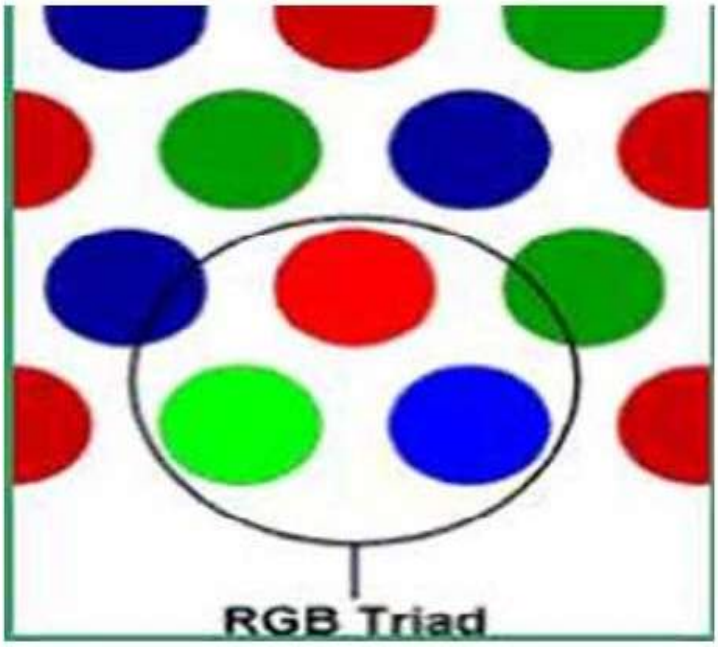


Figure 3.6: Phosphorus dot pattern for a shadow mask CRT

In Shadow Mask CRT modest openings in a metal plate isolate the hued phosphors in the layer behind the front glass of the screen. The openings are set in a way guaranteeing that electrons from each of the tube's three cathode firearms achieve just the suitably hued phosphors on the showcase.

Every one of the three bars go through the same openings in the cover, yet the point of methodology is distinctive for every weapon. The separating of the gaps, the dispersing of the phosphors, and the position of the weapons is masterminded so that for instance the blue firearm just has an unhampered way to blue phosphors. The red, green, and blue phosphors for every pixel are for the most part masterminded fit as a fiddle (in some cases called a "ternion").

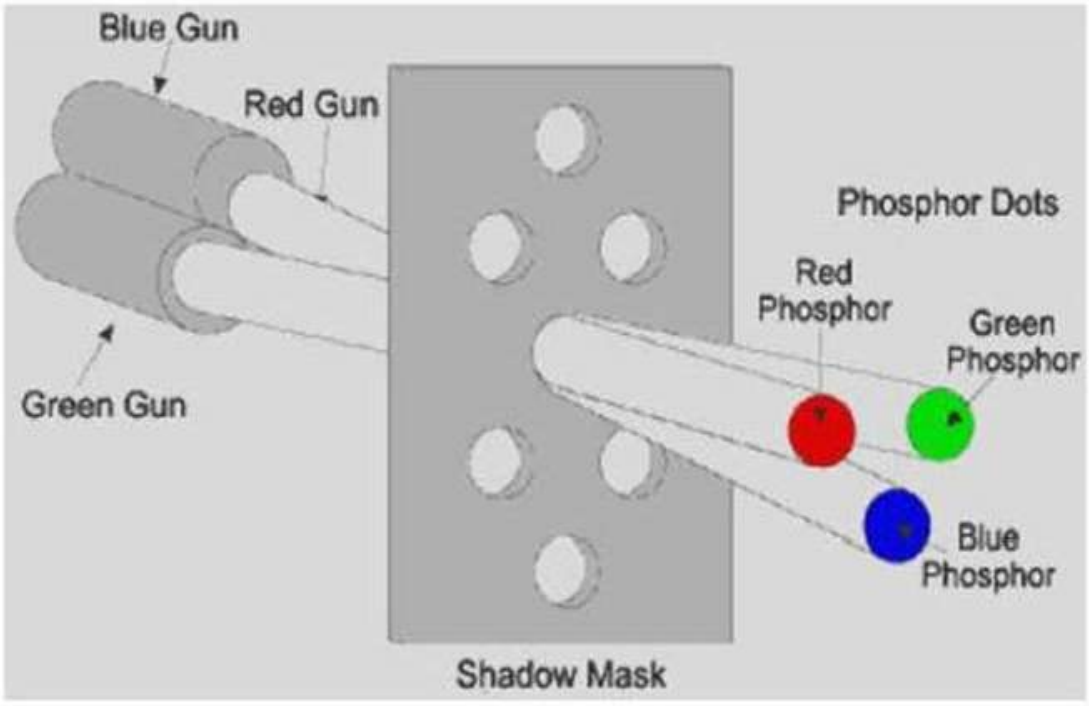


Figure. 3.7

All early shading TVs and the larger part of PC screens, over a wide span of time, use shadow cover innovation. Customarily, shadow covers have been

made of materials which temperature varieties cause to grow and contract to the point of influencing execution.

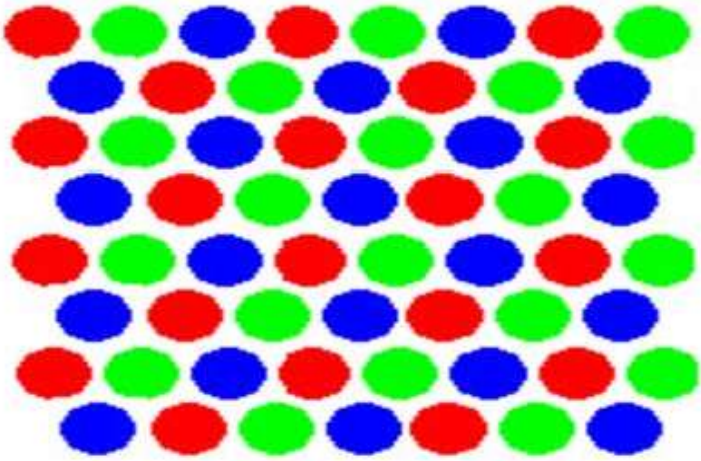


Figure. 3.8

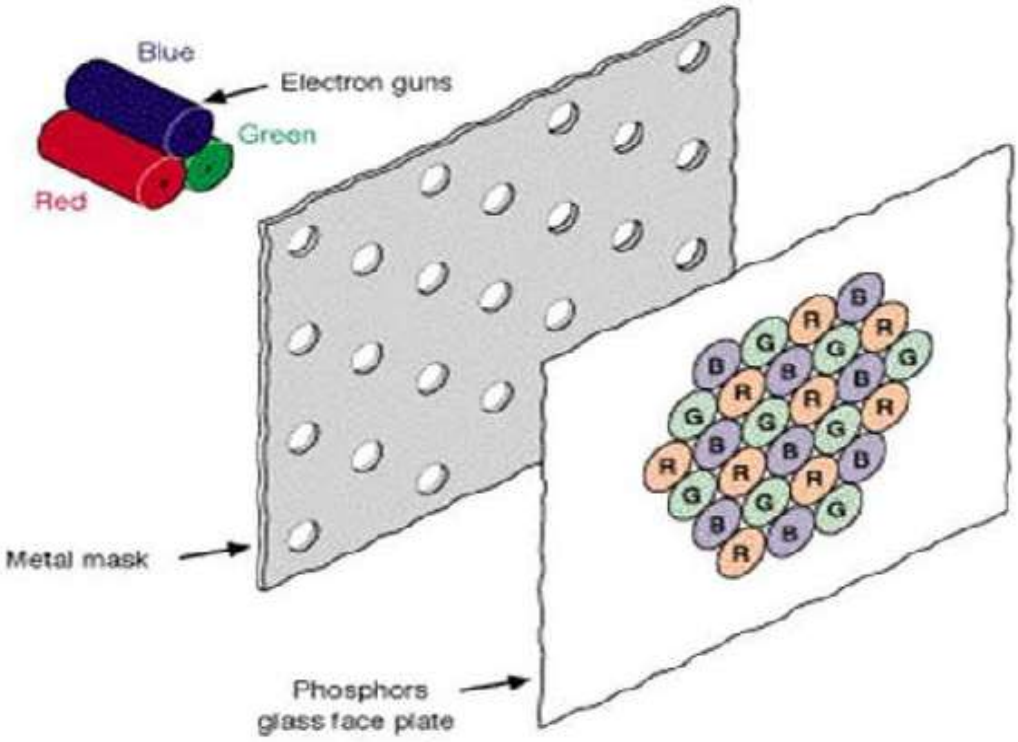


Figure. 3.9

A long these lines it extends and contracts considerably less than different materials in light of temperature changes. This property permits shows made with this innovation to give a clearer, more

The vitality the shadow veil assimilates from the electron weapon in ordinary operation causes it to warm up and grow, which prompts obscured or stained (see doming) pictures. The in var shadow veil is made out of the nickel-iron combination in var.

precise picture. It additionally diminishes the measure of long haul stretch and harm to the shadow veil that can come about because of rehashed extend/contract cycles, in this manner expanding the showcase's future. At the end of the day, In Shadow Mask CRT, before the surge of electrons created by the CRT's cathode achieve the phosphor covered faceplate, it experiences the shadow veil, a sheet of metal scratched with an example of openings.

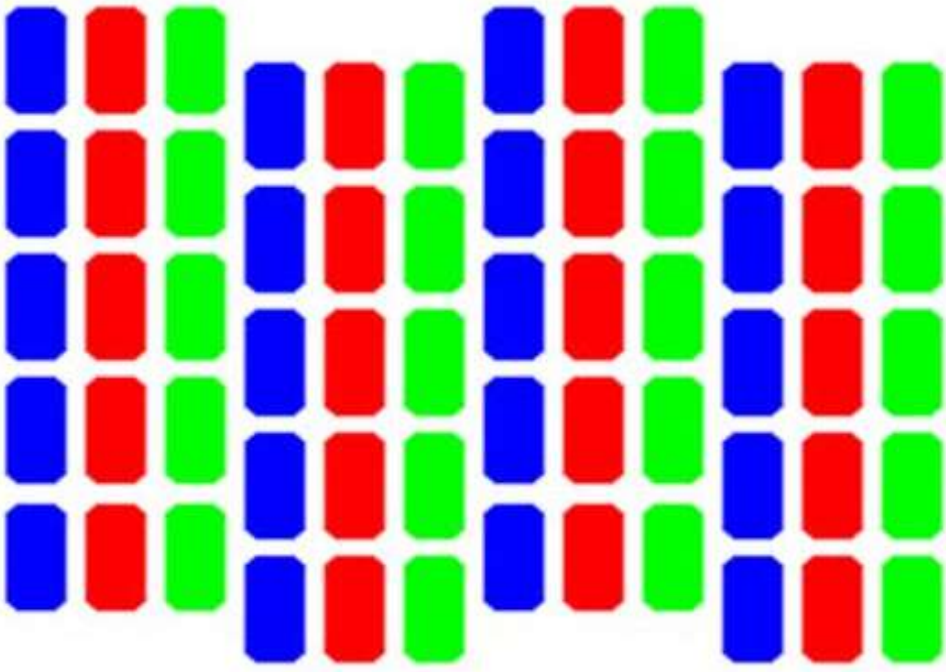


Figure 3.10: In-line Electron Gun Arrangement

Veil situated in glass pipe of CRT amid production and phosphor is cover onto the screen so electron originating from blue red and green firearm position just arrive on the suitable phosphor. Strayed electron strike shadow mask and is absorb by it, producing great deal of heat, which causes metal to expand.

Veil situated in glass pipe of CRT amid production and phosphor is cover onto the screen so electron originating from blue red and green firearm position just arrive on the suitable phosphor. Strayed electron strike shadow mask and is absorb by it, producing great deal of heat, which causes metal to expand.

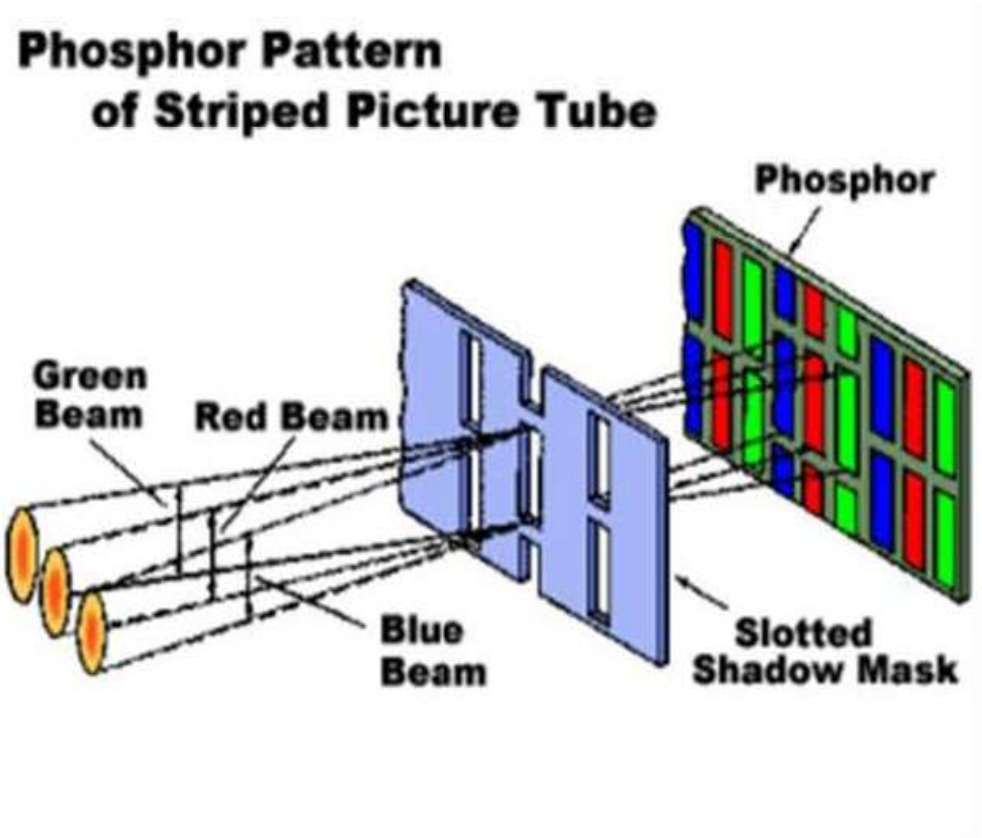


Figure. 3.11: Phosphor Pattern for Stripped Pattern Tube

To allow flat CRT to be made, metals commonly used now for shadow mask is In-var, an alloy of iron and nickel. The metal has a very low coefficient of expansion

and its name derive from the supposed invariability of it dimensions when the heat is applied.

Advantages

- Produce realistic images
- Also produced different colors and shadows scenes.

Disadvantages

- low resolution
- expensive
- electron beam directed to whole screen

3.5 Difference Between Color Monitors

	Beam Penetration	**Shadow Mask**
Which place Used	It use with Random Scan to show color.	It Use With Raster Scan to show color.
Color Display	It displays four colors i.e. Red, Yellow, Orange and Green.	It display Million of colors.
Dependency	Few color are available because the colors in Beam Penetration Strategy depend on speed of electron beam.	Many colors available due to the colors in Shadow Mask strategy depends on type of ray.
Costing	It few Expensive as compare to Shadow Mask strategy	It much expensive than many other methods.
Quality of Picture	Image quality is not good.	This strategy provide realism with shadow effect.

Resolution Of Image	It Provide High Resolution.	It Provide Low Resolution.
Constraint	In this strategy, Color in display depend on distance of electron excites outer Red layer and then Green layer.	There are no such constraint in this strategy for producing colors. This strategy use in computer and color TV etc.

3.6 Summary

CRTs can be helpful for showing photographs with high pixels per unit and adjust color balance. LCDs, as of now the most widely recognized level screen innovation, have for the most part second rate shading version (notwithstanding having more noteworthy general brilliance) because of the bright lights normally utilized as a backdrop illumination.

CRTs are still prominent in the printing and broadcasting enterprises and in addition in the expert video, photography, and illustrations fields because of their more prominent shading constancy, difference and better survey from off-pivot (more extensive review edge). CRTs likewise still discover followers in video gaming due to their higher determination per beginning cost, quick reaction time, and various local resolutions.

3.7 Self Learning Exercise

Q.1 Which of the following type of monitor is common on desktop computers. It looks much like a standard television.

e) Cathode Ray Tube

f) Flat Panel

g) Monochrome

h) Projector

Q.2 The _____ helps to align the electron guns.

a) pixel

b) shadow mask

c) resolution

d) refresh

Q.3 LCD monitors often have a smaller _____ than CRT monitors.

e) Refresh Rate

f) Viewing Rate

g) Color Depth

h) Price

Q.4 This specification of a monitor describes the usable portion of the screen.

a) refresh rate

b) resolution

c) dot pitch

d) viewable area

Q.5 A _____ is similar to the LCD monitor, but has a phosphorescent film between the layers.

e) Electro luminescent displays (ELD)

f) Plasma displays

g) Paper-white displays

h) thin-film transistor

3.8 Review Questions

Q.1 Explain refresh cathode ray tube?

Q.2 Define persistence in terms of CRT Phosphorous.

Q.3 What are the different properties of phosphorus?

Q.4 What is a Beam penetration method?

Q.5 Define shadow masking.

Q.6 Explain color CRT monitors?

Q.7 Give the difference in Beam Penetration and Shadow Mask.

Answers to Self-Learning Exercise

Q.1 (a) Q.2 (b) Q.3 (b) Q.4 (d) Q.5 (a)

References and Suggest Reading

1. "Computer Graphics C version", Donald Hearn and M. Pauline Baker, Pearson Education.
2. "Computer Graphics Principles & Practice", second edition in C, Foley, VanDam, Feiner and Hughes, Pearson Education.

UNIT-4
Output and Input Devices

Structure of the Unit

4.0 Objective

In this chapter, we shall focus on the following topics

- Introduction of input and output devices
- Input Devices
- Output Devices
- Graphical Input Technique

4.1 Introduction

Graphics hardware can be divided into two major categories of devices:

(1) **Input Devices**: By using these devices the user interacts to generate necessary instruction or data for creating graphics.

(2) **Output Devices**: By using these devices the graphics are rendered on the monitor screen or printers through which the tangible graphics output is produced.

4.2 Input Device: An Overview

Input devices are things we use to put information into a computer. An input device is any hardware device that sends data to the computer.

Various hardware devices have been developed to enable the user to interact more naturally. These devices can be separated into two classes.

1. Locators

2. Selectors

1. **Locators:** Locators are the devices which give position information. The computer receives from a locator the co-ordinates for a point. Using a locator, we can indicate a position on the screen.

The different locator's devices are as follows

i. Mouse
ii. Joystick

2. **Selectors**: These devices are used to select the particular graphical object. A selector may pick a particular item, but no information about that item is located on the screen. The different selectors devices are as follows.

i. Light Pen
ii. Keyboard
iii. Trackball
iv. Digitizers
v. Scanner
vi. Touch Panels
vii. Data Globe

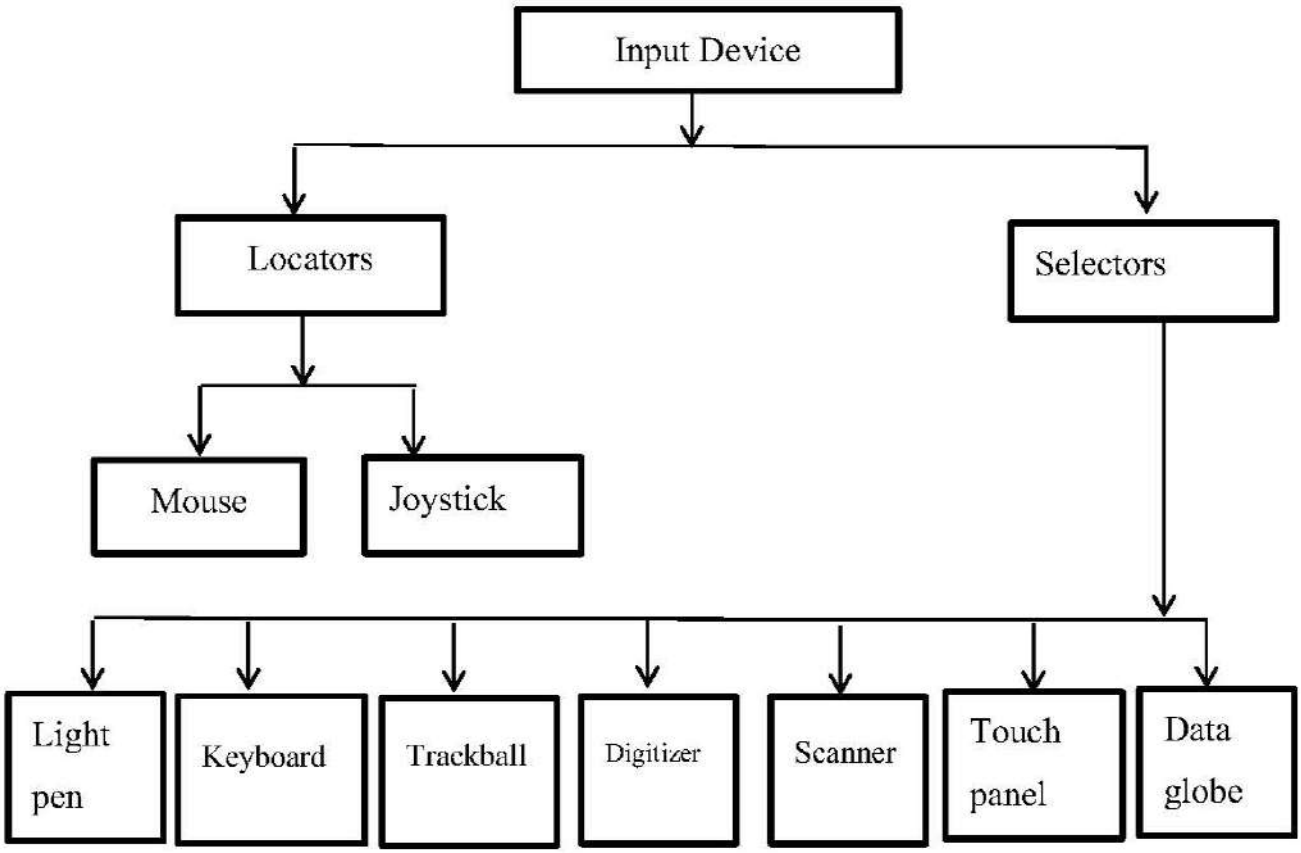


Figure. 4.1

4.3 Various Input Devices

1. **Mouse**:- A mouse is a pointing device that detects two-dimensional motion relative to a surface. The mouse controls the movement of the pointer, also called the mouse pointer, on the screen. Drawing shapes or designing figures using graphic application packages (like AutoCAD, Photoshop, CorelDraw, and Paint) is almost impossible without a mouse.

Figure. 4.2: Mouse

2. **Joystick**: - A joystick is an input device consisting of a stick that pivots on a base and reports its angle or direction to the device it is controlling. A joystick is configured so that moving the stick left or right signals movement along the X-axis, and moving it forward (up) or back (down) signals movement along the Y-axis. Joysticks are used mostly for computer games, but they are also used occasionally for CAD/CAM systems and other applications.

Figure. 4.3: Joystick

3. **Light Pen: -** A light pen is a computer input device in the form of a light sensitive wand used in conjunction with a computer's CRT display. The light pen contains a light-sensitive element (photoelectric cell) which, when placed against the screen, detects the light from the screen enabling the computer to identify the location of the pen on the screen. A light pen can work with any CRT-based monitor, but not with LCD screens.

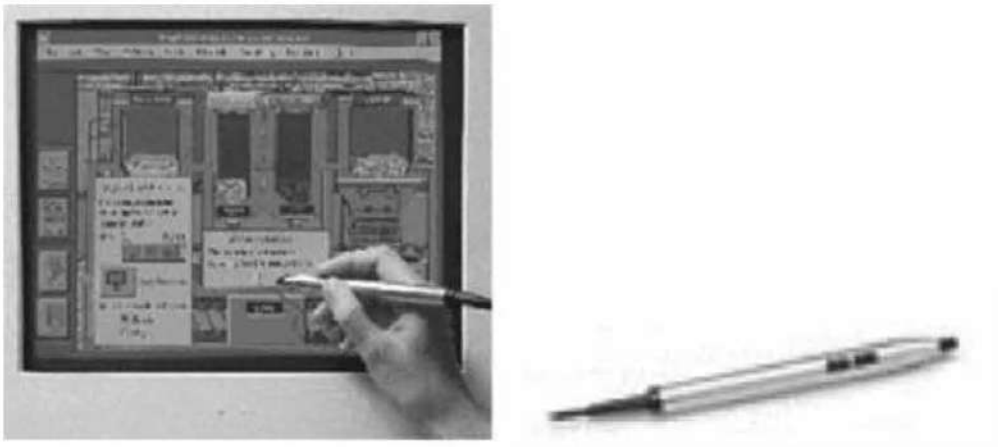


Figure. 4.4: Light Pen

4. **Keyboard**: One of the primary input devices used with a computer. Keyboards allow a computer user to input letters, numbers, and other symbols into a computer. Most keyboards have between 80 and 110 keys, including:

• Alphanumeric keys –all letters and numbers on the keyboard. A-Z and 0-9.

- Punctuation keys – All of the keys associated with punctuation such as the comma, period, semicolon, brackets, and parenthesis and so on. Also, all of the mathematical operators such as the plus sign, minus sign, and equal sign.
- Special keys – All of the other keys on the computer keyboard such as the function keys, control keys, arrow keys, caps lock key, delete key, etc.

Figure.4.5: Keyboard

5. **Trackball**: - A trackball is a pointing input device consisting of a ball held by a socket containing sensors to detect a rotation of the ball about two axes—like an upside-down mouse with an exposed protruding ball. The user rolls the ball with the thumb, fingers, or the palm of the hand to move a cursor. Military mobile anti-aircraft radars and submarine sonar tend to continue using trackballs.

Figure.4.6: **Trackball**

6. **Digitizer**:-It is a common device for drawing, painting, or interactively selecting coordinate positions on an object. Typically, it is used to scan an Object and to input discrete coordinate positions.

A graphics tablet is one such digitizer that consists of a flat surface upon which the user may draw an image using an attached stylus, a pen-like drawing apparatus. Three dimension (3D) Digitizers use sonic or electromagnetic transmissions to record positions.

Figure. 4.7: Digitizer

7. **Scanner**: - A scanner is a device that captures images from photographic prints, posters, magazine pages, and similar sources for computer editing and display. Scanners come in hand-held, feed-in, and flatbed types and for scanning black-and-white only, or color. Very high resolution scanners are used for scanning of high-resolution printing.

Figure. 4.8: Scanner

8. **Touch Panel: -** A touch panel is an electronic visual display that can detect the presence and location of a touch within the display area. Touch screens can sense Finger and other passive objects, such as a stylus. However, if the object sensed is active, as with a light pen.

Touch panels have gained wide acceptance in bank ATMs, video games, and kiosk at railway stations & tourist information centers.

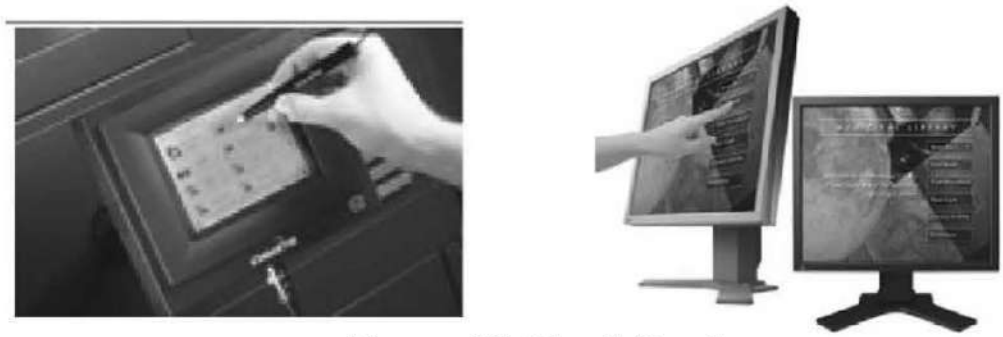


Figure. 4.9: Touch Panel

9. **Data Globe: -** A data glove is an interactive device, resembling a glove worn on the hand, which facilitates tactile sensing and fine-motion control in robotics and virtual reality**.** Fine-motion control involves the use of sensors to detect the movements of the user's hand and fingers, and the translation of these motions into signals that can be used by a virtual hand (for example, in gaming) or a robotic hand (for example, in remote-control surgery). It uses in 3D animation movies and visual effects.

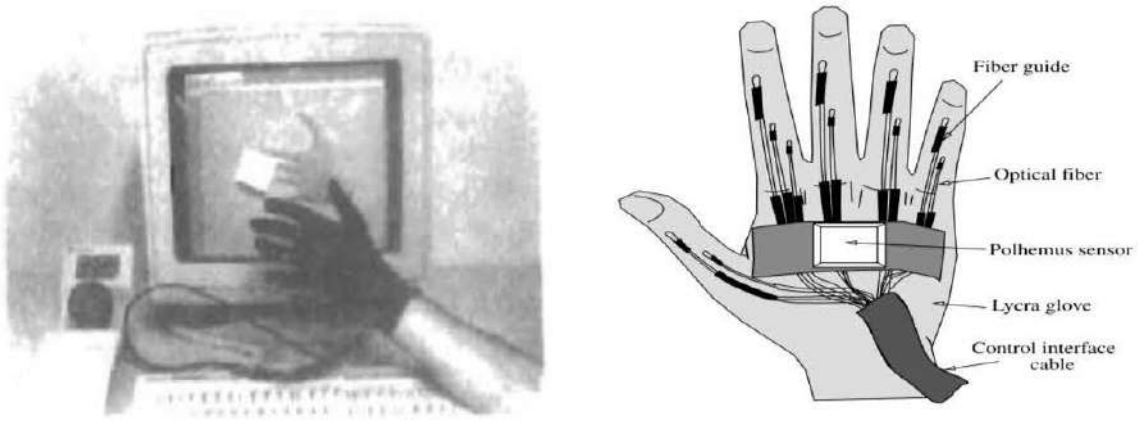


Figure.4.10: Data Globe

4.4 Output Devices

The display devices are known as output devices. The display system may be attached to a PC to display character, picture and video outputs.

The most common output device for interactive computer graphics is the display. Displays consist of different technologies with different characteristics such as size , resolution, color depth, refresh rate, and brightness. Some of the common types of display systems are following.

1. Raster scan display
2. Random scan display
3. Direct view storage tube
4. Flat panel display

Before we discuss the major display systems let us first know about basic terms.

Pixel:- A pixel (short for picture element, using the common abbreviation "pix" for "picture") is one of the many tiny dots that make up the representation of a picture in a computer's memory. Pixels are normally arranged in a regular 2D grid, and are often represented using dots or squares. Each pixel has a particular colour and brightness value. The intensity of each pixel is variable; in color systems, each pixel has typically three or four components such as red, green, and blue, or cyan, magenta, yellow, and black.

Resolution: - Resolution is the term used to describe the number of dots, or pixels, used to display an image. The display, or resolution on a monitor, is composed of thousands of pixels or dots. This display is indicated by a number combination, such as 800 x 600. This indicates that there are 800 dots horizontally across the monitor, by 600 lines of dots vertically, equalling 480,000 dots that make up the

image you see on the screen. Higher resolutions mean that more pixels are used to create the image, resulting in a cleaner image.

Frame Buffer: - A region of memory where the picture definition for the entire screen is stored. A frame buffer is a large, contiguous piece of computer memory. Each screen pixel corresponds to a particular entry in a 2D array residing in memory. This memory is called a frame buffer or a bitmap.

1. Raster scan display: - In a raster-scan system the electron beam is swept across the screen one row at a time from top to bottom.

As the electron beam moves across each row, the beam intensity is turned on and off to create a pattern of illuminated spots. Picture definition is stored in a memory area called the refresh buffer or frame buffer. This memory area holds the set of intensity values for all the screen points.

These stored intensity values are then retrieved from the refresh buffer and used to control the intensity of the electron beam as it moves from spot to spot across the screen. Stored intensity values then retrieved from refresh buffer and "painted" on the screen one row (scan line) at a time.

Sometimes, refresh rates are described in the unit of cycles per second, or Hertz (**Hz**). Refreshing on a raster scan displays is carried out at the rate 60 to 80 frames per second. At the end of each scan line, the electron beam returns to the left side of the screen to begin displaying the next scan line.

Horizontal retrace:- The return to the left of the screen, after refreshing each scan line.

Vertical retrace:- At the end of each frame, the electron beam returns to the top left corner of the screen to begin the next frame.

On some raster-scan systems (and in TV sets), each frame is displayed in two passes using an interlaced refresh procedure. In the first pass, the beam sweeps across every other scan line from top to bottom. Then after the vertical retrace, the beam sweeps out the remaining scan lines. The interlacing of the scan lines in this way allows us to see the entire screen displayed in one-half the time it would have taken to sweep across all the lines at once from top to bottom.

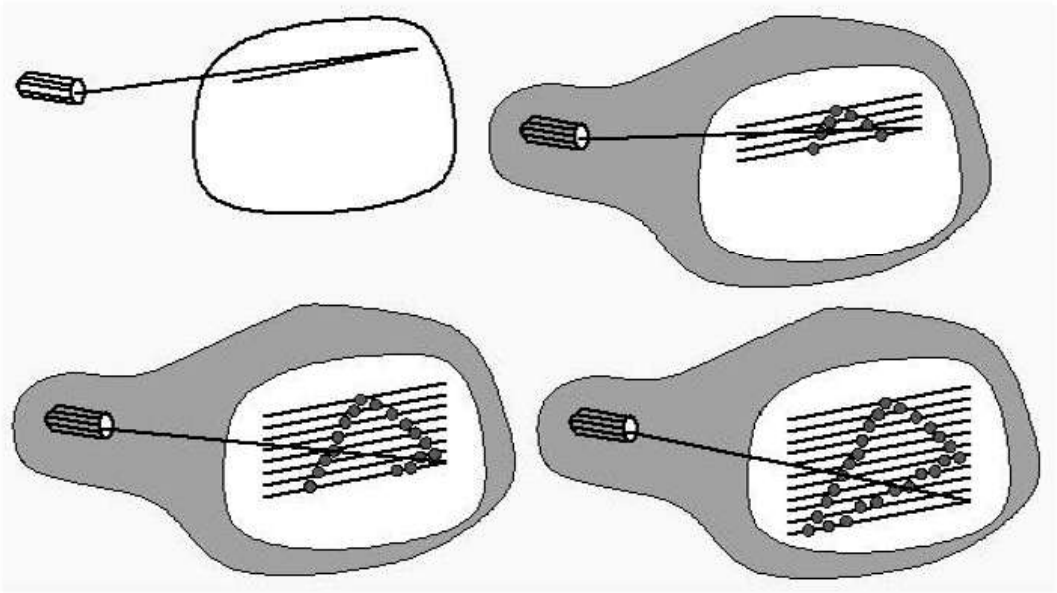


Figure.4.11: Raster scan display

2. **Random scan display:-** Random scan display is the use of geometrical primitives such as points, lines, curves, and polygons, which are all based upon mathematical equation.

In random scan technique, the electron beam is directed straight away to the particular point(s) of the screen where the image is to be produced. It generates the image by drawing a set of random straight lines much in the same way one might move a pencil over a piece of paper to draw an image – drawing strokes from one point to another, one line at a time. Random scan monitors draw a picture one line at a time (Vector display, Stroke –writing or calligraphic displays). The component lines of a picture can be drawn and refreshed.

Random scan displays have higher resolution than raster systems.

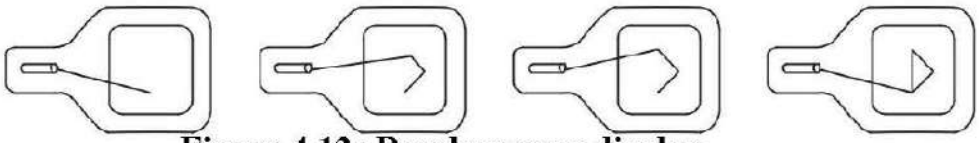


Figure.4.12: Random scan display

3. **Direct view storage tube: -** An alternative method for maintaining a screen image is to store the picture information inside the CRT instead of refreshing the screen. A direct-view storage tube (DVST) stores the picture information as a charge distribution just behind the phosphor-coated screen.

In DVST there is no refresh buffer; the images are created by drawing vectors or line segments with a relatively slow-moving electron beam. The beam is designed not to draw directly on phosphor but on a fine wire mesh (called *storage mesh*) coated with dielectric and mounted just behind the screen. A pattern of positive charge is deposited on the grid, and this pattern is transferred to the phosphor-coated screen by a continuous flood of electrons emanating from a separate flood gun.

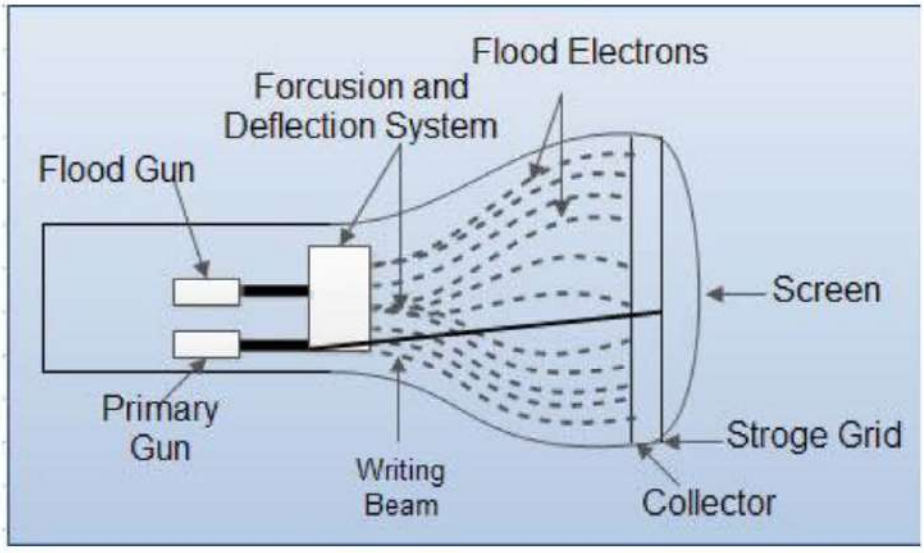


Figure.4.13: **Direct view storage tube**

Just behind the storage mesh is a second grid, the collector, whose main purpose is to smooth out the flow of flood electrons. These electrons pass through the collector at low velocity and are attracted to the positively charged portions of the storage mesh but repelled by the rest. Electrons not repelled by the storage mesh pass right through it and strike the phosphor.

To increase the energy of these slow-moving electrons and thus create a bright picture, the screen is maintained at a high positive potential. The storage tube retains the image generated until it is erased. Thus, no refreshing is necessary, and the image is absolutely flickers free.

A major disadvantage of DVST in interactive computer graphics is its inability to selectively erase parts of an image from the screen. To erase a line segment of the displayed image, one has to first erase the complete image and then redraw it by omitting that line segment. However, the DVST supports a very high resolution which is good for displaying complex images.

4. **Flat panel display:-**The term flat–panel displays refers to a class of video devices that have reduced volume, weight, and power requirements compared to a CRT. A significant feature of flat-panel displayed is that they are thinner than CRTs.

Two categories of Flat-panel displays are:

1. Emissive Displays

2. Nonemissive Displays

1. **Emissive display**:-are devices that convert electrical energy into light.

Examples are Plasma panels, thin-film electroluminescent displays and Light-emitting diodes (LED).

Plasma panel:- A layer of gas (usually neon) is sandwiched between two glass plates. By applying a high voltage to a pair of horizontal and vertical conductors, a small section of the gas (tiny neon bulb) at the intersection of the conductors break down into the glowing plasma of electrons and ions.

Thin-film electroluminescent:- The region between the glass plates is filled with a phosphor, such as zinc sulfide doped with manganese.

Light-emitting diode:- A matrix of diodes is arranged to form the pixel positions in the display, and picture definition is stored in a refresh buffer. Information is read from the refreshed buffer and converted to voltage levels that are applied to the diodes to produce the light patterns in the display.

2.**Nonemissive display**:-In this display, it uses the optical effect to convert sunlight or light from some other sources into a graphic pattern.

Example is LCD(liquid crystal display).

Liquid crystal display:- used in small systems, such as calculators, laptop computers. Produce a picture by passing polarized light (from the surrounding or from an internal light source) through a liquid-crystal material that can be aligned to either block or transmit the light.

Two glass plates, each containing a light polarizer at right angles to the other plate, sandwich the liquid crystal materials. Polarized light passing through the material is twisted so that it will pass through the opposite polarizer. The light is then reflected back t the viewer.

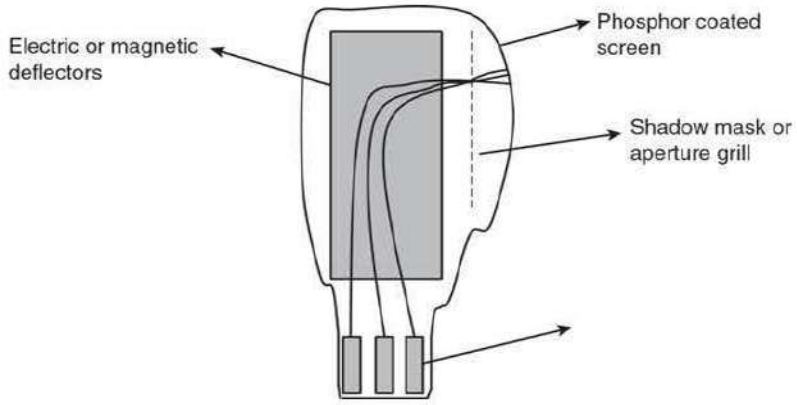


Figure. 4.14: Liquid crystal display

4.5 Graphical Input Technique

Several techniques are incorporated into graphics packages to aid the interactive construction of pictures.

An interaction technique is a way of using a physical input/output device to perform a generic interaction task in a human-computer dialogue. It represents an abstraction of some common class of interactive task, for example, choosing one of several objects shown on a display screen, so it is not bound to a single application.

The basic interaction tasks for interactive graphics are positioning, selecting, entering text and entering numeric quantities.

Various input techniques are following

1. **Basic Positioning Methods**:- This can be considered the most basic of graphical input operations. In its simplest form, it involves choosing a symbol/character on the screen and moving it to another location. One way of using it is to choose the symbol or picture involved, moving the cursor to the position required and pressing a (predetermined) key to place in that position.
   With a text string, for example, the screen point could be taken as the center string position, or the start or end position of the string, or any of the other string-

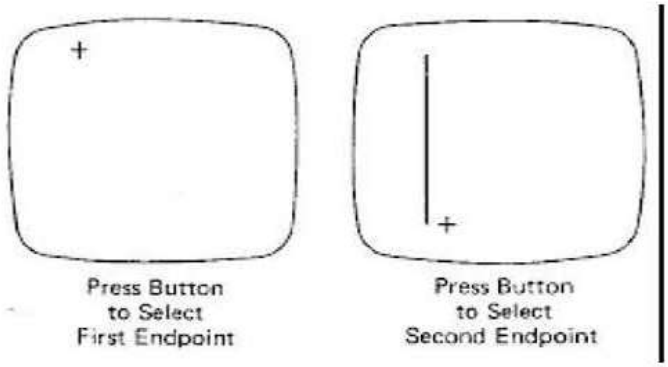
positioning options. For lines, straight line segments can be displayed between two selected screen positions.

As an aid in positioning objects, numeric values for selected positions can be echoed on the screen. Using the echoed coordinate values as a guide, we can make adjustments in the selected location to obtain accurate positioning.

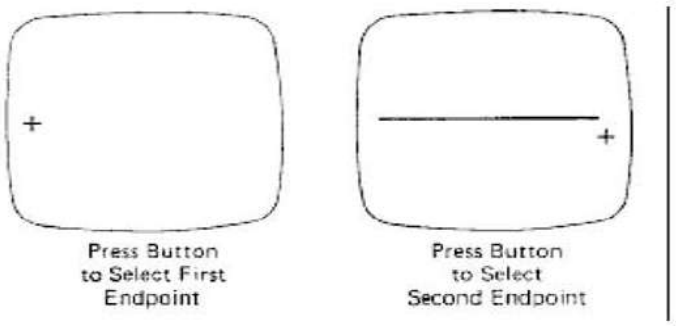
2. **Positional constraint:-**

One of the problems faced by inexperienced users while drawing figures is the concept of positioning. For example, we may want to put an object exactly at the end of a straight-line or across at the center of the circle etc. Because of lack of coordination between the eyes and the hand movements.

In some applications, certain types of prescribed orientations or object alignments are useful. A constraint is a rule for altering input-coordinate values to produce a specified orientation or alignment of the displayed coordinates. There are many kinds of constraint functions that can be specified, but the most common constraint is a horizontal and vertical alignment of straight lines. This type of constraint, shown in following figures, is useful in forming network layouts. With this constraint, we can create horizontal and vertical lines without worrying about precise specification of endpoint coordinates.



Horizontal line constraints



Vertical line constraints

Figure. 4.15: Positional constraint

A horizontal or vertical constraint is implemented by determining whether any two input coordinate endpoints are more nearly horizontal or more near vertical. If the difference in the y values of the two endpoints is smaller than the difference in x values, a horizontal line is displayed. Otherwise, a vertical line is drawn. Other kinds of constraints can be applied to input coordinates to produce a variety of alignments. Lines could be constrained to have a particular slant, such as 45°, and input coordinates could be constrained to lie along predefined paths, such as circular arcs.

3. **Grid**:- The Another technique for constraining vertex positions is to superimpose an "invisible" grid (with user specified spacing) over the drawing area. Now object vertices can be constrained to be at the intersection of a horizontal and vertical grid line. This makes it easier, for example, to guarantee two line segments share a common endpoint vertex.

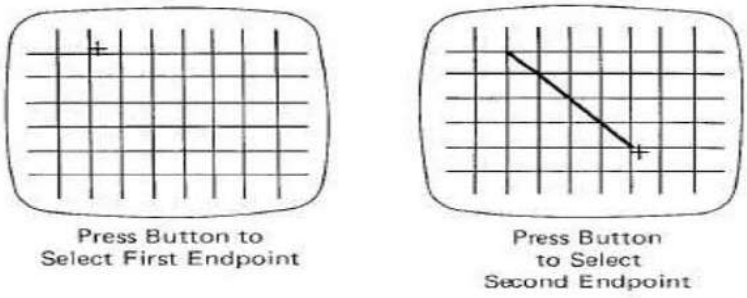


Figure. 4.16: Grid

Grids facilitate object constructions, because a new line can be joined easily to a previously drawn line by selecting any position near the endpoint grid intersection of one end of the displayed line.

3. Gravity Field:-To identify (or select) positions on objects that might not match well-defined positions (like grid points), we might enable an invisible "gravity field" around the objects.

A gravity field is just the region containing all coordinate positions within a specified maximum distance of an object.

Then we want to connect a line (for example) with an object, we only have to select an endpoint that is within the gravitational field of the object; the software automatically connects the line to the object.

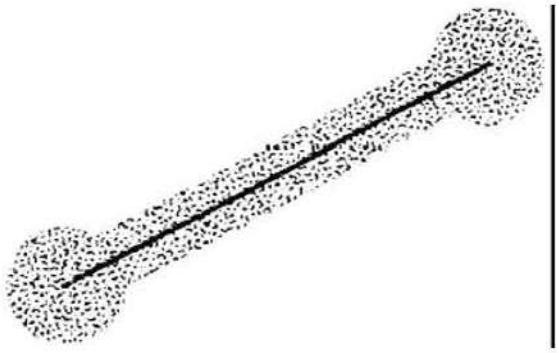


Figure. 4.17: Gravity Field

4. **Rubber band technique**:- It is used for interactive adjusting object sizes. For example, one end-point of a line can be selected.

Then as the cursor moves, the "draft line" is displayed between the start position and the cursor position. Finally, the end position is selected (e.g. a mouse button is released)

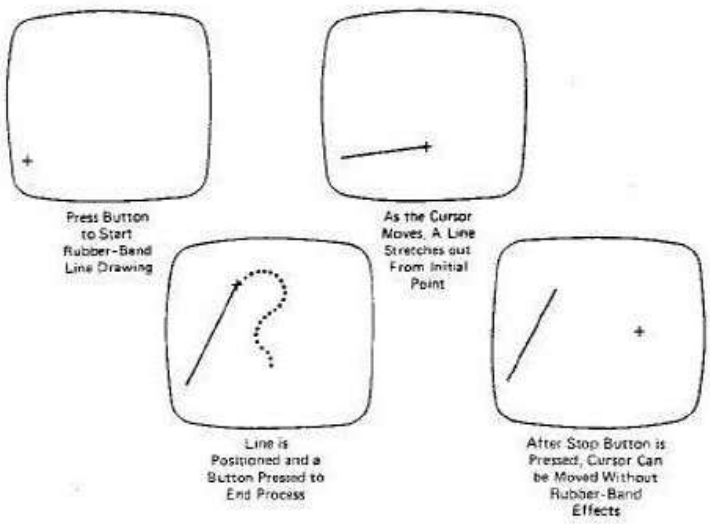


Figure .4.18: Rubber band technique

Rubber-band methods are used to construct and position other objects besides straight lines. The following figure demonstrates the rubber-band construction of a rectangle and rubber-band circle construction.

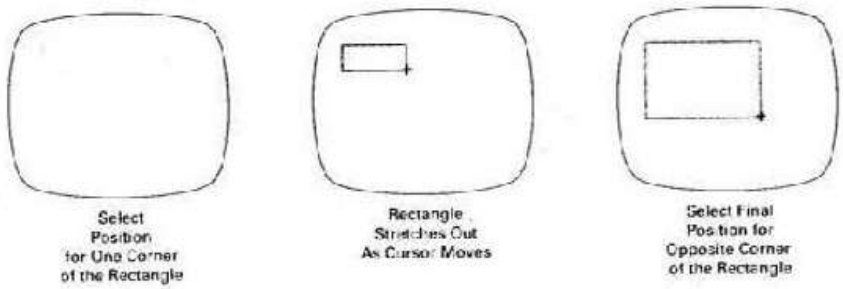
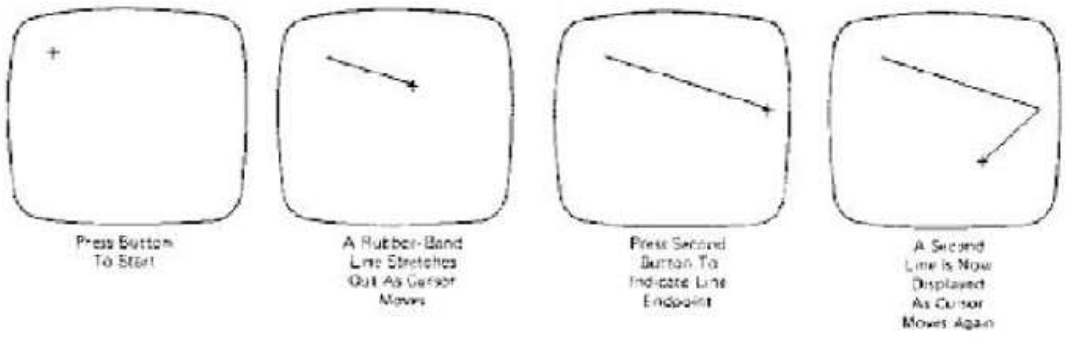


Figure.4.19: Rubber band method for constructing rectangle



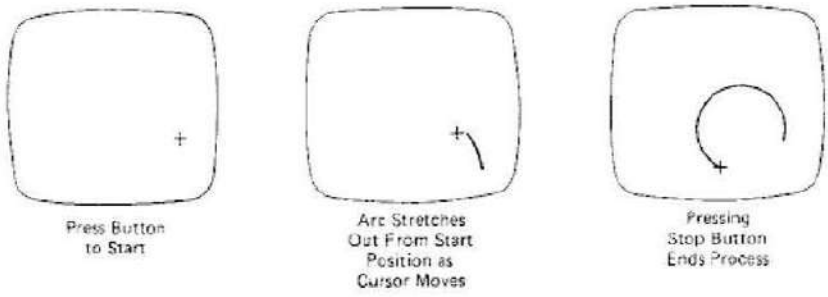


Figure.4.20: Rubber band method for constructing circle

5. **Dragging:-** A technique that is often used in interactive picture construction is to move objects into position by dragging them with the screen cursor. We first select an object, then move the cursor in the direction we want the object to move, and the selected object follows the cursor path. Dragging objects to various positions in the scene is useful in applications, where we might want to explore different possibilities before selecting a final location.
6. **Sketching**:-Options for sketching, drawing, and painting come in a variety of forms. Straight lines, polygons, and circles can be generated with the methods discussed in the previous sections. Curve-drawing options can be provided using standard curve shapes, such as circular arcs and splines, or with freehand sketching procedures. Splines are interactively constructed by specifying a set of discrete screen points that give the general shape of the curve. Then the system fits the set of points with a polynomial curve. In freehand drawing, curves are generated by following the path of a stylus on a graphics tablet or the path of the screen cursor on a video monitor. Once a curve is displayed, the designer can alter the curve shape by adjusting the positions of selected points along the curve path.

4.6 Self Learning Exercise

Q.1 Which devices provides positional information to the graphics system?

a) Input devices

b) Output devices

c) Pointing devices

d) Both a and c

Q. 2 Which stores the picture information as a charge distribution behind the phosphor-coated screen?

a) Cathode ray tube

b) Direct-view storage tube

c) Flat panel displays

d) 3D viewing devices

Q.3 The maximum number of points that can be displayed without overlap on a CRT is referred as?

a) Picture

b) Resolution

c) Persistence

d) Frame buffer

4.7 Summary

The term "computer graphics" refers to anything involved in the creation or manipulation of images on the computer, including animated images. In computer graphics, various input and output devices are used to perform many operations.

4.8 Glossary

Pixel-- The smallest indivisible unit of a digital image. A pixel is always the same color throughout. An image is a two-dimensional array of pixels.

Refresh rate--- The rate at which parts of the image on a CRT are re-painted, or refreshed. The horizontal refresh rate is the rate at which individual scan lines are drawn. The vertical refresh rate is the rate at which fields are drawn in interlaced mode, or whole frames are drawn in non-interlaced mode.

4.9 Answers to Self-Learning Exercise

Q.1 (d)

Q.2 (b)

Q.3 (b)

4.10 Exercise

Q. 1 In the raster scan method for transformation, a 90^0 rotation can be performed by ?

a) reversing the order of bits within each row in the frame buffer

b) by performing XOR on the frame buffer location

c) by coping each row of the block into a column in the new frame buffer location

d) None of above

Q. 2 Beam penetration method is usually used in .....?

a) LCD

b) Raster Scan display

c) Random scan display

d) DVST

Q. 3 Video devices with reduced volume, weight and power consumption are collectively known as .....?

a) Light weight monitors

b) Flat-panel displays

c) CRT

d) Portable display

Q. 4 Which technique is employed for drawing entities using mouse only.?

a) Gravity

b) Rubber band

c) Constraint

d) Painting

Q. 5 ..... is used to connect a new line to a previously drawn line?

a) Gravity field

b) Rubber band method

c) Painting

d) None of the above

Q. 6 Which method are used to get and set the position of a pixel, object or text in active area of a desktop?

a) Drugging method

b) Basic positioning method

c) Sketching method

d) Gravity field method

Q. 7 The division displayed on screen into row and columns is known as ?

a) Rubber band method

b) Gravity field

c) Dragging

d) Grid

4.11 Answers of Exercise

Q. 1 (c)

Q. 2 (b)

Q. 3 (b)

Q. 7 (d)

Q. 4 (b)

Q. 5 (a)

Q. 6 (b)

References and Suggested Readings

1. Computer Graphics by Donald Hearn and M.Pauline Baker, 2nd edition, Prentice Hall,1994.
2. Procedural Element for Computer Graphics by David F.Rogers, Tata McGraw Hill, 2001.
3. Computer Graphics by Dr.Prabhakar Gupta,Vineet Agarwal,Manish Varshnay, Laxmi Publications, 2011.

UNIT-5
Output Primitives

Structure of the Unit

5.0 Objective

In this chapter, we shall focus on the following topics

- Scan Converting Lines
- Scan Converting Circle
- Scan Converting Ellipse

5.1 Introduction

A picture is completely specified by the set of intensities for the pixel positions in the display. Shapes and colors of the objects can be described

internally with pixel arrays into the frame buffer or with the set of the basic geometric structure such as straight line segments and polygon color areas. To describe structure of basic object is referred to as output primitives.

Each output primitive is specified with input co-ordinate data and other information about the way that objects is to be displayed. Additional output primitives that can be used to constant a picture include circles and other conic sections, quadric surfaces, Spline curves and surfaces, polygon floor areas and character string.

Graphic SW and HW provide subroutines to describe a scene in terms of basic geometric structures called output primitives. Output primitives are combined to form complex structures. Simplest output primitives are as follows:

Point (pixel)

Line segment

Points and Lines

Point plotting is accomplished by converting a single coordinate position into appropriate operations for the output device. With a CRT monitor, for example, the electron beam is turned on to illuminate the screen phosphor at the selected location.

Line drawing is accomplished by calculating intermediate positions along the line path between two specified end points positions. An output device is then directed to fill in these positions between the end points of the object.

Digital devices display a straight line segment by plotting discrete points between the two end points. Discrete coordinate positions along the line path are calculated from the equation of the line. For a raster video display, the line color (intensity) is then loaded into the frame buffer at the corresponding pixel coordinates. Reading from the frame buffer, the video controller then plots "the screen pixels".

Pixel positions are referenced according to scan-line number and column number (pixel position across a scan line). Scan lines are numbered consecutively

from 0, starting at the bottom of the screen; and pixel columns are numbered from 0, left to right across each scan line.

5.2 Scan Converting Lines

The basic meaning of scan converting is to convert output primitives into frame buffer updates. In conversion process, we have to choose which pixels contain which intensity value. In the scan conversion process, there are many constraints. First straight line should appear as a straight line. Second constraint states that primitives should start and end accurately. Next constraint shows that primitives should have a consistent brightness along with their length. And they should draw rapidly.

Line Equation

We determine pixel positions along a straight-line path from the geometric properties of the line. The Cartesian slope-intercept equation for a straight line is

$$y = m \cdot x + b \qquad (5\text{-}1)$$

With m as the slope of the line and b as the y intercept.

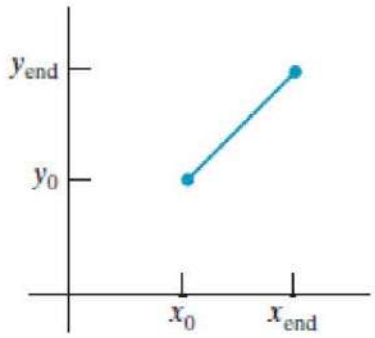


Figure .5.1 : Line path between end points position

Given that the two end points of a line segment are specified at positions (x_0, y_0) and (x_{end}, y_{end}), as shown in Figure 5-1.

We can determine values for the slope m and y intercept b with the following calculations:

$$m = (y_{end} - y_0) / (x_{end} - x_0) \quad (5\text{-}2)$$

$$b = y_0 - m \cdot x_0 \quad (5\text{-}3)$$

Algorithms for displaying straight lines are based on the line equation 3-1 and the calculations given in equations 5-2 and 5-3.

For any given x interval δx along a line, we can compute the corresponding y interval δy from equation 5-2 as

$$\delta y = m \cdot \delta x \quad (5\text{-}4)$$

Similarly, we can obtain the x interval δx corresponding to a specified δy as

$$\delta x = \delta y / m \quad (5\text{-}5)$$

These equations define the basis for determining deflection voltages in analog displays, such as a vector-scan system, where arbitrarily small changes in deflection voltage are possible. For lines with slope magnitudes $|m| < 1$, δx can be set proportional to a small horizontal deflection voltage, and the corresponding vertical deflection is set to proportional to δy as calculated from equation 5-4.

For lines whose slopes have magnitudes $|m| > 1$, δy can be set proportional to a small vertical deflection voltage with the corresponding horizontal deflection voltage set proportional to δx, calculated from Equation 5-5. And finally for lines with $m = 1$, $\delta x = \delta y$ and the horizontal and vertical deflections voltages are equal. In each case, a smooth line with slope m is generated between the specified endpoints.

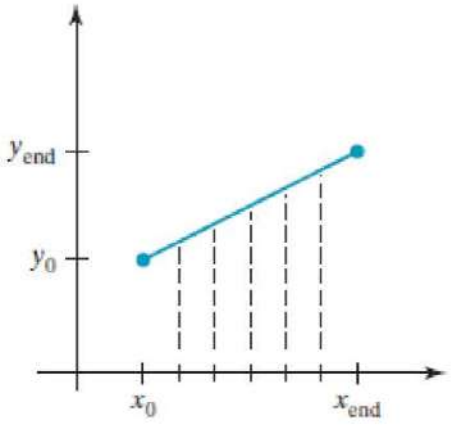


Figure. 5.2: Straight line segments with five sampling position along the x axis

On Raster systems, lines are plotted with pixels, and step sizes in the horizontal and vertical directions are constrained by pixel separations. We must "sample" a line at discrete positions and determine the nearest pixel to the line at each sampled position. This scan-conversion process for straight lines is mentioned in Fig. 5-2 with discrete sample positions along the x axis.

Line Drawing Algorithms

- Digital Differential Analyzer (DDA) Algorithm
- Bresenham's Line Algorithm
- Parallel Line Algorithm

Digital Differential Analyzer (DDA) Algorithm

The digital differential analyzer (DDA) is a scan-conversion line algorithm based on calculation either Δy or Δx.
The line at unit intervals in one coordinate and determine other coordinate by corresponding integer values nearest the line path.

A line with positive slope, if the slope is less than or equal to 1, at unit x intervals ($\Delta x=1$) and compute each successive y values as:

$$y_{k+1} = y_k + m \quad (1)$$

Subscript k takes integer values starting from 1 for the first point and increases by 1 until the final endpoint is reached. m can be any real number between 0 and 1, the calculated y values must be rounded to the nearest integer
For lines with a positive slope greater than 1 we reverse the roles of x and y, while ($\Delta y=1$) and calculate each succeeding x value as

$$x_{k+1} = x_k + 1/m \quad (2)$$

Equation (1) and (2) are based on the assumption that lines are to be processed from the left endpoint to the right endpoint. If we make this processing reversed, Δx=-1 that the starting endpoint is at the right position

$$y_{k+1} = y_k - m \quad (3)$$

When the slope is greater than 1 and Δy = -1 with

$$x_{k+1} = x_k - 1(1/m) \quad (4)$$

If the absolute value of the slope is less than 1 and the start endpoint is at the left, we set Δx = 1 and calculate y values with equation (1).

When the start endpoint is at the right (for the same slope), we set Δx = -1 and obtain y positions from equation (3). Similarly, when the absolute value of a negative slope is greater than 1, we use Δy= -1 and equation (4) or we use Δy = 1 and equation (2).

DDA Algorithm

```
#define ROUND (a) ((int) (a+0.5))
Void lineDDA (int xa, int ya, int xb, int yb)
{
        int dx = xb - xa, dy = yb - ya, steps, k;
        float xIncrement, yIncrement, x = xa, y = ya;
        if (abs (dx) > abs (dy)
                steps = abs (dx) ;
        else
                steps = abs dy);
        xIncrement = dx / (float) steps;
        yIncrement = dy / (float) steps
```

```
        setpixel (ROUND(x), ROUND(y) ) :
        for (k=0; k<steps; k++)
        {
                x = x + xIncrement;
                y = y + yIncrement;
                setpixel (ROUND(x), ROUND(y));
        }
}
```

Algorithm Description:

Step 1: Accept Input as two endpoint pixel positions

Step 2: Horizontal and vertical differences between the endpoint positions are assigned to parameters dx and dy (Calculate dx=xb-xa and dy=yb-ya).

Step 3: The difference with the greater magnitude determines the value of parameter steps.

Step 4: Starting with pixel position (xa, ya), determine the offset needed at each step to generate the next pixel position along the line path.

Step 5: loop the following process for steps number of times

a. Use a unit of increment or decrement in the x and y direction.

b. if xa is less than xb the values of increment in the x and y directions are 1 and m.

c. if xa is greater than xb then the decrements -1 and – m are used.

Example: Consider the line from (0, 0) to (4, 6)

1. xa=0, ya =0 and xb=4, yb=6
2. dx = xb – xa = 4 – 0 = 4 and dy = yb – ya = 6 – 0 = 6
3. x= 0 and y= 0
4. 4 > 6 (false) so, steps= 6
5. Calculate xIncrement = dx/steps = 4 / 6 = 0.66 and yIncrement = dy/steps =6/6=1
6. Setpixel(x, y) = Setpixel (0, 0) (Starting Pixel Position).

7. Iterate the calculation for xIncrement and yIncrement for steps (6) number of times.

8. Tabulation of the each iteration.

K	X	Y	Plotting Points (Rounded to integer)
0	0+0.66=0.66	0+1=1	(1,1)
1	0.66+0.66=1.32	1+1=2	(1,2)
2	1.32+0.66=1.98	2+1=3	(2,3)
3	1.98+0.66=2.64	3+1=4	(3,4)
4	2.64+0.66=3.3	4+1=5	(3,5)
5	3.3+0.66=3.96	5+1=6	(4,6)

Result :

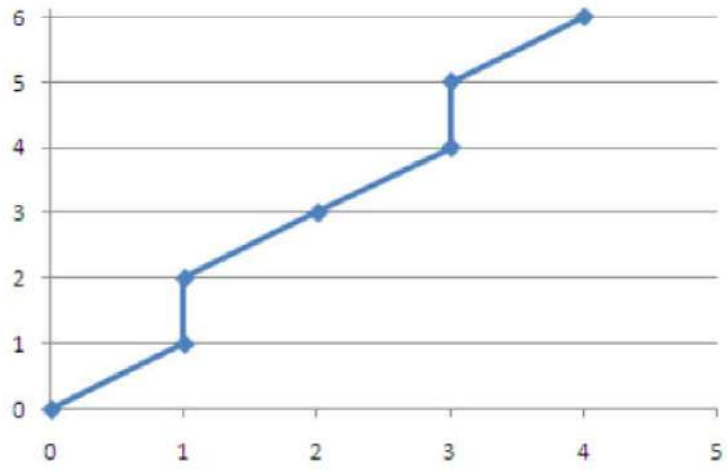


Figure .5.3:

Advantages of DDA Algorithm

1. It is the simplest algorithm.

2. It is a faster method for calculating pixel positions than the direct use of line equations.

Disadvantages of DDA Algorithm

1. Floating point arithmetic in DDA algorithm is still time-consuming
2. End point accuracy is poor
3. Because of rounding operation the some calculated pixels positions away from through line.

Bresenham's Line Algorithm

An accurate and efficient raster line generating algorithm developed by Bresenham that uses only incremental integer calculations.

Also, Bresenham's line algorithm can be designed to display circles and other curves. To illustrate Bresenham's approach, we- first consider the scan-conversion process for lines with a positive slope less than 1.

Pixel positions along a line path are then determined by sampling at unit x intervals. Starting from the left endpoint (x_0, y_0) of a given line, we step to each successive column (x position) and plot the pixel whose scan-line y value is closest to the line path.

To determine the pixel (x_k, y_k) is to be displayed, next to decide which pixel to plot the column $x_{k+1} = x_{k+1}$, (x_{k+1}, y_k) and (x_{k+1}, y_{k+1}). At sampling position x_{k+1}, we mark vertical pixel separations from the mathematical line path as d1 and d2. The y coordinate on the mathematical line at pixel column position x_{k+1} is calculated as follows:

$$y = m(x_{k+1}) + b \quad (1)$$

Then

$$d1 = y - y_k$$
$$= m(x_{k+1}) + b - y_k$$
$$d2 = y_{k+1} - y$$
$$= y_{k+1} - m(x_{k+1}) - b$$

To determine which of two pixels are closest to the line path. Efficient test that is based on difference between two pixels separations.

$$d1 - d2 = 2m(x_{k+1}) - 2y_k + 2b - 1 \qquad (2)$$

A decision parameter P_k for the k^{th} step in the line algorithm can be obtained by rearranging equation (2). By substituting m=Δy/Δx where Δx and Δy are the vertical and horizontal separations of the endpoint positions and defining the decision parameter as follows:

$$P_k = \Delta x\,(d1 - d2)$$
$$= 2\Delta y\, x_k - 2\Delta x.\, y_k + c$$

(3)

The sign of P_k is same as the sign of d1–d2, since Δx>0.

Parameter C is constant and having the value 2Δy + Δx(2b-1) which is independent of the pixel position and will be eliminated in the recursive calculations for P_k.

If the pixel at y_k is "closer" to the line path than the pixel at y_{k+1} (d1< d2) than decision parameter P_k is negative. In this case, plot the lower pixel, otherwise plot the upper pixel.

Coordinate changes along the line occur in unit steps in either the x or y directions.

To obtain the values of successive decision parameters using incremental integer calculations. At steps k+1, the decision parameter is evaluated from equation (3) as follows:

$$P_{k+1} = 2\Delta y\, x_{k+1} - 2\Delta x.\, y_{k+1} + c$$

Subtracting the equation (3) from the preceding equation

$$P_{k+1} - P_k = 2\Delta y(x_{k+1} - x_k) - 2\Delta x(y_{k+1} - y_k)$$

But $x_{k+1} = x_k + 1$ so that

$$P_{k+1} = P_k + 2\Delta y - 2\Delta x(y_{k+1} - y_k)$$

(4)

Where the term $y_{k+1} - y_k$ is either 0 or 1 depending on the sign of parameter P_k. This recursive calculation of decision parameter is performed at each integer x position, starting at the left coordinate endpoint of the line.

The first parameter P_0 is evaluated from equation at the starting pixel position (x_0,y_0) and with m evaluated as $\Delta y/\Delta x$.

$$P_0 = 2\Delta y - \Delta x$$

(5)

Bresenham's line drawing for a line with a positive slope (m) less than 1 in the following outline of the algorithm.

The constants $2\Delta y$ and $2\Delta y$-$2\Delta x$ are calculated once for each line to be scan converted.

Bresenham's Line Drawing Algorithm for |m| < 1

1. Input the two line endpoints and store the left end point in (x_0,y_0)
2. Load (x_0,y_0) into frame buffer, i.e. Plot the first point.
3. Calculate the constants Δx, Δy, $2\Delta y$ and obtain the starting value for the decision parameter as

   $$P_0 = 2\Delta y - \Delta x$$

4. At each x_k along the line, starting at k=0 perform the following test

   If $P_k < 0$,

   The next point to plot is (x_{k1},y_k) and

   $$P_{k+1} = P_k + 2\Delta y$$

   Otherwise,

   The next point to plot is (x_{k1},y_{k+1}) and

   $$P_{k+1} = P_k + 2\Delta y - 2\Delta x$$

5. Perform step4 Δx times

Implementation of Bresenham Line drawing Algorithm

```
Void lineBres (int xa, int ya, int xb, int yb)
{
        int dx = abs( xa – xb) , dy = abs (ya - yb);
        int p = 2 * dy – dx;
        int twoDy = 2 * dy, twoDyDx = 2 *(dy - dx);
        int x , y, xEnd;
/* Determine which point to use as start, which as end * /
        if (xa > x b )
        {
                x = xb;
                y = yb;
                xEnd = xa;
        }
        else
        {
                x = xa;
                y = ya;
                xEnd = xb;
        }
        setPixel(x, y);
        while(x<xEnd)
        {
                x++;
                if (p<0)
                        p+=twoDy;
                else
                {
                        y++;
                        p+=twoDyDx;
```

```
            }
            setPixel(x,y);
        }
}
```

Example: Consider the line with endpoints (20, 10) to (30, 18)

The line has the slope m= (18-10) / (30-20) = 8/10 = 0.8

$\Delta x = 10, \Delta y = 8$

The initial decision parameter has the value

$$P_0 = 2\Delta y - \Delta x = 6$$

And the increments for calculating successive decision parameters are

$2\Delta y = 16$ $2\Delta y - 2\ \Delta x = -4$

We plot the initial point (x_0, y_0) = (20, 10) and determine successive pixel positions along the line path from the decision parameter as follows:

Tabulation

K	P_k	$(x_k + 1, y_k + 1)$
0	6	(21, 11)
1	2	(22, 12)
2	-2	(23, 12)
3	14	(24, 13)
4	10	(25, 14)
5	6	(26, 15)
6	2	(27, 16)
7	-2	(28, 16)
8	14	(29, 17)
9	10	(30, 18)

Result

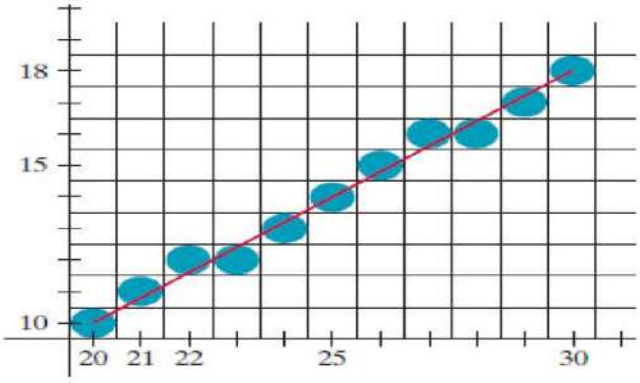


Figure. 5.4:

Advantages Bresenham's Line Algorithm

1. This Algorithm is fast as compared to other line algorithms.
2. Uses only integer calculations

Disadvantages Bresenham's Line Algorithm

1. It is meant only for basic line drawing.

Parallel Line Algorithm

This algorithm allows us to calculate multiple pixel positions along a line path simultaneously by partitioning the computations among the various processors available. There are two methods for implement multiple pixel positions.

First method: To the partitioning problem is to adapt an existing sequential algorithm to take advantage of multiple processors.

Alternatively method: we can look for other ways to set up the processing so that pixel positions can be calculated efficiently in parallel. An important consideration in devising a parallel algorithm is to balance the processing load among the available processors.

Given n_p processors, we can set up a parallel Bresenham line algorithm by subdividing the line path into n_p partitions and simultaneously generating line segments in each of the subintervals. For a line with slope $0 < m < 1.0$ and left endpoint coordinate position is (x_0, y_0), we partition the line along the positive x direction. The distance between beginning x positions of adjacent partitions can be calculated as follows:

$$\Delta x_p = (\Delta \quad x \quad + \quad n_p - 1)/\, n_p$$

(1)

Where Δx is the width of the line, and the value for partition width Δx_p is computed using integer division. Numbering the partitions, and the processors, as 0, 1, 2, up to $n_p - 1$, we calculate the starting x coordinate for the k^{th} partition as follows:

$$x_k = x_0 + k\Delta x_p \qquad (2)$$

For an example, if we have n_p= 4 processors, with Δx=15, the width of the partitions is 4 and the starting x values for the partitions are x_0, x_0+4, x_0+8 and x_0+12.

With above partitioning scheme, the width of the last (rightmost) subinterval will be smaller than the others in some cases. In addition, if the line endpoints are not integers, truncation errors can result in variable width partitions along the length of the line.

- A decision parameter also p_k found from these values.
- Each processor then calculates pixel positions over its assigned sub interval using the decision parameter value and the starting coordinates.
- Another way to set up parallel algorithm is to assign each processor to a particular group of screen pixels with a sufficient number of processor.
- We can assign each processor to one pixel with in some screen pixel.
- These approaches can be adopted to line display by assigning one processor to each of the pixels within the limits of the line coordinate bounding rectangle and calculating pixel distance from the line path.

5.3 Scan Converting Circle

Properties of a circle

A circle is defined as a set of points that are all the given distance (x_c, y_c).

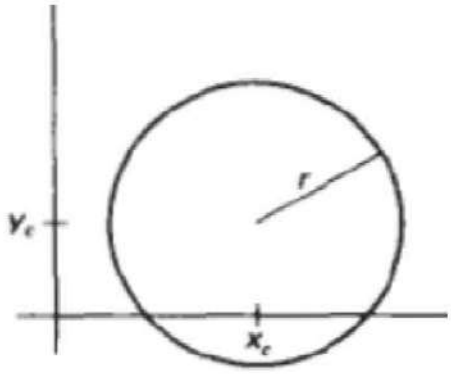


Figure. 5.5: Circle with center coordinates and radius

This distance relationship is expressed by the Pythagorean Theorem in Cartesian coordinates as

$$(x - x_c)^2 + (y - y_c)^2 = r^2 \quad (1)$$

Use above equation to calculate the position of points on a circle circumference by stepping along the x axis in unit steps from $x_c - r$ to $x_c + r$ and calculating the corresponding y values at each position as

$$y = y_c + (-)\,(r^2 - (x_c - x)^2)^{1/2} \quad (2)$$

This is not the best method for generating a circle for the following reason:

- Considerable amount of computation
- Spacing between plotted pixels is not uniform

To eliminate the unequal spacing calculate points along the circle boundary using polar coordinates r and θ. Show the circle equation in parametric polar form as:

$$x = x_c + r\cos\theta \qquad y = y_c + r\sin\theta$$

When a display is generated with these equations using a fixed angular step size, a circle is plotted with equally spaced points along the circumference. To reduce calculations use a large angular separation between points along the

circumference and connect the points with straight line segments to approximate the circular path.

Set the angular step size at 1/r. This plots pixel positions that are approximately one unit apart. The shape of the circle is similar in each quadrant. To determine the curve positions in the first quadrant, to generate the circle section in the second quadrant of the xy plane by noting that the two circle sections are symmetric with respect to the y axis and circle section in the third and fourth quadrants can be obtained from sections in the first and second quadrants by considering symmetry between octants.

Circle sections in adjacent octants within one quadrant are symmetric with respect to the 45° line dividing the two octants. Where a point at position (x, y) on a one-eight circle sector is mapped into the seven circle points in the other octants of the xy plane. To generate all pixel positions around a circle by calculating only the points within the sector from x=0 to y=0. The slope of the curve in this octant has an magnitude less than or equal to 1.0. At x=0, the circle slope is 0 and at x=y, the slope is -1.0.

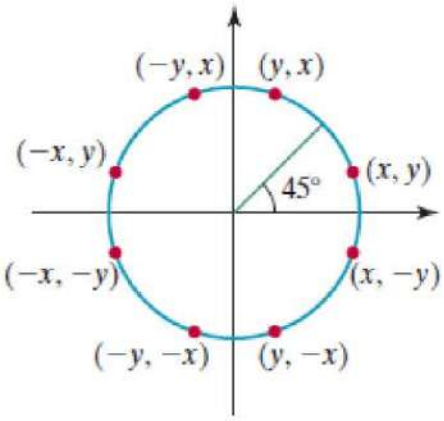


Figure. 5.6: Symmetry of a Circle

Bresenham's line algorithm for raster displays is adapted to circle generation by finding the closest pixel to the circumference at each sampling step. Square root evaluations would be required to computer pixel distances from a circular path.

Bresenham's circle algorithm avoids these square root calculations by comparing the squares of the pixel separation distances. It is possible to perform a direct distance comparison without a squaring operation. In this approach is to test the halfway position between two pixels to determine if this midpoint is inside or outside the circle boundary. This method is more easily applied to other conics and for an integer circle radius the midpoint approach generates the same pixel positions as the Bresenham circle algorithm.

Midpoint Circle Algorithm

In the raster line algorithm at unit intervals and determine the closest pixel position to the specified circle path at each step for a given radius r and screen center position (x_c, y_c) set up our algorithm to calculate pixel positions around a circle path centered at the coordinate position by adding x_c to x and y_c to y.
To apply the midpoint method, we define a circle function as:

$$f_{circle}(x, y) = x^2 + y^2 - r^2$$

Any point (x, y) on the boundary of the circle with radius r satisfies the equation $f_{circle}(x, y) = 0$. If the point is in the interior of the circle, the circle function is negative. And if the point is outside the circle the, circle function is positive.

$f_{circle}(x, y) < 0$, if (x, y) is inside the circle boundary
$=0$, if (x, y) is on the circle boundary
>0, if (x, y) is outside the circle boundary

The tests in the above equations are performed for the mid-positions between pixels near the circular path at each sampling step. The circle function is the decision parameter in the midpoint algorithm.

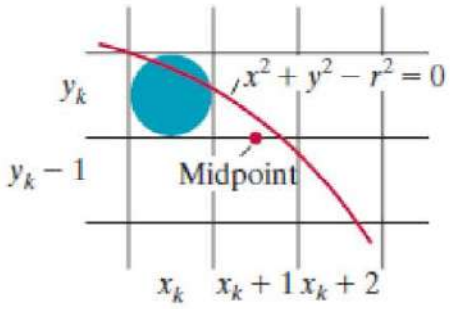


Figure.5.7: Midpoint between candidate pixels at sampling position along with a circular path

Midpoint between candidate pixels at sampling position x_{k+1} along a circular path. Fig 5-5 shows the midpoint between the two candidate pixels at sampling position x_{k+1}. To plot the pixel at (x_k, y_k) next need to determine whether the pixel at position (x_{k+1},y_k) or the one at position (x_{k+1},y_{k-1}) is circular to the circle.

Our decision parameter is the circle function evaluated at the midpoint between these two pixels

$$P_k = f_{circle}(x_{k+1},y_k\text{-}1/2)$$
$$= (x_{k+1})^2 + (y_k\text{-}1/2)^2 - r^2$$

If $P_k < 0$, this midpoint is inside the circle and the pixel on scan line y_k is closer to the circle boundary. Otherwise the mid position is outside or on the circle boundary and select the pixel on scan line y_k-1.

Successive decision parameters are obtained using incremental calculations. To obtain a recursive expression for the next decision parameter by evaluating the circle function at sampling position $x_{k+1+1} = x_{k+2}$

$$P_k = f_{circle}(x_{k+1}+1, y_{k+1}\text{-}1/2)$$
$$= [(x_{k+1}) +1]^2 + (y_{k+1} -1/2)^2 - r^2$$

or

$$P_{k+1} = P_k + 2(x_{k+1}) + ((y_{k+1})^2 - (y_k)^2) - (y_{k+1} - y_k) + 1$$

Where y_{k+1} is either y_k or y_{k-1} depending on the sign of P_k.

Increments for obtaining P_{k+1} are either $2x_{k+1}+1$ (if P_k is negative) or $2x_{k+1}+1 - 2y_{k+1}$.

Evaluation of the terms $2x_{k+1}$ and $2y_{k+1}$ can also be done incrementally as

$$2x_{k+1} = 2x_{k+2}$$

$$2y_{k+1} = 2y_{k-2}$$

At the Start position (0, r) these two terms have the values 0 and 2r respectively. Each successive value for the $2x_{k+1}$ term is obtained by adding 2 to the previous value and each successive value for the $2y_{k+1}$ term is obtained by subtracting 2 from the previous value.

The initial decision parameter is obtained by evaluating the circle function at the start position $(x_0, y_0) = (0, r)$

$$P_0 = f_{circle}(1, r-1/2)$$
$$= 1+(r-1/2)^2 - r^2$$

Or

$$P_0 = (5/4) - r$$

If the radius r is specified as an integer. $P_0=1-r$ (for r an integer)

Algorithm

1. Input radius r and circle center (x_c, y_c) and obtain the first point on the circumference of the circle centered on the origin as

$$(x_c, y_c) = (0, r)$$

2. Calculate the initial value of the decision parameter as $P_0 = (5/4) - r$

3. At each x_k position, starting at k=0, perform the following test. If $P_k < 0$ the next point along the circle centered on (0, 0) is (x_{k+1}, y_k) and $P_{k+1}=P_k+2x_{k+1}+1$.

Otherwise the next point along the circle is (x_{k+1}, y_{k-1}) and $P_{k+1}=P_k+2x_{k+1}+1-2y_{k+1}$.

Where $2x_{k+1} = 2x_{k+2}$ and $2y_{k+1} = 2y_{k-2}$.

4. Determine symmetry points in the other seven octants.

5. Move each calculated pixel position (x, y) onto the circular path centered at (x_c, y_c) and plot the coordinate values.

$$X = x + x_c \qquad y = y + y_c$$

6. Repeat step 3 through 5 until x >= y.

Example: Midpoint Circle Drawing

Given a circle radius r = 10

The circle octant in the first quadrant from x = 0 to x = y. The initial value of the decision parameter is P_0=1- r = - 9

For the circle centered on the coordinate origin, the initial point is (x_0, y_0) = (0, 10) and initial increment terms for calculating the decision parameters are:

$$2x_0 = 0, \qquad 2y_0 = 20$$

Successive midpoint decision parameter values and the corresponding coordinate positions along the circle path are listed in the following table.

K	P_k	(x_{k+1}, y_{k-1})	$2x_{k+1}$	$2y_{k+1}$
0	-9	(1, 10)	2	20
1	-6	(2, 10)	4	20
2	-1	(3, 10)	6	20
3	6	(4, 9)	8	18
4	-3	(5, 9)	10	18
5	8	(6, 8)	12	16
6	5	(7, 7)	14	14

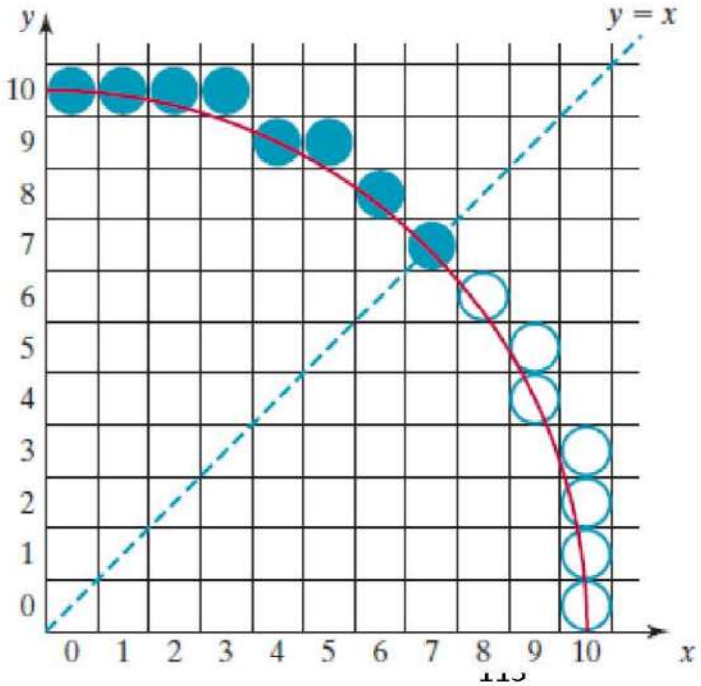


Figure.5.8: Implementation of Midpoint Circle Algorithm

```
Void circleMidpoint (int xCenter, int yCenter, int radius)
{
        int x = 0;
        int y = radius;
        int p = 1 - radius;
        void circlePlotPoints (int, int, int, int);
        /* Plot first set of points */
        circlePlotPoints (xCenter, yCenter, x, y);
        while (x < y)
        {
                x++ ;
                if (p < 0)
                        p +=2*x +1;
                else
                {
                        y--;
                        p +=2* (x - Y) + 1;
                }
                circlePlotPoints(xCenter, yCenter, x, y)
        }
}
Void circlePlotPolnts (int xCenter, int yCenter, int x, int y)
{
        setpixel (xCenter + x, yCenter + y ) ;
        setpixel (xCenter - x. yCenter + y);
        setpixel (xCenter + x, yCenter - y);
        setpixel (xCenter - x, yCenter - y ) ;
        setpixel (xCenter + y, yCenter + x);
```

```
        setpixel (xCenter - y , yCenter + x);
        setpixel (xCenter t y , yCenter - x);
        setpixel (xCenter - y , yCenter - x);
}
```

5.4 Scan Converting Ellipse

An ellipse is an elongated circle. So, elliptical curves can be generated by altering circle-drawing procedures to take into account the different dimensions of an ellipse along the major and minor axes.

Properties of Ellipse

An ellipse can be given in terms of the distances from any point on the ellipse to two fixed positions called the **foci** of the ellipse. The sum of these two distances is the same values for all points on the ellipse. If the distances to the two focus positions from any point p = (x, y) on the ellipse are labeled d1 and d2, then the general equation of an ellipse can be stated as

d1+d2=constant

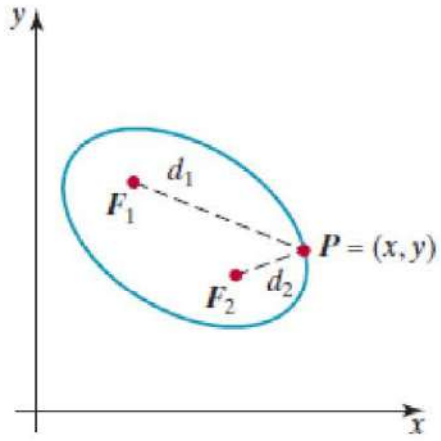


Figure. 5.9: Ellipse generated about foci F1 and F2

Expressing distances d1 and d2 in terms of the focal coordinates $F_1 = (x_1, y_1)$ and $F_2 = (x_2, y_2)$

$$\text{Sqrt}\left((x-x_1)^2 + (y-y_1)^2\right) + \text{sqrt}\left((x-x_2)^2 + (y-y_2)^2\right) = \text{constant}$$

By squaring this equation isolating the remaining radical and squaring again. The general ellipse equation in the form

$$Ax^2 + By^2 + Cxy + Dx + Ey + F = 0$$

The coefficients A, B, C, D, E and F are evaluated in terms of the focal coordinates and the dimensions of the major and minor axes of the ellipse.

The major axis is the straight line segment extending from one side of the ellipse to another through the foci. The minor axis spans the shorter dimension of the ellipse, perpendicularly bisecting the major axis at the halfway position (ellipse center) between the two foci.

An interactive method for specifying an ellipse in an arbitrary orientation is to input the two foci and a point on the ellipse boundary.

Ellipse equations are simplified if the major and minor axes are oriented to align with the coordinate axes. The major and minor axes oriented parallel to the x and y axes parameter r_x for this example labels the semi major axis and parameter r_y labels the semi minor axis.

$$((x-x_c) / r_x)^2 + ((y - y_c) / r_y)^2 = 1$$

Using polar coordinates r and θ, to describe the ellipse in Standard position with the parametric equations

$$x = x_c + r_x \cos\theta$$

$$y = y_c + r_x \sin\theta$$

Angle θ called the eccentric angle of the ellipse is measured around the perimeter of a bounding circle.

Midpoint ellipse Algorithm

The midpoint ellipse method is applied throughout the first quadrant in two parts.

The below figure show the division of the first quadrant according to the slope of an ellipse with $r_x < r_y$.

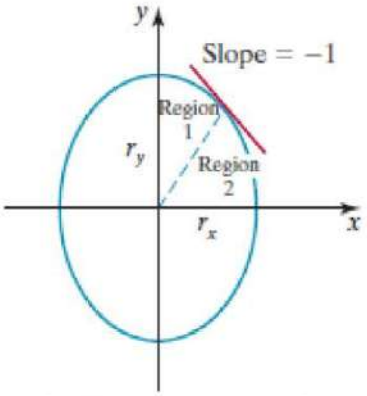


Figure. 5.10: Ellipse processing Regions shows the value of magnitude

In the x direction where the slope of the curve has a magnitude less than 1 and unit steps in the y direction where the slope has a magnitude greater than 1.

Region 1 and 2 can be processed in various ways which are as follow:

1. Start at position $(0, r_y)$ and step clockwise along the elliptical path in the first quadrant shifting from unit steps in x to unit steps in y when the slope becomes less than -1.

2. Start at $(r_x, 0)$ and select points in a counter clockwise order.

    2.1 Shifting from unit steps in y to unit steps in x when the slope becomes greater than -1.0

    2.2 Using parallel processors calculate pixel positions in the two regions simultaneously.

3. Start at $(0, r_y)$ step along the ellipse path in clockwise order throughout the first quadrant ellipse function $(x_c, y_c) = (0, 0)$

$$f_{ellipse}(x, y) = ry^2x^2 + rx^2y^2 - rx^2\, ry^2$$

Which has the following properties:

$f_{ellipse}(x, y) < 0$, if (x, y) is inside the ellipse boundary

$= 0$, if(x, y) is on ellipse boundary

> 0, if(x, y) is outside the ellipse boundary

Thus, the ellipse function $f_{ellipse}(x, y)$ serves as the decision parameter in the midpoint algorithm.

Starting at (0, r_y):

Unit steps in the x direction until to reach the boundary between region 1 and region 2. Then switch to unit steps in the y direction over the remainder of the curve in the first quadrant.

At each step to test the value of the slope of the curve. The ellipse slope is calculated

$$dy / dx = -(2ry^2x / 2rx^2y)$$

At the boundary between region 1 and region 2,

$dy / dx = -1.0$ and $2ry^2x = 2rx^2y$

to more out of region 1 whenever

$2ry^2x >= 2rx^2y$

The following figure denotes the midpoint between two candidate pixels at sampling position x_{k+1} in the first region.

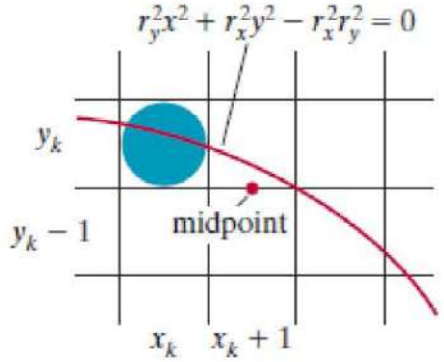


Figure. 5.11: Mid-Point between candidate pixels at sampling position along with elliptical path

To determine the next position along the ellipse path by evaluating the decision parameter at this mid-point

$$P1_k = f_{ellipse}(x_{k+1}, y_k - 1/2)$$
$$= ry^2 (x_{k+1})^2 + rx^2 (y_k - 1/2)^2 - rx^2 ry^2$$

If $P1_k < 0$, the midpoint is inside the ellipse and the pixel on scan line y_k is closer to the ellipse boundary. Otherwise the midpoint is outside or on the ellipse boundary and select the pixel on scan line y_{k-1}.

At the next sampling position (x_{k+1}+1=x_k+2) the decision parameter for region 1 is calculated as:

$$P1_{k+1} = f_{ellipse}(x_{k+1}+1, y_{k+1} - 1/2)$$
$$= \text{ry}^2 (x_{k+1}+1)^2 + \text{rx}^2 (y_{k+1} - 1/2)^2 - \text{rx}^2\ \text{ry}^2$$

Or

$$P1_{k+1} = P1_k + 2\, r_y^2\, (x_k + 1) + r_y^2 + r_x^2\, [(y_{k+1} - 1/2)^2 - (y_k - 1/2)^2]$$

Where y_{k+1} is [(y_k or y_{k-1} depending on the sign of $P1_k$.

Decision parameters are incremented by the following amounts:

$$\text{increment} = \{\ 2r_y^2\, (x_k + 1) + r_y^2 \text{ if } P1_k < 0\ \}$$
$$\{\ 2r_y^2\, (x_k + 1) + r_y^2 - 2\, r_x^2\, y_{k+1} \text{ if } P1_k \geq 0\ \}$$

Increments for the decision parameters can be calculated using only addition and subtraction as in the circle algorithm.

The terms $2r_y^2$x and $2r_x^2$y can be obtained incrementally. At the initial position $(0, r_y)$ these two terms evaluate to:

$$2r_y^2\text{x} = 0$$
$$2r_x^2\text{y} = 2r_x^2\, r_y$$

The updated increment values are compared at each step and more from region 1 to region 2. In region 1 the initial value of the decision parameter is obtained by evaluating the ellipse function at the start position:

$$(x_0, y_0) = (0, r_y)$$

Region 2 at unit intervals in the negative y direction and the midpoint is now taken between horizontal pixels at each step for this region the decision parameter is evaluated as:

$$P1_0 = f_{ellipse}(1, r_y - 1/2)$$
$$= r_y^2 + r_x^2\, (r_y - 1/2)^2 - r_x^2\, r_y^2$$

Over region 2, we sample at unit steps in the negative y direction and the midpoint is now taken between horizontal pixels at each step. For this region, the decision parameter is evaluated as:

$$P2_k = f_{ellipse}(x_k + 1/2, y_k - 1)$$
$$= r_y^2(x_k + 1/2)^2 + r_x^2 (y_k - 1)^2 - r_x^2 r_y^2$$

1. If $P2_k > 0$, the mid-point position is outside the ellipse boundary, and select the pixel at x_k.
2. If $P2_k <= 0$, the mid-point is inside the ellipse boundary and select pixel position x_{k+1}.

To determine the relationship between successive decision parameters in region 2 evaluate the ellipse function at the sampling step: y_{k+1} - 1= y_{k-2}.

$$P2_k = f_{ellipse}(x_{k+1} + 1/2, y_{k+1} - 1)$$
$$= r_y^2(x_{k+1} + 1/2)^2 + r_x^2 ((y_{k+1} - 1) - 1)^2 - r_x^2 r_y^2$$

Or

$$P2_{k+1} = P2_k - 2 r_x^2(y_{k+1} - 1) + r_x^2 + r_y^2[(x_{k+1} + 1/2)^2 - (x_k + ½)^2]$$

With x_{k+1} set either to x_k or x_{k+1}, depending on the sign of $P2_k$. when we enter region 2, the initial position (x_0, y_0) is taken as the last position. Selected in region 1 and the initial decision parameter in region 2 is then equation would be as follows:

$$P2_0 = f_{ellipse}(x_0 + 1/2, y_0 - 1)$$
$$= r_y^2(x_0 + 1/2)^2 + r_x^2 ((y_0 - 1)^2 - r_x^2 r_y^2$$

To simplify the $P2_0$, select pixel positions in counter clock wise order starting at $(r_x, 0)$. Unit steps would then be taken in the positive y direction up to the last position selected in region 1.

Mid-point Ellipse Algorithm

1. Input (r_x, r_y) and ellipse center (x_c, y_c) and obtain the first point on an ellipse centered on the origin as

$$(x_0, y_0) = (0, r_y)$$

2. Calculate the initial value of the decision parameter in region 1 as

$$P1_0 = r_y^2 - r_x^2 r_y + (1/4) r_x^2$$

3. At each x_k position in region1 starting at k=0 perform the following test. If $P1_k$<0, the next point along the ellipse centered on (0,0) is (x_{k+1}, y_k) and

$$P1_{k+1} = P1_k + 2r_y^2 x_{k+1} + r_y^2$$

Otherwise the next point along the ellipse is (x_k+1, y_k-1) and

$$P1_{k+1} = P1_k + 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1} + r_y^2$$

with

$2r_y^2 x_k + 1 = 2r_y^2 x_k + 2r_y^2$

$2r_x^2 y_k + 1 = 2r_x^2 y_k + 2r_x^2$

And continue until $2r_y^2$x >= $2r_x^2$y.

4. Calculate the initial value of the decision parameter in region 2 using the last point (x_0, y_0) is the last position calculated in region 1.

$$P2_0 = r_y^2 (x_0 + 1/2)^2 + r_x^2 ((y_0 - 1)^2 - r_x^2 r_y^2$$

5. At each position y_k in region 2, starting at k=0 perform the following test, If $P2_k$>0 the next point along the ellipse centered on (0, 0) is (x_k, y_{k-1})) and

$$P2_{k+1} = P2_k - 2r_x^2 y_{k+1} + r_x^2$$

Otherwise the next point along the ellipse is (xk+1,yk-1) and

$$P2_{k+1} = P2_k + 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1} + r_x^2$$

Using the same incremental calculations for x any y as in region 1.

6. Determine symmetry points in the other three quadrants.

7. Move each calculate pixel position (x, y) onto the elliptical path centered on (x_c, y_c) and plot the coordinate values

$$x = x + x_c, \qquad y = y + y_c$$

8. Repeat the steps for region1 unit $2r_y^2$x >= $2r_x^2$y.

Example: Mid-point ellipse drawing

Input ellipse parameters r_x=8 and r_y=6 the mid-point ellipse algorithm by determining raster position along the ellipse path is the first quadrant. Initial values and increments for the decision parameter calculations are

$2r_y^2$x =0 (with increment $2r_y^2$ = 72)

$2r_x^2$y = $2r_x^2r_y$ (with increment -$2r_x^2$= -128)

For region 1 the initial point for the ellipse centered on origin is (x_0, y_0) = (0, 6) and the initial decision parameter value is

$P1_0 = r_y^2 - r_x^2r_y + (1/4)r_x^2$ = -332

Successive midpoint decision parameter values and the pixel positions along the ellipse are listed in the following table.

K	$P1_k$	(x_{k+1}, y_{k+1})	$2r_y^2x_{k+1}$	$2r_x^2y_{k+1}$
0	-332	(1, 6)	72	768
1	-224	(2, 6)	144	768
2	-44	(3, 6)	216	768
3	208	(4, 5)	288	640
4	-108	(5, 5)	360	640
5	288	(6, 4)	432	512
6	244	(7, 3)	504	384

Move the region 1, $2r_y^2$x > $2r_x^2$y.

For a region 2 the initial point is (x_0, y_0) = (7, 3) and the initial decision parameter is

$$P2_0 = f_{ellipse}(7+1/2, 2) = -151$$

The remaining positions along the ellipse path in the first quadrant are then calculated as

K	$P2_k$	(x_{k+1}, y_{k+1})	$2r_y^2x_{k+1}$	$2r_x^2y_{k+1}$
0	-151	(8, 2)	576	256
1	233	(8, 1)	576	128
2	745	(8, 0)	-	-

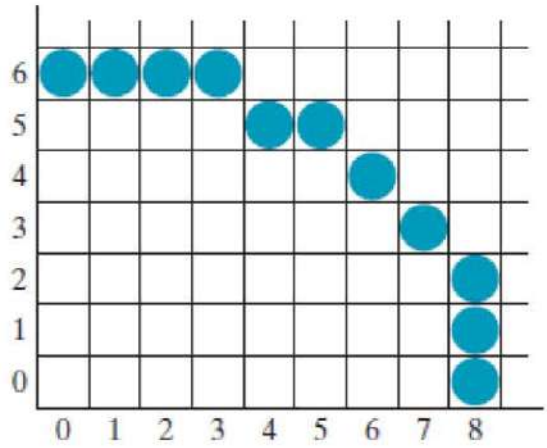


Figure.5.12:

Implementation of Midpoint Ellipse drawing

```
#define Round (a) ((int) (a+0.5))
Void ellipseMidpoint (int xCenter, int yCenter, int Rx, int Ry)
{
int Rx2=Rx*Rx;
int Ry2=Ry*Ry;
int twoRx2 = 2*Rx2;
int twoRy2 = 2*Ry2;
int p;
int x = 0;
int y = Ry;
int px = 0;
int py = twoRx2* y;
Void ellipsePlotPoints ( int , int , int , int ) ;
/* Plot the first set of points */
ellipsePlotPoints (xcenter, yCenter, x, y ) ;
/ * Region 1 */
p = ROUND (Ry2 - (Rx2* Ry) + (0.25*Rx2));
While (px < py)
{
x++;
px += twoRy2;
if (p < 0)
```

```
p += Ry2 + px;
else
{
y - - ;
py -= twoRx2;
p+= Ry2 + px - py;
}
ellipsePlotPoints(xCenter, yCenter, x, y);
}
/* Region 2 */
p = ROUND (Ry2*(x+0.5)*' (x+0.5)+ Rx2*(y- 1 )* (y- 1 ) - Rx2*Ry2);
while (y > 0 )
{
y--;
py -= twoRx2;
if (p > 0)
p += Rx2 - py;
Else
{
x++;
px+=twoRy2;
p+=Rx2-py+px;
}
ellipsePlotPoints(xCenter, yCenter, x, y);
}
}
void ellipsePlotPoints(int xCenter, int yCenter, int x, int y);
{
setpixel (xCenter + x, yCenter + y);
setpixel (xCenter - x, yCenter + y);
setpixel (xCenter + x, yCenter - y);
setpixel (xCenter- x, yCenter - y);
}
```

5.5 Self Learning Exercise

Q.1. Distinguish between lines, curve and ellipse drawing algorithms.

Q.2. Extend Bresenham's line algorithm to generate lines with any slope, taking symmetry between quadrants into account.

Q.3. Set up a parallel version of Bresenham's line algorithm for slopes in the range $0 < m < 1$.

5.6 Summary

We have discussed various scan conversion algorithms of output primitives. These output primitives provide the basic tools for constructing pictures with individual points, straight line and curves. Three methods that can be used to locate pixel positions along a straight-line path are the DDA algorithm, Bresenham's algorithm, and the midpoint method.

Bresenham's line algorithm and the midpoint line method are equivalent, and they are the most efficient. Color values for the pixel positions along the line path are efficiently stored in the frame buffer by incrementally calculating the memory addresses. Any of the line-generating algorithms can be adapted to a parallel implementation by partitioning the line segments and distributing the partitions among the available processors. Circles and ellipses can be efficiently and accurately scan converted using midpoint methods and taking curve symmetry into account.

These algorithms are also explained with suitable examples.

5.7 Glossary

DDA: Digital Differential Analyzer.

Ellipse: An ellipse is an elongated circle along with major and minor axis.

5.8 Answers to Self-Learning Exercise

Ans.1, Ans.2 and **Ans.3** were discussed in chapter in details. See respective contents.

5.9 Exercise

Q.1. The syntax of Line color is:

(A) Persistence, resolution and aspect ratio

(B) Persistence, resolution and pixel ratio

(C) Perseverance, resolution and pixel ratio

(D) Perseverance, resolution and aspect ratio

Q.2. DDA and Bresenham algorithm are _____ drawing algorithms.

(A) Circle (B) Ellipse

(C) Line (D) None of these

Q.3. The equation of circle is $(x-x_c)^2+(y-y_c)^2=r^2$

(A) True

(B) False

Q.4. A spline is a:

(A) Straight curve through a designed set of points

(B) Smooth curve through a designed set of points

(C) Straight curve through a designed set of pixels

(D) Smooth curve through a set of points

Q.5. Explain the Bresenham's line drawing algorithm with suitable example.

Q.6. Explain the mid-point circle drawing algorithm with suitable example. Write its implementation process in C language.

5.10 Answers to Exercise

Ans.1: A **Ans.2:** C **Ans.3:** A

Ans.4: B

References and Suggested Readings

1. J. Foley, A. Van Dam, S. Feiner, J. Hughes: Computer Graphics- Principles and Practice, Pearson.
2. Hearn and Baker: Computer Graphics, PHI.
3. R. K. Chauhan and Abhishek Taneja "Computer Graphics and Multimedia" Galgotia, 2009.

UNIT-6
Color Filling Algorithm

Structure of the Unit

6.0 Objective

In this chapter, we shall focus on the following topics

- Filled-Area Primitives
- Scan Line Polygon fill Algorithm Inside-Outside Tests
- Seed Fill Algorithm

6.1 Introduction

Before using filled area primitive we should able to draw certain shapes with combinations of lines like a triangle, rectangle, square and other polygons. For fill area, we use polygon as a shape. For area filling, we follow various approaches:

i.e. Flood fill algorithm, Boundary fill algorithm and scan line polygon fill algorithm.

6.2 Filled Area Primitives

In general graphics packages, a standard output primitive is a solid color or patterned polygon area. Polygons are easier to process as compare to other area primitives. So polygon consists linear boundaries. There are two ways for filling an area on raster system. First is to start from a given interior position and fill color corresponding to this position until we get the specified boundary conditions. To fill polygons, ellipse, circle and other curves we use scan line approach as general graphics packages. Fill process starting from an interior point with more complex boundaries. While the Second method is to fill area by determining the overlap intervals for scan lines that cross the area.

6.3 Scan Line Polygon fill Algorithm

Before start discussion about the algorithm, one should know about what is polygon? Besides polygon definition we will know about following topics one by one, which are as follows:

- Polygon Definition
- Filled v/s Unfilled Polygon
- Parity Definition
- Scan-Line Polygon Fill Algorithm
- Special Cases
- Polygon Fill Example

Polygon

A Polygon is formed by line segments that are placed end to end, making a continuous closed path. Polygon is classified into three basic types:

Convex, Concave and **Complex**.

I. **Convex polygons** are the simplest type of polygon to fill. A polygon is said to be convex if it fulfills following aspects:
   – All interior angles < 180 graden, and
   – All line segments between 2 interior points in polygon, and
   – All points on the same side of line through edge, and
   – From each interior point complete boundary visible

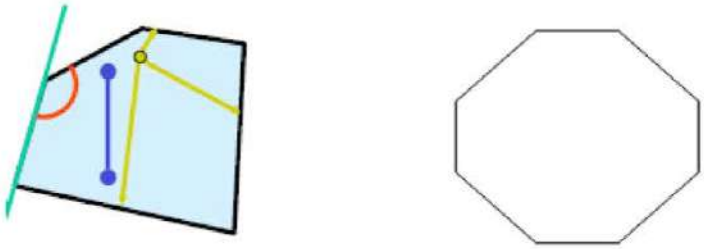


Figure.6.1:convex polygon

II. **Concave Polygons** are a superset of convex polygons, with certain restrictions than convex polygons. The line connecting any two points that lie inside the polygon may intersect more than two edges of the polygon. Thus, more than two edges may intersect any scan line that passes through the polygon. The polygon edges may also touch each other, but they may not cross one another.

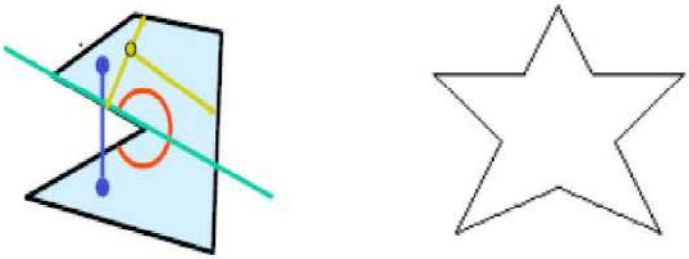


Figure. 6.2: concave polygon

III. Complex Polygons are concave polygons that may have self-intersecting edges. The complexity arises from distinguishing which side is inside the polygon when filling it.

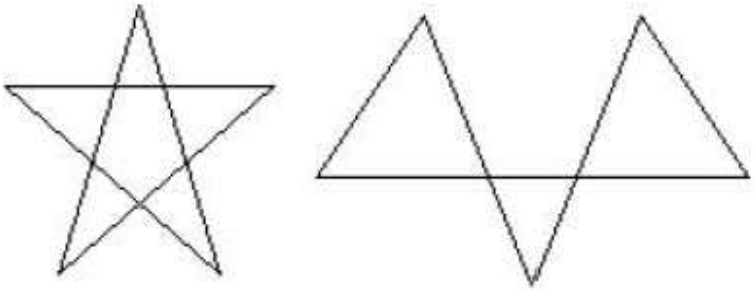


Figure. 6.3: complex polygon

Filled v/s Unfilled Polygon

If an unfilled polygon is rendered, only the points on the perimeter of the polygon are drawn. However, if a polygon is filled, the polygon's interior is considered. All the pixels within the boundaries of the polygon must be set to the specified color or pattern. The following figure shows the difference between filled and unfilled polygons.

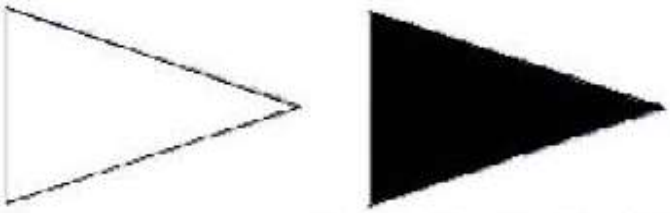


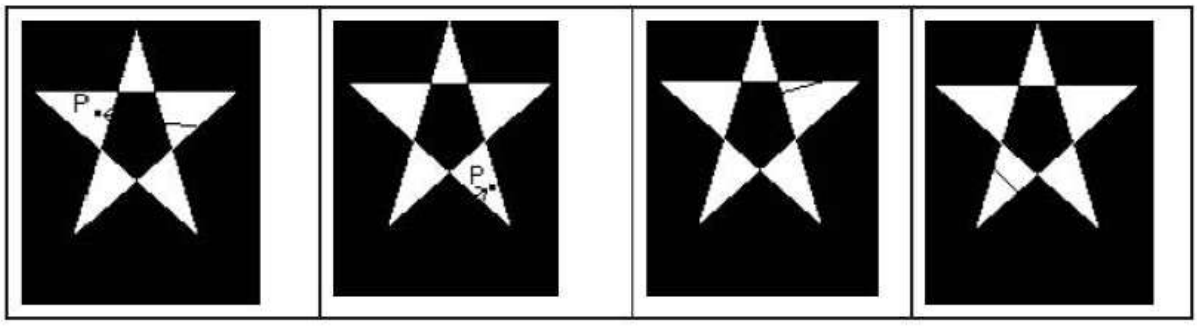
Figure.6.4: Filled and Unfilled Polygon

To check which pixels are inside the polygon, the odd-parity rule is used within the scan-line polygon fill algorithm.

Parity

Parity is a concept used to determine which pixels lie within a polygon, i.e. which pixels should be filled for a given polygon.

Principle: Conceptually, the odd parity test means drawing a line segment from any point that lies outside the polygon to a point P that we wish to determine whether it is inside or outside of the polygon. Count the number of edges that the



line crosses.

Figure.6.5: Parity for pixels

If the number of polygon edges crossed is odd, then P lies within the polygon. Similarly, if the number of edges is even, then P lies outside of the polygon. There are special methods for counting the edges when the line crosses a vertex. This will be discussed in the algorithm section.

Scan-Line Polygon Fill Algorithm

In scan line polygon fill algorithm:

- We check each scan-line which crosses the polygon boundary.
- For each scan line do-
  - Get intersection points of scan-line with polygon edges.
  - Sort these intersection points from left to right with increasing value of x.
  - From these intersection pairs, No. of intersections points are even, so we can easily form pairs.
- Between each pair, fill the pixels with the desired color using the draw-pixel function.
- Update the list of intersection points for each scan line.
- The above process would be completed when scan-line has reached y_{max} value for a polygon.

Before filling pixels, we should check whether pixels are within polygon or not. So we look from left to right of pixels that are to be plotted. If the count of pixels from left to right and vice versa is odd, then the pixel is inside the polygon. If the count is even on both sides, the pixel is at the background or outside the polygon. In the example of fig 6-6, the four pixel intersection positions with polygon boundaries define two stretches of interior pixels from x=10 to x=14 and from x=18 to x=24.

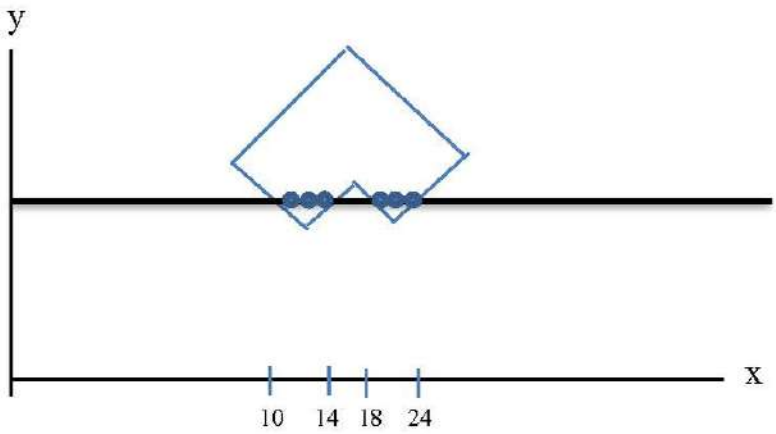


Figure.6.6: Interior pixel along a scan-line passing through a polygon area

Some scan line intersection at polygon vertices requires special handling procedure. Fig 6-7 denotes two scan lines at y and y' positions that intersect edge end points. Scan-line y intersects 5 polygon edges. Scan line y' intersects an even no. of edges although it also passes through a vertex. Intersection point along scan-line y' correctly identify the interior pixel spans. For scan-line y, two intersecting edges sharing a vertex are on opposite sides of the scan-line. But for scan-line y', two intersecting edges are both above the scan-line.

Thus, the vertices that require additional processing are those that have connecting edges on opposite sides of the scan line. For find out such type of vertices:

- We trace polygon boundary either in clockwise or anticlockwise.
- Observe relative changes in vertex y coordinates as we move from one edge to another.

- If the end point y value of two consecutive edges increase or decrease monotonically, we count the middle vertex as single intersection point for scan-line passing through it.
- If the end point y value does not increase or decrease monotonically, it represents local extremum on polygon boundary.

Two edge intersections with the scan-line passing through that vertex can be added to intersection list.

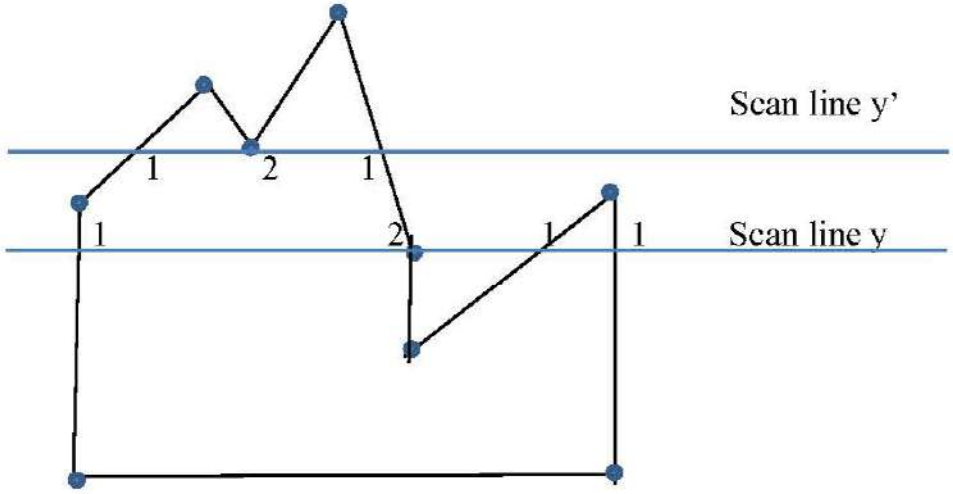
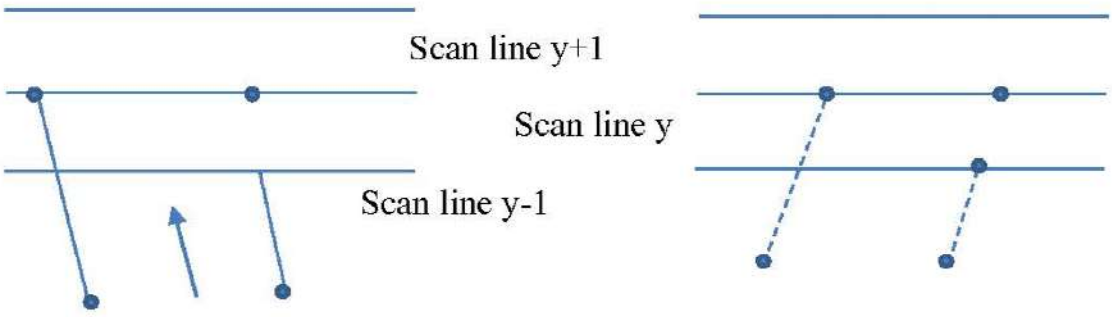


Figure. 6.7: intersection of scan-line y (odd intersections) and y' (even intersections)

A way to solve the problem of whether the vertex point should be counted as one or two is to shorten polygon edges in the case when vertex can be counted as one intersection.

When the endpoint y of two edges are increasing, the y values of the upper-end point for the current edge is decreased by 1. Fig 6-8 (a).

And if the y value is monotonically decreasing, we decrease y coordinate of upper end point of the edge following the current edge. Fig 6-8 (b).



(a) (b)

Figure. 6.8: Adjusting endpoint y values for a polygon. Edge currently being processed is indicated by a solid line.

Now we are going to handle some special cases to make it sure it works fast and correct. There are two important features of scan-line based polygon fill algorithm.

1. Scan-line coherence: shows that value does not change much from one scan-line to next. Example, Coverage (visibility) of a face on one scan-line typically differs little from the previous.
2. Edge coherence: shows that slope of the edge intersected by the scan-line `i` is almost same as intersected by scan-line `i+1`.It will change only if you pass through a vertex.

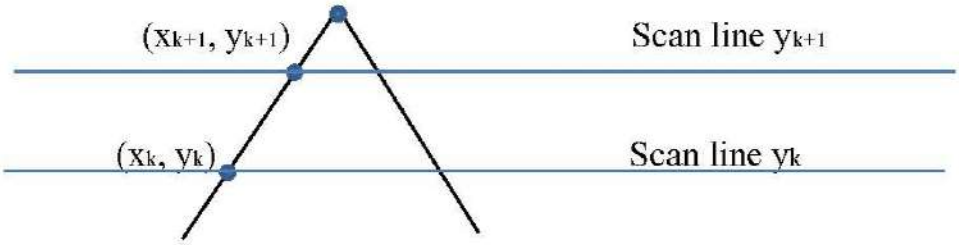


Figure. 6.9: Two successive scan lines intersecting a polygon boundary

The slope of polygon boundary line in the form of scan line intersection coordinates:

$$m = \frac{y_{k+1} - y_k}{x_{k+1} - x_k} \qquad \ldots\ldots\ldots\ldots (1)$$

Since change in y coordinates between two scan line is

$$y_{k+1} - y_k = 1 \qquad \text{..... (2)}$$

Along an edge with slope m, the intersection x_k value for scan line k above the initial scan line can be calculated as:

$$x_k = x_0 + \frac{k}{m} \qquad \text{..... (3)}$$

Slope of line can be recalled using two integer values i.e.

$$m = \frac{\Delta x}{\Delta y}$$

So, incremental calculation of x intercepts along an edge for successive scan line can be termed as:

$$x_{k+1} = x_k + \frac{\Delta x}{\Delta y} \qquad \text{..... (4)}$$

Example 1. Given $m = \frac{\Delta x}{\Delta y} = \frac{7}{3}$, Counter C is set to C=0 and C increment $\Delta C = \Delta x = 3$.

So next 3 scan-lines successive values of C are 3, 6, 9. At 3rd scan line C >Δy. Whenever counter value equal or greater than Δy, increment x by 1 and decrease counter by value Δy. at 3rd scan line x_k is incremented by 1 and counter C is:

C= C - Δy = 9 - 7= 2

the process would be continued until we reach y_{max}.

For calculating next intersection point for next scan-line, we define SET (sorted edge table) and AEL (Active edge list). SET contains all information necessary to process the scan lines effectively. SET is built using bucket sort. All edges are sorted by their y_{min} coordinates with a separate y bucket for each scan line. Within each bucket, edges are sorted by increasing value of x of the y_{min} point. The edges are stored in the SET at scan-line position. Each entry in edge table contains y_{max}, x_{min}, Δx/Δy and pointer to next edge.

AEL contains all edges crossed by a scan-line at the current stage of iteration. It is a list of edges that are active for a current scan lines, sorted by increasing x intersections. AES is also called as AET(Active Edge Table).

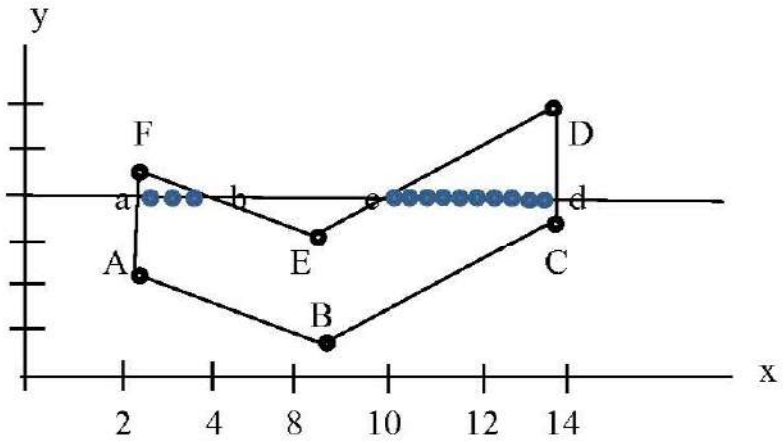
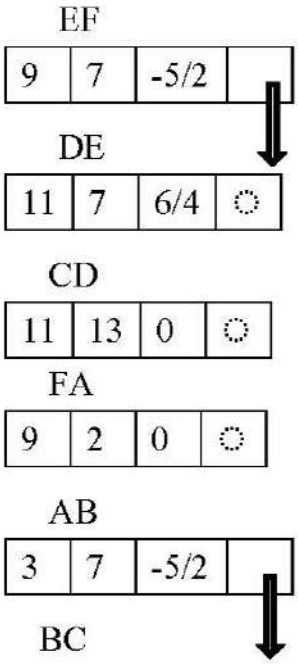


Figure.6.10: SET data structure solve using scan fill algorithm

To start with data structures, SET is built using bucket sort with sorted y_{min} coordinates. for every edge find out y_{min} and put it in the corresponding scan-line bucket number. In Fig 6-9, for scan-line 1, it passes through vertex B, AB and BC edges values stored for scan-line 1 in the corresponding bucket has values for y_{max}, x_{min}, $\Delta x/\Delta y$ and next edge pointer.

For edge AB, y_{max}= 3, x_{min}= 7, $\Delta x/\Delta y$= -5/2 and ◌ shows null pointer.



5	7	6/4	◌

Now active edge list for fig 6-9 for each scan-line and edge sorted with increasing of x values in each scan-line can be made now.

AEL for scan-line 8 is:

AET pointer ⟶ FA ⟶ EF ⟶ DE ⟶ CD

Scan-line 8 crosses the edges FA, EF, DE, and CD. So, they are sorted with increasing value of x in AEL. Making pairs first then filling pixels between those pairs in the next step. FA and EF forms pairs and their x_{min} value 2 and 4 are x-values between which we have to fill the pixels. Similarly between 9 and 13 we fill pixels.

AEL for scan-line 9 is:

Here y=9 which is y_{max} for FA and EF. So we remove edges FA and EF from AEL.

Now we fill pixels between x=10 to 13. So this process is repeated filling other scan-lines. During each iteration with scan-line AET is updated.

Scan-line Algorithm (scan – fill polygon)

```
Construct SET
        y_min = min of all y in SET
        AET = NULL
For y = y_min to y_max
        Merge sort ET[y] into AET by x values
        Fill between pairs of x in AET
        For each edge in AET
                If edge. y_max = y
                        Remove edge from AET
                Else
                        edge. x = edge. x + dx/dy
        End if
  Sort AET by x values
```

And loop.

Inside and outside tests:

All area filling algorithm need to the interior region of objects. Identifying the interior region of the standard polygon is a straightforward process. Graphic packages normally use either odd-even rule or nonzero winding numbers rule to identify the interior region of an object.

Odd even rule, draw a line from any position P to a distant point outside the coordinates of a polygon and count number of edges crossing the line. If the no. of edges crossed by this line is odd, then P is an interior point. Otherwise, it is an exterior point. We should also keep in mind that the line path we choose does not intersect any polygon vertices.

Non-zero winding number rule counts the number of times polygon edges wind around a particular point in counter clockwise direction. Such count is winding number. And interior points of a 2D object are defined to be those that have a non- zero value for winding number.

Processing NZWN rule:

- We apply this rule for a polygon by set window number to 0 and again assuming a line drawn from any position P to a distant point i.e. beyond to the coordinate extent of the object.
- The line i.e. chosen must not pass through any vertices. We move along the line from position P to distant point, and count number of edges that cross the line each direction.
- Add 1 to winding number when polygon edge intersects and crosses line from right to left.
- We subtract 1 whenever intersects edge that crosses the line from left to right.
- The final value of window number is considered after all edges crossing have been counted, determine the relative position of P.
- If the winding number is non-zero P is defined to the interior point, otherwise P is taken as exterior point.

- This method is applicable for std. polygon or simple shapes and produces the same result as odd even rule.

For complex shapes, the way is to determine directional edge crossing is to take the vector cross product of a vector u along with the line from P to the distant point with edge vector E for each edge that crosses the line.

- If the z component of cross product u×E for a particular edge is positive, that edge crosses from right to left we add 1 to window number.
- Otherwise, the edge crosses from left to right then we subtract 1 from window number.
-
- Now edge vector is calculated by subtracting starting vertex position for that edge from the ending vertex position.

$$E_{AB} = V_B - V_A$$

- Where V_A and V_B denotes point vectors for vertices A and B.

6.4 Seed fill Algorithm

The second approach to filling area is to start a point inside a region. It is assumed that at least one pixel is interior to a polygon or region. The region may be either interior defined or boundary defined. In interior defined region, all the pixels in interior region have one color and pixels in the exterior region have another color. For boundary filling region, all pixels on the boundary have a unique color and pixels exterior to boundary may also have boundary color. The algorithm that fills interior region is called flood fill algorithms and algorithm for fill boundary defined region is called boundary fill algorithms.

Boundary fill algorithm

This algorithm takes as an input an interior point say (x, y) fill color and a boundary color. The process starts with point (x, y), we test for each neighboring pixel point of (x, y) to determine whether they are of boundary color. If not, they are filled with fill color. And their neighbors are tested. This will remain continue until all the pixels up to the boundary color for the area have been tested. Testing of the neighboring pixels can be done using two methods: 4-connected method and 8-connected method. The first method tests the pixels on left, right, top and bottom of the current pixel. The second method tests the pixels on the four diagonal position with the left, right, top and bottom positions. These methods are shown below in fig 6-10.

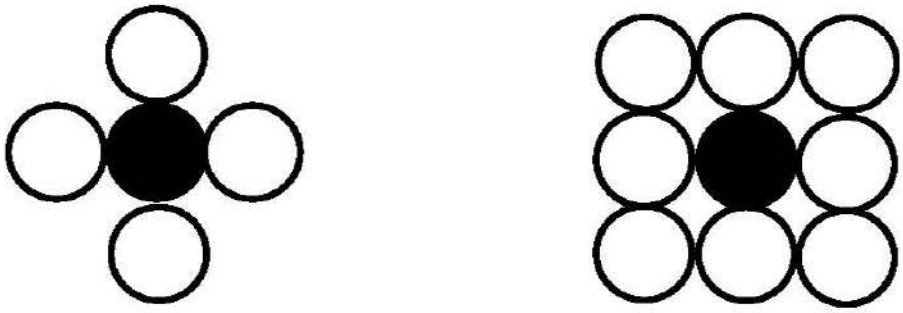


Figure.6.11: Filled method applied to 4-connected and 8-connected area

Boundary fill algorithm

Boundary fill
Begin
    Initialize fill color and boundary color,
    Get current pixel value at position (x, y),
    If (current value ≠ fill color and current value ≠ boundary color)
    Begin
        Set current with pixel fill color;

Recursively fill neighboring pixels;

End if

End boundary fill.

Stack based Boundary fill algorithm

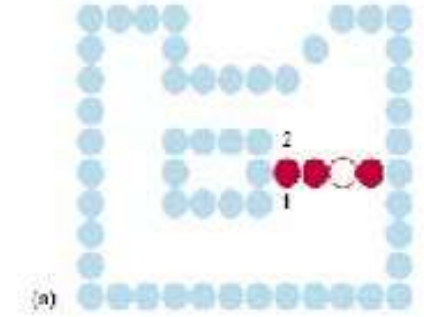
A simple scheme for a boundary defined region can be developed using a stack. That is referred to as FILO (First In Last Out) algorithm.

- Push the seed pixel, chosen from the interior of the region on to the stack.
- While the stack is not empty,
  - Pop a pixel from the stack.
  - Set the pixel to the required value (color).
- For each of 4-connected pixels adjacent to the current pixel, check whether it is boundary pixel or if it has been already filled. If it is so, ignore, else push the pixel on to the stack.

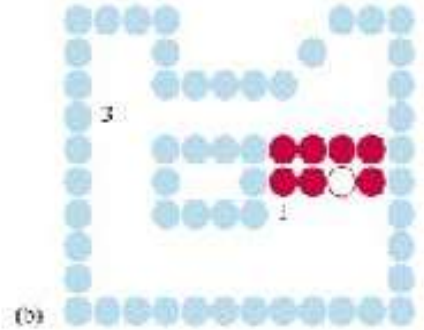
There is a drawback in above method. We can fill the region partially starting with the initial position. This is because we cannot fill region pixels diagonally.

The better way of filling polygon is to fill each line horizontally. It limits tack size by using only one seed pixel in scan line span. This horizontal span is a group of contiguous pixels covered by pixels in border color area.

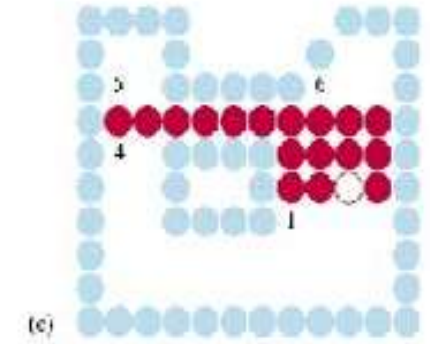
- Seed pixel is popped from the stack and placed on the span.
- Extreme pixels i.e. left and right in the span are designed as x_l and x_r.
- This span is filled on both sides using x_l and x_r.
- Scan line above and below the current line are examined with fill and boundary color. If they are not filled with any color, extreme left pixel in each span is marked as seed pixel and pushed on to the stack.
- Method is better to other because we push on the stack only beginning position for each scan line, instead of pushing all unprocessed neighbor pixels.

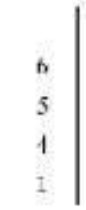

2
1
(a)

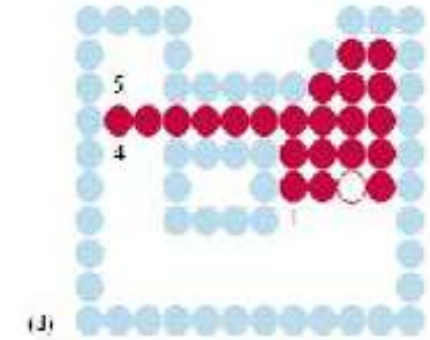

2
1


3
1
(b)

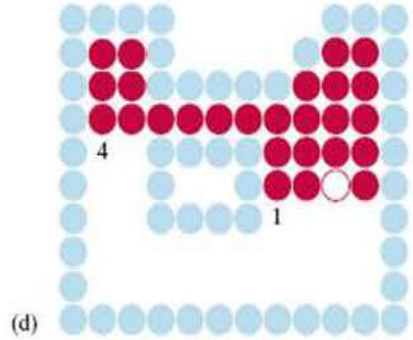

3
1

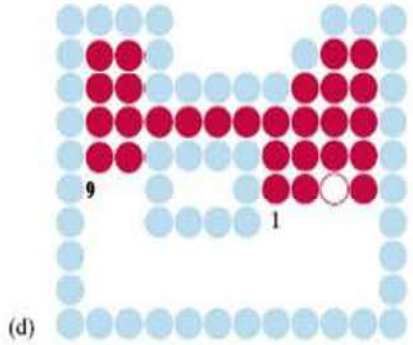

5
6
4
1
(c)

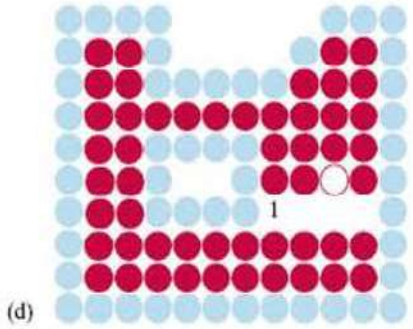

6
5
4
1

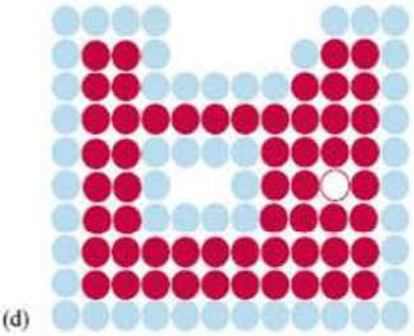

5
4
1
(d)


5
4
1









Flood fill Algorithm

Flood fill

Begin

Initialize fill color,

```
Get color value at current pixel position;
If (current value = = old color)
Begin
        Set current pixel with fill color
        Recursively fill the neighboring pixels;
End if.
End flood fill.
```

6.5 Self Learning Exercise

Q.1. Write a short note on:

(a) Boundary fill algorithm

(b) Flood fill algorithm

Q.2. Write boundary fill algorithm to fill an 8-connected region.

6.6 Summary

We have discussed filled area primitives to fill polygon. A common method for polygon fill on raster system is scan-line fill algorithm that determines interior pixel span across scan-line that intersect the polygon. This algorithm is also used to fill the interior of objects with curved boundaries. Two other methods for interior region filling are boundary fill and flood fill algorithm. These algorithm paint the interior, one pixel at a time. The scan-line fill algorithm is an example of filling object interiors using odd even rule to locate the interior region.

6.7 Glossary

Nonzero winding number rule: It counts the number of times polygon edges wind around a particular point in counter clockwise direction.

Parity: Parity is a concept used to determine which pixels lie within a polygon,

6.8 Answers to Self-Learning Exercise

Ans.1 and **Ans.2** discussed in chapter in details. See respective contents.

6.9 Exercise

Q.1. The process of coloring the area of a polygon is called

(A) Polygon filling (B) Polygon flow

(C) Aliasing (D) None of these

Q.2. The algorithm used for filling the interior of a polygon is called

(A) Flood fill algorithm (B) Boundary fill algorithm

(C) Scan-line Polygon fill algorithm

(D) None of these

Q.3. If the pixel is already filled with desired color then leaves it otherwise fills it this is called

(A) Flood fill algorithm (B) Boundary fill algorithm

(C) Scan-line Polygon fill algorithm

(D) None of these

Q.4. A vector can be defined as

(A) Intersection b/w two point position

(B) Difference b/w two point position

(C) Comparison b/w two point position

(D) None of these

Q.5. Write boundary fill algorithm to fill a 4-connected region.

Q.6. Explain the algorithm for scan line polygon filling.

Q.7. Explain odd even parity rule in detail.

6.10 Answers to Exercise

Ans.1: A

Ans.2: A

Ans.3: B

Ans.4: B

References and Suggested Readings

1. J. Foley, A. Van Dam, S. Feiner, J. Hughes: Computer Graphics- Principles and Practice, Pearson.
2. Hearn and Baker: Computer Graphics, PHI.
3. Additional programming examples and information on PHIGS primitive can be found in Howard, et al. 1991
4. Filled-Area_Primitives_I-Computer_Graphics-Lecture_Notes.pdf

UNIT-7
Attributes of Output Primitives

Structure of the Unit

7.0 Objective

In this chapter, we shall focus on the following topics

- Line Attributes
- Curve Attributes
- Character Attributes
- Antialiasing

7.1 Introduction

Any parameter that affects primitives to be displayed is treated as an attribute parameter. Some attributes like color and size determine fundamental characteristics of a primitive. Other show how primitives work under special conditions. In this unit, we include those attributes that control the basic display properties of primitives. For example, line can be dotted or dashed, fat or thin and blue or orange. The area can be filled with single or multiple colors. Text may display reading from left to right, slanted diagonally across the screen or vertical columns. Text character can be displayed in different colors, fonts and sizes.

To get appropriate attributes, we extend the parameter list of each output primitives in graphics package. For example, a line drawing function could contain parameters to set the color, width and other parameters, in addition to end point coordinates. Another way is to maintain the system list of current attribute values. To generate an output primitive, system checks the relevant attributes and invoke the display routine for that primitive using the current attribute setting. Some graphics packages provide attribute function and attribute parameters in the output primitives.

7.2 Line Attributes

Basic attributes of a line segment are its width and color. Some graphic packages denote line can be drawn using a selected pen or brush option. Fundamental line attributes are Line Type, Line Width, Pen and Brush Options, Line Color etc.

Line type

Possible selection of line type attribute includes solid lines, dashed lines and dotted lines. To set line type attributes in a PHIGS application program, a user invokes the function

setLinetype (lt)

Where parameter 1t is assigned a positive integer value of 1, 2, 3 or 4 to generate lines that are solid, dashed, dash dotted respectively. Other values for line type parameter it could be used to display variations in dot-dash patterns.

Raster line algorithms denote line-type attributes by plotting pixel spans. For dashed, dotted, and dot-dashed patterns, the line-drawing procedure outputs sections of contiguous pixels along the line path, skipping over a number of intervening pixels between the solid spans. Where pixel counts for the span length and inter-span spacing can be specified in a pixel mask, i.e. a pattern of binary digits indicating which positions to plot along the line path. The linear mask 11111100, for instance, could be used to display a dashed line with a dash length of six pixels and an inter-dash spacing of two pixels. Pixel positions corresponding to the 1 bits are assigned the current color, and pixel positions corresponding to the 0 bits are displayed in the background color.

(a)

Figure 7-1 Unequal length dashes
displayed with the same number of pixels.

(b)

Figure.7.1: Unequal length dashes displayed with the same number of pixels.

Drawing dashes with a fixed number of pixels results in unequal length dashes for different line orientations, as shown in Fig. 7-1. Both dashes shown are plotted with four pixels, but the diagonal dash is longer by a factor of $\sqrt{2}$. We can display approximately equal length dashes by reducing the diagonal dash to three pixels.

Line Width

Implementation of line width option depends on the capabilities of the output device to set the line width attributes.

setLinewidthScaleFactor (lw)

Line width parameter lw is assigned a positive number to indicate the relative width of line to be displayed. A value of 1 specifies a standard width line. A user could set lw to a value of 0.5 to plot a line whose width is half that of the standard line. Values greater than 1 produce lines thicker than the standard. In the Brenham algorithm a standard width line is generated with single pixel at each sample position.

- Other width lines can be plotted with additional pixels as positive integer multiples of the standard line.
- Line with slope magnitude less than 1, we modify thick line by plotting a vertical span of pixel at each x position along the line.
- The number of pixels in each span is equal to integer magnitude of parameter lw.
- We plot a double width line by generating a parallel line above the original line path as shown in Fig 7-2.
- At each sample position of x we calculate corresponding y value and plot pixel with screen coordinates (x, y) and (x, y+1).
- We display line having $lw \geq 3$ by alternately plotting pixels above and below single width line path.

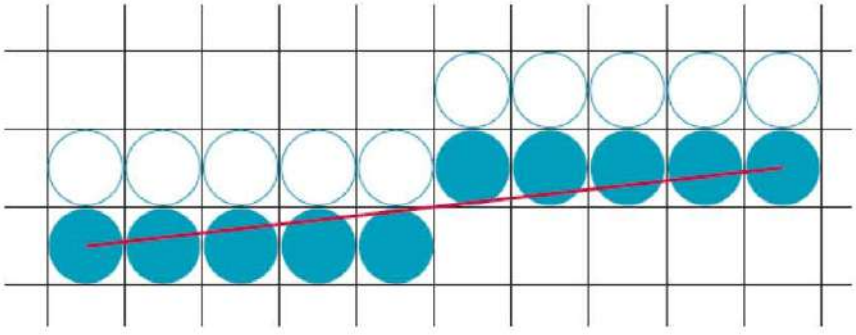


Figure .7.2: Double width line with slope |m| < 1 generated with vertical pixel span

- Now for line with slope magnitude greater than 1, we plot thick line with horizontal spans, by picking up pixels to right and left of line path i.e. shown in Fig 7-3.
- We can implement by comparing the magnitudes of the horizontal and vertical separations (Δx and Δy) of the line endpoints.
- If $|\Delta x| \wedge |\Delta y|$, pixels are replicated along columns. Otherwise, multiple pixels are plotted across rows.

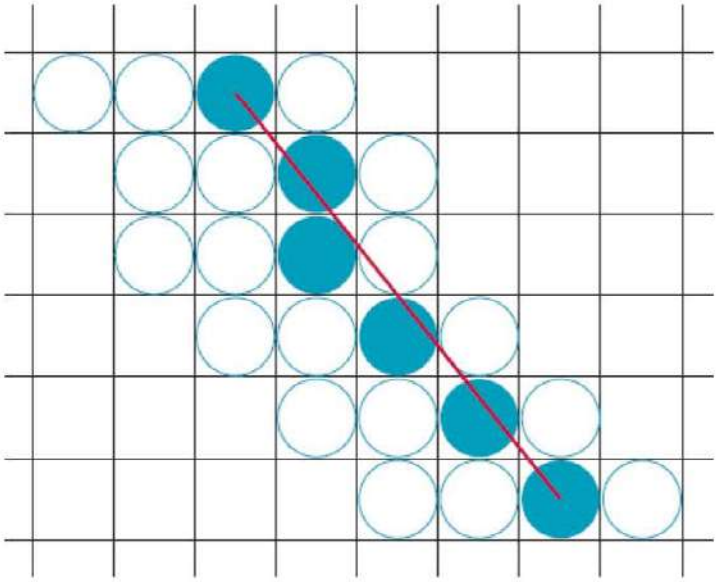


Figure.7.3: Line with slope |m| > 1 and line width parameter 1 w=4 horizontal pixel span.

Line Caps

We adjust the shape of line ends to give them a better display by adding line caps. There are three types of line caps which are as follows:

1. Butt cap
2. Round cap
3. Projecting square cap

Butt cap obtained by adjusting the end positions of the component parallel lines so that the thick line is displayed with square ends that are perpendicular to the line path.

Round cap obtained by adding a filled semicircle to each butt cap. The circular arcs are centered on the line endpoints and have a diameter equal to the line thickness.

Projecting square cap extend the line and add butt caps that are positioned one-half of the line width beyond the specified endpoints.

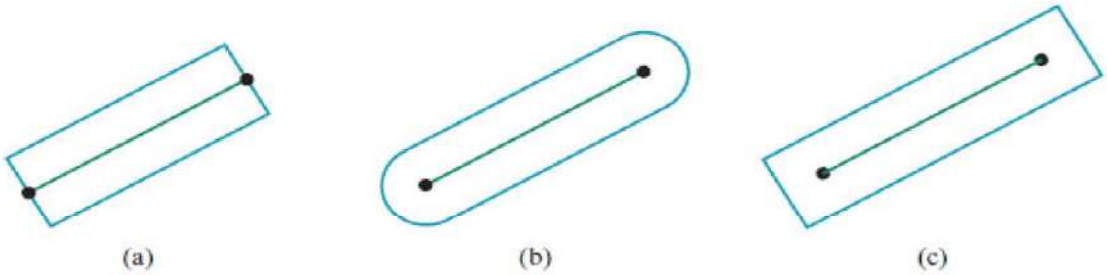


Figure. 7.4: Tick line (a) butt cap, (b) round cap, (c) projecting square cap

There are three possible methods for smoothly joining two line segments which are

1. Miter Join
2. Round Join
3. Bevel Join

A miter join designed by extending the outer boundaries of each of the two lines until they meet.

A round join is produced by capping the connection between the two segments with a circular boundary whose diameter is equal to the width.

A bevel join is generated by displaying the line segment with butt caps and filling in the triangular gap where the segments meet.

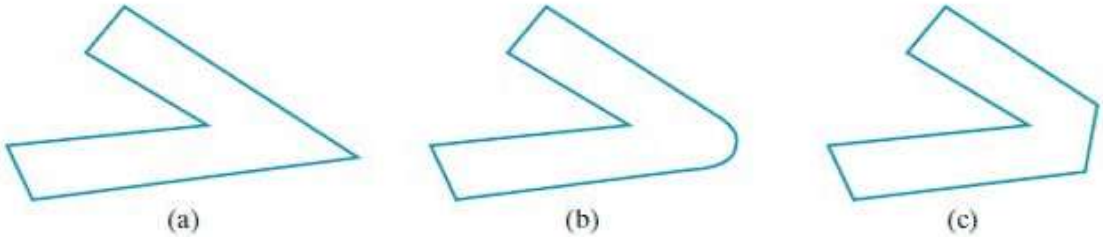


Figure.7.5: Thick line connected via (a) miter join, (b) round join, (c) bevel join

Pen and Brush Options

In painting and drawing systems, we directly select different pen and brush styles. In this category we include options shape, size and pattern for the pen or brush. Some example pen and brush shapes are shown in Fig. 7-6. These shapes can be stored in a pixel mask that identifies the array of pixel positions that are to be set along the line path.

For example, a rectangular pen can be implemented with the mask shown in Fig. 7-7 by moving the center (or one corner) of the mask along the line path, as in Fig. 7-8. To avoid setting pixels more than once in the frame buffer, we simply accumulate the horizontal spans generated at each position of the mask and keep track of the beginning and ending x positions for the spans across each scan line.

Lines generated by pen or brush shapes can be displayed in various widths by changing the size of the mask. Rectangular pen line can be narrowed with a 2 by 2 rectangular mask or widened with a 4 by4mask. Or line can be displayed with desired patterns by superimposing the pattern values onto the pen or brush mask.

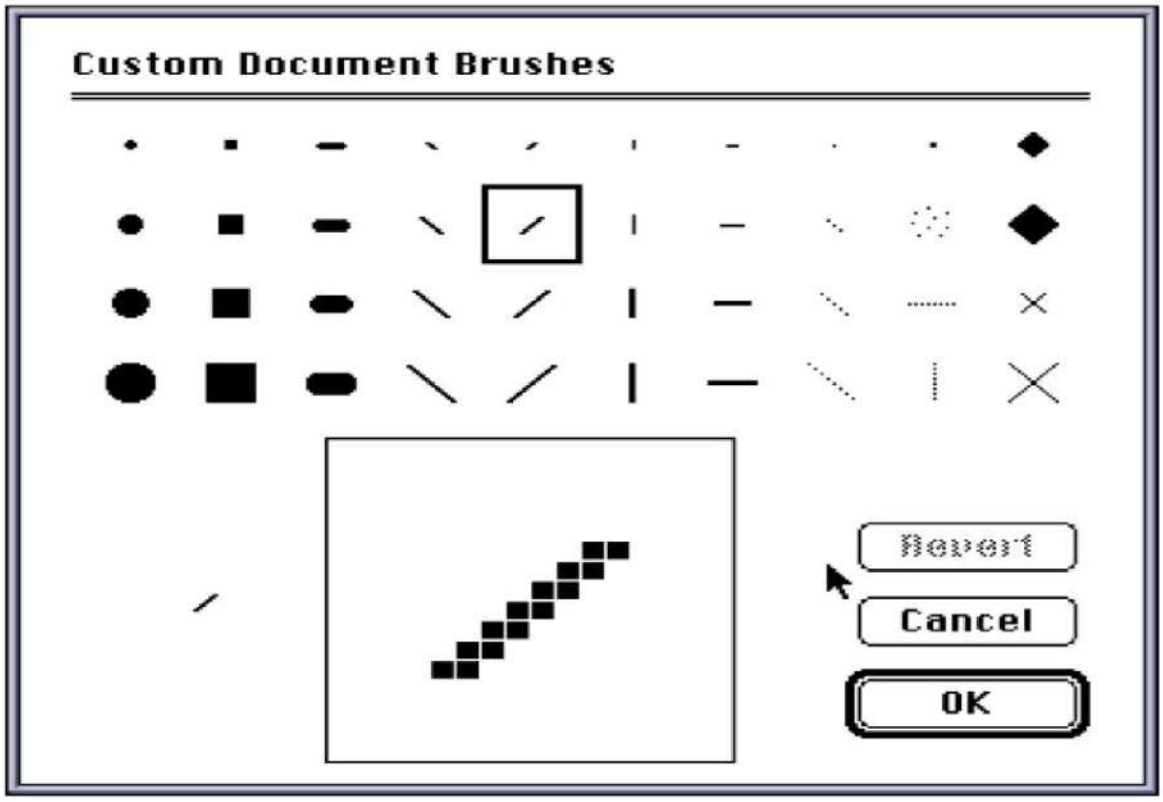


Figure.7.6: for display lines various pen and brush shapes

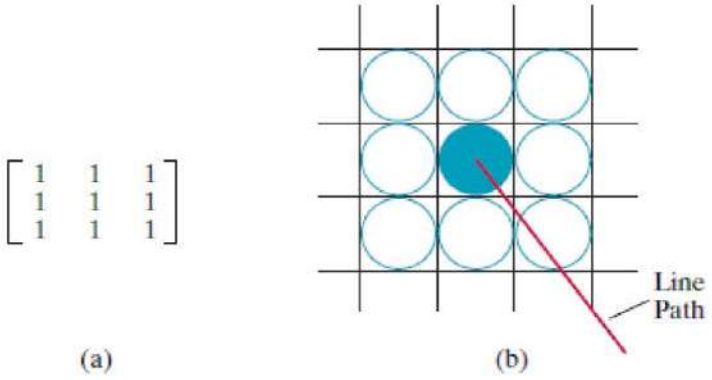


Figure.7.7: (a) Pixel mask for a rectangular pen, (b) associated array of pixels displayed by centering mask over a specified pixel position

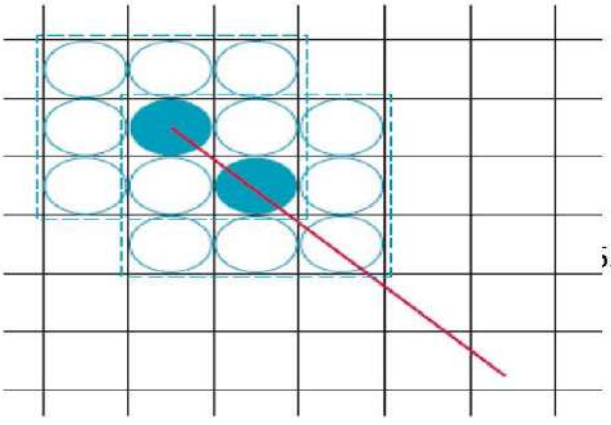


Figure.7.8: Generation of a line with the pen shapes

Line Color

A poly line routine displays a line in the current color by setting this color value in the frame buffer at pixel locations along the line path using the set pixel mechanism. We set the line color value in PHlGS with the function

setPolylineColourIndex (lc)

Nonnegative integer values, corresponding to allowed color choices, are assigned to the line color parameter lc

7.3 Curve Attributes

Parameters for curve attribute are same as those for line segments. Curves are displayed with varying colors, widths, dot - dash patterns and available pen or brush options.

First method for displaying curve with various width is **horizontal or vertical pixel spans**. Where the magnitude of the curve slope is less than or equal to 1.0, we plot vertical spans, where the slope magnitude is greater than 1.0, we plot horizontal spans. Figure 7-9 explain this method for displaying a circular arc of width 4 in the first quadrant.

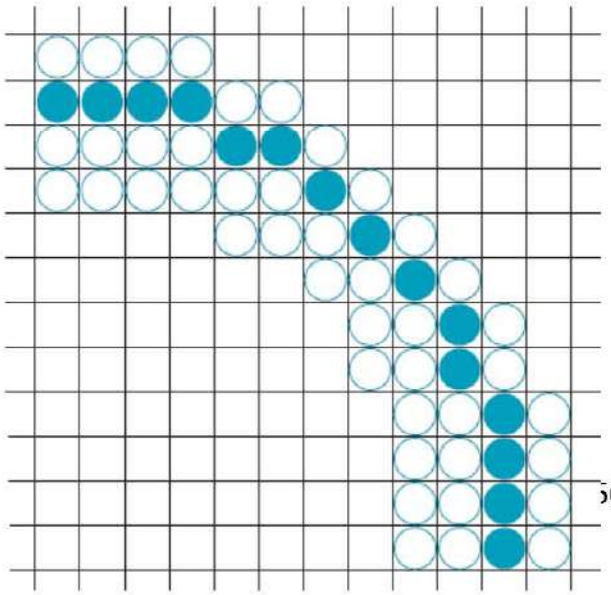


Figure.7.9: A circular arc of width 4 plotted with either vertical or horizontal pixel spans, depending on the slope.

- Using circle symmetry, we generate the circular path with vertical spans in the octant from x = 0 to x = y and then reflect pixel positions about the line y = x to obtain the remainder of the curve shown.
- Circle sections in the other quadrants are obtained by reflecting pixel positions in the first quadrant about the coordinate axes.
- The thickness of curves displayed with this method is again a function of curve slope. Circles, ellipses, and other curves will appear thinnest where the slope has a magnitude of 1.

Another method for displaying thick curves is to **fill area between two parallel curved paths**, whose separation distance is equal to the desired width.

- We specify curve path as one boundary and setting up the second boundary either inside or outside the original curve path.
- We maintain the original curve position by setting the two boundary curves at a distance of one-half the width on either side of the specified curve path.
- Figure 7-10 shown for a circle segment with radius 16 and a specified width of 4. The boundary arcs are then set at a separation distance of 2 on either side of the radius of 16.
- To maintain the proper dimensions of the circular arc, we set the radii for the concentric boundary arcs at r = 14 and r = 17. Although this method is accurate for generating thick circles.

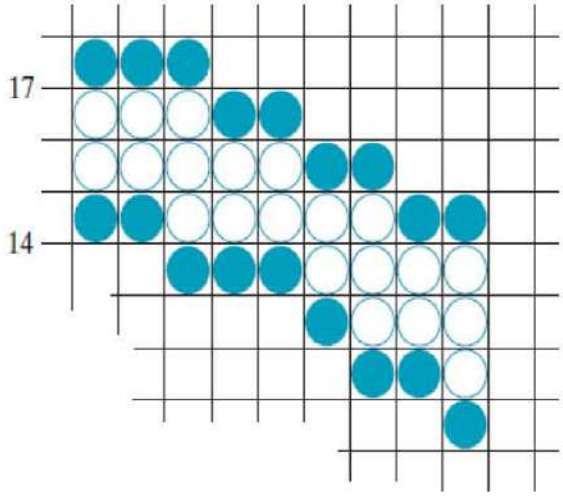


Figure.7.10: A circular arc of width 4 and radius 16 displayed by filling the Region between two concentric arcs.

We can generate dashes in the various octants using circle symmetry, but we must shift the pixel positions as we move from one octant to another with the correct sequence of dashes and spaces. If we want to display constant length dashes, we have to adjust the number of pixels plotted in each dash with equal angular arc as we move around the circle circumference. As shown in Fig 7-11.

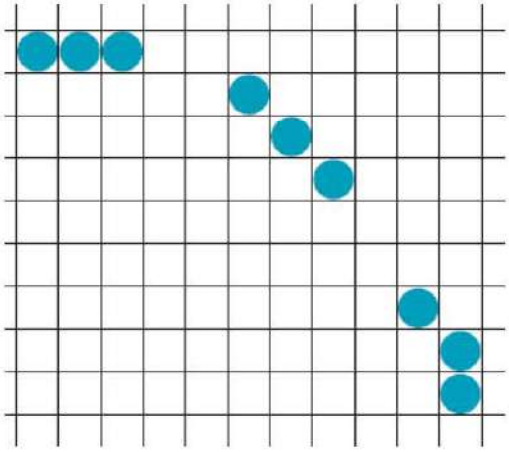


Figure.7.11: A dashed circular arc displayed with a dash span of 3 pixels and an inter-dash spacing of 2 pixels.

A circular arc can be displayed using a rectangular pen. The center of the rectangular pen is moved with successive curve positions to produce curve shape shown in Fig 7-12. Curves displayed with a rectangular pen would be thicker where the magnitude of the curve slope is 1. A uniform curve thickness can be achieved by rotating the rectangular pen to align it with the slope direction as we move around the curve.

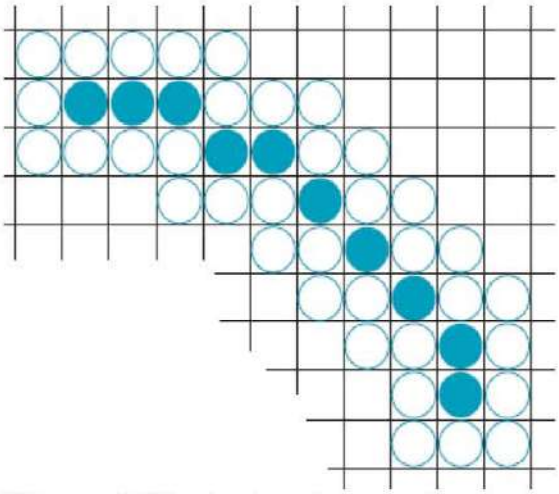


Figure.7.12: A circular arc drawn using a rectangular pen.

7.4 Character Attributes

Display of character is controlled by attributes such as font, size, color and orientation. Attributes can be set for entire character strings (text) and individual characters as well i.e. defined as marker symbols. Character attributes are classified into two categories: one is text attributes and another is marker attributes.

Text Attributes

The choice of font or type face is set of characters with a particular design style as Courier, Helvetica, Times Roman, and other symbol groups. The selected font character can be displayed with styles (solid, dotted, double) in bold face, in italics, and in outline or shadow styles.

In a PHIGS program by setting an integer code for the text font parameter tf in the function

SetTextFont (tf)

Control of text color (or intensity) is managed from an application program with

SetTextColourIndex (tc)

Where text color parameter tc specifies an allowable color code. Text size can be adjusted without changing the width to height ratio of characters with

SetCharacterHeight (ch)

Parameter ch is assigned a real value greater than 0 to set the coordinate height of capital letters. We can adjust overall dimensions of character. Character size is denoted as printer and compositors in points i.e. 1 point 0.013837" (inch). Character Height is defined as the distance between baseline and capline of the characters as shown in Fig 7-13.

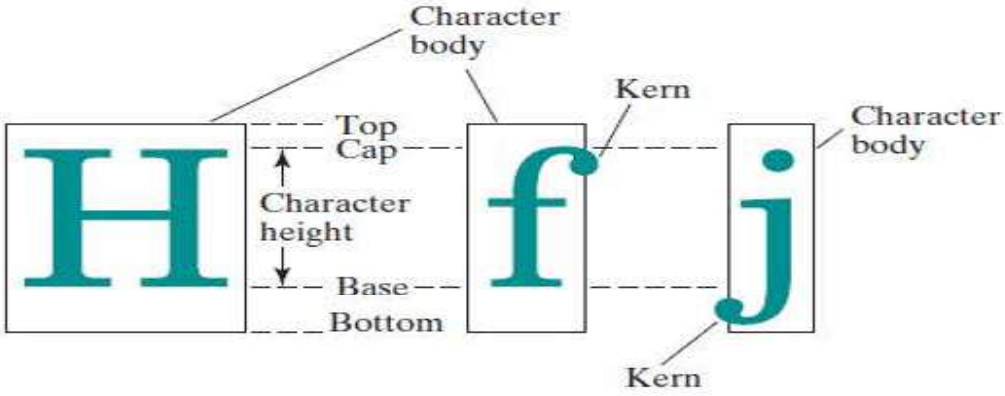


Figure 7-13 Character body

To maintain same text proportion, width and spacing of character is adjusted. The width of text can be set with function:

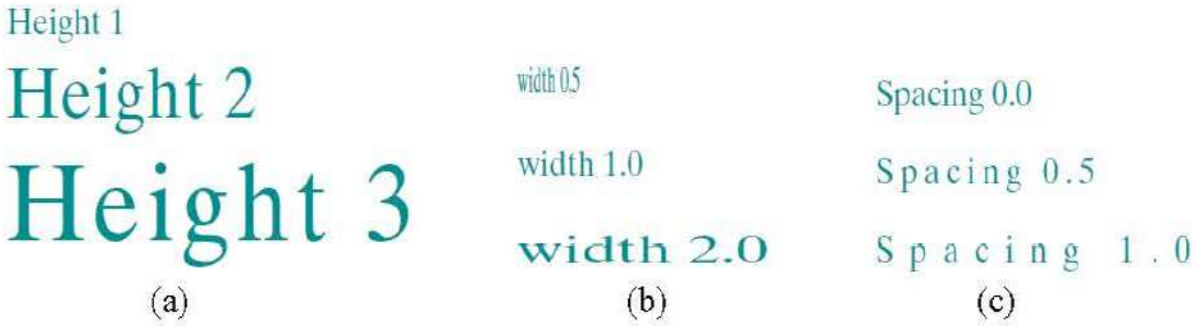
SetCharacterExpansionFactor (cw)

Where the character width parameter cw is set to a positive real value that scales the

Body width of character. Spacing between characters is controlled separately with function:

SetCharacterSpacing (cs)

Where the character-spacing parameter cs can he assigned any real value. Value of cs 0 denote no spacing between two characters. The amount for spacing is determined by multiplying the value of cs by character height.



(a) (b) (c)

Figure.7.14: (a) Text strings displayed with different ch settings and a constant width-to-height ratio. (b) Text strings displayed with varying size for cw and constant height. (c) Text strings displayed with different cs values.

The orientation for a displayed character string is set according to the direction of the character up vector

SetCharacterUpVector (upvect)

Parameter upvect in this function is assigned two values that specify the (x, y) vector components. For example, with upvect = (1, 1), the direction of the up vector is 45° and text would be displayed as shown in Figure 7-15.

To arrange character strings vertically or horizontally

SetTextPath (tp)

Where tp can be assigned the value: right, left, up, or down

Another attribute for character strings is alignment. That specifies how text is set with respect to start coordinates. Alignment attributes are set with function:

SetTextAlignment (h, v)

Where parameters h and v control horizontal and vertical alignment. The value of h may be left, center, or right. And v is set by the value of top, cap, half, base or bottom.

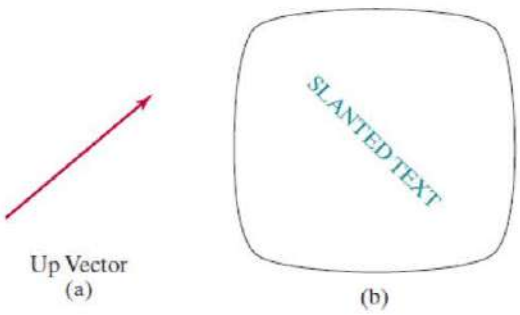


Figure.7.15: (a) direction of up vector (b) Orientation of text

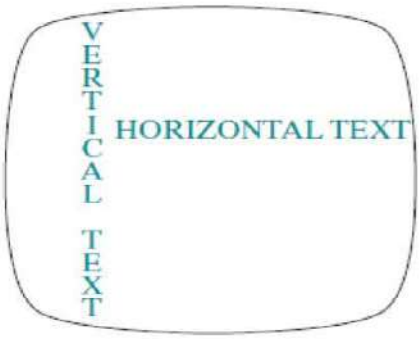


Figure.7.16: Text path is set with horizontal and vertical

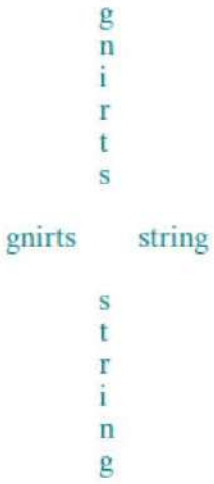


Figure.7.17: Text with four path

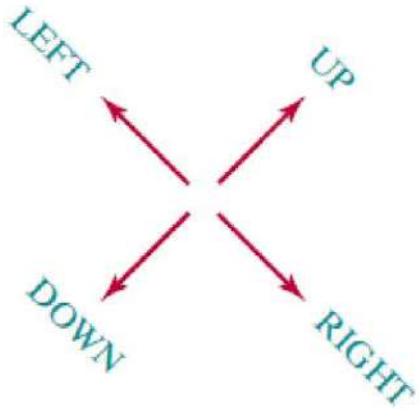


Figure.7.18: Associated direction with up Vector

Marker Attributes

A marker symbol is a single character that can he displayed in different colors and sizes. Marker attributes are implemented by selected character into the raster at the defined positions with the specified color and size.

We choose a particular character to be the marker symbol with function:

SetMarkerType (mt)

Where marker type (mt) is set to an integer code i.e. 1 through 5, specifying, respectively, a dot (.) a vertical cross (+), an asterisk (*), a circle (o), and a diagonal cross (X).

We set the marker size with function:

SetMarkerSizeScaleFactor (ms)

Parameter marker size (ms) assigned a positive number.

This scaling parameter is applied to the nominal size for the particular marker symbol chosen. Values greater than 1 produce character enlargement. Values less than 1 reduce the marker size.

Marker color is specified with function:

SetPolymarkerColourIndex (mc)

A selected color code parameter mc is stored in the current attribute list and used to display subsequently specified marker primitives.

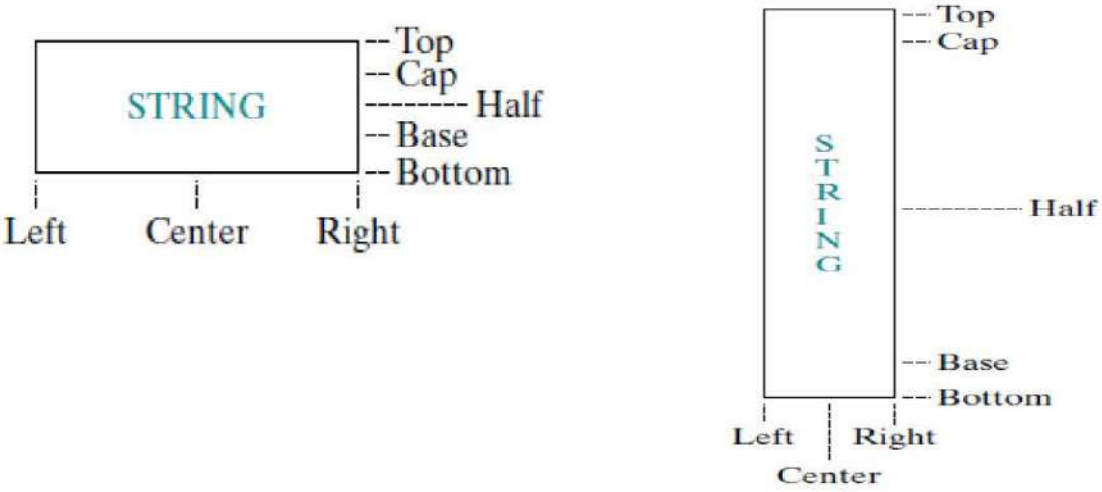


Figure.7.19: Character alignment for horizontal and vertical string.

7.5 Antialiasing

The sampling process digitizes coordinate points on an object to discrete integer pixel positions. This distortion of information due to low-frequency sampling (undersampling) is called aliasing. We can improve the appearance of displayed raster lines by applying anti-aliasing methods that compensate for the undersampling process.

To avoid losing information from periodic objects, we need to set the sampling frequency to at least twice that of the highest frequency occurring in the object, referred as the Nyquist sampling frequency f_s:

$$f_s = 2f_{max} \qquad \text{.....7.1}$$

In other words, sampling interval should be no larger than one-half the cycle interval (called the Nyquist sampling interval). For x-interval sampling, the Nyquist sampling interval Δx_s is

$$\Delta x_s = \frac{\Delta x_{cycle}}{2} \qquad \text{.....7.2}$$

Where $\Delta x_{cycle} = 1/f_{max}$.

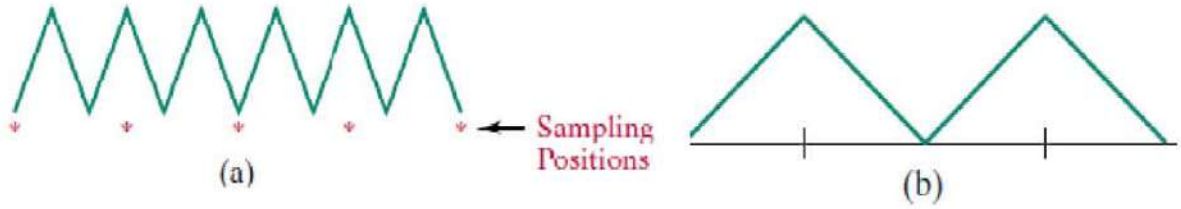


Figure.7.20: sampling (a) periodic shape (b) Aliased lower frequency display

In raster system, one way to increase sampling rate is display object at high resolution. To achieve this, we have to limit frame buffer and maintain refresh rate at 30 to 60 frames per second. But screen resolution is not a complete solution to aliasing problem. We use anti-aliasing method to modify pixel intensities. There are three methods for anti-aliasing for pixel intensities. Supersampling, area sampling and pixel phasing.

Supersampling: A straight forward antialiasing method is to increase sampling rate by treating the screen covered with a finer grid. We can then use multiple sample points across this finer grid to determine an appropriate intensity level for each screen pixel. This technique of sampling object characteristics at a high resolution and displaying the results at a lower resolution is called supersampling (or postfiltering).

Area sampling: to determine pixel intensity by calculating the areas of overlap of each pixel with the objects to be displayed. Antialiasing by overlap areas is referred to as area sampling (prefiltering).

Pixel phasing: in pixel phasing method antialiasing is done by shifting the display location of pixel area.

Supersampling straight line segment

- For a straight-line segment in supersampling, we divide each pixel into a number of subpixels and count the number of subpixels that overlap the line path.
- The intensity level for each pixel is set to a value that is proportional to this subpixel count.
- In fig 7-21 each square pixel area is divided into nine equal-sized square subpixels, and the shaded subpixels would be selected by Bresenham's algorithm. It show three intensity settings above zero, since the maximum number of subpixels that can be selected within any pixel is three.
- For this example, the pixel at position (10, 20) is set to the maximum intensity (level 3), pixels at (11, 21) and (12, 21) are each set to the next highest intensity (level 2), and pixels at (11, 20) and (12, 22) are each set to the lowest intensity above zero (level 1). It display blurred line.

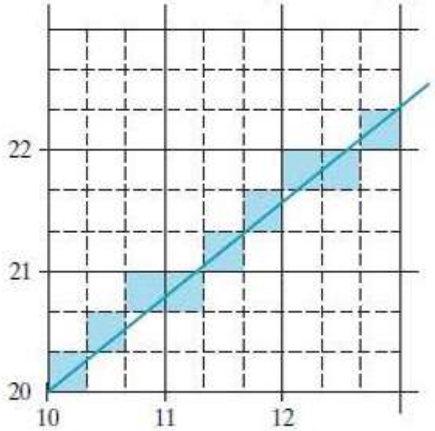


Figure.7.21: Supersampling of a straight line whose left end point is at screen coordinates (10, 20).

- For finite size of pixel area, advantage of this supersampling procedure is that the number of possible intensity levels for each pixel is equal to the total number of subpixels within the pixel area as shown in Fig 7-22.
- If we have a color display, we extend the method to take background colors into account. If five subpixels within a particular pixel area are determined to be inside the boundaries for a red line and the remaining four subpixels fall within a blue background area, we can calculate the color for this pixel as:

$$\text{pixel}_{\text{color}} = \frac{(5 \cdot \text{red} + 4 \cdot \text{blue})}{9}$$

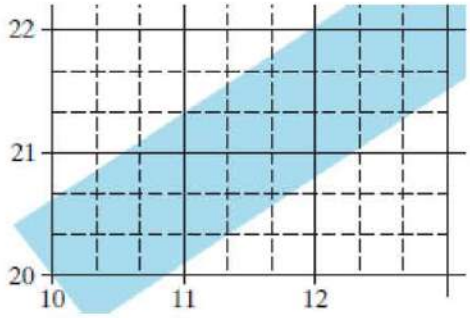


Figure.7.22: Supersampling subpixel positions in relation to the interior of a line of finite width.

Area sampling straight line segment

- We denote area sampling for a straight line by setting pixel intensity proportional to the area of overlap of the pixel with the finite-width line.
- The line is treated as a rectangle, and the section of the line area between two adjacent vertical (or two adjacent horizontal) screen grid lines is then a trapezoid.
- Overlap areas for pixels are calculated by determining how much of the trapezoid overlaps each pixel in that column or row. As shown in Fig. 7-22, the pixel with screen grid coordinates (10, 20) is about 90 percent covered by the line area, so its intensity is set to 90% of the maximum intensity.

Filtering

It is a more accurate method for antialiasing of a line. The method shows imagine a continuous weighting surface (or filter function) covering the pixel. Examples of rectangular, conical, and Gaussian filter functions.

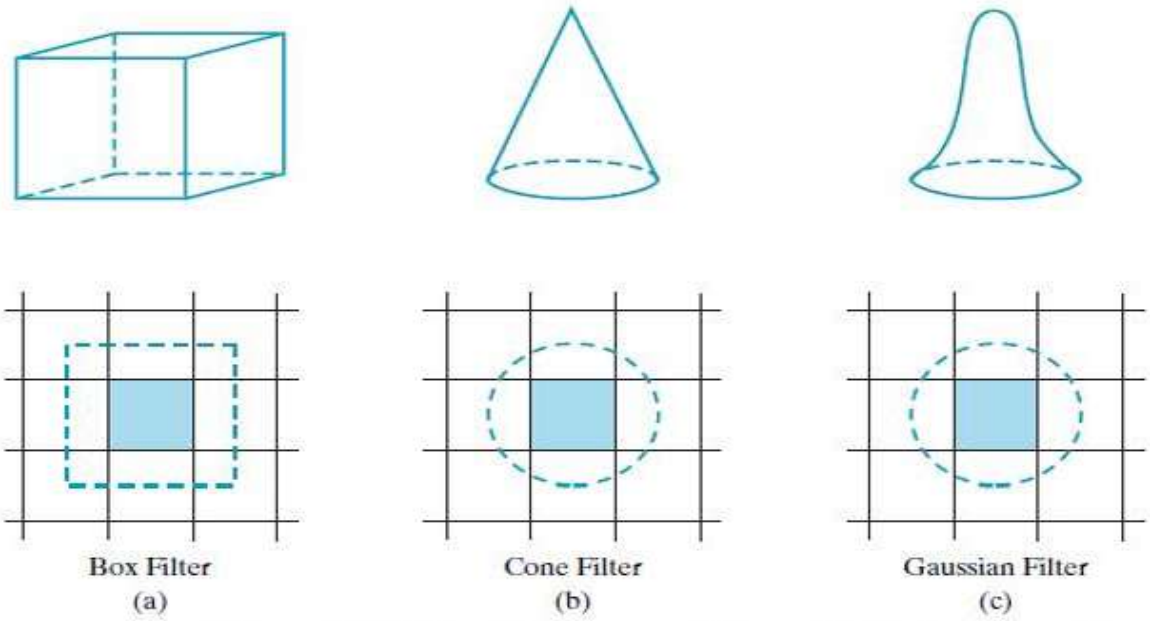


Figure.7.23: filter functions used for anti-aliasing line paths.

Pixel phasing

Pixel phasing can be used to antialias objects. A line display is smoothed with this technique by moving (micro positioning) pixel positions closer to the line path. In the system for pixel phasing the electron beam is shifted by a fraction of a pixel diameter i.e. typically shifted by 14, 12 or 34 of a pixel diameter to plot points closer to the true path of a line or object edge.

7.6 Self Learning Exercise

Q.1. What do you mean by line cap in line attributes? Explain its types.

Q.2. What are the possible methods for smoothly joining two line segments?

Q.3. Explain supersampling, area sampling and pixel phasing in antialiasing.

7.7 Summary

We have discussed various attributes of output primitives. Line attributes are color, width, and style. Specifications for line width are given in terms of multiples of a standard, one-pixel-wide line. The line-style attributes include solid,

dashed, and dotted lines, as well as various brush or pen styles. These attributes are common for line and curve as well. Character attributes are text and marker. That shows different types of functions like spacing, width, height, color, position of text etc. Antialiasing for a straight line can be achieved using supersampling, area sampling and pixel phasing.

7.8 Glossary

PHIGS: It's Programmer Hierarchical Interactive Graphics System, that allow the user to display and interact with 2-D and 3-D graphics. It is an international standard.

7.9 Answers to Self-Learning Exercise

Ans.1, Ans.2 and **Ans.3** were discussed in chapter in details. See respective contents.

7.10 Exercise

Q.1. Polyline is assigned a table index as 3 would be displayed using

(A) Dotted line (B) Dashed line

(C) Same index (D) All of these

Q.2. A dashed line could be displayed by generating

(A) Inter dash spacing (B) Very short dashes

(C) Both A and B (D) None of these

Q.3. Pixel mask means

(A) A string containing only 1's (B) A string containing only 0's

(C) A string containing 1 and 0 (D) A string containing 0 and 0

Q.4. The algorithm which displays line type attributes by plotting pixel span is

(A) Raster line algorithm

(B) Raster scan algorithm

(C) Random line algorithm

(D) Random scan algorithm

Q.5. If the angle between 2 connected line segments is very small then which join generate a long spike that distorts the appearance of the polyline.

(A) Miter join (B) Round join

(C) Bevel join (D) None of these

Q.6. Explain the curve attributes as output primitives.

Q.7. Explain text attributes of character in detail.

7.11 Answers to Exercise

Ans.1: B **Ans.2:** A **Ans.3:** C

References and Suggested Readings

1. J. Foley, A. Van Dam, S. Feiner, J. Hughes: Computer Graphics- Principles and Practice, Pearson.
2. Hearn and Baker: Computer Graphics, PHI.
3. Additional programming examples and information on PHIGS primitive can be found in Howard, et al. 1991.

UNIT-08
Curves and Surfaces

Structure of the Unit

8.0 Objective

In this chapter, we shall focus on the following topics

- Spline Representation
- Cubic Spline
- Beizer Curves
- B-Spline Curves
- Quadric Surfaces
- Beizer Surfaces

8.1 Introduction

There is verity of techniques are available for drawing and designing curves manually, like pencils, pens, brushes, etc. each tool has its function and use. No single tool is sufficient for all tasks. Similarly, a verity of techniques and tools are useful for curve design and generation in computer science. A curve is 2-Dimensional if it lies in its entirety in a single plane.

A graphics system typically uses a set of primitives or geometric forms that are simple enough to be efficiently, implemented on the computer but flexible enough to be easily manipulated to represent or model a variety of objects. Curve and surfaces equations can be expressed in either a parametric or non-parametric

8.2 Curves

The intersection of 2 surfaces in 3D gives a ***curve***. A curve may be represented as a collection of points. Provided the points are properly spaced, connection of points by short straight line segment yields an adequate visual representation of the curve. Spline representations are examples of this class of

curve and surfaces. These methods are used to design new objects shapes, to digitize drawing and to describe animation paths. Curve and surfaces equations can be expressed in entire a parametric or a non-parametric equation.

Representation of space curves

3-D space curve are represented in explicit non parametric form as

$x=x;\ y=f(x);\ z=g(x)$

Non-parametric implicit representation of the curve as the intersection of 2 surfaces is given by:

$f(x,y,x)=0$

$g(x,y,z)=0$

A parametric space curve is expressed as

$x=x(t) : y=y(t);\ z=z(t)$

Where parameter t varies over the range $t1 \leq t \leq t2$

8.3 Spline Representation

In computer graphics, a spline is a curve that connects two or more specific points, or that is defined by two or more points. A physical spline is a long, narrow strip of wood or plastic used to fit curves through specified data points. The splines are shaped by lead weights called ducks.

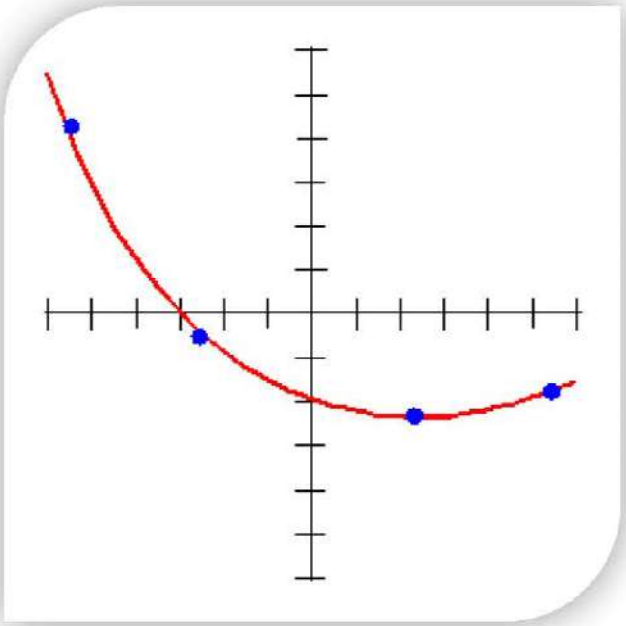


Figure. 8.1: Spline Representations

We specify spline curve by set of coordinate positions called ***control points***. These control points are fitted in two ways. When polynomial sections are fitted so that curve passes through each control point, curve is said to interpolate the set of control points and method is called ***curve fitting*** method whereas, if polynomials are fitted to control point path without necessarily passing through each control point, then curve is said to approximate the set of control points and technique is called as ***curve fairing*** technique.

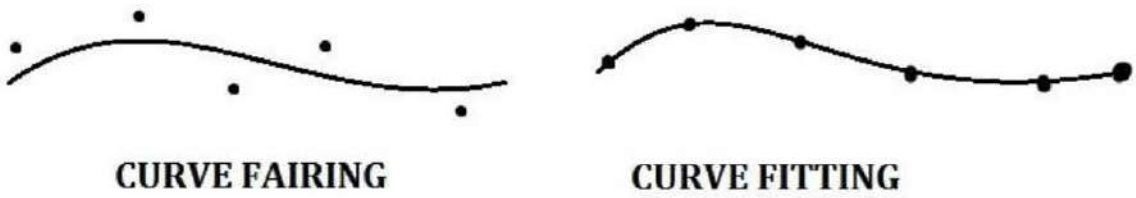


Figure.8.2: Curve fairing and Curve fitting

In computer graphics, the term spline curve refers to any composite curve formed with polynomial sections satisfying specified continuity conditions (parametric continuity and geometric continuity conditions) at the boundary of the pieces, without fulfilling these conditions no two curves can be joined smoothly.

Each sections of spline can be described with parametric coordinate functions of the form

$x=x(t) : y=y(t); z=z(t)$

Where parameter t varies over the range $t1 \leq t \leq t2$

There can be zero order parametric continuity, first-order parametric continuity, second order parametric continuity and so on. Parametric derivatives of the adjoining sections of the curve are matched at their boundary to find out parametric continuity.

Zero order parametric continuity (C^0) simply means that x, y, z values at the boundary of the two adjoining sections are same.

First-order parametric continuity (C^1) means the first derivatives of parametric coordinated functions is same at the boundary of two adjoining sections of the curve. It cannot be used for animation applications.

Second order parameter continuity (C^2) ensures that both first and second derivatives of the two adjoining curve sections are same. Second order parameter continuity is useful for animation applications.

8.4 Cubic Spline

Cubic spline interpolation is a special case for Spline interpolation. It is an interpolating piecewise cubic polynomial. This class of spline is most often used to set up paths for objects motions or to provides a representation for existing object or drawing, but interpolation spline are sometimes used to design objects shapes. Cubic polynomial offers a reasonable compromise between flexibility and speed of computation. Compared to higher order polynomials, cubic splines require less calculations and memory and they are more stable. Compared to lower order polynomials, cubic splines are more flexible for modeling arbitrary curve shapes.

Mathematically, spline is a piecewise polynomial of degree K with continuity of derivatives of order K-1 at the common joints between segments. Accordingly, cubic spline has a second order or C^2 continuity at the joints.

The equation for a single parametric cubic spline segment is given by:-

$$P(t) = \sum_{i=1}^{4} B_i \, t^{i-1} \qquad t_1 \leq t \leq t_2$$

Where B_i is the polynomial blending function. Blending function are called so because they blend the boundary constraint values to obtain each coordinate position along curve t_1 and t_2 are the parameter values at the beginning and end of the segment. *P(t)* is the position vector of any point on the cubic spline segment.

8.5 Beizer Curves

A Bezier curve is a mathematically defined curve used in two-dimensional graphic applications. The curve is defined by four points: the initial position and the terminating position (which are called "anchors") and two separate middle points (which are called "handles"). The shape of a Bezier curve can be altered by moving the handles. Bezier curve was discovered by the French engineer Pierre Bezier. These curves can be generated under the control of other points. Cubic splines curves pass through existing data points, they are curve fitting techniques. Examples for which are aircraft wings, mechanical and structural parts. Other

techniques in which curves do not necessarily pass through each data or control points called curve fairing techniques can be used for other class of problems. It helps in the manufacture of skin of car bodies, aircraft fuselages, ship hulls, furniture and glassware.

As it is considered for cubic spline fitting method, obvious relationship between number of control points and the curve shape does not always exist.

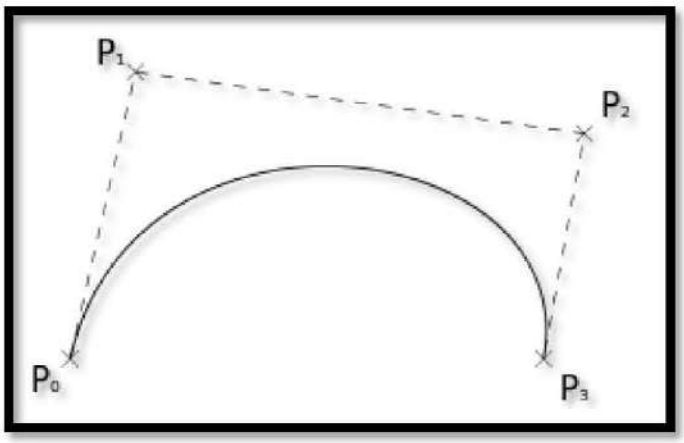


Figure. 8.3: Beizer Curve

Beizer curve is determined by defining a polygon.

Beizer curves have following characteristics:

- The basic functions or Beizer or blending functions are real.
- Degree of polynomial defining the curve segment is one less than the number of control points.
- Curve generally follows the shapes of the defining polygon.
- First and last points on the curve coincide with the and last points of the defining polygon.

- Tangent vectors at the ends of the curve have the same directions as the first and last polygon spans
- Curve is contained within the convex hull of the defining polygon.
- Curve is invariant under an affine transformation.

Mathematically, parametric Beizer curve is defined by:-

$$P(t) = \sum_{i=0}^{n} B_i \, J_{n,i} \qquad 0 \leq t \leq 1$$

Where blending or basis function is given by

$$J_{n,i}(t) = \frac{n!}{i!\,(n-i)!} t^i (1-r)^{n-i}$$

In this equation $j_{n,i}(t)$ is the i^{th} n^{th}- order blending function and n is the degree of the blending function. The value of 'n' is one less than the number of control points in the defining Beizer polygon.

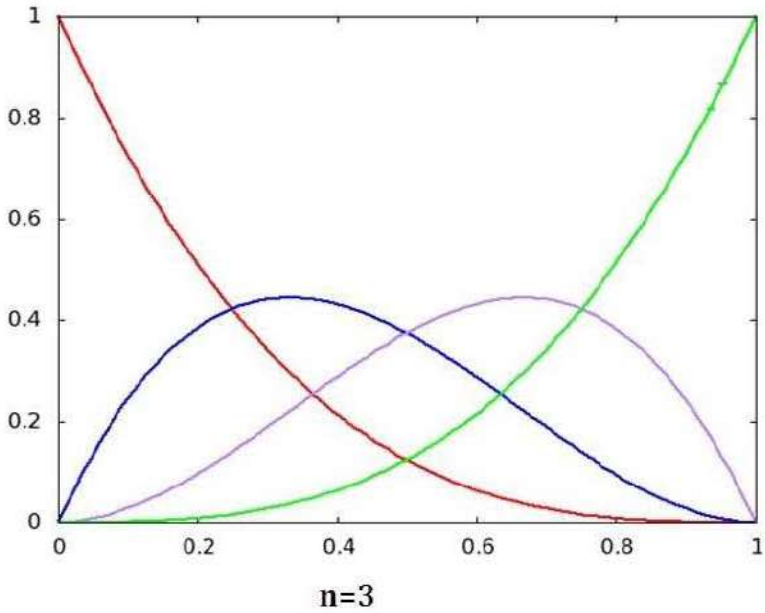


Figure. 8.4: Beizer blending function for n=3

Four blending functions are shown in FIGURE 8.4 for n=3. All the four are cubic blending functions. Maximum value of blending functions is at ***t=i/n***

8.6 B-Spline Curves

These are the most used class of approximating splines. This method is better than Beizer spline curves method because of two characteristics of Beizer method which limits the flexibility of the resulting curves. First is, number of polygon vertices fixes the order of the resulting polynomial which defines the curve. Example cubic curve is defined by four vertices of the polygon. To reduce the degree of the curve, we have to reduce the number of vertices. Second limiting characteristic says that Beizer curve is the result of blending the values of all defining vertices. A change in one vertex is felt in the entire curve. This global nature of Beizer curves eliminating the ability to produces local change within the curve.

B-Spline basis is generally non-global. This behavior of B-Spline curve is because each vertex Bi is associated with a unique basis function. Each vertex affects the shape of the curve locally i.e. over a range of parameter values where it's associated basis without changing the number of defining polygon vertices.
A B-Spline curve is given by:-

$$\boldsymbol{P(t) = \sum_{i=1}^{n+1} B_i \; N_{i,K}(t) \qquad t_{min} \leq t \leq t_{max}, 2 \leq K \leq n+1}$$

P(t) is the position vector along the curve, B_i are the position vectors of n+1 defining polygon vertices and $N_{i,k}$ are normalized B-Spline basis functions.

SO to define B-spline curve, it is defined as a polynomial spline function of order K (degree K-1) and it should satisfy two conditions:-

- *Function P(t) is a polynomial of degree K-1 on each interval* $x_i \leq t \leq x_{i+1}$
- *P(t)* and its derivatives of order 1,2,……*K-2* are all continuous over the entire curve.

Properties of B-Spline curves:-

- Sum of B-spline basis functions for any parameter value t can be shown as:

$$\sum_{i=1}^{n+1} N_{i,K}\ (t) = 1$$

  Each basis function or positive or zero for all parameter values i.e. $N_{i,k} \geq \mathbf{0}$

- Maximum order of curve of the curve is equal to the number of defining polygon vertices.
- Curve generally follows the shape of the defining polygon.
- Any transformations can be applied to the curve by transformations the defining polygon vertices.
- Curve lies within the convex hull of its defining polygon

8.7 Surfaces

Surfaces and their description play a critical role in design and manufacturing, examples include automobiles bodies, ships hulls, tribunes, compressor, glassware, bottles, furniture etc. A surface can be represented using multiple orthogonal

projections. The surface is defined by a net or mesh of orthogonal plane curve lying in plane sections plus multiple orthogonal projections of certain 3-Dimensional feature lines.

In computer graphics, it is advantages to develop a true three dimensional model of surface. Surface characteristics can be easily analyzed like curvature, volume, surface area etc. visual rendering of the surface is simplified. Generation of necessary information required to fabricate the surface e.g. numerical control codes is simplified as compared to a traditional net of lines approach.

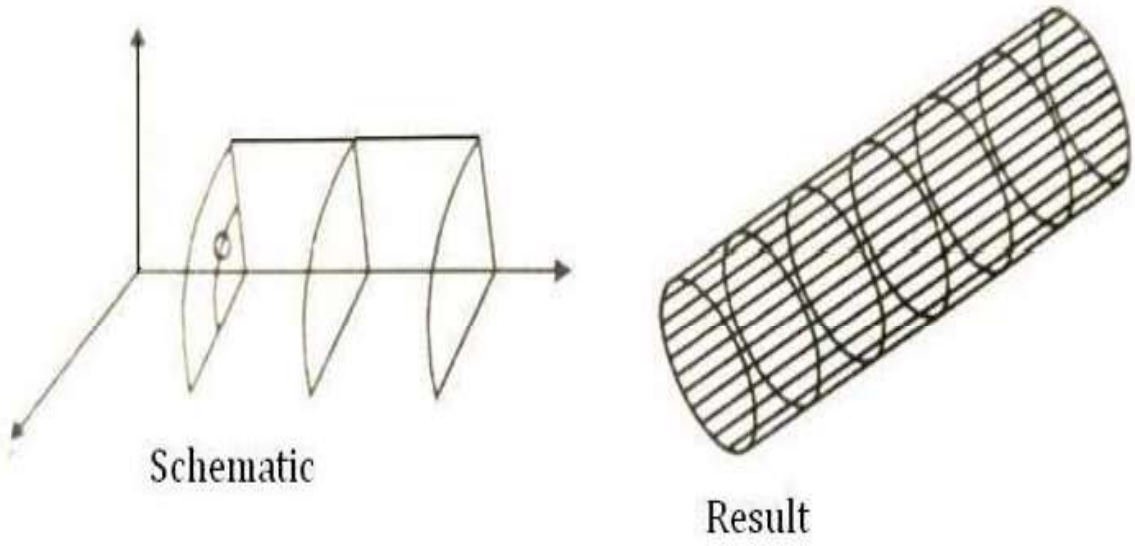


Figure. 8.5: Cylindrical surface of revolution

Quadric Surfaces:-

Quadric surfaces are described with second-degree equations. It includes spheres, ellipsoids etc. A spherical surface with radius r centered in the coordinate origin is defined as the set of points (x,y,z) represented by the equation

$$x^2 + y^2 + z^2 = r^2$$

Sweep Surfaces:-

A 3-D surface is also obtained by traversing an entity e.g. a line, polygon or curve, along a path in space. The resulting surfaces are called sweep surfaces. The simplest sweep entity is a point. Though, sweeping a point-along its path gives curve and not surface.

Sweep representations are useful for constructing three-dimensional objects that process translational, rotational or other symmetries.

Beizer Surfaces:-

Two sets of Beizer curve can be used to design an object surface specified by an input mesh of control points. The Beizer surface is formed as the Cartesian product of the blending functions of two Beizer curves.

8.8 Self Learning Exercise

Q.1 _____curve is one of the sp line approximation methods

(A) Bezier
(B) Ellipsoid
(C) Shearing
(D) None of these

Q.2 A Bezier curve is a polynomial of degree _____the no of control points used

(A) One more than
(B) One less than
(C) Two less than
(D) None of these

Q.3 What is the difference between interpolation splines and approximation splines.

8.9 Summary

In this unit, we have seen a spline that is a flexible strip used to produce a smooth curve through a designed set of points. Beizer spline curve are widely used CAD systems, in graphics packages, drawing, etc. Beizer curve ensures that the polynomial smoothly follows the control points without erratic oscillations. Spline is used graphics applications to design curve and surfaces shapes to digitize drawings for computer storage, and to specify animation paths for the objects or the camera in a scene.

8.10 Glossary

Control Points: A Bezier curve is defined in terms of a number of control points.

8.11 Answers to Self-Learning Exercise

Ans.1: A **Ans.2:** B

8.12 Exercise

Q.1 Bezier spline always passes through

(A) First and second control point

(B) Does not pass from First and second control point

(C) Both a & b

(D) None of these

Q.2 The object refers to the 3D representation through linear, circular or some other representation are called

(A) Quadric surface

(B) Sweep representation

(C) Torus

(D) None of these

Q3. The Bezier curve obtained from the four control points is called a

(A) Square Bezier curve

(B) Cubic Bezier curve

(C) Hectare Bezier curve

(D) Rectangle Bezier curve

Q4. The shape of a Bezier curve primarily depends upon the

(A) Position of control points

(B) Distance of control points

(C) Position of control panel

(D) None of these

Q5. Given $B_0[1,1]$, $B_1[2,3]$, $B_2[4,3]$ and $B_3[3,1]$ and the vertices of Beizer polygon, determine seven points on the Beizer curve.

8.13 Answers to Exercise

Ans.1: A **Ans.2:** B

Ans.3: B **Ans.4:** A

References and Suggested Readings

1. Computer Graphics by Donald Hearn and M. Pauline Baker; Pearson Education, Seventh Edition 2005
2. Computer Graphics by Apurva A. Desai;PHI Learning, Third Edition 2012

3. Computer Graphics Principles & practice in C by James D. Foley, Steven K. Feiner, Andries van Dam and F. Hughes John; Pearson Education, Second Edition 2003
4. Computer Graphics A Programming Approach by Harrington and Steven; McGraw Hill.

UNIT-9
Geometric 2D Transformation

Structure of the Unit

9.0 Objective

In this chapter, we shall focus on the following topics

- Basic Transformation
  - Scaling
  - Translation
  - Rotation
- Homogeneous Coordinates
- Other Transformation
  - Reflection
  - Shearing
- Composite Transformation

9.1 Introduction

We can create many two dimensional objects and images using lines, curves, polygons, etc. In most applications, there is also a need for altering or manipulating displays or images. In animations, pictures are produced by moving the camera or the objects in a scene along animation paths. To add reality or feel like reality, many times these objects are needed to be moved, rotated or even in some applications, enlarged or reduced. This change in accomplished by geometric transformation or object transformations. Here we will discuss the basic transformations such as translation, rotation, scaling and other transformations which uses the concept of this basic transformations.

9.2 Basic 2-D Transformation

Transformation means changing some graphics into something else by applying rules. There are three basic general procedures for applying movements, moving, scaling and rotating any object on a plane. When a transformation takes place on a 2D plane, it is called 2D transformation. These movements are

performed through some basic geometry. So, there are three basic transformations, translation, scaling, and rotation.

Let us discuss all these transformations one by one in detail.

Translation

A common requirement is to relocate a picture or an object to a new position along a straight line path from one coordinate location to another. Translation of an object is possible in any one of the following three directions;

1. Translation of an object parallel to X-axis, i.e. horizontal direction.
2. Translation of an object parallel to Y-axis, i.e. vertical direction.
3. Translation of an object in any direction either vertical or horizontal.

We translate a 2D point by adding translation distances, t_x and t_y, to the original coordinate position (x, y) to get new position(x', y') as shown in Figure 09.1,

$$x' = x + t_x, \qquad y' = y + t_y \tag{9.1}$$

The translation distance pair (t_x, t_y) used in equation 9.1is called a translation vector or shift vector.

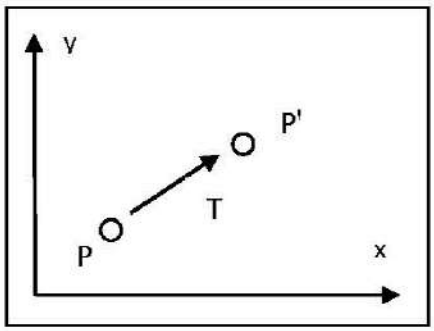


Figure.9.1: Translating a point from position P to new position P' with translation vector T.

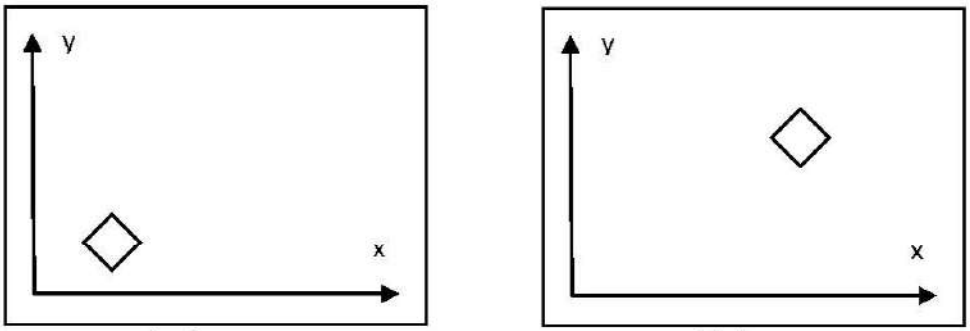
The translation equation 9.1 can be expressed as a single matrix equation by using column vectors to represent coordinate positions and the translation vector.

$$P = \begin{bmatrix} x \\ y \end{bmatrix} \qquad P' = \begin{bmatrix} x' \\ y' \end{bmatrix} \qquad T = \begin{bmatrix} t_x \\ t_y \end{bmatrix} \qquad (9.2)$$

Now we can write the two-dimensional translation equations in the matrix form;

$$P' = T + P \qquad (9.3)$$

The translation is a rigid body transformation. A transformation is called rigid body transformation if the Euclidean distance between any two coordinates remains unchanged by the transformation. That is, every point on the object is moved by the same amount. Figure 9.2 illustrates the generalized translation of an object.



(a) (b)

Figure.9.2: Translation a polygon from position (a) to position (b) in both X and Y direction.

Scaling

Scaling transformation is used to alter the size of an object. The alteration of size of the object is defined by the scaling factor. This operation can be carried out for polygons by multiplying he coordinate values (x, y) of each vertex by scaling factor S_x and S_y to produce the transformed coordinates (x', y') as shown in Figure 9.3.

$$x' = x \cdot S_x, \qquad y' = y \cdot S_y \qquad (9.4)$$

Scaling factor S_x scales objects in the x direction, where S_y scales in the y direction. The matrix representation of equation 9.4 is as follow:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} \tag{9.5}$$

Which is equivalent to,

$$P' = S \cdot P \tag{9.6}$$

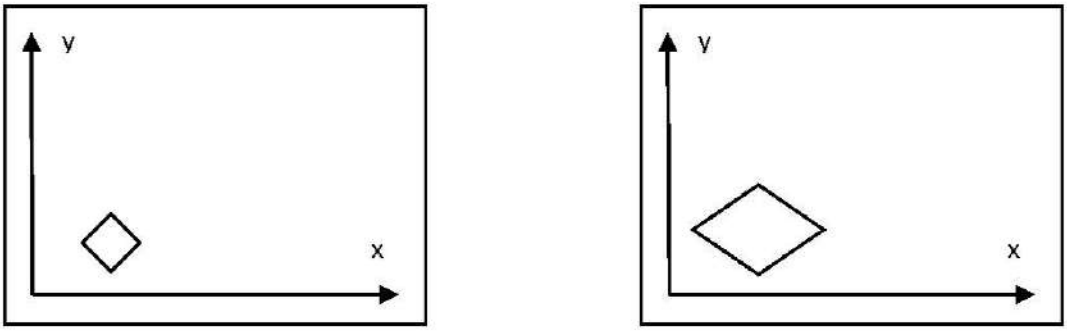


Figure.9.3: Scaling Transformation

S_x and S_y are scaling factor they are always positive. Any positive number can be assigned to the scaling factors. A value greater than 1 will enlarge the object in horizontal or vertical directions respectively whereas their values between 0 and 1 will reduce the object in the respective directions. When both the scaling factors have the same value, a uniform scaling transformation is performed otherwise it is called differential scaling transformations as shown in Figure 9.4.

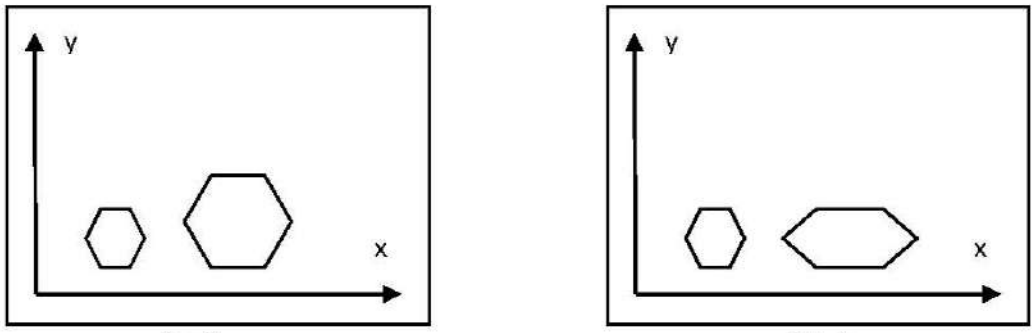


Figure.9.4: (a) Uniform Scaling (b) Differential scaling

Object transformed by equation 9.5 are both scaled and repositioned. When the scaling factor is greater than one, then the object will move away from the origin, but when the scaling factor is less than one, then the object will move near to the origin. That is, the object changes its position when we apply scaling transformation. Almost all the graphical application need to keep the object at its original position or at some fixed position this is known as **fixed point scaling.**

We achieve fixed point scaling at a fixed point (X_f, Y_f) by the following set of equations.

$$x' = x \cdot S_x + (X_f - X_f S_x)$$
$$y' = y \cdot S_{y+} (Y_f - Y_f S_y) \qquad (9.7)$$

Or

$$x' = x \cdot S_x + X_f (1 - S_x)$$
$$y' = y \cdot S_y + Y_f (1 - S_y) \qquad (9.8)$$

Rotation

Another common type of transformation is rotation. This is used to orientate objects. This transformation moves an object on a circular path about an origin. This transformation requires **rotation angle α** and the position (X_r, Y_r) of the **rotation point** or known as **pivot point** about which the object is to be rotated.

Any object can be rotated either in a clockwise direction or anti-clockwise direction by a given value of angle **α.** A positive value of angle define counterclockwise rotation, and negative value rotates the object in clockwise.

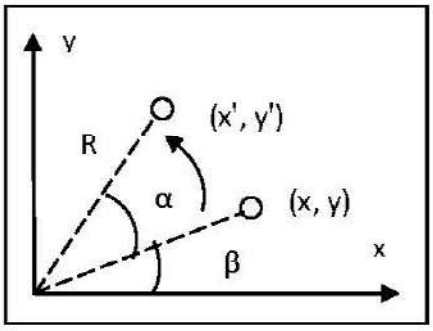


Figure.9.5: Rotating a point about the origin

The rotation of one point in the object is illustrated in Figure 9.5. A line joining the point with the origin makes an angle b with the x-axis and has length R, hence.

$$x = R.\cos\beta$$
$$y = R.\sin\beta \qquad (9.9)$$

After rotation, the point has coordinates x' and y' with values

$$x'=R.\cos(\alpha+\beta)$$
$$y'=R.\sin(\alpha+\beta) \qquad (9.10)$$

Expanding these formulae for $\cos(\alpha+\beta)$ and $\sin(\alpha+\beta)$ and rearranging gives

$$x'=R.\cos\alpha\cdot\cos\beta - R.\sin\alpha\cdot\sin\beta$$
$$y'=R.\sin\alpha\cdot\cos\beta + R.\sin\beta\cdot\cos\alpha \qquad (9.11)$$

Finally, substituting for $R.\cos\beta$ and $R.\sin\beta$ gives

$$x' = x.\cos\alpha - y.\sin\alpha$$
$$y' = x.\sin\alpha + y.\cos\alpha \qquad (9.12)$$

With the column vector representation for coordinate position, we can write the rotation equation in the matrix form:

$$P' = R.P$$

Where the rotation matrix is

$$R = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \qquad (9.13)$$

R is a rotation transformation matrix to rotate a point in clockwise direction by α angle relative to the origin.

9.3 Homogeneous coordinate system

In animation some time it might require an object to be translated and rotated at each increment of motion. To achieve this we perform translations, rotations, and scaling's to fit the picture components into their proper positions. Here we consider how the matrix representations discussed in the previous section can be reformulated so that such transformations can be processed.

We can combine the multiplicative and translations terms for two-dimensional geometric transformations into a single matrix representation by changing 2 by 2 matrix into 3 by 3 matrices. To obtain square matrices, an additional row was added to the matrix and an additional coordinate, the w-coordinate, was added to the vector for a point. In this way a point in 2D space is expressed in three-dimensional homogeneous coordinates. This technique of representing a point in a space whose dimension is one greater than that of the point is called homogeneous representation. It provides a consistent, uniform way of handling affine transformations. So, we represent each Cartesian coordinate position (x, y) with the homogeneous coordinate triple (x_w, y_w, w).

Here

$$x = \frac{x_w}{w}, \qquad y = \frac{y_w}{w}$$

The newly generated homogeneous coordinates can be represented as (w.x, w.y, w) where w can assume any non-negative value. A convenient choice is simply to set w =1. On converting a 2D point (x, y) to homogeneous coordinates the w-coordinate is set to 1, giving the corresponding homogeneous coordinate point (x, y, 1).

Now rewrite all the matrices derived in the previous sections again with homogeneous rows and columns.

Scaling transformation matrix

$$S = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{9.14}$$

For example, to scale a triangle with respect to the origin, with vertices at original coordinates (10, 20), (10, 10), (20, 10) by S_x=2, S_y=1.5, we compute as followings:

Scaling of vertex (10, 20):

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 \\ 20 \\ 1 \end{bmatrix} = \begin{bmatrix} 2*10+0*20+0*1 \\ 0*10+1.5*20+0*1 \\ 0*10+0*20+1*1 \end{bmatrix} = \begin{bmatrix} 20 \\ 30 \\ 1 \end{bmatrix}$$

Scaling of vertex (10, 10):

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 2*10+0*10+0*1 \\ 0*10+1.5*10+0*1 \\ 0*10+0*10+1*1 \end{bmatrix} = \begin{bmatrix} 20 \\ 15 \\ 1 \end{bmatrix}$$

Scaling of vertex (20, 10):

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 20 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 2*20+0*10+0*1 \\ 0*20+1.5*10+0*1 \\ 0*20+0*10+1*1 \end{bmatrix} = \begin{bmatrix} 40 \\ 15 \\ 1 \end{bmatrix}$$

The resultant coordinates of the triangle vertices are (20, 30), (20, 15), and (40, 15) respectively.

Rotation transformation matrix

$$R = \begin{bmatrix} cos\alpha & -sin\alpha & 0 \\ sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{9.15}$$

For example, to rotate a triangle about the origin with vertices at original coordinates (10, 20), (10, 10), (20, 10) by 30 degrees, we compute as follow:

$$\begin{bmatrix} cos\alpha & -sin\alpha & 0 \\ sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} cos30 & -sin30 & 0 \\ sin30 & cos30 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.866 & -0.5 & 0 \\ 0.5 & 0.866 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation of vertex (10, 20):

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0.866 & -0.5 & 0 \\ 0.5 & 0.866 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 \\ 20 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.866 * 10 - 0.5 * 20 + 0 * 1 \\ 0.5 * 10 + 0.866 * 20 + 0 * 1 \\ 0 * 10 + 0 * 20 + 1 * 1 \end{bmatrix} = \begin{bmatrix} -1.34 \\ 22.32 \\ 1 \end{bmatrix}$$

Rotation of vertex (10, 10):

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0.866 & -0.5 & 0 \\ 0.5 & 0.866 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.866 * 10 - 0.5 * 10 + 0 * 1 \\ 0.5 * 10 + 0.866 * 10 + 0 * 1 \\ 0 * 10 + 0 * 10 + 1 * 1 \end{bmatrix} = \begin{bmatrix} 3.66 \\ 13.66 \\ 1 \end{bmatrix}$$

Rotation of vertex (20, 10):

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0.866 & -0.5 & 0 \\ 0.5 & 0.866 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 20 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.866 * 20 - 0.5 * 10 + 0 * 1 \\ 0.5 * 20 + 0.866 * 10 + 0 * 1 \\ 0 * 20 + 0 * 10 + 1 * 1 \end{bmatrix} = \begin{bmatrix} 12.32 \\ 18.66 \\ 1 \end{bmatrix}$$

The resultant coordinates of the triangle vertices are (-1.34, 22.32), (3.6, 13.66), and (12.32, 18.66) respectively.

Translation transformation matrix

$$T = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \tag{9.16}$$

For example, to translate a triangle with vertices at original coordinates (10, 20), (10, 10), (20, 10) by t_x=5, t_y=10, we compute as followings:

Translation of vertex (10, 20):

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 \\ 20 \\ 1 \end{bmatrix} = \begin{bmatrix} 1*10+0*20+5*1 \\ 0*10+1*20+10*1 \\ 0*10+0*20+1*1 \end{bmatrix} = \begin{bmatrix} 15 \\ 30 \\ 1 \end{bmatrix}$$

Translation of vertex (10, 10):

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 1*10+0*10+5*1 \\ 0*10+1*10+10*1 \\ 0*10+0*10+1*1 \end{bmatrix} = \begin{bmatrix} 15 \\ 20 \\ 1 \end{bmatrix}$$

Translation of vertex (20, 10):

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 20 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 1*20+0*10+5*1 \\ 0*20+1*10+10*1 \\ 0*20+0*10+1*1 \end{bmatrix} = \begin{bmatrix} 25 \\ 20 \\ 1 \end{bmatrix}$$

The resultant coordinates of the triangle vertices are (15, 30), (15, 20), and (25, 20) respectively.

9.4 Other Transformation

The transformation we studied till is included in most graphics packages. Other than these transformations there are some more exciting transformations are also in the market which is used in graphics applications. These two transformations are reflection and shearing.

Reflection

Reflection is nothing more than a rotation of the object by 180o. In case of reflection, the image formed is on the opposite side of the reflective medium with the same size. The axis around which reflection takes place is called *axis of reflection*. We use the identity matrix with positive and negative signs according to the situation respectively. Let see few very common reflections.

Reflection about the line y=0, (i.e. about x axis). The transformation matrix as follows:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{9.17}$$

This transformation keeps x value same but changes the y values of coordinates positions. The resulting orientation of an object after reflection about the x axis shown in Figure 09.6.

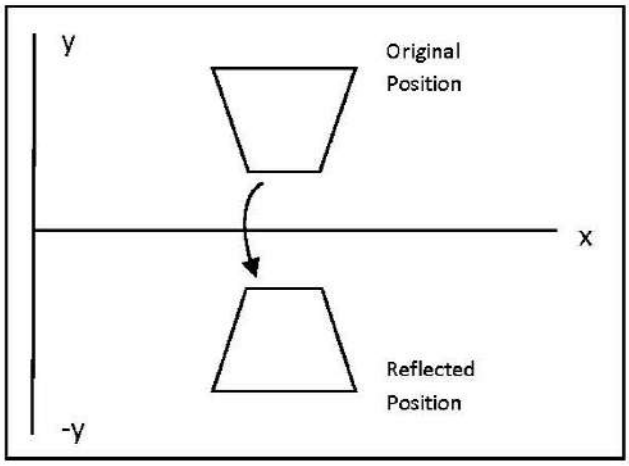


Figure. 9.6: Reflection about x-axis.

Similarly, reflection about the line x=0, (i.e. about y axis). The transformation matrix as follows:

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{9.18}$$

This transformation keeps y value same but changes the x values of coordinates positions. The resulting orientation of an object after reflection about the y axis shown in Figure 09.7.

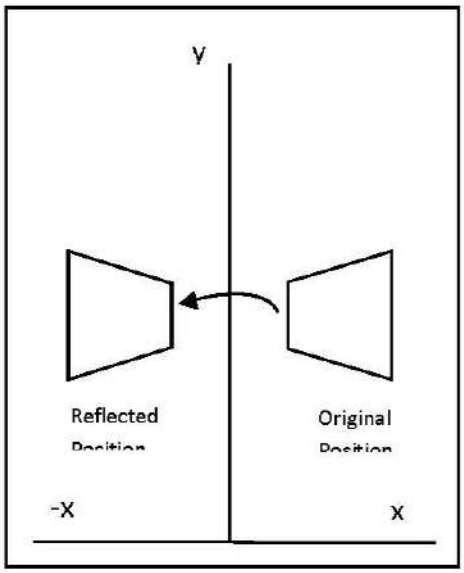


Figure.9.7: Reflection about y-axis.

When both the x and y coordinates are flipped then the reflection produced is relative to an axis that is perpendicular to x-y plane and that passes through the coordinate origin. This transformation is referred as a reflection relative to coordinate origin and can be represented using the matrix below.

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{9.19}$$

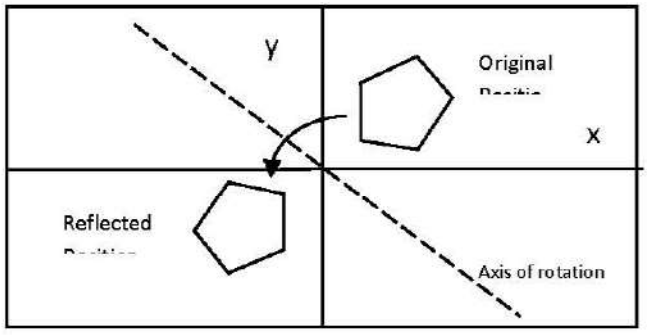


Figure.9.8: Reflection of an object relative to an axis perpendicular to the x-y plane.

Reflection about an arbitrary line y= mx + b can be accomplished with a combination of translate-rotate-reflect transformations.

Steps are as follows

1. Translate the working coordinate system (WCS) so that the line passes through the origin.
2. Rotate the WCS such that one of the coordinate axis lies onto the line.
3. Reflect about the aligned axis
4. Restore the WCS back by using the inverse rotation and translation transformation.

Shear

A shear is a transformation that distorts the shape of an object along either or both of the axis. Like scale and translate, a shear can be done along just one or along both of the coordinate axes. A shear along one axis (say, the x-axis) is performed in terms of the point's coordinate in the other axis (the y-axis). Thus a shear of 1 in the x-axis will cause the x-coordinate of the point to distort by 1*(y-coordinate).

The generalized shearing matrix is:

$$\begin{bmatrix} 1 & a & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (9.20)$$

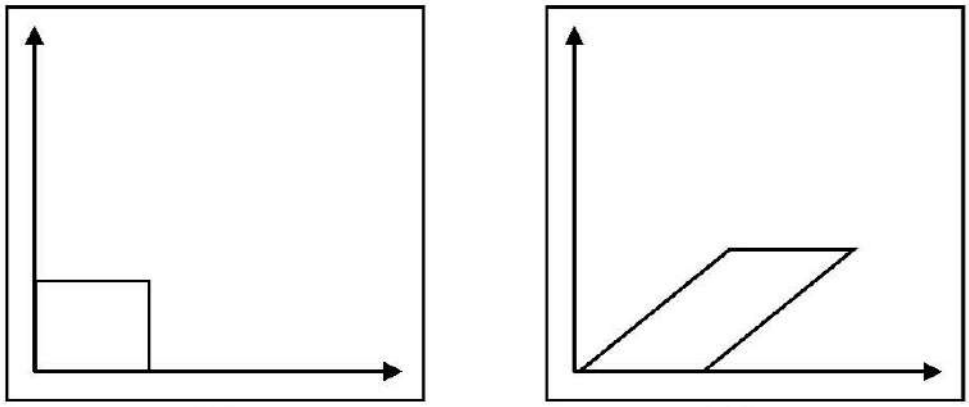
To shear in the x direction the equation is:

$$x' = x + ay$$
$$y' = y \qquad \text{Where } b = 0$$

Where x' and y' are the new values, x and y is the original values, and a is the shearing factor in the x direction. The matrix is as follows.

$$\begin{bmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (9.21)$$

An example of this shearing transformation is given in Figure 9.9 for a shear parameter value of a = 2.



(a) (b)

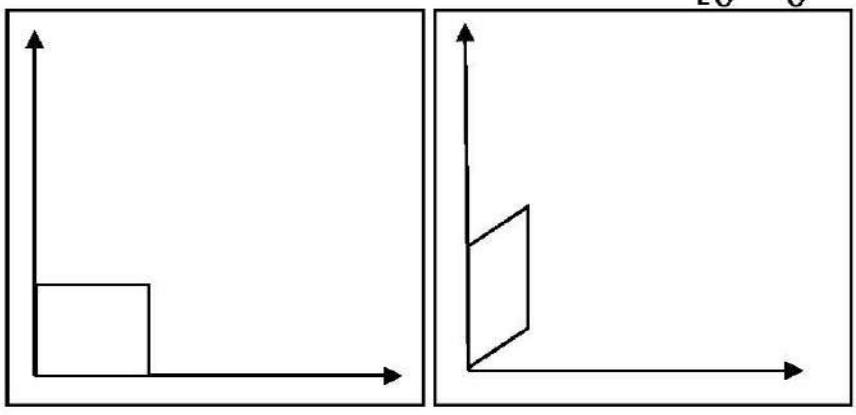
Figure.9.9: A unit square (a) is converted into a parallelogram (b) using the x-direction shear.

Shearing in the y direction is similar except the roles are reversed.

$$x' = x$$

$$y' = y + bx \qquad \text{Where, } a = 0.$$

Where, x' and y' are the new values, x and y are the original values, and b is the scaling factor in the y direction. The matrix is. $\begin{bmatrix} 1 & 0 & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ (9.22)



(a) (b)

Figure.9.9: A unit square (a) is converted into a parallelogram (b) using the y-direction shear.

9.5 Composite transformation

We saw that the basic scaling and rotating transformations are always with respect to the origin. To scale or rotate about a particular point (the fixed point) we must first translate the object so that the fixed point is at the origin. We then perform the scaling or rotation and then the inverse of the original translation to move the fixed point back to its original position. Forming products of transformation matrices is often referred to as a concatenation, or composition of matrices. We form composite transformations by multiplying matrices in order from right to left.

Translations

By common sense, if we translate an object with 2 successive translation vectors: (t_{x1}, t_{y1}) and (t_{x2}, t_{y2}), it is equal to a single translation of $(t_{x1}+ t_{x2}, t_{y1}+ t_{y2})$. The final transformed location P' is calculated as

$$P' = T(t_{x2}, t_{y2}).\ \{T(t_{x1}, t_{y1}).\ P\}$$
$$= \{T(t_{x2}, t_{y2}).\ T(t_{x1}, t_{y1})\}.\ P$$

This additive property can be demonstrated by composite transformation matrix:

$$\begin{bmatrix} 1 & 0 & t_{x2} \\ 0 & 1 & t_{y2} \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & t_{x1} \\ 0 & 1 & t_{y1} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_{x1} + t_{x2} \\ 0 & 1 & t_{y1} + t_{y2} \\ 0 & 0 & 1 \end{bmatrix} \qquad (9.23)$$

or

$$T(t_{x2}, t_{y2}).\ T(t_{x1}, t_{y1}) = T(t_{x1} + t_{x2}, t_{y1} + t_{y2}) \qquad (9.24)$$

This demonstrates that 2 successive translations are additive.

Rotations

By common sense, if we rotate a shape with 2 successive rotation angles: α and β, about the origin, it is equal to rotating the shape once by an angle $\alpha + \beta$ about the origin

The final transformed location P' is calculated as

$$P' = R(\alpha).\ \{R(\beta).\ P\}$$
$$= \{R(\alpha).\ R(\beta)\}.\ P$$

By multiplying the two rotation matrices, we can verify that two successive rotations are additive:

$$R(\alpha).\ R(\beta) = R(\alpha + \beta)$$
$$P' = R(\alpha + \beta).\ P$$

Similarly, this additive property can be demonstrated by composite transformation matrix:

$$\begin{bmatrix} cos\alpha & -sin\alpha & 0 \\ sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} cos\beta & -sin\beta & 0 \\ sin\beta & cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos(\alpha+\beta) & -sin(\alpha+\beta) & 0 \\ sin(\alpha+\beta) & cos(\alpha+\beta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

This demonstrates that 2 successive rotations are additive.

Scaling

Concatenating transformation matrices for two successive scaling factor: (sx1, sy1) and (sx2, sy2), with respect to the origin, it is equal to a single scaling of (sx1* sx2, sy1* sy2) with respect to the origin.

$$P' = S(S_{x2}, S_{y2}).\ \{S(S_{x1}, S_{y1}).\ P\}$$

$$= S(S_{x1}. S_{x2}, S_{y1}. S_{y2}). P$$

Composite transformation matrix for 2 successive scaling:

$$\begin{bmatrix} S_{x1} & 0 & 0 \\ 0 & S_{y1} & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} S_{x2} & 0 & 0 \\ 0 & S_{y2} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} S_{x1} . S_{x2} & 0 & 0 \\ 0 & S_{y1} . S_{y2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

This demonstrates that 2 successive scaling with respect to the origin are multiplicative.

9.6 Self Learning Exercise

Q.1 In which transformation, the mirror image of an object can be seen with respect to x-axis, y-axis and z-axis as well as with respect to an arbitrary line?

(A) Translation (B) Rotation

(C) Reflection (D) Scaling

Q.2 The transformation in which the dimension of an object are changed relative to a specified fixed point is called

(A) Rotation (B)Reflection

(C) Translation (D) Scaling

Q.3 Write down Reflection matrix about x-axis (2 Dimensional).

9.7 Summary

Here we have seen that the polygons, line segments, circles, arcs etc. will create image or pictures. On these images, we can do some manipulations or restructurings for many reasons. These manipulations are performed on basic geometry of these shapes. The basic geometric transformations are translation, rotation, and scaling. When we move an object from one position in a straight line path it is known as translation, when an object moves from one point to another point in a circular path around a specified pivot point, then it is known as rotation, when dimensions of an object changes than it is known as scaling.

Here we study who we can express two-dimensional geometric transformation as 3 by 3 matrix, use of Homogeneous Coordinator, composite transformations of any combinations of translations, rotations and scaling matrices. Other transformations include reflections and shears. Transformations between Cartesian coordinate system are accomplished with a sequence of translate-rotate transformation. After studying this unit in detail, the reader will be able to manipulate or restructure any images.

9.8 Glossary

2-D: computer graphics is the computer-based generation of digital images—mostly from two-dimensional models and by techniques specific to them.

9.9 Answers to Self-Learning Exercise

Ans.1: C **Ans.2:** D

9.10 Exercise

Q.1 What will be Homogeneous form of point (9, 6, 3, 3)?

(A) (1, 2, 3) (B) (3, 2, 1)

(C) (3, 2, 1, 1) (D) (9, 6, 3)

Q.2 If you rotate the point (20, 30) by 90 degrees anticlockwise and then translate it by (-20, 0) and then scale it by (2, 1), where will the point be?

(A) (100, -20) (B) (100, 20)

(C) (100, 10) (D) (-100, 20)

Q.3 Find out the final co-ordinates of a figure bounded by the co-ordinates (1, 1), (3, 4), (5, 7), (10, 3) when rotated about a point (4, 8) by 60° in clockwise direction.

Q.4 Find out the final co-ordinates of a figure bounded by the co-ordinates (1, 1), (3, 4), (5, 7), (10, 3) when rotated about a point (8, 8) by 30° in anticlockwise direction & scaled by 2 units in x-direction & 3 units in y-direction.

Q.5 Find the transformation matrix that transforms the square ABCD to half its size with center still remaining in the same position. The coordinates of square are A(1, 1), B(3, 1), C(3, 3) and D(1, 3) and center at (2, 2).

9.11 Answers to Exercise

Ans.1: C **Ans.2:** D

References and Suggested Readings

1. Computer Graphics by Donald Hearn and M. Pauline Baker; Pearson Education, Seventh Edition 2005
2. Computer Graphics by Apurva A. Desai; PHI Learning, Third Edition 2009
3. Computer Graphics Principles & practice in C by James D. Foley, Steven K. Feiner, Andries van Dam and F. Hughes John; Pearson Education, Second Edition 2003
4. Computer Graphics A Programming Approach by Harrington and Steven; McGraw Hill.

UNIT-10
Geometric 3D Transformation

Structure of the Unit

10.0 Objective

In this chapter, we shall focus on the following topics

- Basic Transformations in 3D
  - Scaling
  - Translation
  - Rotation
- Other Transformations
- Composite Transformations
- Concatenation Properties of composite Matrix

10.1 Introduction

In the previous unit, we discussed about the computer graphics concepts in 2D only. But real world objects have a third dimension which is depth. To simulate any real world scene it needed objects with the third dimension. Unlike 2D graphics, 3D scenes are more complex. Methods for geometric transformation and object modelling in three dimensions are extended from 2D methods by including consideration for the z coordinate. Now translate on an object by specifying a 3D translation vector that determines how much the object is to be moved in each of the three coordinate directions. Similar, we scale an object with three coordinate scaling factors. The extension for three-dimensional rotation is less straight forward.

In this unit, we will discuss the issues associated with the 3D computer graphics.

10.2 Basic Transformation in 3D

Once the 3D objects are created, then like the 2D images, they are required to be processed or manipulated. Like 2D images, these 3D objects can also be translated, scaled and rotated.

TRANSLATION

In a three-dimensional homogeneous coordinate representation, a point is translated form position P = (x, y, z) to P'= (x', y', z') with the matrix operation.

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad (10.1)$$

Or

$$P' = T.\ P$$

Parameters t_x, t_y, and t_z specifying translation distances for the coordinate directions x, y, and z, are assigned any real values.

The Matrix representation in equation 10.1 is equivalent to the three equations

$$x' = x + t_x \qquad y' = y + t_y \qquad z' = z + t_z \qquad (10.2)$$

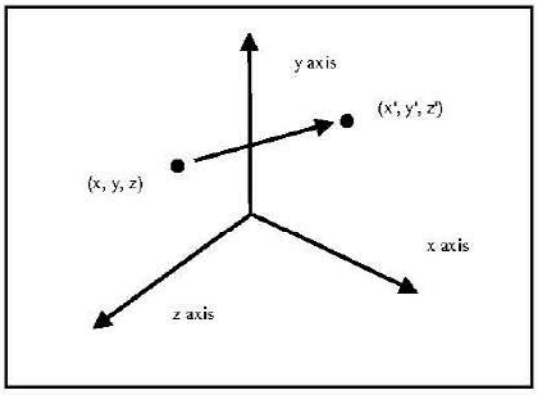


Figure.10.1: Translating an object with translation vector T = (t_x, t_y, t_z)

An object is translated in three dimensions by transforming each of the defining points of the object. For an object represented as a set of polygon surfaces, translate each vertex of each surface Figure 10.1 and redraw the polygon facets in the new position.

We obtain the inverse of the translation matrix in equation 10.1 by negating the translation distance t_x, t_y, and t_z. This produces a translation in the opposite direction, and the product of a translation matrix and its inverse produces the identity matrix.

SCALING

Matrix expression for the scaling transformation of a position P = (x, y, z) relative to the coordinate origin can be written as,

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad (10.3)$$

Or

$$P' = S.\ P$$

Where, scaling parameters S_x, S_y and S_z are assigned any positive values. Explicit expressions for the coordinate transformations for scaling relative to the origin are

$$x' = x\ .\ S_x,\quad y' = y\ .\ S_y,\quad z' = z\ .\ S_z \qquad (10.4)$$

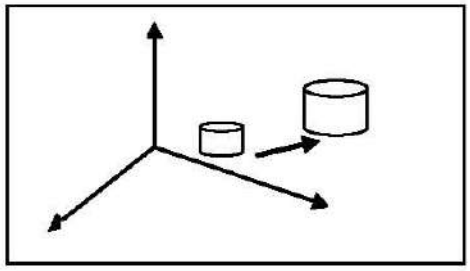


Figure.10.2: Scaling of the object.

Scaling an object with transformation 10.3 changes the size of the object and repositions the object relative to the coordinate origin. Also, if the transformation changed parameters are not all equal, relative dimensions of the object are changed. We preserve the original shape of an object with a uniform scaling ($S_x = S_y = S_z$). The result of scaling an object uniformly with each scaling parameter set to 2 is shown in Figure 10.2.

Scaling with respect to a selected fixed position (x_f, y_f, z_f) can be represented with the following transformation sequence:

1. Translate the fixed point to the origin.
2. Scale the object relative to the coordinate origin using Eq. 10.3.
3. Translate the fixed point back to its original position.

The matrix representation for an arbitrary fixed-point scaling can then be expressed as the concatenation of these translate-scale-translate transformations as

$$T(x_f, y_f, z_f) \cdot S(S_x, S_y, S_z) \cdot T(-x_f, -y_f, -z_f) = \begin{bmatrix} S_x & 0 & 0 & (1-S_x)x_f \\ 0 & S_y & 0 & (1-S_y)y_f \\ 0 & 0 & S_z & (1-S_z)z_f \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (10.5)$$

We form the inverse scaling matrix for either equation 10.3 or equation 10.5 by replacing the scaling parameters S_x, S_y, and S_z with their reciprocals. The inverse matrix generates an opposite scaling transformation, the concatenation of any scaling matrix and its inverse produces the identity matrix.

ROTATION

To generate a rotation transformation for an object, we must designate an axis of rotation (about which the object is to be rotated) and the amount of angular rotation. Unlike a two-dimensional application, where all transformations are carried out in the xy plane, a three-dimensional rotation can be specified around any line in space. The easiest rotation axes to handle are those that are parallel to

the coordinate axes. Also, we can use a combination of coordinate-axis rotation (along with appropriate translations) to specify any general rotation.

By convention, positive rotation angles produce counter clockwise rotation about a coordinate axis, if we are looking along the positive half of the axis toward the coordinate origin Figure 10.3 this agrees with our earlier discussion of rotation in two dimensions, where positive rotations in the xy plane are counter clockwise about axes parallel to the z axis.

Coordinate-Axes Rotations

The two-dimensional z-Axis rotation equations are easily extended to three dimensions:

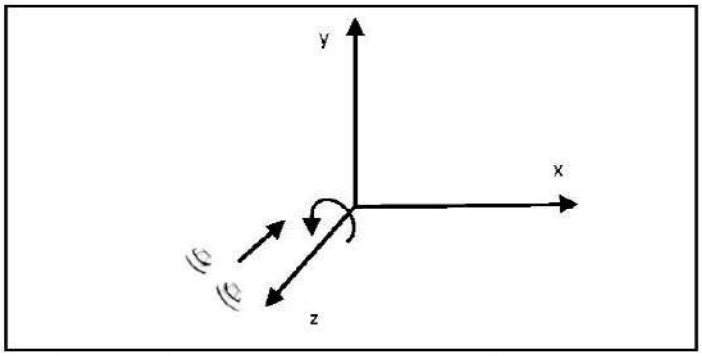


Figure.10.3: Positive rotation directions about the coordinate axes.

$$\begin{aligned} x' &= x\cos\theta - y\sin\theta \\ y' &= x\cos\theta + y\sin\theta \\ z' &= z \end{aligned} \qquad (10.6)$$

Parameter θ specifies the rotation angle. In homogeneous coordinate form, the three-dimensional z-axis rotation equations are expressed as

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta & 0 & 0 \\ sin\theta & cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad (10.7)$$

This can be written more compactly as

$$P' = R_z(\theta) \cdot p \tag{10.8}$$

Transformation equations for rotations about the other two coordinate axes can be obtained with a cyclic permutation of the coordinate parameters x, y, and z in equation 10.6. That is, we use the replacements as illustrated in Figure. 10.4.

$$x \rightarrow y \rightarrow z \rightarrow x \tag{10.9}$$

Substituting permutations 10.9 in equation 10.6, we get the equations for an x-axis rotation:

$$\begin{aligned} y' &= y\cos\theta - z\sin\theta \\ z' &= y\sin\theta + z\cos\theta \\ x' &= x \end{aligned} \tag{10.10}$$

This can be written in the homogeneous coordinate for

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\theta & -sin\theta & 0 \\ 0 & sin\theta & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{10.11}$$

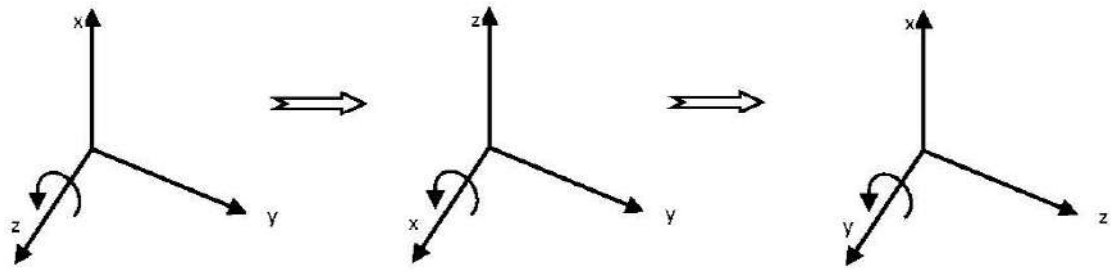


Figure.10.4: cyclic permutations of the Cartesian-coordinate axes to produce the three sets of coordinate-axis rotation equations.

Or

$$P' = R_x(\theta) \cdot P \tag{10.12}$$

Cyclically permuting coordinates in equation 10.10 give us the transformation equations for a y-axis rotation:

$$\begin{aligned} z' &= z\cos\theta - x\sin\theta \\ x' &= z\cos\theta + x\sin\theta \\ y' &= y \end{aligned} \tag{10.13}$$

The matrix representation for y-axis rotation is

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} cos\theta & 0 & sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -sin\theta & 0 & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{10.14}$$

Or

$$P' = R_y(\theta) \cdot P \tag{10.15}$$

An inverse rotation matrix is formed by replacing the rotation angle θ by $-\theta$. Negative values for rotation angles generate rotations in a clockwise direction, the identity matrix is produced when any rotation matrix is multiplied by tis inverse, since only the sine function is affected by the change in sign of the rotation angle, the inverse matrix can also be obtained by interchanging rows and columns. That is, we can circulate the inverse of any rotation matrix R by evaluating its transpose ($R^{-1} = R^T$). This method for obtaining an inverse matrix holds also for any composite rotation matrix.

General Three-Dimensional Rotation

A rotation matrix for any axis that does not coincide with a coordinate axis can be set up as composite transformation involving combination of translations and the coordinate-axes rotations. Obtain the required composite matrix by first setting up the transformation sequence that moves the selected rotation axis onto one of the coordinate axes. Then set up the rotation matrix about that coordinate axis for the specified rotation angle. The last step is to obtain the inverse transformation sequence that returns the rotation axis to its original position.

In the special case where an object is to be rotated about an axis that is parallel to one of the coordinate axes, we can attain the desired rotation with the following transformation sequence.

1. Translate the object so that the rotation axis coincides with the parallel coordinate axis.
2. Perform the specified rotation about that axis.
3. Translate the object so that the rotation axis in moved back to its original position.

The steps in this sequence are illustrated in Figure 10.5. Any coordinate position P on the object in this figure is transformed with the sequence shown as

$$P' = T^{-1}. R_x(\theta). T. P \qquad (10.16)$$

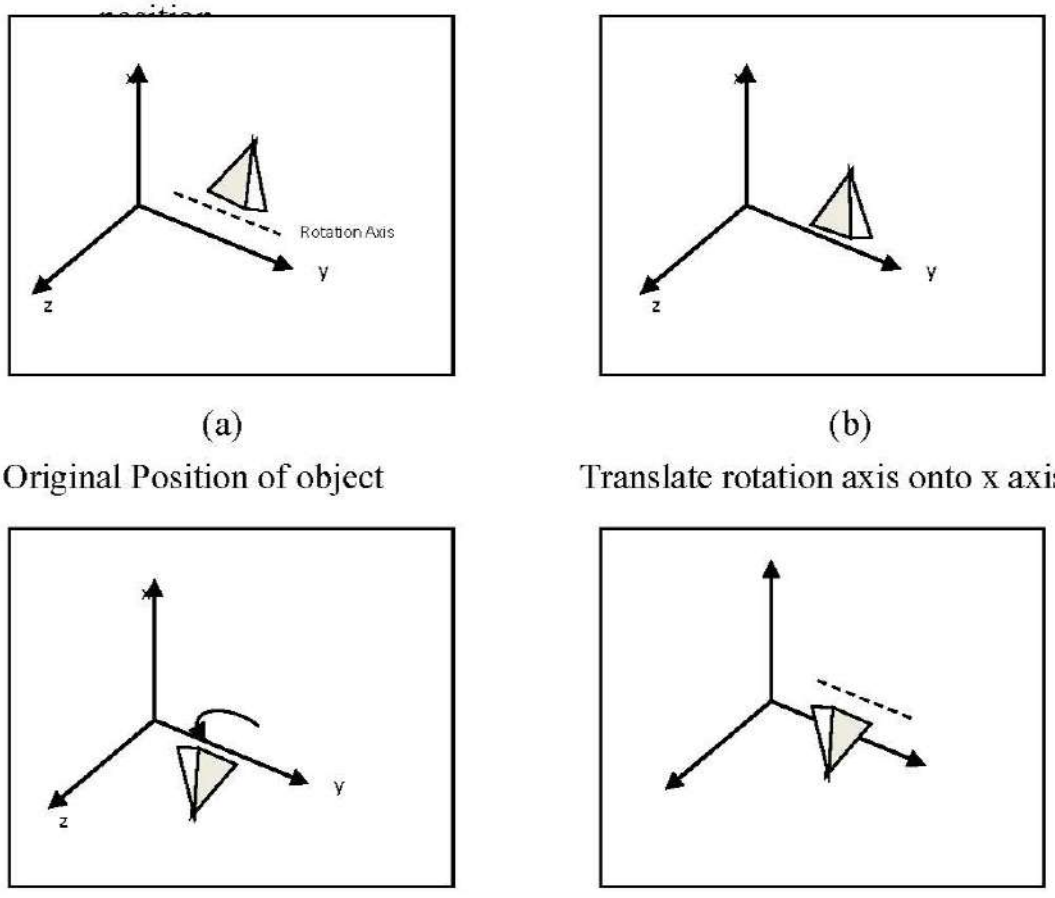
Where the composite matrix for the transformation is

$$R(\theta) = T^{-1} . R_X(\theta) . T \qquad (10.17)$$

This has same form, as the two-dimensional transformation sequence for rotation about an arbitrary pivot point.

When an object is to be rotated about an axis that is not parallel to one of the coordinate axes, we need to perform some additional transformations. In this case, we also need rotations to align the axis with a selected coordinate axis and to bring the axis back to its original orientation. Given the specification for the rotation axis and the rotation angle. We can accomplish the required rotation in five steps:

1. Translate the object so that the rotation axis passes through the coordinate origin.
2. Rotate the object so that the axis of rotation coincides with one of the coordinate axes.
3. Perform the specified rotation about that coordinate axis.
4. Apply inverse rotations to bring the rotation axis back to its original orientation.
5. Apply the inverse translation to bring the rotation axis back to its original position.



(a) Original Position of object

(b) Translate rotation axis onto x axis

(c) Rotate object through angle

(d) Translate rotation axis to original position

Figure.10.5: Transformations sequence for rotating object about an axis that is parallel to the x axis.

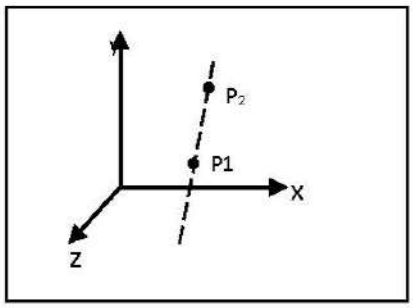
We can transform the rotation axis onto any of the three coordinate axes. The z axis is a reasonable choice, and the following discussion shows how to set up the transformation matrices for getting the rotation axis onto the z axis and returning the rotation axis to its original position Figure 10.6.

A rotation axis can be defined with two coordinate positions, as in Figure 10.7, or with one coordinate point and direction angles (or direction cosines) between the rotation axis and two of the coordinate axes. We will assume that the rotation axis is defined by two points, as illustrated, and that the direction of rotation is to be counter clockwise when looking along the axis from P_2 to P_1. An axis vector is then defined by the two points as,

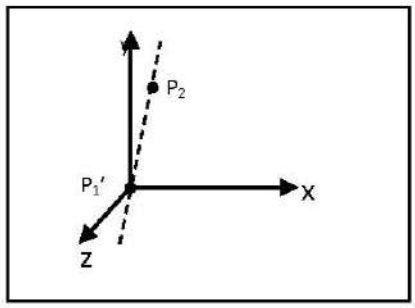
$$\begin{aligned} V &= P_2 - P_1 \\ &= (x_2 - x_1, y_2 - y_1, z_2 - z_1) \end{aligned} \tag{10.18}$$

A unit vector u is then defined along the rotation axis as

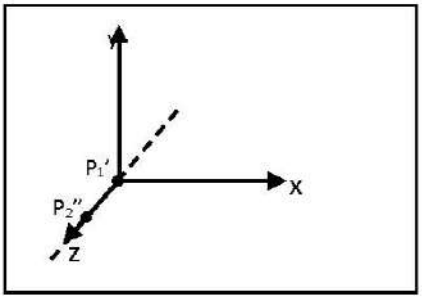
$$u = \frac{V}{[v]} = (a, b, c) \tag{10.19}$$



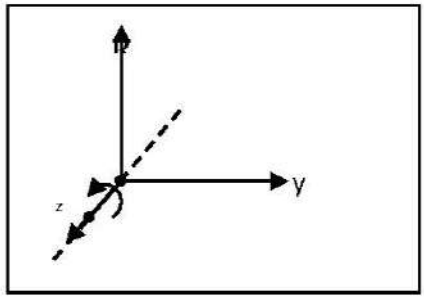
Initial Position



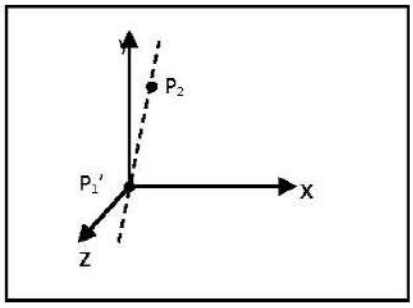
Step 1 Translate P_1 to the Origin



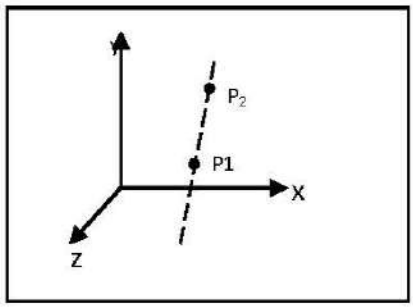
Step 2 Rotate P_2' onto the z axis



Step 3 Rotate the object around the z axis



Step 4 Rotate the object around the z axis



Step 5 Translate the rotation axis to the original position

Figure.10.6: Five transformation steps for obtaining a composite matrix for rotation about an arbitrary axis, which the rotation axis projected onto the z axis.

Where the components a, b, and c of unit vector u are the direction cosines for the rotation axis;

$$a = \frac{x_2 - x_1}{|V|}, \qquad b = \frac{y_2 - y_1}{|V|}, \; c = \frac{z_2 - z_1}{|V|}, \tag{10.20}$$

If the rotation is to be in the opposite direction (clockwise when viewing from (P_2 to P_1), we would reverse axis vector **V** and unit vector **u** so that they point from P_2 to P_1.

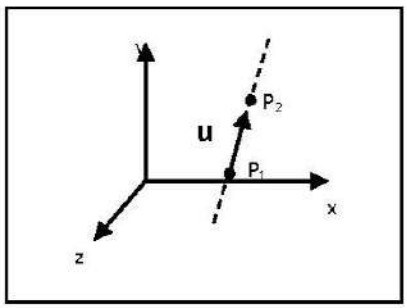


Figure.10.7: An axis of rotation defined with points P_1 and P_2.

The first step in the transformation sequence for the desired rotation is to set up the translation matrix that repositions the rotation axis so that it passes through the coordinate origin. For the desired Direction of rotation Figure 10.7, we accomplish this by moving point P_1 to the origin (if the rotation direction had been specified in the opposite direction, we would move P_2 to the origin.) This translation matrix is

$$T = \begin{bmatrix} 1 & 0 & 0 & -x_1 \\ 0 & 1 & 0 & -y_1 \\ 0 & 0 & 1 & -z_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (10.21)$$

This repositions the rotation axis and the object, as shown in Figure 10.8.

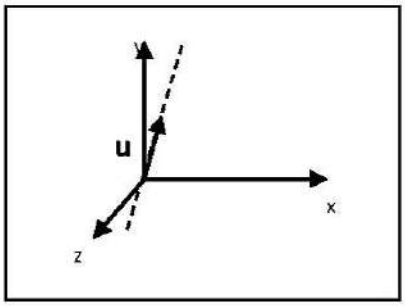


Figure.10.8: Translation of the rotation axis to the coordinate origin

10.3 Other Transformation

Apart from the basic three-dimensional transformations, we have two additional transformations that are often used in various three-dimensional graphics applications. These transformations are reflection and shear.

REFLECTIONS

Reflection in a three-dimensional can be performed relative to a selected reflection axis or with respect to a selected reflection plane. 3D reflection matrices are set up similarly to those for two dimensions. Reflections with respect to a plane are equivalent to 180° rotations in four-dimensional space. Reflections relative to a given axis are equivalent to 180° rotations about that axis. When we do the conversion between left-handed and right-handed system, actually we are reflecting the plane with the coordinate plane as xy, xz, or yz.

For example reflection that converts coordinate specifications from a left-handed system to a right-handed system (or vice versa) is shown in Figure 10.9. In

this transformation the value of x and y coordinates were unchanged and there is changes in the sign of the z coordinates. The matrix representation for this reflection of points relative to the xy plane is.

$$RF_z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

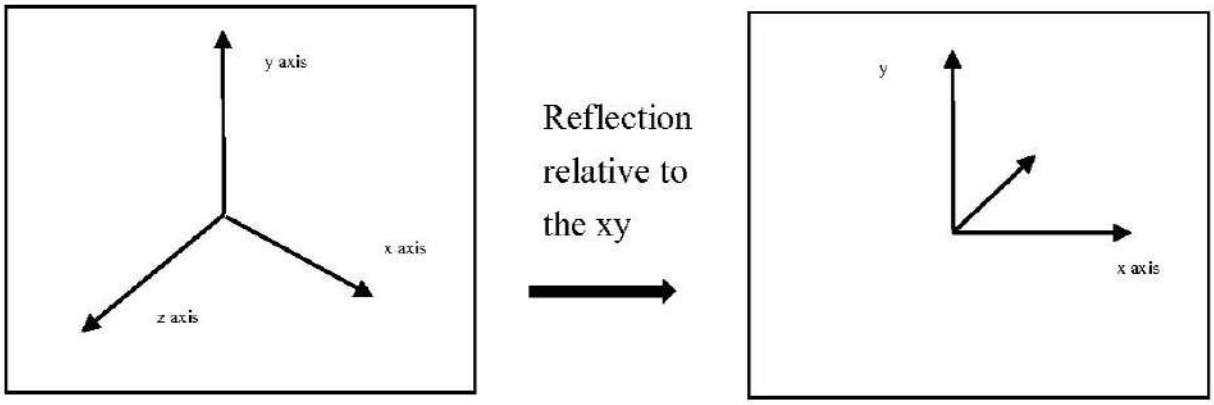


Figure. 10.9: Converting coordinate from a right handed to left handed system.

Transformation matrices for inverting x and y values are defined similarly. Reflections relative to yz and xz plane, respectively. Reflections about other planes can be obtained as combination of rotations and coordinate-plane reflections.

SHEARS

To modify the shape of any object in graphics we use shear transformation. To get a general projection, we can use shearing to view a three-dimensional object. In two dimensions, we discussed transformations relative to the x or y axes to produce distortions in the shapes of the object. In three dimensions, we can also generate shears relative to the z axis.

As an example of three-dimensional shearing, the following transformation produces a z-axis shear:

$$SH_z = \begin{bmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

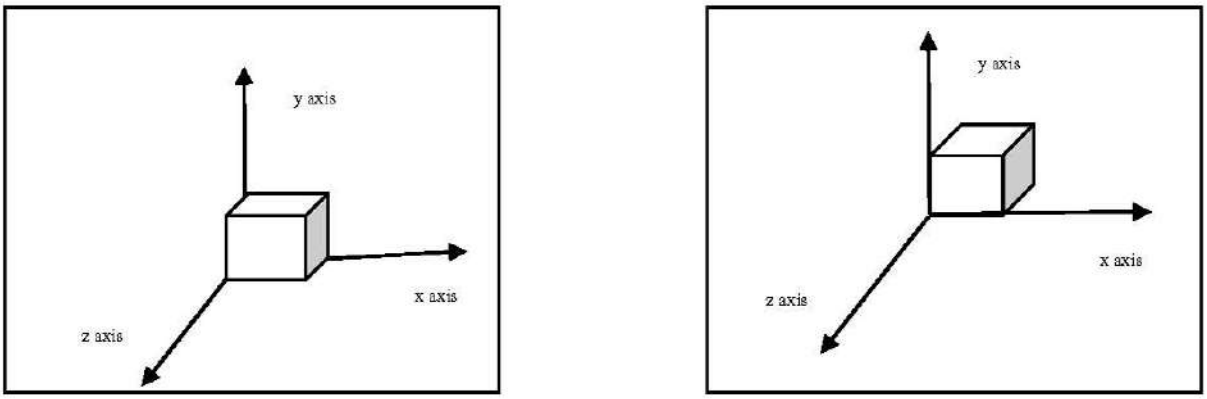


Figure.10.10: A cube is sheared by transformation matrix given above.

Here in this matrix parameters, a and b can be assigned any real values. The effect of this transformation matrix is to alter x-and y-coordinate values by an amount that is proportional to the z value, while leaving the z coordinate unchanged. Boundaries of planes that are perpendicular to the z axis are thus shifted by an amount proportional to z. An example of the effect of this shearing matrix on a cube is shown in Figure 10.10, for shearing values a= b = 1, Shearing matrices for the x axis and y axis are defined similarly.

10.4 Composite Transformation

In 2D transformation, we have seen that the composite transformation we got is basically the matrix multiplication of various homogeneous matrixes of transformations. We can form a composite three-dimensional transformations by the same methods. For composite three dimensional transformations multiply the matrix representations of the individual operation in the transformation sequence.

This concatenation is carried out from right to left, where the rightmost matrix is the first transformation to be applied to an object and the left most matrix is the last transformation. To get the final coordinates of the transformed object we

apply the sequence of basic three-dimensional geometric transformations which are combination of a single composite transformation.

10.5 Concatenation Properties of composite matrix

I. Matrix multiplication is associative:

For any three matrices, A, B and C the matrix product A*B*C can be performed by first multiplying A and B or by first multiplying B and C:

$$A * B * C = (A * B) * C = A * (B * C)$$

Therefore, we can evaluate matrix products using either a left to right or a right to left associative grouping.

For example, we have a triangle, we want to rotate it with the matrix B, and then we translate it with matrix A.
Then, for a vertex of that triangle represented as C, we compute its transformation as:

$$C'=A\cdot(B\cdot C)$$

But we can also change the computation method as:

$$C' = (A\cdot B)\cdot C$$

The advantage of computing it using $C' = (A\cdot B)\cdot C$ instead of $C'=A\cdot(B\cdot C)$ is that, for computing the 3 vertices of the triangle, C1, C2, C3, the computation time is shortened:
Using $C'=A\cdot(B\cdot C)$:

1. compute B · C1 and put the result into I1
2. compute A · I1 and put the result into C1'
3. compute B · C2 and put the result into I2
4. compute A · I2 and put the result into C2'
5. compute B · C3 and put the result into I3
6. compute A · I3 and put the result into C3'

Using C' = (A·B)·C:

1. compute A · B and put the result into M
2. compute M · C1 and put the result into C1'
3. compute M · C2 and put the result into C2'
4. compute M · C3 and put the result into C3

II. Matrix multiplication may not be commutative:

The matrix product A*B is not equal to B*A.

If we want to translate and rotate an object we must be careful about the order in which the composite matrix is evaluated. Using the previous example, if you compute

C' = (A·B)·C

You are rotating the triangle with B first, then translate it with A.
But if you compute

C' =(B·A)·C

You are translating it with A first, then rotate it with B. The result is different.

In special cases, as a sequence of transformation all of the same kind, the multiplication of transformation matrices is commutative. Two successive rotations could be performed in either order and the final position would be same.

10.6 Self Learning Exercise

Q.1 The equation for describing surface of 3D plane is.

(A) Ax+ By+ Cz+ D= 0 (B) Ax+ By+ Cz = 0
(C) Ax+ By+ D= 0 (D) Ax+ By+ Cz+ D= 1

Q.2 The sweep representation of an object refers to the

(A) 2D representation (B) 3D representation
(C) Both a & b (D) None of these

Q.3 A circle, if scaled only in one direction becomes…….

(A) Hyperbola (B) Ellipse

(C) Parabola (D) remains a circle

Q.4 What is affine transformation?

10.7 Summary

In this unit, we have seen the concepts for creating the 3D graphical objects were discussed. Here we discussed the 3D transformation which is used in various computer graphics applications. The basic geometric transformations are translation, rotation and scaling. Two additional transformations are reflections and shears. This unit deals with the creation, representation and manipulation of 3D objects. The operations are represented with 4 by 4 matrices. By concatenation the matrix representations for the individual components we will get the composite transformations which are used in animations.

The concatenation properties of various composite matrixes are explained. Finally, the object modelling often requires a hierarchical transformation structure that ensures that the each component of object move or modified in the proper manner.

10.8 Glossary

3D: Three dimension.

10.9 Answers to Self-Learning Exercise

Ans.1: A **Ans.2:** B **Ans.3:** B

10.10 Exercise

Q.1 A composite transformation matrix can be made by determining the _____ of matrix of the individual transformation.

(A) Sum (B) Product

(C) Difference (D) None of the above

Q.2 _____refers to the result obtained by multiplying the matrix of the individual transformation representation sequences

(A) Wire frame model (B) Constructive solid geometry methods

(C) Composite transformation (D) None of these

Q.3 Forming products of transformation matrices is often referred as.

(A) Concatenation (B) Composition

(C) Both a & b (D) None of these

Q.4 A translation that takes (x, y, z) to (x, y-3, z) is given by which translation matrix?

Q.5 What is composite transformation? Explain two successive translation and rotations with the final composite transformation matrixes in 3D.

Q.6 Find a transformation of triangle A (1, 0, 1), B (0, 1, 1), C (1, 1, 0)

i) Rotating 45° about the origin & than translating 1 unit in x, y & z direction.

ii) Translating 1 unit in x, y & z direction & than rotating 45° about the origin.

Q.7 Prove that the multiplication of 3-dimensional transformation matrices for each of the following sequence operations is commutative:

a) Any 2 successive translations

b) Any 2 successive scaling operations

c) Any 2 successive rotations about any of the co-ordinate axis

10.11 Answers to Exercise

Ans.1: B

Ans.2: C

Ans.3: C

References and Suggested Readings

1. Computer Graphics by Donald Hearn and M. Pauline Baker; Pearson Education, Seventh Edition 2005
2. Computer Graphics by Apurva A. Desai; PHI Learning, Third Edition 2010
3. Computer Graphics principles & practice in C by James D. Foley, Steven K. Feiner, Andries van Dam and F. Hughes John; Pearson Education, Second Edition 2003
4. Computer Graphics A Programming Approach by Harrington and Steven; McGraw Hill.

UNIT-11
Viewing Transformation

Structure of the Unit

11.0 Objective

In this chapter, we shall focus on the following topics

- Window to Viewport Transformation
- Viewing in 3D
- Perspective Projection
- Parallel Projections

11.1 Introduction

To display the view of a picture on to the display device, we make use of viewing transformation. Our picture is defined in one coordinate system which has to be mapped on device or screen coordinates. In short, many times the user wants to work only on a part of the image or object also he wants only a particular part of the image.

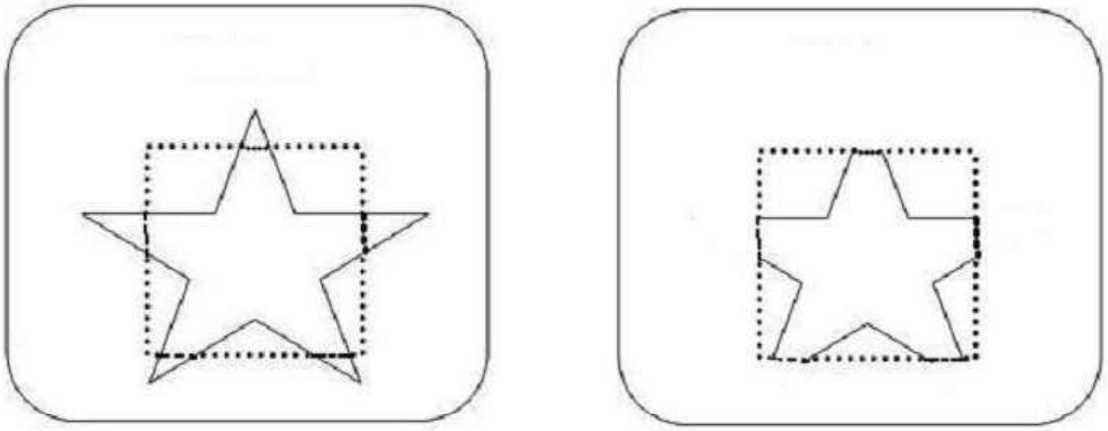
11.2 Coordinate Systems

The coordinate system is used to address the screen are called **screen coordinates**. This is also called **device coordinates**. Another coordinates system called as **world coordinates system** are user defines application specific coordinated system having its own units of measure, axis, origin, etc.

The rectangular region of the world that is visible is called **window** and the method or selecting only that part of the image is called **windowing**. The rectangular region of screen space that is used to display the window is called **viewport**.

Note that when we put a window on any object or image, the part selected in that window can be called as object space and when selected part is displayed in a view port then it is called as image space. In general, the mapping of the world coordinate scene on the device coordinates is called ***viewing transformation*** or

converting the image from object space to image space is called viewing transformation.



Window Viewport

Figure.11.1: Window and Viewport

11.3 Window to Viewport Transformation

Window and viewport generally have standard rectangle shapes, though they can have any orientation. The transformation has few basic steps as shown in figure 11.2.

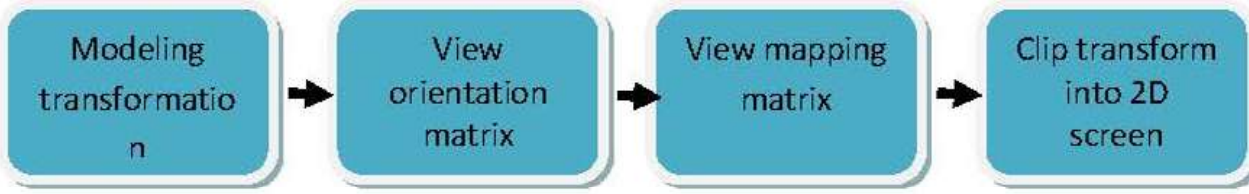


Figure.11.2: Process of Window to Viewport Transformation

Window is denoted by (x, y_{space}) with the coordinate's x_{min}, y_{min}, x_{max}, y_{max}. Viewport is denoted by (u, v_{space}) with the coordinated u_{min}, v_{min}, u_{max}, v_{max}. The transformation steps for mapping from window to viewport are:-

- Translate the window to the origin
- Scale it to the size of the view port
- Translate it to the view port location.

Matrix representation for window to viewport transformation, M_{wv}:-

$$M_{wv} = T(u_{min}, v_{min}).\ S(S_x, S_y).\ T(-x_{min}, -y_{min})$$

Where,

$$S_x = \frac{umax - umin}{xmax - xmin} S_y = \frac{vmax - vmin}{ymax - ymin}$$

11.4 Viewing in 3D

Viewing transformations for 3D objects are more complex and challenging than 2D objects. The reason behind is that in 3D third dimension and requirement for the 3D object to appear realistic in 2D display device. So the mapping from the 3D objects on 2D display device projection is performed. ***Projection*** is the process of transformation of 3D coordinated system to 2D coordinates.

Projections are classified into perspective and parallel projections. This classification is based on whether rays coming from the object converge at the ***Center of projection (COP)*** or not. For perspective projection, rays converge at COP and for parallel projection rays do not converge at COP. So an assumption for parallel projection is that rays do converge but at infinity. Center of projection is also called ***perspective reference point (PRP).***

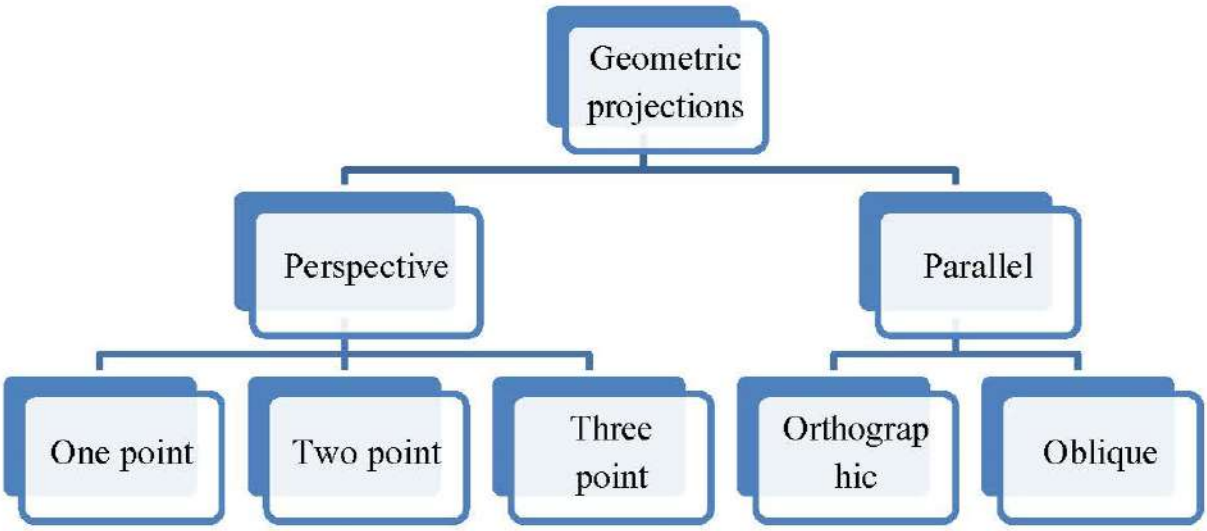


Figure.11.3: Classification of Geometric Projection

11.5 Perspective Projection

In perspective projection, the distance from the center of projection to project plane is finite and the size of the object varies inversely with distance which looks more realistic

.

Figure.11.4: Perspective Projection in real world

The distance and angles are not preserved and parallel lines do not remain parallel. Instead, they all converge at a single point at center of projection or projection reference point.

In figure 11.4 we can see that parallel roads getting narrower and narrower and seem to be meeting at a point.

In perspective projection lines converge into a point and these lines are not parallel to each other. In real world the eyes of the viewer is the center of projection and the lines of projection are the light rays that are coming into viewer's eyes as shown in below figure 11.5.

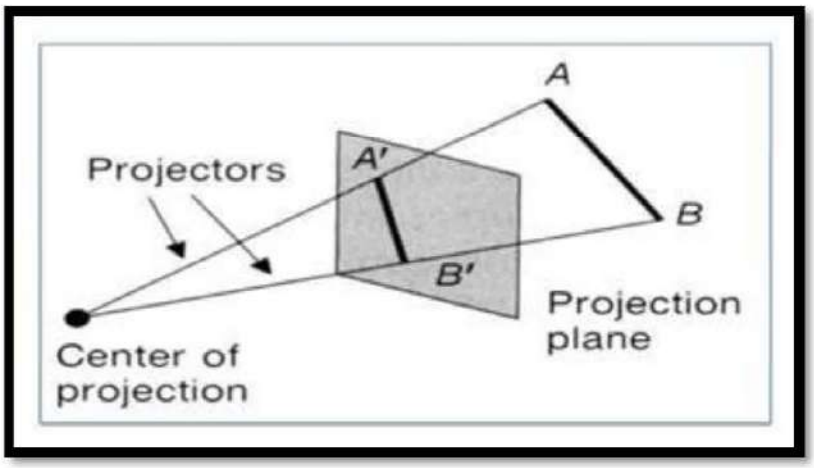


Figure.11.5: Perspective Projection

The rate at which the parallel lines converge to the Centre of projection is called ***Perspective Angle***. This angle is determined by the distance the Centre of projection and the object. It is obvious that a larger perspective angle will result into a larger projected image.

Types of perspective projection

- One point perspective

- Two point perspective
- Three point perspective

11.5.1One-point perspective

One-point perspective when it contains only one vanishing point on the horizon line. The vanishing point is a point in the image where a parallel line through the Centre of projection intersects the view plane or we can say that vanishing point is a point from where the projection line intersects the view plane. This type of perspective is typically used for images of roads, railway tracks, hallways, or buildings viewed so that the front is directly facing the viewer. The parallel lines converge at the vanishing point.

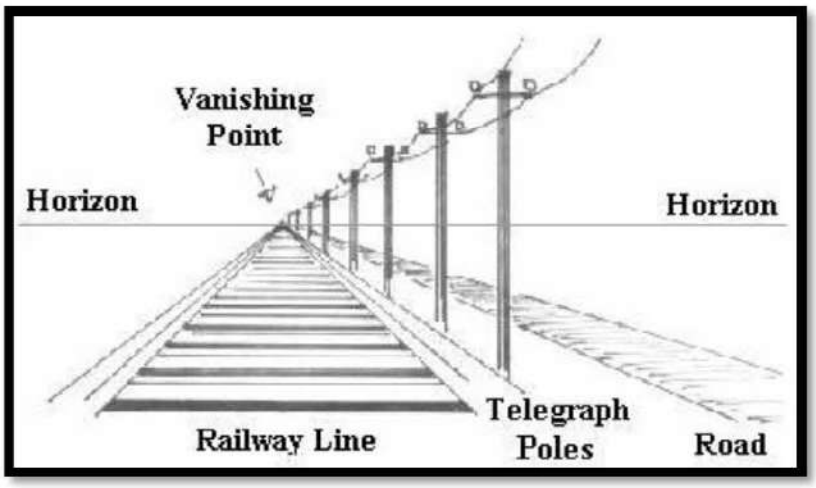


Figure.11.6: One-point perspective in real world

In one point perspective parallel lines will not intersect to the view plane only the edges which are parallel to z-axis will only intersect the view plane and therefore such edges will create vanishing point.

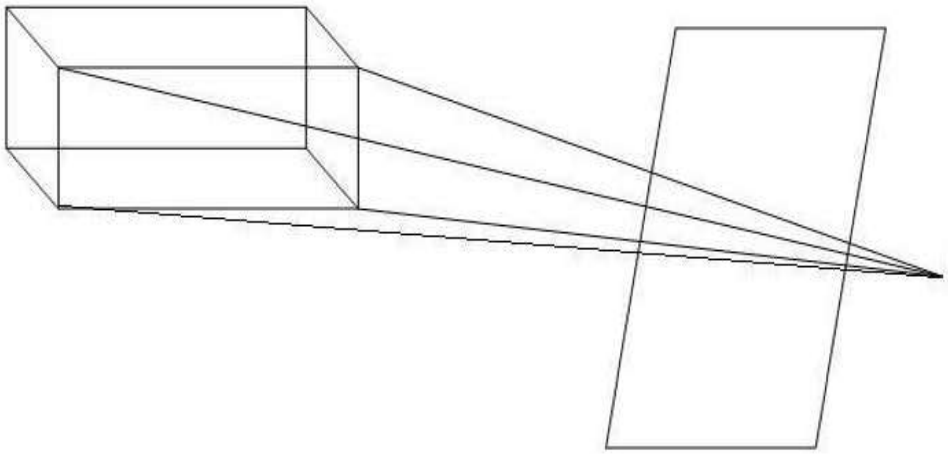


Figure.11.7: One-point perspective with one vanishing point

11.5.2 Two-point perspective

In two-point perspective when it contains two vanishing points on the horizon line.

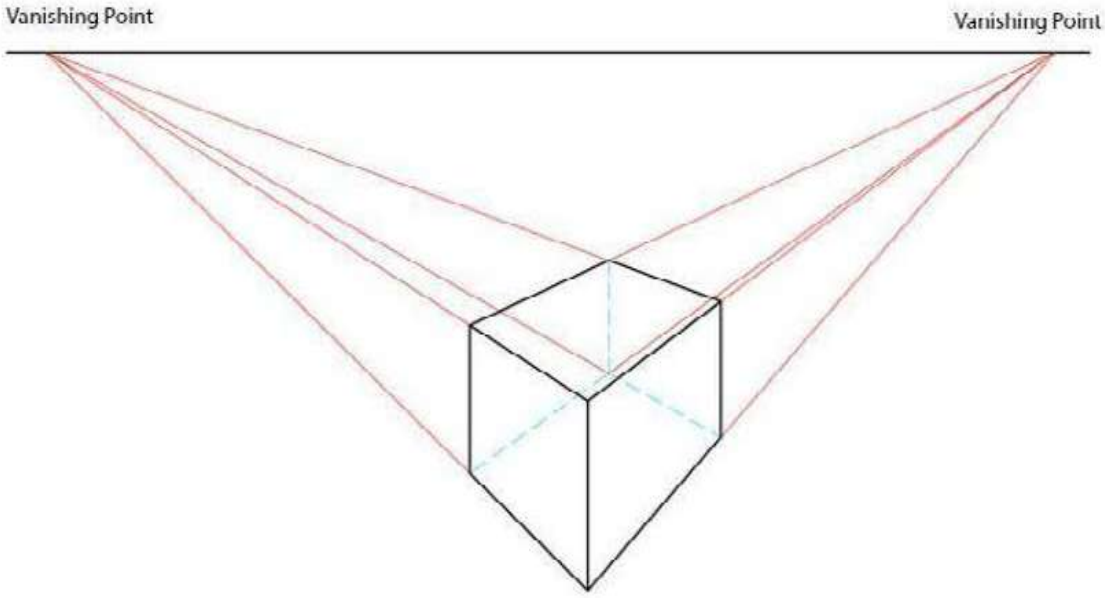


Figure.11.8: two-point perspective with two vanishing point

Two-point perspective has one set of lines parallel to the picture plane and two sets oblique to it. Parallel lines oblique to the picture plane converge to a vanishing point, which means that this set-up will require two vanishing points.

11.5.3 Three-point perspective

In three point perspective none of the edges of the object is parallel to the view plane. All the edges intersect to the view plane. This creates a three-point perspective projection.

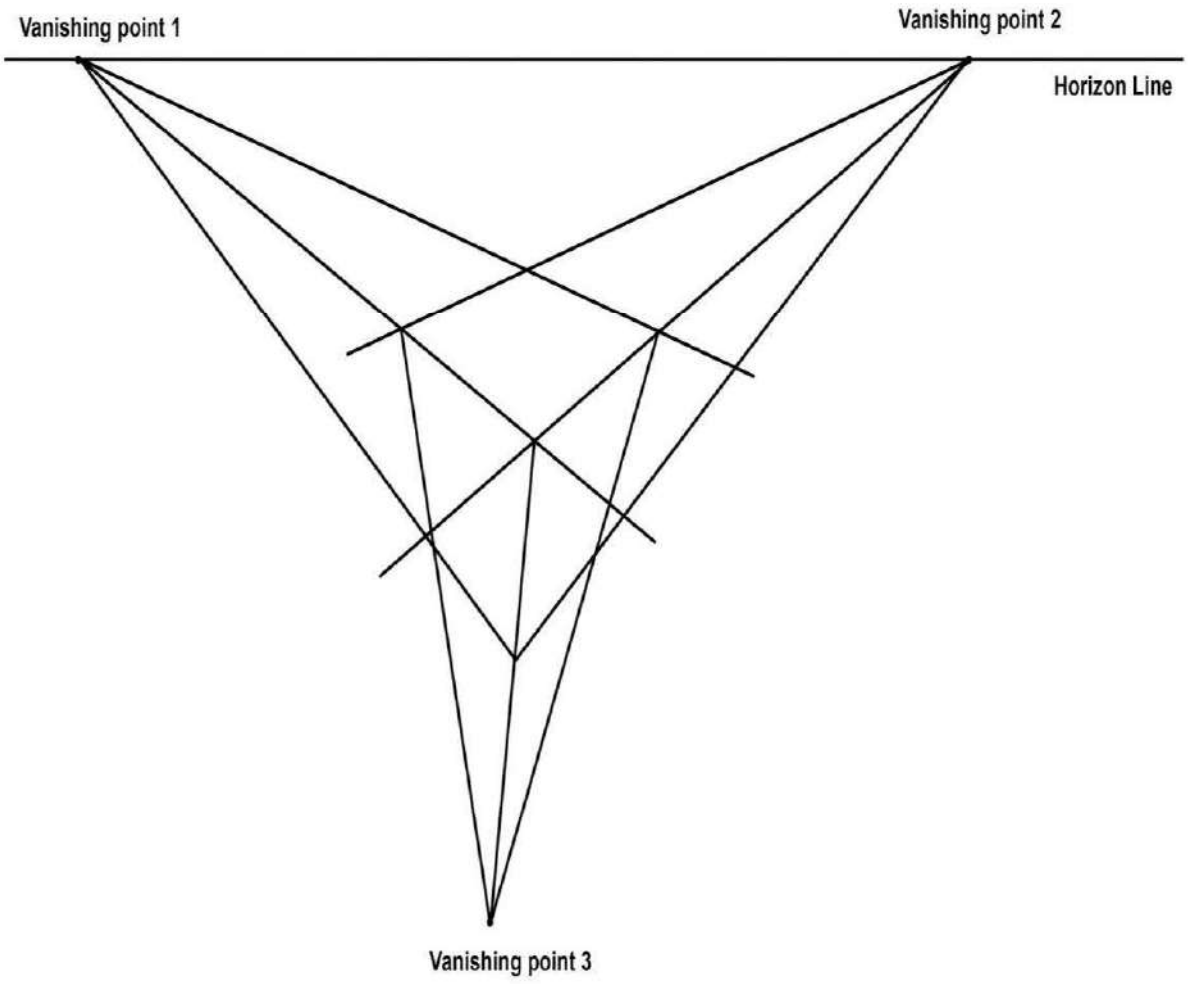


Figure.11.9: Three-point perspective with three vanishing point

11.6 Parallel Projections

In parallel projection, rays are coming from the object converge at infinity, i.e. the distance from the Centre of projection to the projection plane is infinity. Therefore, projectors are parallel lines and we need to specify a direction of projection (DOP) or direction cosines. Parallel projection preserves the relative proportions of objects thus the view of object obtained is accurate but not realistic like perspective projection.

Parallel projection is classified as orthographic and oblique projection. This classification is based on the angle between of projection and projection plane.

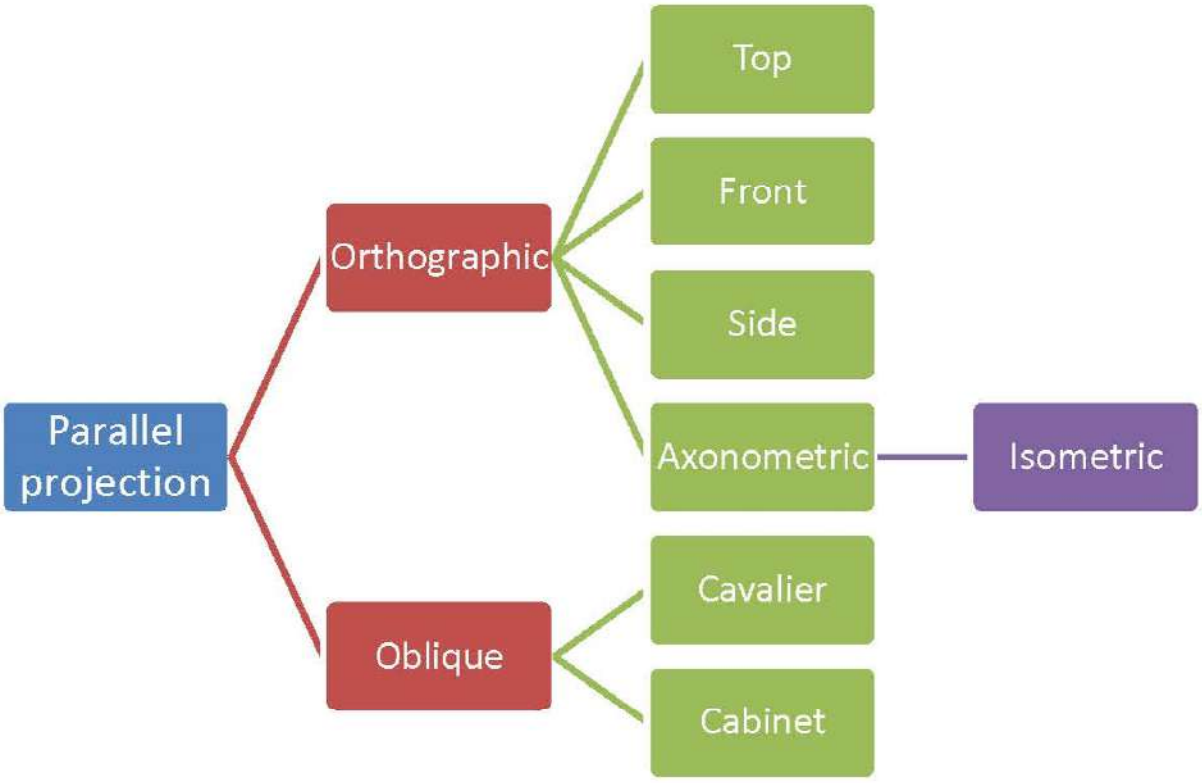


Figure.11.10: Types of parallel projection

11.6.1 Orthographic Projection

For orthographic projection, the angle is of 90° and it produces top plane view, front elevation and side elevation. Also it includes only two dimensions: length and width. Figure 11.11 shows the orthographic projection showing a front view, side view and top view.

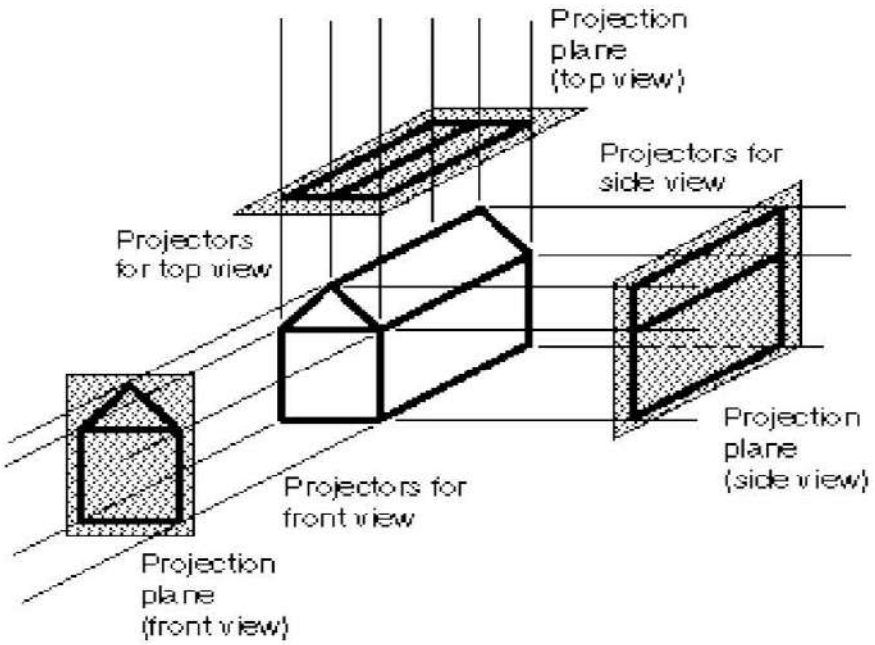


Figure.11.11: Orthographic projection

Another type of Orthographic projection is axonometric projection which is used in projection plane that are not normal to the principal axis and they show multiple faces of an object. Isometric projection is a special case of axonometric projection. For this projection, projection plane intersects each coordinates axis, in which object is defined, at the same distance from the origin.

Orthographic projections are generally used in engineering and architectural dewing.

11.6.2 Oblique Projection

For orthographic projection, the angle is of 90° and for all another angle it is oblique parallel projection. Oblique projection produces three dimensions length, width, and height. Thus oblique projection shows all the dimensions in a single view.

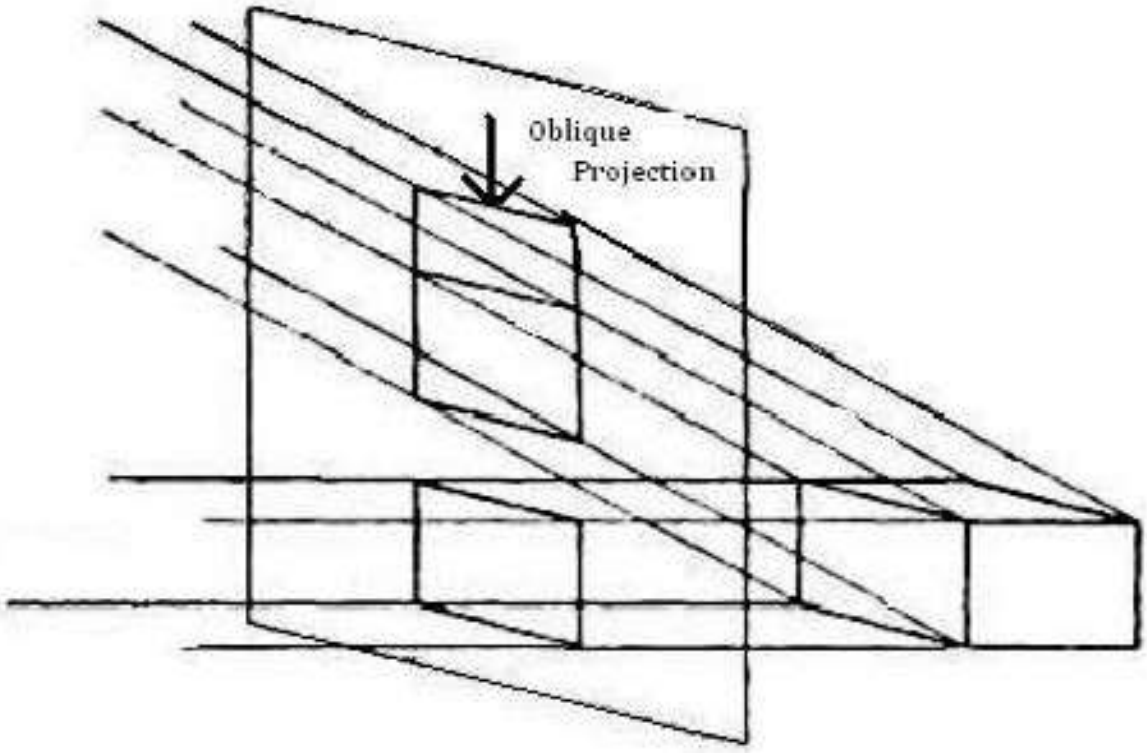
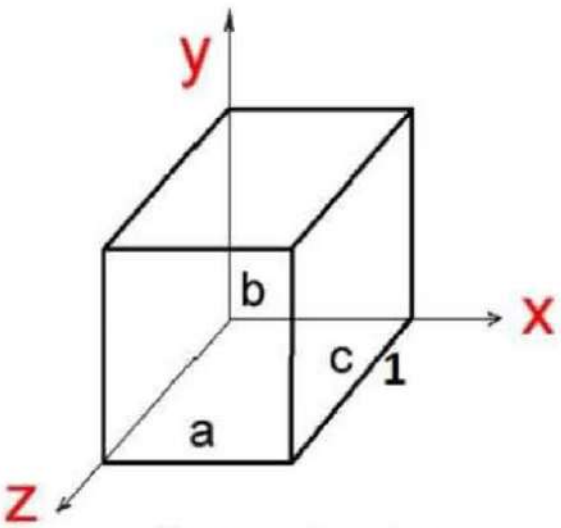


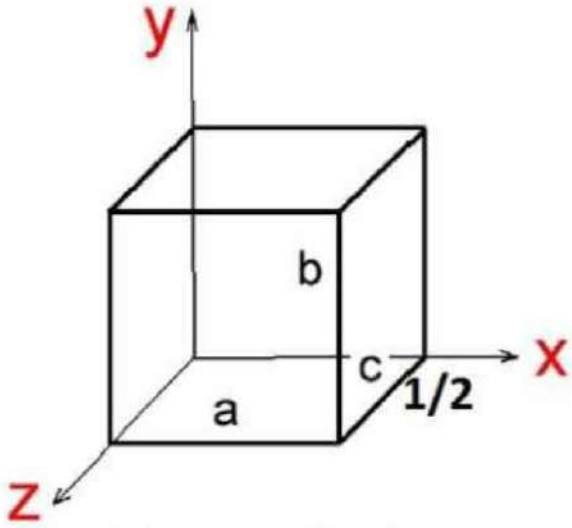
Figure.11.12: Oblique Projections

There are two types of oblique projections – ***Cavalier*** and ***Cabinet***. The Cavalier projection makes 45° angle with the projection plane. The projection of a line perpendicular to the view plane has the same length as the line itself in Cavalier projection. In a cavalier projection, the foreshortening factors for all three principal directions are equal.

The Cabinet projection makes 63.4° angle with the projection plane. In Cabinet projection, lines perpendicular to the viewing surface is projected at ½ their actual length. Both the projections are shown in the below figure.



Cavalier Projection



Cabinet Projection

11.7 Self Learning Exercise

Q.1 In an oblique drawing, the projection rays are drawn _____ to each other and _____ to the plane of projection.

(A) Oblique.....oblique
(B) Oblique.....parallel
(C) Parallel.....oblique
(D) Parallel....parallel

Q.2 What two types of projections give a pictorial view of the object without convergence?

(E) Orthographic and perspective
(F) Oblique and axonometric
(G) Perspective and oblique

(H) Isometric and orthographic

Q.3 What is the difference between parallel and perspective projection. Explain with suitable example.

11.8 Summary

In this unit, we have seen how we can display 3D object into a 2D object by projecting it onto a plane. Projecting a three dimensional object onto a plane is similar to casting.

Perspective projections are at their best when:

- Realism counts
- We want to move through the scene and have a view like a human viewer would have
- We do not need to measure or align parts of the image

Orthographic projections are at their best when:

- Items in the scene need to be checked to see if they line up or are the same size
- Lines need to be checked to see if they are parallel
- We do not care that distance is handled unrealistically
- We are not trying to move through the scene

11.9 Glossary

Viewport: An area on display device to which a window is mapped.

COP: The centre of projection is the origin or source of the stream of projecting rays.

Window: An area of a world coordinate scene that has been selected for display.

11.10 Answers to Self-Learning Exercise

Ans.1: C **Ans.2:** B

11.11 Exercise

Q.1 By which, we can take a view of an object from different directions and different distances

(A) Projection
(B) Rotation
(C) Translation
(D) Scaling

Q.2 Projection rays (projectors) emanate from a

(A) COP (Centre of projection)
(B) Intersect projection plane
(C) Both a & b
(D) None of these

Q.3 The Centre of projection for parallel projectors is at

(A) Zero
(B) Infinity
(C) One
(D) None of these

Q4. What is the major difference(s) between perspective and parallel projection?

(A) Parallel projection can only be used with objects containing parallel edges.

(B) Perspective projection gives a more realistic representation of an object.

(C) Parallel projection is equivalent to a perspective projection where the viewer is standing infinitely far away.

(D) B and C

Q5. Find the principal vanishing points, when the object is first rotated with respect to y-axis by -30° and x-axis by 45° and projected onto z=0 plane, with the Centre of projection being (0,0,-5).

11.12 Answers to Exercise

Ans.1: A

Ans.2: C

Ans.3: B

Ans.4: D

References and Suggested Readings

1. Computer Graphics by Donald Hearn and M. Pauline Baker; Pearson Education, Seventh Edition 2005
2. Computer Graphics by Apurva A. Desai; PHI Learning, Third Edition 2012
3. Computer Graphics principles & practice in C by James D. Foley, Steven K. Feiner, Andries van Dam and F. Hughes John; Pearson Education, Second Edition 2003
4. Computer Graphics a Programming Approach by Harrington and Steven; McGraw Hill.

UNIT-12
Clipping

Structure of the Unit

12.0 Objective

In this chapter, we shall focus on the following topics

- Point Clipping
- Line Clipping
- Polygon Clipping
- Text Clipping

12.1 Introduction

Any method that identifies those parts of the image which is either inside or outside of a specified region of space is known as a clipping algorithm or clipping. The area against which an image is clipped is known as clip window. Here, we are considering clipping window as a rectangular in shape. There are other clipping window shape also. For the viewing transformation, we want to display only those parts of the image which are within the window region. Anything which is outside the window is discarded.

For any picture or image, there could be three possibilities. First, the complete image is outside the clipping window, and therefore the complete image will be discarded and will not be saved for display in the viewport. Second, the complete image is inside the clipping window and therefore the entire image will be displayed in the viewport. Third and Final possibility, that an image is partially inside and partially outside the clipping window. For third case which is partially inside and outside, we use some algorithm to clip or cut the outer part from the image.

Different algorithms are available for different kinds of images and shapes. In this unit, we will study the algorithms used for clipping the following primitive's types.

- Point Clipping
- Line Clipping (straight-line segments)

- Area Clipping (polygons)
- Text Clipping

12.2 Point Clipping

Suppose we are given a point A(x, y) and the standard clipping window (rectangular in shape) see Figure 12.1.

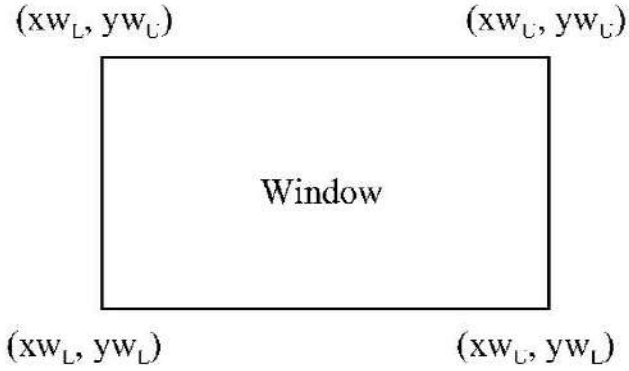


Figure.12.1: Window coordinates.

Now point A will be considered within the clipping area or within the window if the point A(x, y) satisfies the following conditions.

$$xw_L \leq x \leq xw_U$$

$$yw_L \leq y \leq yw_U$$

Where the edges of the clip window (xw_L, xw_U, yw_L, yw_U) can be either the world coordinates boundaries or viewport boundaries. If any one of these four inequalities if not satisfied, the point is clipped. Point clipping algorithm is less often used then that of line or polygon clipping. However, point clipping can be applied to scenes involving small particles (points) in the image. This algorithm can be used for background clipping, the background that is created by a dotted pattern.

12.3 Line Clipping

In this section, we discuss Line clipping algorithms which are mostly used to clip the straight lines or line segment. The following are the major line clipping algorithms.

i. Cohen-Sutherland Algorithm
ii. Liang-Barsky Algorithm
iii. Nicholl-Lee_Nicholl Algorithm
iv. Mid-Point Subdivision Algorithm

There are many possible relationships between lines and clipping windows. A line clipping method involves several parts. As illustrated in Figure 12.2.

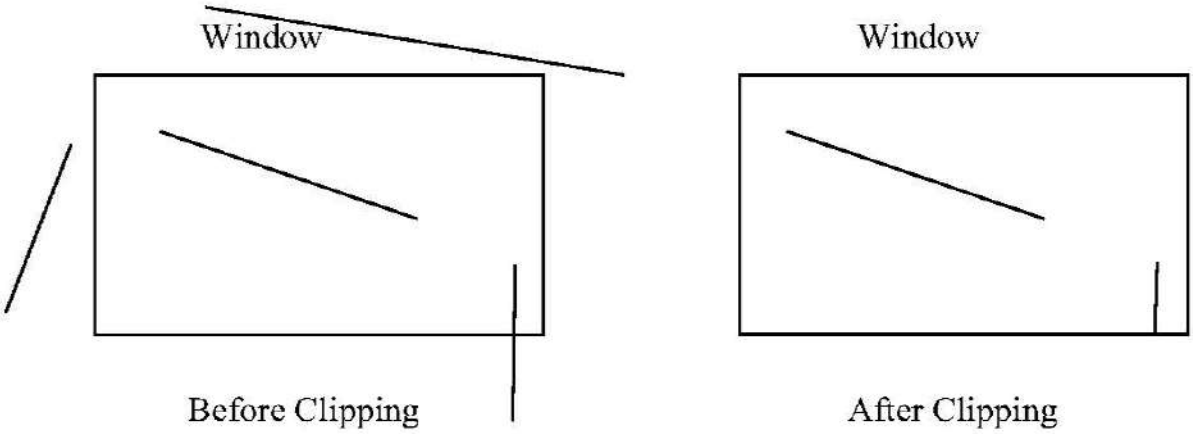
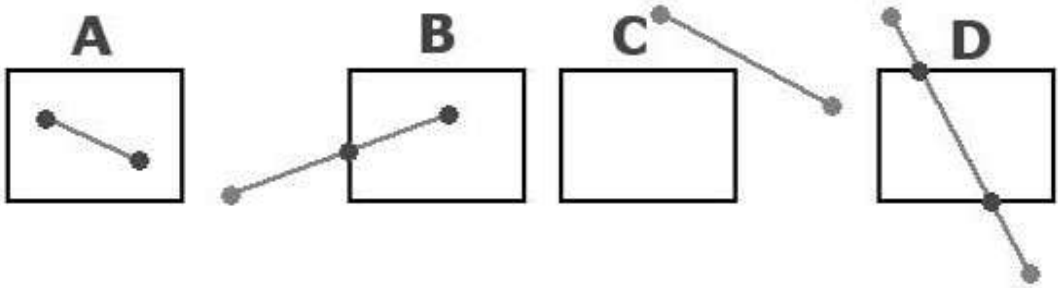


Figure.12.2: Line clipping against a rectangular clip window coordinates.

When drawing a line, if one endpoint of the line is outside the screen, and the other inside, you have to clip the line so that only the part of it that's inside the screen remains. Even if both endpoints are outside the screen, it's still possible that

a part of the line should be visible. The clipping algorithm needs to find new endpoints of the lines that are inside or on the edges of the screen. Here are a few cases, where the black rectangle represents the screen, in red are the old endpoints, and in blue the ones after clipping:



- Case A: both endpoints are inside the screen, no clipping needed.
- Case B: one endpoint outside the screen, that one had to be clipped
- Case C: both endpoints are outside the screen, and no part of the line is visible, don't draw it at all.
- Case D: both endpoints are outside the screen, and part of the line is visible, clip both endpoints and draws it.

There are many different cases, each endpoint can be inside the screen, left of it, right of it, above, below, etc. The Cohen Sutherland Clipping Algorithm can recognize these cases quite efficiently and do the clipping.

Cohen-Sutherland Line Clipping Algorithm

This is mostly used most popular and the oldest line clipping algorithm. It uses the concept of initial testing which speed up the procedure of clipping. The algorithm divides the space (window) area in nine regions. The center region is the

screen (window), and the other eight regions are different sides outside the window. Every line endpoint in a picture is assigned a four digit binary code, called **region code**. The codes are chosen as follows:

- If the region is above the screen, the first bit is 1
- If the region is below the screen, the second bit is 1
- If the region is to the right of the screen, the third bit is 1
- If the region is to the left of the screen, the fourth bit is 1

Obviously, an area can't be to the left and the right at the same time, or above and below it at the same time, so the third and fourth bit can't be 1 together, and the first and second bit can't be 1 together. The screen itself has all 4 bits set to 0.

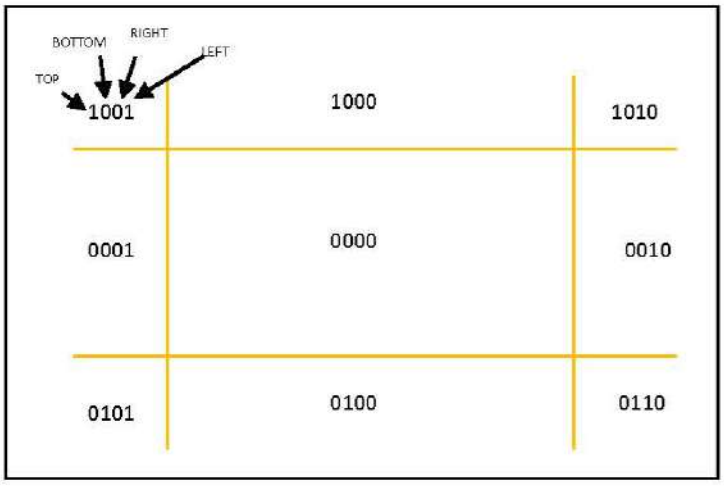


Figure.12.3: Bit position according to window.

A value of 1 in any bit position indicates that the point is in that relative position; otherwise, the bit position is set to 0. Bit values in the region code are determined by comparing endpoint coordinates values (x, y) to the clip window boundaries. Bit 1 is set to 1 if $x < xw_L$. The other three bit values can be determined using similar comparisons. Once we have decided region codes for all

line endpoints, we can quickly determine which lines are completely inside the clip window and which are clearly outside.

If line segment is completely within the window, then both the end points will get region code 0000 and we trivially accept these lines. The line segment that have a 1 in the same bit position in the region codes for each endpoint are completely outside the clipping window, and we trivially reject these lines. The rule to test the given line is completely inside or outside, is performing logical AND operation on region code of two endpoints of the line segment. If the result is not 0000, the line is completely outside the clipping window as shown in Figure 12.3.

Algorithm

Step 1	– Assign a region code for each line endpoints.
Step 2	– If both endpoints have a region code 0000 then accept this line.
Step 3	– Else, perform the logical AND operation for both region codes.
Step 3.1	– If the result is not 0000, then reject the line.
Step 3.2	– Else we need clipping.
Step 3.2.1	– Choose an endpoint of the line that is outside the window.
Step 3.2.2	– Find the intersection point at the window boundary (base on region code).
Step 3.2.3	– Replace endpoint with the intersection point and update the region code.
Step 3.2.4	– Repeat step 2 until we find a clipped line either trivially accepted or trivially rejected.
Step 4	– Repeat step 1 for other lines.

Liang-Barsky Line Clipping Algorithm

This line clipping algorithm was introduced in the year 1984 which is an extension of Cyrus and Beck work. It is faster algorithm as compared to Cohen Sutherland algorithm. This algorithm is based on the parametric form of a line.

The parametric line equation is as follow:

$$x = x1 + u(x2 - x1) = x1 + \Delta x$$

$$y = y1 + u(y2 - y1) = y1 + \Delta y$$

Where the value of u lies between 0 and 1 i.e. $0 \leq u \leq 1$

It uses the condition of point clipping algorithm. Let us assume the two corner vertices, left lower and right upper of window: (xw_L, yw_L) and (xw_U, yw_U) respectively. Let assume (x1, y1) and (x2, y2) be the coordinates of two end points of line. The according to the condition of point clipping algorithm:

$$xw_L \leq x \leq xw_U$$

$$yw_L \leq y \leq yw_U$$

Using parametric equation, new condition of point clipping become

$$xw_L \leq x1 + u\,\Delta x \leq xw_U$$

$$yw_L \leq y1 + u\,\Delta y \leq yw_U$$

Inequalities of above equation can be rewritten as

$$-xw_L + x1 \geq -u\,\Delta x$$
$$xw_U - x1 \geq u\,\Delta x$$

$$-yw_L + y1 \geq -u\,\Delta y$$

$$yw_U - y1 \geq u\,\Delta y$$

So, the general form of these equation is

$$up_i \leq q_i \text{ where } i = 1, 2, 3, 4.$$

Where i indicates sides of the window boundaries, 1 left, 2 right, 3 bottom and 4 top boundary. We use these in our algorithm which is as follow.

Algorithm

Step 1 – Read two end points of line P1 (x1, y1) and P2 (x2, y2).

Step 2 – Read two corner vertices, left lower and right upper of window: (xwL, ywL) and (xwU, ywU).

Step 3 – Calculate values of parameters pi and qi for I =1, 2, 3, 4 such that

p1 = –Δx p2 = Δx

q1 = x1 – xwL q2 = xwU – xi

q1 = – Δy q2 = Δy

q3 = y1 – ywL q4 = ywU – y1

Step 4 – Check the value of pi.

Step 4.1 – If pi = 0 then, Line is parallel to ith boundary.

Step 4.2 – If qi < 0 then, Line is completely outside the boundary. Therefore, discard line segment and Go to Step 10.

Step 4.3 – Else Check line is horizontal or vertical and accordingly check line end points with corresponding boundaries. If line endpoints lie within the bounded area then use them to draw line. Otherwise use boundary coordinates to draw line. Go to Step 10.

Step 5 – Initialize t1 = 0 and t2 = t1.

Step 6 – Calculate values for qi/pi for i= 1, 2, 3, 4.

Step 7 – Select values of qi/pi where pi<0 and assign maximum out of them as t1.

Step 8– Select values of qi/pi where pi>0 and assign maximum out of them as t2.

Step 9 – If (t1 < t2) Calculate endpoints of clipped line:

xx1 = x1 + t1 Δx

xx2 = x1 + t2 Δx

yy1 = y1 + t1 Δy

yy2 = y1 + t2 Δy

Draw line (xx1, yy1, xx2, yy2).

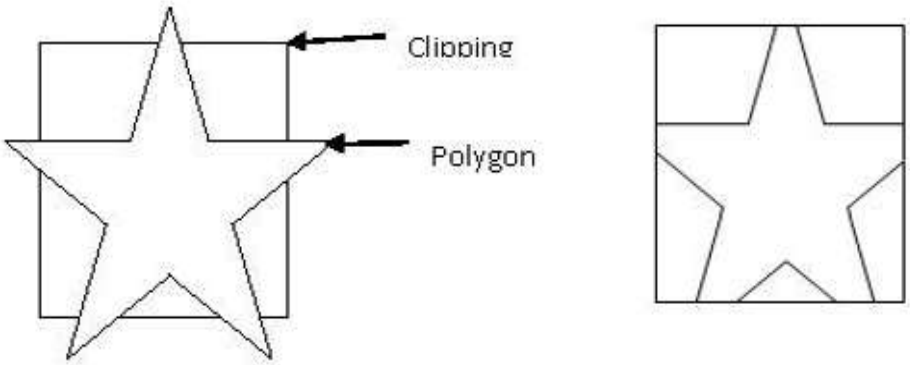
Step 10 – Stop.

12.4 Polygon Clipping

A polygon is generally stored as a collection of vertices. Any clipping algorithm takes one collection, and outputs a new collection. A clipped polygon, after all, is also a polygon. Notice that the clipped polygon often will have more vertices than the unclipped one, but it can also have the same number, or less. If the unclipped polygon lies completely outside the clipping boundary, the clipped polygon even has zero vertices.

A polygon can also be clipped by specifying the clipping window. An algorithm that clips a polygon must deal with many different cases. The case is particularly noteworthy in that the concave polygon is clipped into two separate polygons. All in all, the task of clipping seems rather complex. Each edge of the polygon must be tested against each edge of the clip rectangle; new edges must be added, and existing edges must be discarded, retained, or divided. Multiple polygons may result from clipping a single polygon. We need an organized way to deal with all these cases.

The following example illustrates a simple case of polygon clipping.



Before Clipping After Clipping

Figure.12.4: A polygon clipping against a rectangular clip window coordinates.

Sutherland-Hodgman Polygon Clipping Algorithm

There are several well-known polygon clipping algorithms, each having its strengths and weaknesses. The oldest one is called the Sutherland-Hodgman algorithm. It uses a divide-and-conquer strategy to solve the problem. First, it clips the polygon against the left edge of the polygon window to get new vertices of the polygon. These new vertices are used to clip the polygon against right edge, top edge, bottom edge, of the clipping window as shown in the following figure. (Of course, it also works in another order also.)

While processing an edge of a polygon with clipping window, an intersection point is found if edge is not completely inside clipping window and the a partial edge from the intersection point to the outside edge is clipped. The following figures show left, right, top and bottom edge clippings –

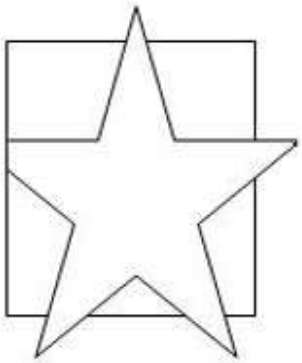


Figure.12.5: Clipping left edge

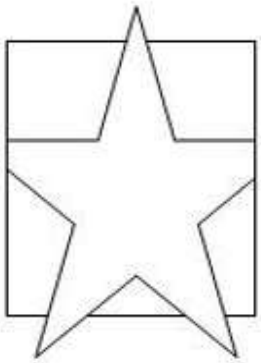


Figure.12.6: Clipping right edge

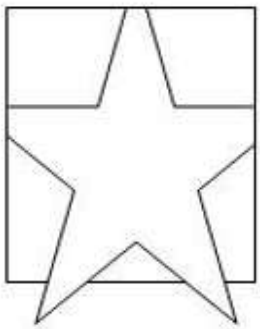


Figure.12.7: Clipping top edge

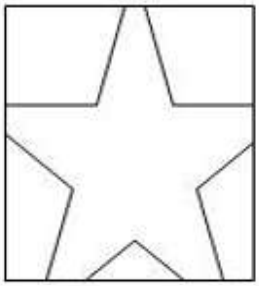


Figure.12.8: Clipping bottom edge

Algorithm

Step 1 – Read coordinates of all vertices of polygon.

Step 2 – Read coordinates of clipping window.

Step 3 – Consider left edge of window.

Step 4 – Compare vertices of each edge of polygon individually with clipping plane.

Step 5 – Save resulting intersection and vertices in new list of vertices according to 4 possible relationships between edge and clipping boundary:

Step 5.1 – If 1st vertex of edge is outside window boundary and second vertex of edge is inside, then, intersection point of polygon edge with window boundary and second vertex are added to output vertex list.

Step 5.2 – If both vertices of edge are inside window boundary, only second vertex is added to output vertex list.

Step 5.3 – If 1st vertex of edge is inside window boundary and second vertex of edge is outside, then, only edge intersection with window boundary is added to output vertex list.

Step 5.4 – If both vertices of edge are outside window boundary, nothing is added to output vertex list.

Step 6 – Repeat Steps 4 to 5 for remaining edges of clipping window. Each time successively pass the resultant list of vertices to process next edge of clipping window.

12.5 Text Clipping

Various techniques are used to provide text clipping in a computer graphics. It depends on the methods used to generate characters and the requirements of a particular application. There are three methods for text clipping which are listed below –

- All or none string clipping
- All or none character clipping
- Text clipping

The following figures shows all or none string clipping –

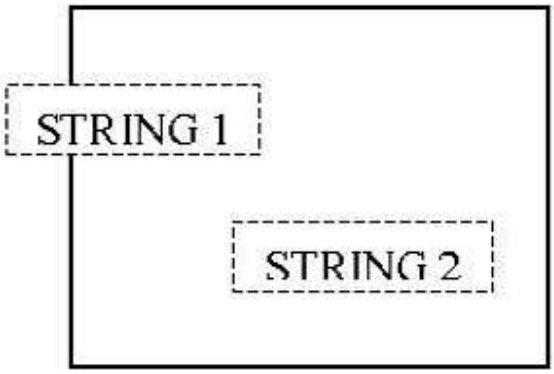


Figure.12.9: Before Clipping

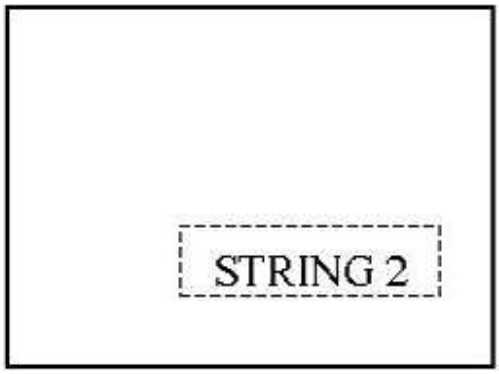


Figure.12.10: After Clipping

In all or none string clipping method, either we keep the entire string or we reject entire string based on the clipping window. As shown in the above figures, STRING2 is entirely inside the clipping window so we keep it and STRING1 being only partially inside the window, we reject. The following figures shows all or none character clipping –

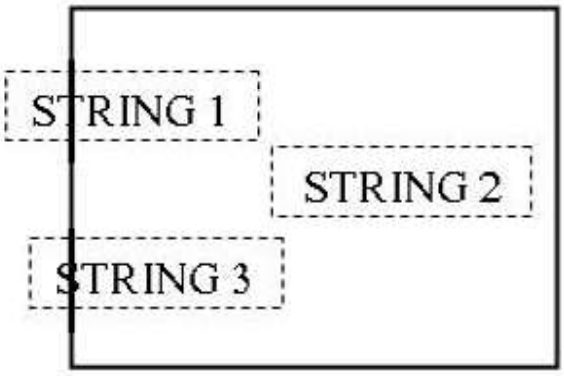


Figure.12.11: Before Clipping

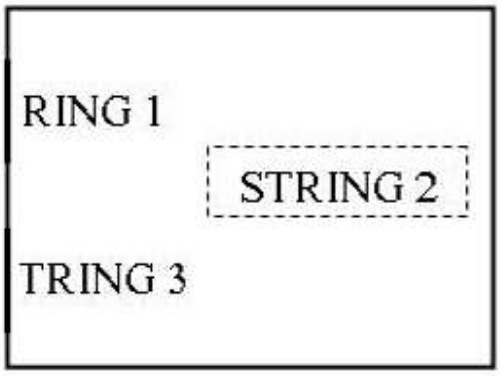


Figure.12.12: After Clipping

This clipping method is based on characters rather than entire string. In all or none character clipping method if the string is entirely inside the clipping window, then we keep it. If it is partially outside the window, then –

- We reject only the portion of the string being outside the window.
- If the character is on the boundary of the clipping window, then we discard that entire character and keep the rest string as shown in Figure 12.12.

The following figures show text clipping –

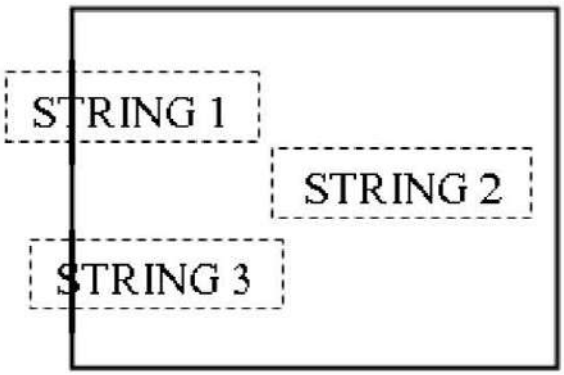


Figure.12.13: Before Clipping

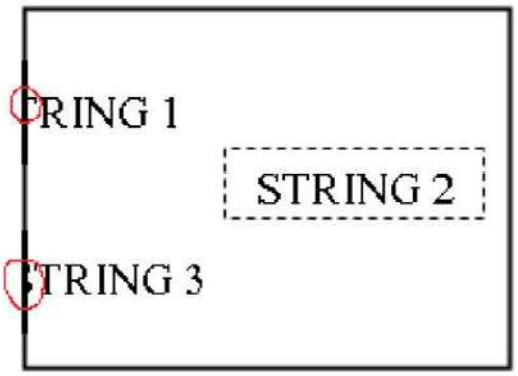


Figure.12.14: After Clipping

This clipping method is based on characters rather than the entire string. In **text clipping method** if the string is entirely inside the clipping window, then we keep it. If it is partially outside the window, then

- We reject only the portion of string being outside the window.
- If the character is on the boundary of the clipping window, then we discard only that portion of character that is outside of the clipping window as shown in Figure 12.14.

12.6 Self Learning Exercise

Q.1 If XL, XR, YB, YT represent the four parameters of x-left, x-right, y-bottom and y-top of the clipping window and (x, y) is a point inside the window then……..

(A) XL≤ x ≤ XR and YB ≤ y ≤ YT

(B) XL≤ x ≤ XR and YB ≥ y ≥ YT

(C) XL ≥ x ≥ XR and YB ≤ y ≤ YT

(D) XL ≥ x ≥ XR and YB ≥ y ≥ YT

Q.2 If (x, y) is a point inside the clipping window then its code according to the Cohen-Sutherland algorithm is……

(A) 0001 (B) 0000

(C) 1000 (D) 1111

Q.3 What is line clipping?

12.7 Summary

In this unit, we have seen how we can display various types of shapes on the output devices, all parts of a picture outside the window (or viewport) are clipped off by using various clipping algorithms. The clipping region is commonly referred to as the clipping window or as the clipping rectangle when our output devices are standard rectangle. Many algorithms have been developed for clipping images against the clip window.

For Line-Clipping we have many algorithms includes the Cohen-Sutherland and Liang-Barsky method.

Polygon clipping algorithms include the Sutherland-Hodgemen method and Liang-Barsky method. Curve equations are used to calculate the intersection point between curved objects boundaries and clipping window. In text clipping method clip a string if any part of the string is outside any window boundary, it is the fastest method to clip a text. Other than this we can use another approach of all-or-none with the individual character in a string.

12.8 Glossary

Viewport: An area on display device to which a window is mapped.

Window: An area of a world coordinate scene that has been selected for display.

12.9 Answers to Self-Learning Exercise

Ans.1: A **Ans.2:** B

12.10 Exercise

Q.1 In Cohen-Sutherland line clipping algorithm, if the bitwise logical AND of the region codes is 0000, then line is

(A) Visible (B) Not visible
(C) A candidate for clipping (D) None of these

Q.2 If two bits are zeros and two bits are ones in a code of a sub region in Cohen-Sutherland line clipping algorithm the sub region is .....

(A) Corner region (B) Middle region
(C) Central region (D) None of these

Q.3 In the Cohen-Sutherland line clipping algorithm, if the codes of the two points P and Q are 0000 and 0000 then the line segment joining the points P and Q will be ..... the clipping window

(A) Totally outside (B) Partially outside
(C) Totally inside (D) None

Q.4 Compare Liang-Barsky algorithm with Cohen-Sutherland algorithm.

Q.5 How can polygons be clipped. Explain Sutherland-Hodgeman polygons clipping algorithm.

Q.6 Consider a clipping window A (0, 0), B (30, 0), C (30, 20), D (0, 20). Using the out codes of the end points of the line X (–10, 30) and Y (35, 8). Show that the line is partially visible.

Q.7 Explain Cohen-Sutherland line clipping algorithm with region code details.

12.11 Answers to Exercise

Ans.1: A **Ans.2:** A

Ans.3: C

References and Suggested Readings

1. Computer Graphics by Donald Hearn and M. Pauline Baker; Pearson Education, Seventh Edition 2005
2. Computer Graphics by Apurva A. Desai; PHI Learning, Third Edition 2012
3. Computer Graphics principles & practice in C by James D. Foley, Steven K. Feiner, Andries van Dam and F. Hughes John; Pearson Education, Second Edition 2003
4. Computer Graphics A Programming Approach by Harrington and Steven; McGraw Hill.

UNIT-13
Visible surface detection

Structure of the Unit

13.0 Objective

In this chapter, we shall focus on the following topics

- Various visible surface detection methods
- Object space algorithm
- Image space algorithm

13.1 Introduction

In reality when we generate a realistic object or scene in 2-dimensional view. The viewer can see only the front surfaces and edges. So, here our major consideration is to identifying those parts of an image which are visible from a chosen viewing direction. The surfaces and edges which are at the back of object are not seen. When we view an image containing non-transparent objects and surfaces, then we cannot see those objects or surfaces, which are behind the objects or surfaces closer to eyes.

We must remove these hidden surfaces to get a realistic screen image. The identification and removal of these surfaces is called **Hidden-surface problem**. In this chapter, we will learn how to find out these Hidden surfaces and also algorithms to remove or find such surface. The various algorithms are referred to as **Visible-surface detection methods** and also referred to as **Hidden-surface elimination or removing algorithms**

13.2 Visible surface detection methods

Algorithms used to detect visible surface are broadly categorized according whether algorithms deals with object definitions directly or with their projected images. Based on the methodology used in the algorithm, the algorithms are classified into two categories:

i. Object space algorithm
ii. Image space algorithm

An object space method, use comparison technique, it compare object and parts of objects to each other within the image definition to find which surface in the image is visible. In an image space method, the visibility of an object is determined as point by point at each pixel position on the projection plane. Most of the algorithms use image space method.

The following is the list of some algorithms, which are used in graphics for the finding of hidden surfaces.

- Back Face Removal Algorithm
- Depth Buffer Algorithm
- Scan-Line Algorithm
- Depth Sorting Algorithm
- Area Subdivision Algorithm
- Binary Space Partition Tree Algorithm
- Ray Casting Algorithm

We will discuss these algorithms in the following sections.

13.3 Back face culling (removal)

In a solid object, there are surfaces which are facing the viewer (front faces) and there are surfaces which are opposite to the viewer (back faces).

These back faces contribute to approximately half of the total number of surfaces. Since we cannot see these surfaces anyway, to save processing time, we can remove them before the clipping process with a simple test.

A fast and simple object space method for identifying the back faces of a polygon is based on the test known as "inside-outside" test. A point (x, y, z) is inside a polygon surface with the plane parameters A, B, C and D if

$$Ax + By + Cz + D < 0$$

There is a normal vector **N** to a polygon surface. If this vector is pointing in the direction of the center of projection, it is a front face and can be seen by the viewer. If it is pointing away from the center of projection, it is a back face and cannot be seen by the viewer. In general if **V** is a vector in the viewing direction then this polygon is back face if

$$\mathbf{V}.\mathbf{N} > 0$$

The test is very simple, if the z component of the normal vector is positive, then, it is a back face. If the z component of the vector is negative, it is a front face.

13.4 Depth buffer algorithm

The Depth buffer or Z-buffer algorithm, developed by Catmull (1975). It is an algorithm that operates in image or screen space. This is known as Z-buffer because the depth of object is usually measured from the view plane along the z-axis of viewing system. The basic idea is to test the Z-depth of each surface to determine the closest (visible) surface. Each surface of an image is processed separately, one point at a time across the surface. The depth values for a pixel are compared and the closest (smallest z) surface determines the color to be displayed in the frame buffer.

It is applied very efficiently on surfaces of polygon. Surfaces can be processed in any order. The method is usually applied to scene containing only polygon surface. It is easy and quick to find depth of polygon surface.

Here in this algorithm, two buffer areas are used. The two buffers are named as **Frame buffer (Refresh buffer)** and **Depth buffer.**

The **Depth buffer** is used to store depth values for (x, y) position, as surfaces are processed ($0 \leq$ depth ≤ 1).

The **Frame buffer** is used to store the intensity value of color value at each position (x, y). Each surface listed in the polygon tables is then processed, one scan line at a time calculating the depth at each pixel position (x, y). The steps of algorithm to find the visible surface is as follows.

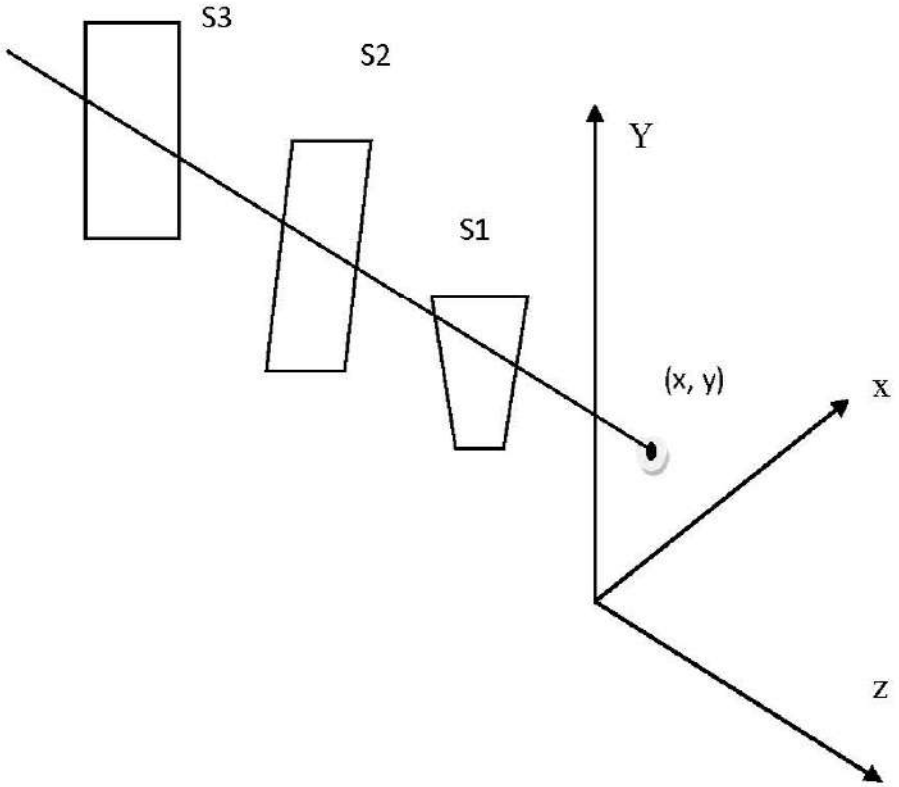


Figure. 13.1: Position (x, y) has three surface and surface S1 has the smallest depth from the view plane. So, S1 is visible.

Algorithm

Step-1 – Set the buffer values – Depthbuffer (x, y) = 0

Framebuffer (x, y) = background color

Step-2 – Process each polygon (One at a time) for each pixel position.

For each projected (x, y) pixel position of a polygon, calculate depth z.

If Z > depthbuffer (x, y)

Compute surface color,

set depthbuffer (x, y) = z,

framebuffer (x, y) = surfacecolor (x, y)

Depth of the surfaces are calculated from the equation of each surface. Remember the equation of a plane, is

$$Ax + By + Cz + D = 0$$

Therefore,

$$z = -(Ax + By + D) / C$$

There are few advantages of using this algorithm are, it is simple to process, and it reduces the speed problem if implemented in hardware. The limitation of the algorithm is that, it requires large memory to store values of two buffers. As it process all the pixels in the image, it is time consuming process also. The next problem with this algorithm is that it can work only with the opaque surfaces not with the transparent surface.

A-BUFFER ALGORITHM

The A-buffer method is an extension of the depth-buffer method. The A-buffer method is a visibility detection method developed at Lucas film Studios for the rendering system Renders Everything You Ever Saw (REYES).

This is an algorithm, which takes care of not only the opaque surfaces but also considers transparent surfaces. Thus, this algorithm shows the true distance of each and every pixel. This is an algorithm which falls under the image space method. The A-buffer expands on the depth buffer method to allow transparencies. The key data structure in the A-buffer is the accumulation buffer.

In A-buffer algorithm, a link list is attached with each pixel position and this link list carries the intensity information of each surface associated with that position as show in Figure 13.2.

Each position in the A-buffer has two fields –

- **Depth field** – It stores a positive or negative real number
- **Intensity field** – It stores surface-intensity information or a pointer value

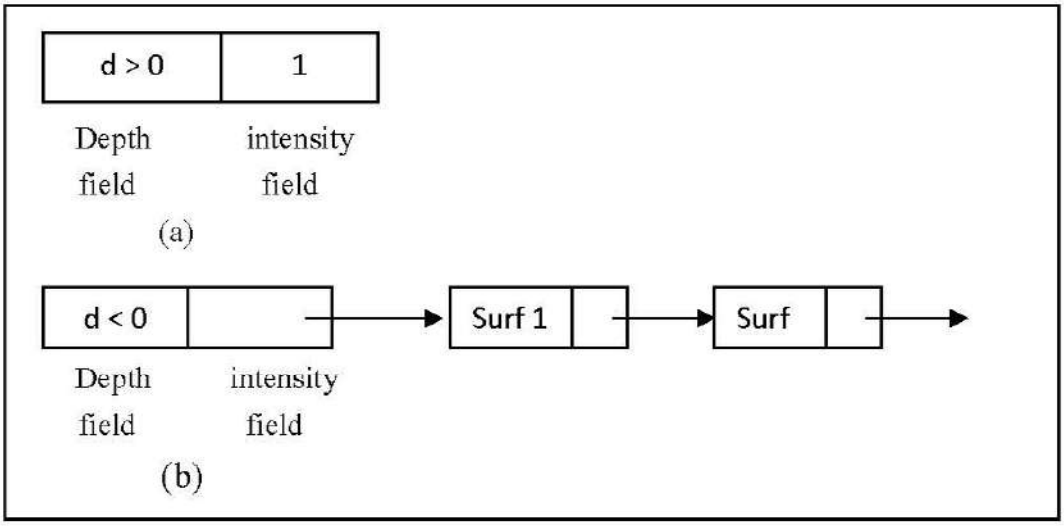


Figure. 13.2: Organization of an A buffer pixel position: (a) single surface (b) multiple surface overlap.

If depth >= 0, the number stored at that position is the depth of a single surface overlapping the corresponding pixel area. The intensity field then stores the RGB components of the surface color at that point and the percent of pixel coverage.

If depth < 0, it indicates multiple-surface contributions to the pixel intensity. The intensity field then stores a pointer to a linked list of surface data. The surface buffer in the A-buffer includes –

- RGB intensity components

- Opacity Parameter
- Depth
- Percent of area coverage
- Surface identifier
- Pointer to next surface

The algorithm proceeds just like the depth buffer algorithm. The depth and opacity values are used to determine the final color of a pixel.

13.5 Self Learning Exercise

Q.1 The surfaces that is blocked or hidden from view in a 3D scene are known as

(A) Hidden surface (B) Frame buffer

(C) Quad tree (D) Area buffer

Q.2 The name of a visible surface detection algorithm is _____.

(A) Back face detection (B) Back face removal

(C) Ray tracing (D) Area tracing

Q.3 What is Hidden Surface Problem?

13.6 Scan line algorithm

It is an image-space method to identify visible surface. It is an extension of the scan-line algorithm for filling polygon. In this method, as each scan line is processed, all polygon surfaces intersecting that line are examined to determine which are visible.

In order to require one scan-line of depth values, we must group and process all polygons intersecting a given scan-line at the same time before processing the next scan-line. Two important tables, **edge table** and **polygon table**, are maintained for this.

Edge Table − It contains coordinate endpoints of each line in the scene, the inverse slope of each line, and pointers into the polygon table to connect edges to surfaces.

Polygon Table – It contains the plane coefficients, surface material properties, other surface data, and may be pointers to the edge table.

Figure 13.3 illustrate the scan line algorithm for locating visible portion of surfaces for pixel positions along the line. Here we define a flag for each surface that is set on or off to indicate whether a position along a scan line is inside or outside of the surface.

Any number of overlapping polygon surfaces can be processed with this scan line algorithm. Flags for the surfaces are set to indicate whether a position is inside or outside and depth calculation are performed when surfaces overlap.

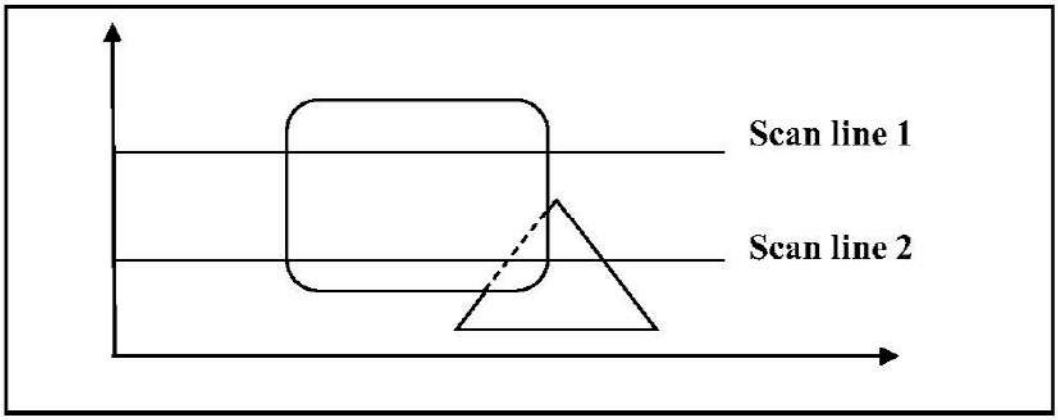


Figure.13.3: Scan lines on two surfaces. Dashed line indicate the boundaries of hidden surface.

Algorithm

Step-1 – For each scan line do

```
Begin
        For each pixel (x, y) along the scan line do
        Begin
        z_buffer(x, y) = 0
        Image_buffer(x, y) = background_color
        End
```

Step-2 – For each polygon in the scene do

```
Begin
    For each pixel (x, y) along the scan line that is covered by the
    polygon do
    Begin
        2a. Compute the depth or z of the polygon at pixel location (x,
        y).
        2b. If z < z_buffer(x, y) then
                Set z_buffer(x, y) = z
                Set Image_buffer(x, y) = polygon's colour
    End
End
```

The basic idea of this method is simple. When there are only few objects in the scene, this method can be very fast. However, as the number of objects increases, the sorting process can become very complex and time consuming.

Depth sorting method uses both image space and object-space operations. The depth-sorting method performs two basic functions –

- First, the surfaces are sorted in order of decreasing depth.
- Second, the surfaces are scan-converted in order, starting with the surface of greatest depth.

13.7 Depth sorting algorithm

The scan conversion of the polygon surfaces is performed in image space. This method for solving the hidden-surface problem is often referred to as the **Painter's algorithm**. Hewells introduced the Painter's algorithm in the year 1972. It is

known as Painter's algorithm as it work like a painter creating an oil painting. When an artist paints, he starts with an empty canvas, and first he would create the background layer or the painting. Then after that background layer, he starts creating another layer of objects one-by-one by this way he would be completing this painting. In Figure 13.4 see the step by step process.

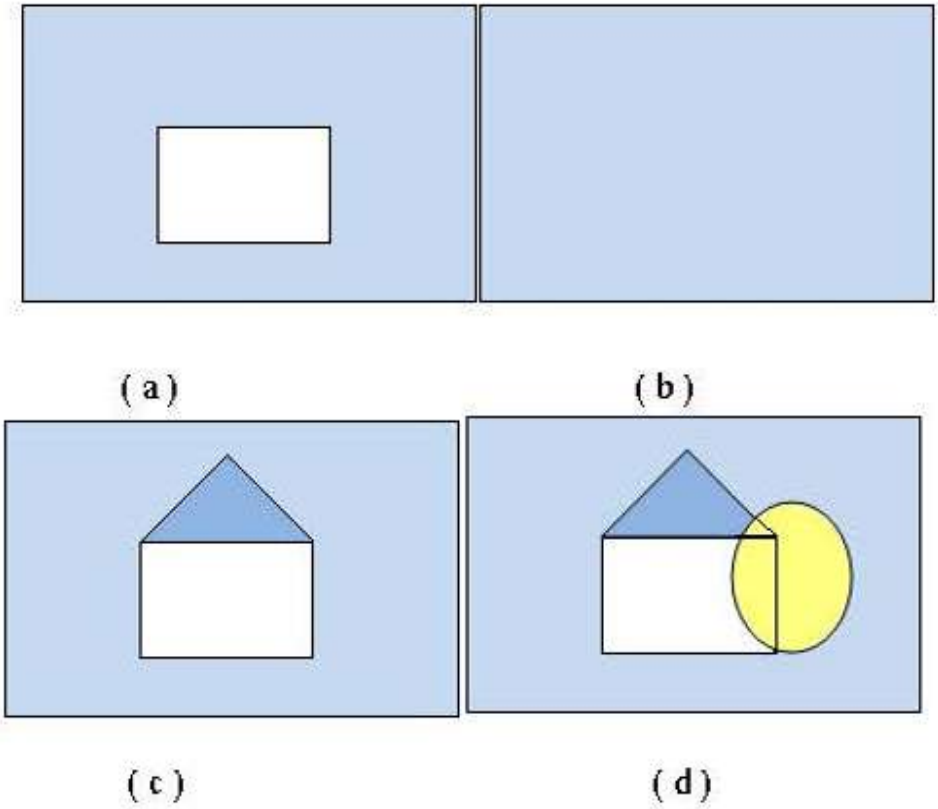


Figure. 13.4: Step by step series of process for Painter's algorithm.

The algorithm begins by sorting by depth. For example, the initial "depth" estimate of a polygon may be taken to be the closest z value of any vertex of the polygon. The frame buffer is first painted with the background color. Then the farthest polygon is entered in the frame buffer. The pixel information of the background associated with the farthest polygon will be replaced with that of the

farthest polygon. This process will be repeated for all the surfaces one by one until the nearest surface is painted in the frame buffer.

Algorithm

Step-1 – Sort all the objects in the order of their depth coordinate from the largest to the smallest. Smallest means the surface nearest to the view plan.

Step-2 – Resolve where z overlaps with the other objects. That is, who's the comparison of polygons takes place here.

Step-3 – Scan convert the surfaces from the largest z to the smallest z. That is, the innermost surface is placed first and on the surfaces which are nearer to the viewer will be placed one by one.

13.8 Area Subdivision algorithm

Warnock introduced this algorithm in the year 1969. This is a very primitive and basic algorithm and it is also a very length algorithm. In this technique to find hidden surface we use image space method, but object space operations can be used to accomplish depth ordering of surfaces. The **area subdivision algorithm** takes advantages of area coherence in an image by locating those view areas that represent part of a single surface. Divide the total viewing area into smaller and smaller rectangles until each small area is the projection of part of a single visible surface or no surface at all.

This algorithm divides the image into some viewports, and then each of the viewport is checked for the surfaces falling in the viewport. If the number of surface is 0 or 1 the visibility of the surface for that viewport is trivial. This means for the considered viewport, there is only one surface which is visible.

Continue this process until the subdivisions are easily analyzed as belonging to a single surface or until they are reduced to the size of a single pixel. An easy way to do this is to successively divide the area into four equal parts at each step as shown in Figure 13.5.

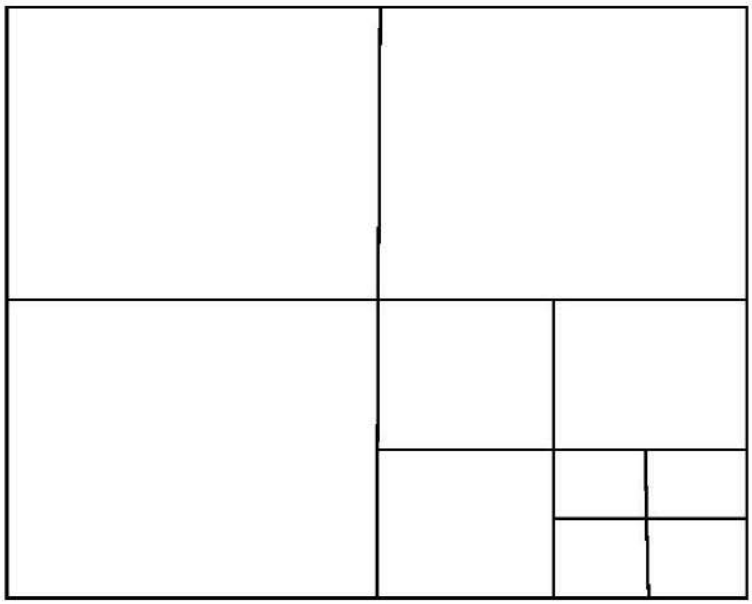


Figure. 13.5: Dividing a square are into equal sized quadrants at each step.

There are four possible relationships that a surface can have with a specified area boundary.

a) **Surrounding surface** – One that completely encloses the area.
b) **Overlapping surface** – One that is partly inside and partly outside the area.
c) **Inside surface** – One that is completely inside the area.
d) **Outside surface** – One that is completely outside the area.

The tests for determining surface visibility within an area can be stated in terms of these four classifications. No further subdivisions of a specified area are needed if one of the following conditions is true –

- All surfaces are outside surfaces with respect to the area.

- Only one inside, overlapping or surrounding surface is in the area.
- A surrounding surface obscures all other surfaces within the area boundaries.

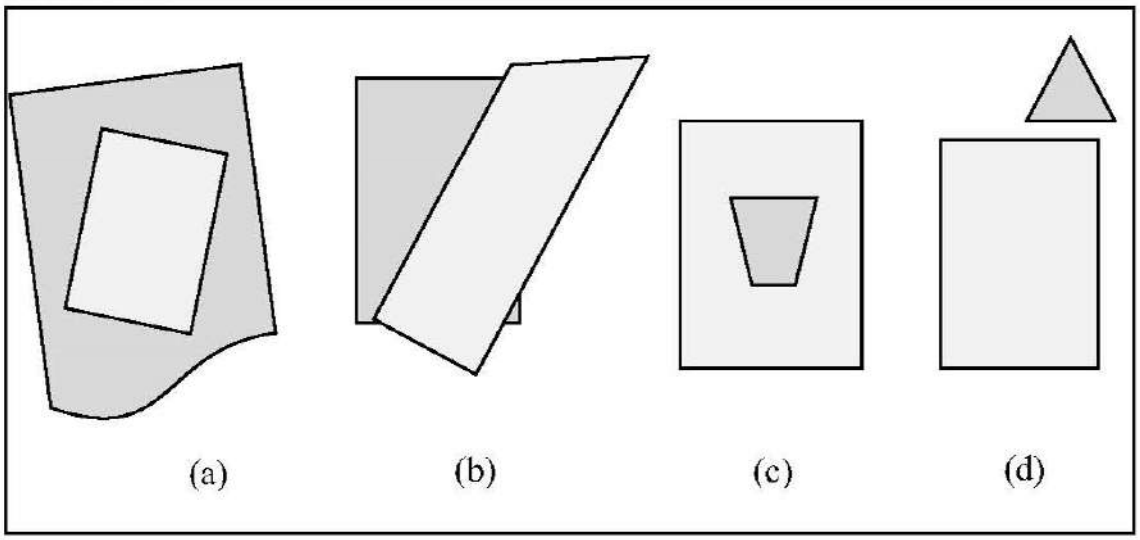


Figure. 13.6: Possible relationship between polygon and rectangular area.

13.9 Self Learning Exercise

Q.4 All the hidden surface algorithms employee image space approach except ___.

(A) Back face removal (B) Depth buffer method

(C) Scan line method (D) Depth sort method

Q.5 Consider the following statements about Hidden Surface removal algorithms.

(i) A Depth Buffer algorithm is not an object-space hidden surface removal algorithm.

(ii) Partially hidden surfaces cannot be determined by the Back-Face Culling algorithm.

(iii) In the Painter's Algorithm, objects are ordered back-to-front and then rendered in that order.

Which of the above statement(s) is/are true?

(A) All (B) Only (i) and (ii)

(C) Only (ii) and (iii) (D) Only (i) and (iii)

Q.6 _____ surface algorithm is based on perspective depth.

(A) Depth comparison (B) Z-buffer or depth-buffer algorithm

(C) Subdivision method (D) back-face removal

13.10 Binary space partition tree algorithm

A binary space partitioning (BSP) tree is an efficient method for determining object visibility by painting surfaces onto the screen from back to front, Kadem and Fuchs introduced this algorithm, which is based on the construction of binary tree. The BSP tree is particularly useful when the view reference point changes, but the object in a scene are at fixed position.

This algorithm is based on the principle of sorting of objects as in quick sort algorithm. Using this algorithm, the surfaces are sorted from back to front. Applying a BSP tree to visibility testing involves identifying surfaces that are inside and outside the partitioning plane at each step of the space subdivision, relative to the viewing direction.

In this algorithm, first an object is considered and then a partition plane is considered in relation to the object. Now the objects in scene will be divided into two parts. Part one contains the objects that are in front of the given object and the second part contain the objects which are in the back of the given object. This will be done recursively. This process will create a binary tree representation.

In this tree, the objects are represented as terminal nodes, with front objects as left branches and back objects as right branches. Figure 13.7 show the process of deriving binary tree from given scene.

VIEWPORT BINARY TREE

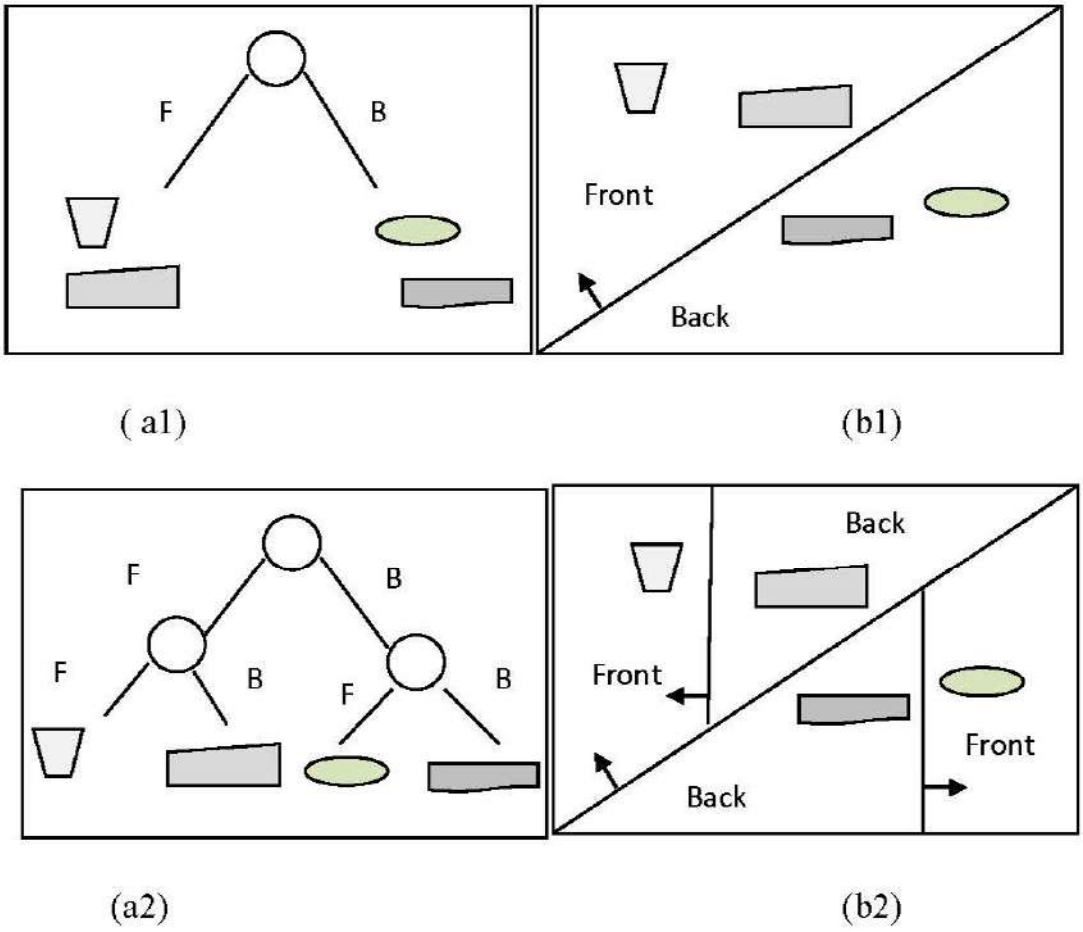


Figure.13.7: A region of space is partitioned in (a1 and a2); BSP tree representation in (b1 and b2). [F = Front and B = Back]

Here, we use concept of painter's algorithm. Now the sorted surfaces will be painted one by one from back to front. The surface that is nearest and visible will be painted last.

Algorithm

Step-1 – Spit the scene along the selected plane with reference to a surface of the scene.
Step-2 – Classify all the surfaces into front and back half space.
Step-3 – Iterate the front half with same steps.
Step-4 – Iterate the back half with same steps.

The biggest advantage of using this algorithm is that, even if we change the view reference points, this algorithm will work efficiently. If any splitting plane intersects any surface, then that surface is divided into two parts and will be classified into the respective halves. Fast hardware implementation for construction and processing BSP trees are used in some systems.

13.11 Ray casting

The intensity of a pixel in an image is due to a ray of light, having been reflected from some objects in the scene, pierced through the center of the pixel. So, visibility of surfaces can be determined by tracing a ray of light from the center of projection (viewer's eye) to objects in the scene.

If we consider the line of sight from a pixel position on the view plane through a scene as shown in Figure 13.8, we can determine which object in the scene intersect this line. In this method, a pixel is taken and from that pixel, the distances of surfaces that are associated with the pixel are calculated.

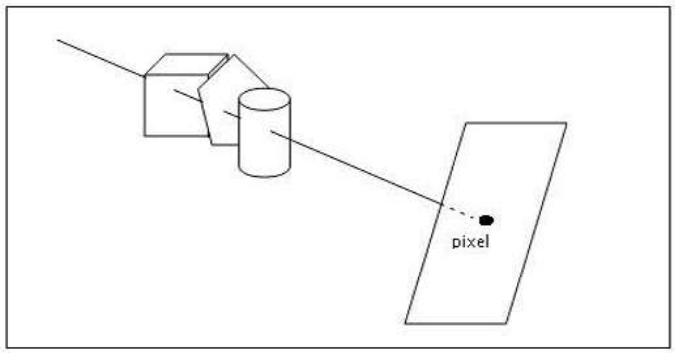


Figure. 13.8: A ray along the line of sight passing through surfaces.

For calculating depth a ray of light or line of sight is taken that is originating from the pixel and passes through all the objects that come in the way of the ray before reaching the viewing plane. Since there are an infinite number of light rays in a scene and we are interested only in those rays that pass through pixel

position, we can trace the light ray paths backward from the pixels through the scene.

We can think of ray casting as a variation of the depth buffer method. In the z-buffer algorithm one surface at a time is taken and its depth calculated from the pixel. When the intersection points between the ray of light and the surfaces are sorted in the increasing order of z values, they are place from the back to front in order. Since, we have assumed that the ray is originating from the pixel. We do the same for each and every pixel in the scene.

The ray casting approach is an effective visibility detection method for scenes with curved surfaces, particularly spheres. For Speeding up the intersection calculation in ray casting following methods can be used.

- Bounding Volume Approach
- Using Hierarchies
- Space Partitioning Approach

13.12 Summary

Here in this unit, we discussed various methodologies of the visibility-detection to locate the surface or a part of surface which is not visible to the viewer in the image. Various methods are discussed at length, which work efficiently in different situations. For a single convex polyhedron, back-face detection eliminates all hidden surfaces, but in general, back-face detection cannot completely identify all hidden surfaces.

Depth buffer method is fast and simple for identifying visible surface in an image. We can also use these methods for displaying line drawings in three dimension. The effectiveness of a visible surface detection method depends on the characteristics of a particular application. We can combine and implement various visible surface detection methods that we study in this unit to identify hidden surface or a part of hidden surface in various ways.

13.13 Glossary

Projection Plane: Is a type of view in which graphical projections from an object intersect.

Area Coherence: A group of adjacent pixels tend to be covered by the same face.

13.14 Answers to Self-Learning Exercises

Ans.1: A **Ans.2: B**

Ans.4: A **Ans.5: A**

Ans.6: B

13.15 Exercise

Q.1 Z -buffer algorithm are _____.

(A) Simplest algorithm (B) Complex algorithm

(C) Largest algorithm (D) Poor algorithm.

Q.2 The method which is based on the principle of checking the visibility point at each pixel position on the projection plane are called _____.

Q.3 Explain depth buffer algorithm for visible surface detection.

Q.4 What is Ray Casting algorithm for hidden surface removal? Explain mathematically how do we find which planes is visible using Ray Casting algorithm.

Q.5 What are the two spaces in which hidden surface algorithm works? How does sorting and coherence speed up calculation in such algorithm?

Q.6 Explain depth buffer algorithm to display visible surfaces of a given polyhedron. Also explain the any relation in object and storage requirement of the depth buffer?

13.16 Answers to Exercise

Ans.1: A **Ans.2:** Image Space method

References and Suggested Readings

1. Computer Graphics by Donald Hearn and M. Pauline Baker; Pearson Education, Seventh Edition 2005.
2. Computer Graphics by Apurva A. Desai; PHI Learning, Third Edition 2012.
3. Computer Graphics principles & practice in C by James D. Foley, Steven K. Feiner, Andries van Dam and F. Hughes John; Pearson Education, Second Edition 2003.
4. Computer Graphics A Programming Approach by Harrington and Steven; McGraw Hill.

UNIT-14
Illumination Model and Shading

14.0 Objective

In this chapter, we shall focus on the following topics

- Ambient Reflection
- Diffuse Reflection

- Specular Reflection and Phong Model
- Gouraud Shading
- Phong Shading
- Ray Tracing

14.1 Introduction

From Physics we can derive models, called "illumination models," of how light reflects from surfaces and produces what we perceive as color. An illumination model also called a lighting model, is used to calculate the intensity of light that we should see at a given point on the surface of an object.

Illumination - the luminous flux per unit area on an intercepting surface at any given point.

Lighting - the process of computing the luminous intensity reflected from a specified 3-D points.

Shading - the process of assigning colours to pixels.

Realistic images are important for several reasons, but the most important is that people generally find them easier to understand. Hidden surface removal, illumination (lighting) and shading, texture mapping, shadows, transparency, improved synthetic cameras and better models all contribute to increasing visual realism. A long standing goal of computer graphics has been photorealism — the production of synthetic images indistinguishable from photographs of real scenes.

14.2 **Illumination and Shading**

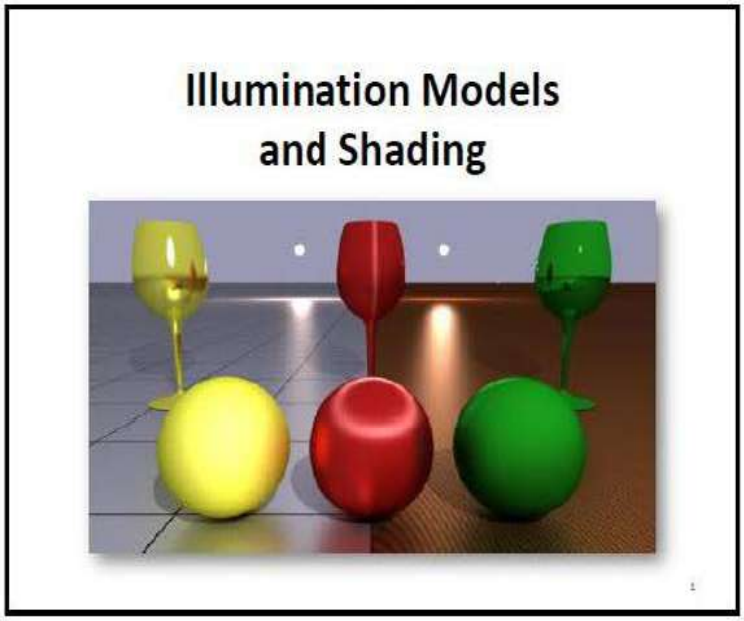


Figure 14.1: Illumination Models and shading

To produce images which look,at least to some objects must be lit and shaded. An illumination or lighting model is a model or technique for determining the color of a surface of an object at a given point. A shading model is a broader framework which determines how an illumination model is used and what parameters it receives. For instance, the illumination model may be used for every pixel covered by an object or just for its vertices. The basis of the calculations for shading objects is the interaction of light and the objects in an environment.

Illumination models in computer graphics are often loosely derived from the physical laws that describe surface light intensities. To minimize intensity calculation, most packages use empirical models based on simplified photometric calculations

14.3 A Simple Illumination Model

A simple illumination model (called a lighting model in OpenGL) can be based on three components: ambient reflection, diffuse reflection and specular reflection. The model we look at is known as the Phong model. The OpenGL lighting model has the components of the Phong model.

14.4 Ambient Reflection

Imagine a uniform intensity background light which illuminates all objects in the scene equally from all directions. This is known as ambient light.

Ignore colour for a while and assume monochrome (grey) ambient light. The ambient reflection can be expressed as –

$$I = L_a k_a$$

Where L_a is the intensity of the ambient light and k_a is the coefficient of ambient reflection of the object's material and ranges between [0, 1].

The object's coefficient of ambient reflection is a material property. The ambient light model attempts to capture the effect of light which results from many object - object reflections without detailed calculations, that is, achieve computational efficiency. Those detailed calculations are performed in some rendering techniques such as radiosity which attempt to achieve higher quality images, but not real-time.

14.5 Diffuse Reflection

Ambient – light reflection is an approximation of global diffuse lighting effects. Objects illuminated by only ambient light have equal intensity everywhere.

If there are light sources in a scene then different objects should have different intensities based on distance and orientation with respect to the light source and the viewer.

A point on a diffuse surface appears equally bright from all viewing positions because it reflects light equally in all directions. That is, its intensity is independent of the position of the viewer.

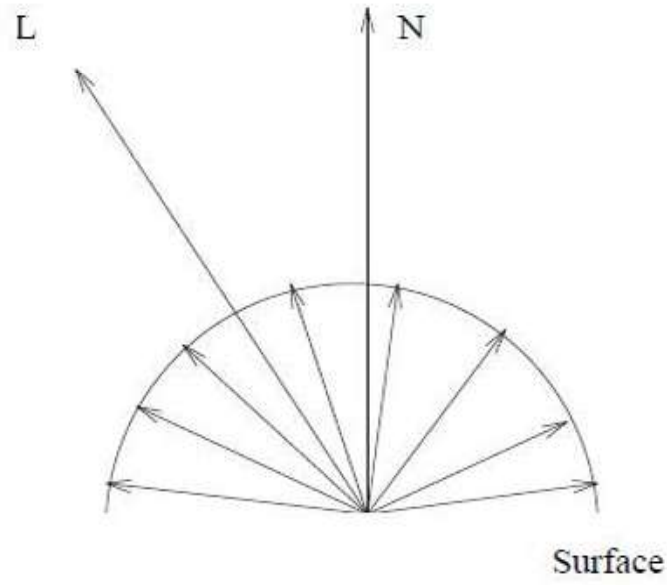


Figure 14.2(a)

Whilst independent of the viewer, the intensity of a point on a diffuse surface does depend on the orientation of the surface with respect to the light source and the distance to the light source.

A simple model for diffuse reflection is Lambertian reflection. Assuming the following geometry

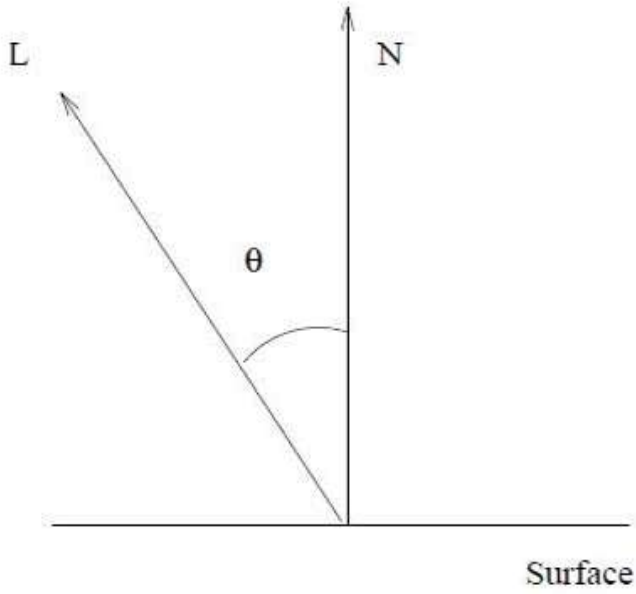


Figure 14.2(b)

Then the intensity of the diffuse reflection from a point light source is

$$I = L_d k_d \cos\theta$$

Where L_d is the intensity of the (point) light source, k_d is the diffuse reflection coefficient of the object's material, and θ is the angle between the normal to the surface and the light source direction vector. If the normal vector to the surface N and the light source direction vector L are both normalised then the above equation can be simplified to

$$I = L_d k_d (N.L)$$

If a light source is an infinite distance from the object then L will be the same for all points on the object — the light source becomes a directional light source. In this case less computation can be performed. Adding the ambient reflection and diffuse reflection contributions together we get

$$I = L_a k_a + L_d k_d (N.L)$$

14.6 Light Source Attenuation

The brightness of an object should depend not just on orientation but also on distance. This is called light source attenuation. From physics we know that the intensity of a light source follows an inverse square law. If we introduce a light source attenuation factor

$$f_{att} = 1/d^2_L$$

then the illumination model becomes

$$I = Iaka + fattIpkd(N.L)$$

However, this attenuation factor gives results which are too severe in practice. Instead, an
attenuation factor of the form is typically used. –

$$fatt = \min(1/c1 + c2dL + c3d2\ L\ ,\ 1)$$

where c3 is often 0.0 (meaning it is not an inverse square relationship but a linear or constant one). Colour So far our illumination equation has not included any mention of colour. We introduce colour by giving each object (or its material) an ambient and diffuse colour and each light source a colour. There are many different ways of specifying colour, or colour models. The most common is the RGB colour model where a colour is specified in terms of red, green and blue colour components.

If we use the RGB colour model then the ambient colour (reflectivity) of an object is (kaR, kaG, kaB), the diffuse colour (reflectivity) of an object kd is (kdR, kdG, kdB) and the colour of the light emitted from the point light source as (LdR, LdG, LdB).
Then the lighting equation becomes for say the red component

$$IR = LaRkaR + fattLdRkdR(N.L)$$

More generally, using wavelength λ for colour component gives –

$$I\lambda = L_{a\lambda}k_{a\lambda} + f_{att}L_{d\lambda}k_{d\lambda}(N.L)$$

14.7 Specular Reflection

Specular reflection occurs on hard, shiny surfaces and is seen as highlights. Specular highlights are strongly directional

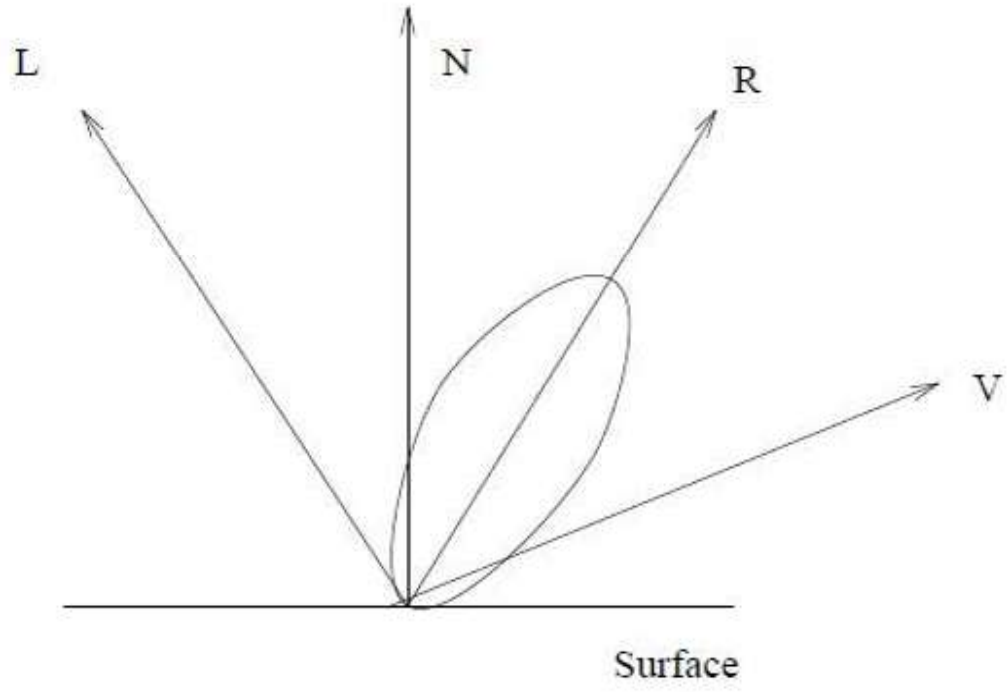


Figure 14.3(a): Specular – reflection

The approach to specular reflection in the Phong model is that the specular reflection intensity drops off as the cosine of the angle between the normal and the specular reflection direction raised to some power n which indicates the shininess of the surface.

The higher the power of n the smaller and brighter the highlight.

The specular component of the illumination model may thus be given as

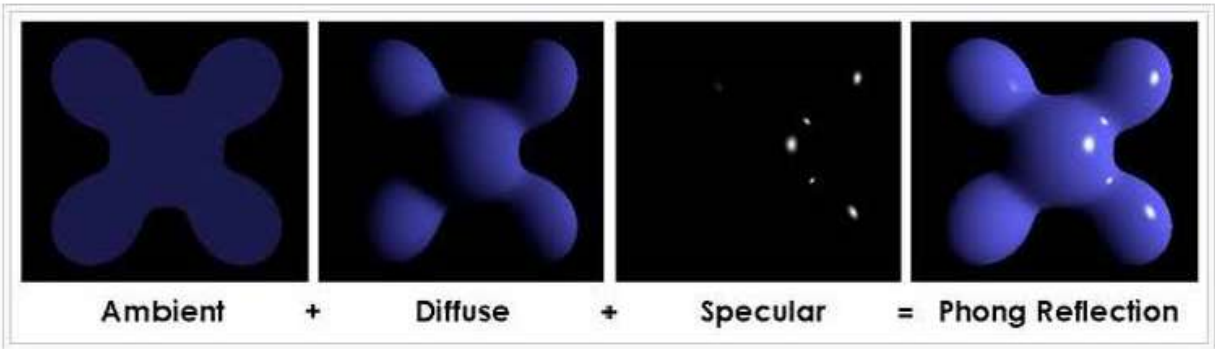
$$I = f_{att} L_{s\lambda} k_{s\lambda} \cos^n \alpha$$

If the direction of (specular) reflection R and the viewpoint direction V are normalised then the equation becomes

$$I = f_{att} L_{s\lambda} k_{s\lambda} (R.V)^n$$

The full lighting equation becomes

$$I_\lambda = L_{a\lambda} k_{a\lambda} + f_{att}[L_{d\lambda} k_{d\lambda} (N.L) + L_{s\lambda} k_{s\lambda} (R.V)^n]$$



When we look at an illuminated shiny surface, such as a polygon.

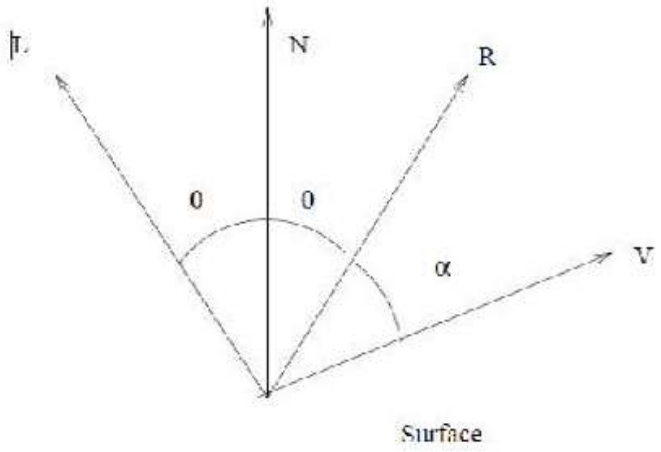
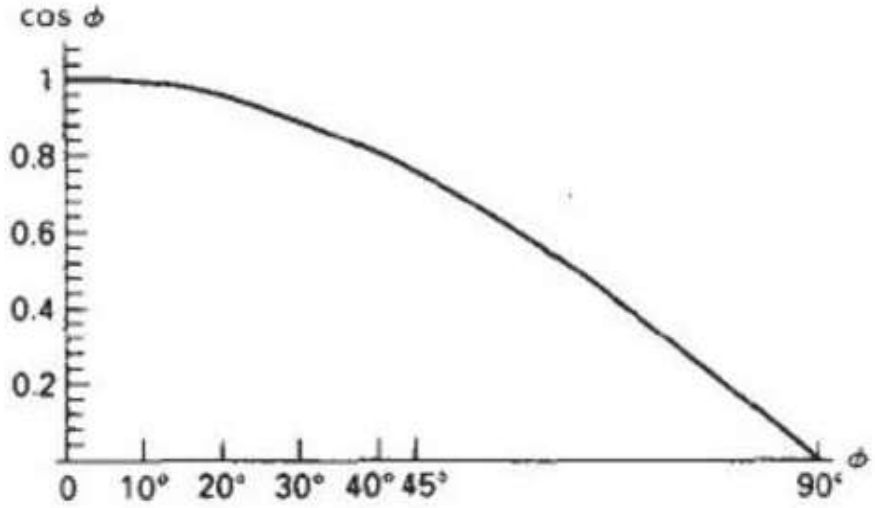


Figure 14.3(b)

Figure 14.3(b) shows the specular reflection direction at a point on the illuminated surface. The specular – reflection angle equals the angle of the incident light, with the two angles measured on opposite sides of the unit normal surface vector N. In this figure, we use R to represent the unit vector in the direction of ideal specular reflection; L to represent the unit vector directed toward the point light source; and V as the unit vector pointing to the viewer forms the surface position. Angle α is the viewing angle relative to the specular reflection direction R. In this case, we would only see reflected light when vectors V and R coincide ($\alpha = 0$)

Objects other than ideal reflectors exhibit specular reflections over a finite range of viewing positions around vector R. Shiny surfaces have a narrow specular-reflection range, and dull surfaces have a wider reflection range. An empirical model for calculating the specular – reflection range, developed by Phong Bui Tuong and called the Phong specular – reflection model, or simply the Phong model, sets the intensity of specular reflection proportional to $\cos^n \alpha$. Angle α can be assigned the value in the range of 0 to 900, so that $\cos\alpha$ varies from 0 to 1.

Figure shows the effect of ns on the angular range for which we can expect to see specular reflections.



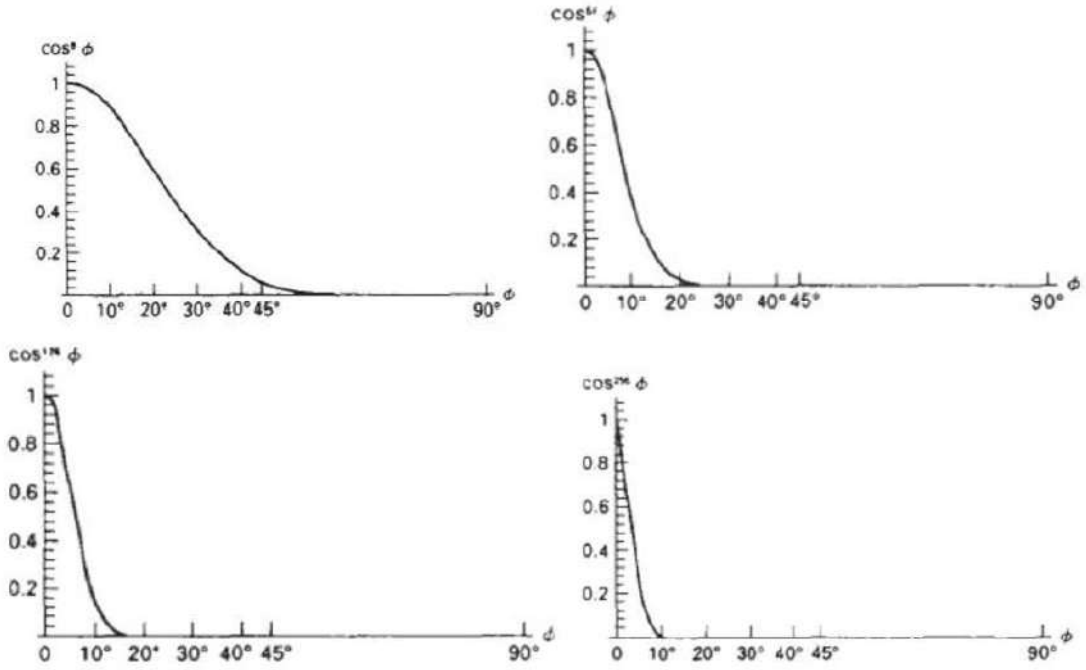


Figure 14.3(c): plots of $\cos^{ns}\alpha$ for several values of specular parameters ns.

As seen in 14.3(c) transparent material such as glass, only exhibit appreciable specular reflections as θ approaches 900. At $\theta = 0^0$, about 4 percent of the incident light on glass surface is reflected.

Since V and R are unit vectors in the viewing and specular – reflection directions, we can calculate the value of $\cos\alpha$ with the dot product V. R. Assuming the specular reflection coefficient is a constant, we can determine the intensity of the specular reflection at a surface point with the calculation.

$$I_{spec} = K_s I_l (V.R)^{ns}$$

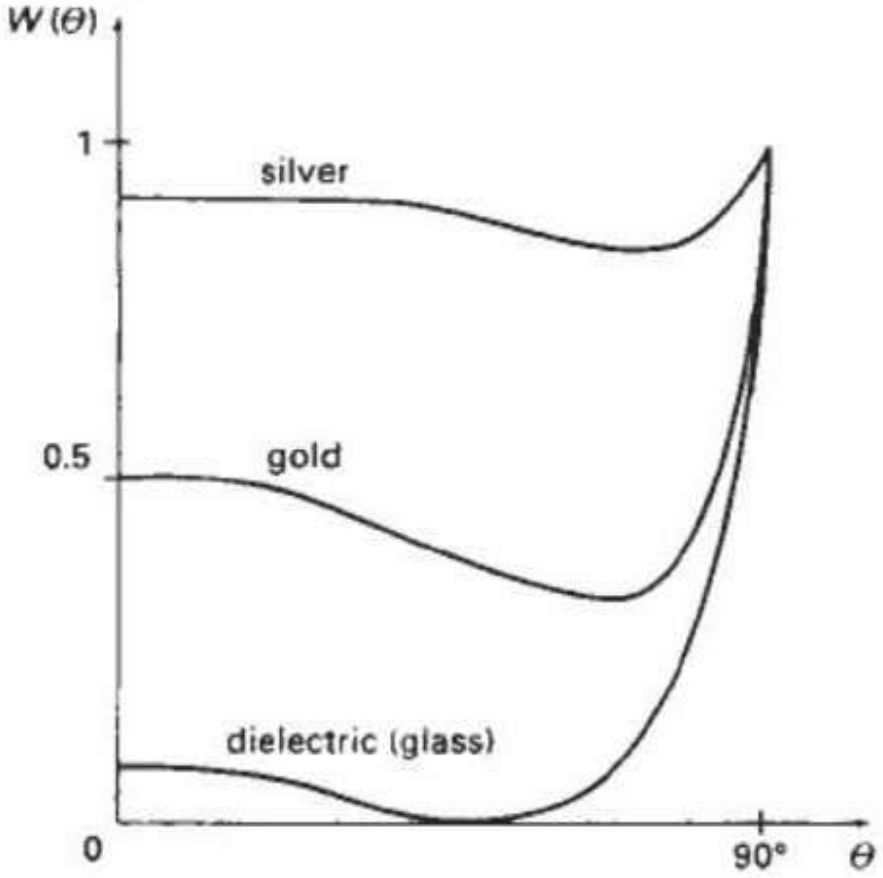


Figure 14.4: Approximation variation of the specular – reflection coefficient as a function of angle of incidence for different materials.

Vector R in this expression can be calculated in terms of vectors L and N. As seen in fig 5, the projection of L onto the direction of the normal vector is obtained with the dot product N.L. Therefore, from the diagram, we have

$$R + L = (2N.L)N$$

And the specular – reflection vector is obtained as

$$R = (2N.L)N-L$$

A simplified Phong model is obtained by using the halfway vector H between L and V to calculate the range of specular reflections. If we replace V.R in the Phong model with the dot product N.H, this simply replaces the empirical $\cos\alpha$

calculation with the empirical cosα calculating. The halfway vector is obtained as
–

$$H = \frac{L+V}{|L+V|}$$

14.8 Gouraud Shading

Illumination or lighting models determine the colour of a point on the surface of an object.
A shading model determines where the lighting model is applied. Most common 3D graphics libraries are polygon based.

Gouraud shading the lighting model is applied at each vertex of the polygon. The polygon is filled by bi-linear interpolation of the resulting values. This is known as smooth shading in OpenGL.

Each polygon surface is rendered with Gouraud shading by performing the following calculations:

- Determine the average unit normal vector at each polygon vertex.
- Apply an illumination model to each vertex to calculate the vertex intensity.
- Linearly interpolate the vertex intensities over the surface of the polygon.

At each polygon vertex, we obtain a normal vector by averaging the surface normal of all polygons sharing that vertex, as illustrated in fig – 14.5. Thus, for any vertex position V, we obtain the unit vertex normal with the calculation.

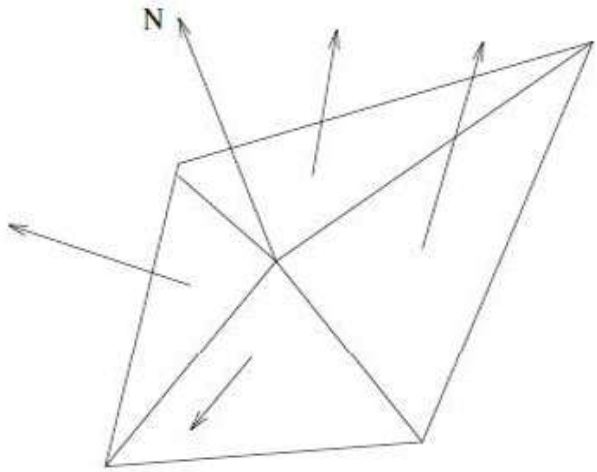


Figure 14.5(a)

$$Nv = \frac{\sum_{k=1}^{n} Nk}{\left|\sum_{k=1}^{n} Nk\right|}$$

Figure 14.5 (b) demonstrate the next step: interpolation intensities along the polygon edge with endpoint vertices at position 1 and 2 is intersected by the scan line at point 4. A fast method for obtaining the intensity at point 4 is to interpolate between intensities I1 and I2 using only the vertical displacement of the scan line.

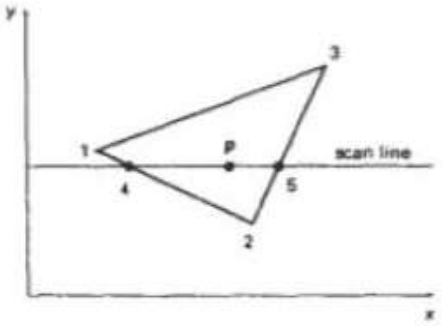


Figure 14.5(b):

Fig –14.5 (b) For Gouraud shading the intensity at point 4 is linearly interpolated from the intensities at vertices 1and 2. The intensity at point 5 is linearly interpolated from

intensities at vertices 2 and 3. Is An interior point P is then assigned an intensity value that is linearly interpolated from intensities at positions 4 and 5.

$$I_4 = \frac{y_4 + y_2}{y_1 - y_2} + \frac{y_1 - y_4}{y_1 - y_2} I_2$$

Similarly, intensity at the right intersection of this scan line (point 5) is interpolated from intensity values at vertices 2 and 3.

$$I_p = \frac{x_5 + x_p}{x_5 - x_4} I_2 + \frac{x_p - x_4}{x_5 - x_4} I_5$$

Incremental calculations are used to obtain successive edge intensity values between scan lines and to obtain successive intensities along a scan line. As showing in fig -, if the intensity at edge position (x, y) is interpolated as –

$$I = \frac{y - y_2}{y_1 - y_2} I_1 + \frac{y_1 - y}{y_1 - y_2} I_2$$

Then we can obtain the intensity along this edge for the next scan line, y-1, as –

$$I' = I + \frac{I_2 - I_1}{y_1 - y_2}$$

Similar calculations are used to obtain intensities at successive horizontal pixel positions along each scan line.

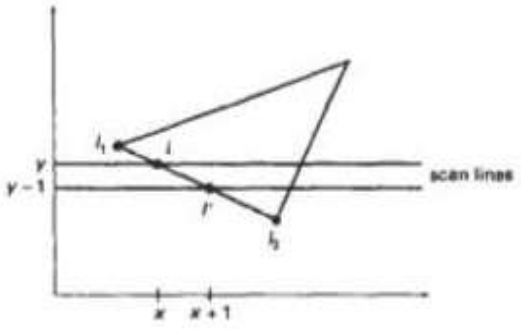


Figure – 14.5 (c)

Incremental interpolation of intensity values along a polygon edge for successive scan lines.

14.9 Phong Shading

Phong Shading The lighting model is applied at every pixel covered by the polygon. The normal vector at each pixel is computed by bi-linear interpolation.

A polygon surface is rendered using Phong shading by carrying out the following steps –

- Determine the average unit normal vector at each polygon vertex.
- Linearly interpolate the vertex normal over the surface of the polygon.
- Apply an illumination model along each scan line to calculate projected pixel intensities for the surface points.

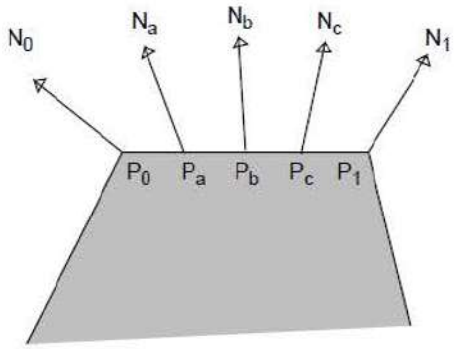


Figure 14.6:

$$N \quad = \frac{y \ - y_2}{y_1 - y_2} N_1 + \frac{y_1 - y}{y_1 - y_2} N_2$$

Gouraud and Phong shading are used to make polygonal objects — which are inherently faceted — appear smooth.

14.10 Ray Tracking

Ray tracing was first developed in the 1960s by scientists at an organization known as Mathematical Applications Group. Ray tracing is used extensively in computer gaming and animation, television and DVD programming and movie production.

In computer graphics, ray tracing is a technique for generating an image by tracing the path of light through pixels in an image plane and simulating the effects of its encounters with virtual objects.

Ray Tracking is an extension of this basic idea. Instead of merely looking for the visible surface for each pixel, we continue to bounce the ray around the scene in figure – 14.7

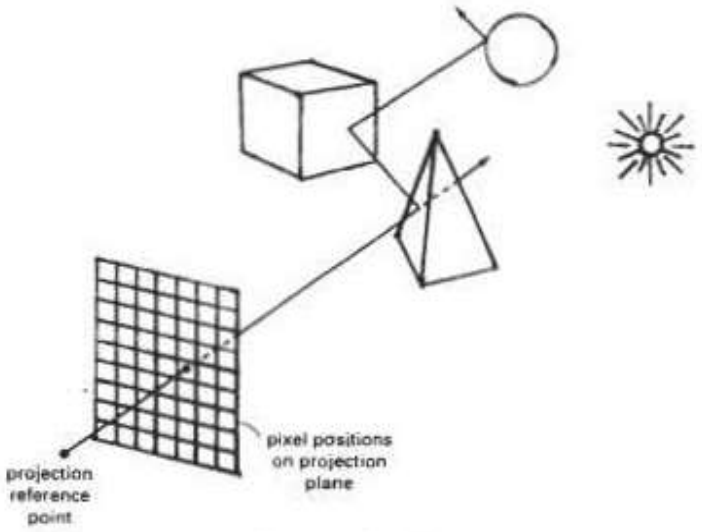


Figure 14.7

Tracking a ray from the projection reference point through a pixel position with multiple reflections and transmissions.

Ray tracing is one of the numerous techniques that exist to render images with computers. The idea behind ray tracing is that physically correct images are composed by light and that light will usually come from a light source and bounce around as light rays (following a broken line path) in a scene before hitting our eyes or a camera.

14.11 Summary

In general, an object is illuminated with radiant energy from light emitting sources and from the reflective surfaces of other objects in the scene. Light sources can be modeled as point sources or as distributed sources. Lighting effects can be described in terms of diffuse and specular components for both reflections and refractions.

Illumination model can be used to describe diffuse reflections with Lambert's cosine and to describe specular reflections with the Phong model.

Gouraud shading approximates light reflections from curved surfaces by calculating intensity values at polygon vertices and interpolating these intensity values across the polygon facets. A more accurate, but slower surface rendering procedure is Phong shading, which interpolates the average normal vectors for polygon vertices over the polygon facets. Ray tracking provides an accurate method for obtaining global, specular reflection and transmission effects.

14.12 Exercise

Q.1 Differentiate between Gouraud Shading and Phong Shading?

Q.2 Explain All illumination Model?

14.13 References

1. Computer Graphics (C Version) Second Edition "Donald Hearn M. Pauline Baker."
2. Additional sources of information on visibility algorithm include Elber and Cohen (1990), Franklin and Kankanhalli (1990), Glassner (1990), Naylor Amantides, and Thibault (1990), and Segal (1990).

UNIT-15
Color Model and Applications

Structure of the Unit

15.0 Objective

In this unit we shall be focused on the following topics:

- Color Models
- RGB Color Model
- CMY Color Model
- YIQ Color Model
- HSV Color Model

15.1 Introduction

The light or colors are the narrow frequency band with the electromagnetic spectrum, called visible band. Each frequency value within the visible band corresponds to a distinct color.

Since the light is an electromagnetic spectrum, it also has a wavelength. So the light can be described either in terms of frequency or wavelength. The relationship between the wavelength and frequency can be represented as

$$c = \lambda f,$$

where c=Speed of light λ =Wavelength and f=frequency. In Vacuum the velocity of light is $3X10^{10}$ cm/sec.

The table below shows the color and associated wavelength which further could be converted into frequency as per the formula given.

color	**Wavelength interval**	**Frequency interval**
Red	~ 700–635 nm	~ 430–480 THz
Orange	~ 635–590 nm	~ 480–510 THz
Yellow	~ 590–560 nm	~ 510–540 THz
Green	~ 560–520 nm	~ 540–580 THz
Cyan	~ 520–490 nm	~ 580–610 THz
Blue	~ 490–450 nm	~ 610–670 THz
Violet	~ 450–400 nm	~ 670–750 THz

In this unit, we shall be concentrating on the mechanism used for generating color into the display.

15.2 Color Model

A color model is a method for explaining the properties or behavior of color within some particular context. The color model uses various color components (usually three or four) for describing different colors.

A Color model usually uses three dimensional (3D) coordinate system to specify a particular color. Any color that can be specified in color model will correspond to a single point within the subspaces it defines.

The range of colors that can be described by the color model called ***color gaunt.*** The two or three colors used to describing other colors are referred to as ***Primary Color***.

There are several color model used in computer graphics but the two most commonly used models are RGB and CMYK model for printing.

Some of the color models we will discuss in this section are:

(i) RGB Color Model
(ii) CMY Color Model
(iii) YIQ Color Model
(iv) HSV Color Model

Since no finite set of color light source can be combined to display all possible colors. So, three standard primaries were defined in 1931 by International Commission on Illumination CIE (Commission International delEclarage) by using XYZ model.

XYZ model:- The set of CIE primaries (X, Y, Z) is commonly referred to as XYZ color model.

Where X, Y, Z represent the vectors in 3D, additive color spaces.

Any color in XYZ model is expressed as

$C \lambda = Xx + Yy + Zz$

Where X, Y, Z are the amount of standard primaries needed to match $C\lambda$

15.3 RGB Color Model

RGB stands for Red, Green and, Blue. This model was inspired by the cones of the retina. As the retina's visual pigments have a peak sensitivity at a wavelength of Red, Green and Blue colors. In this model, the primary colors are Red, Green and Blue. It is an additive model, in which colors are produced by adding components with white having all colors and black, when no color added.

This model is widely used for display units like Television and computer screens.

The RGB model is usually represented by a unit cube with one color located at the origin of 3D color coordinate system. Each color point within the bounds of the cube can be represented as (R, G, B) where values of R, G, B are assigned in the range from 0 to 1. Thus the color C_λ is expressed as

$$C_\lambda = Rr + Gg + Bb$$

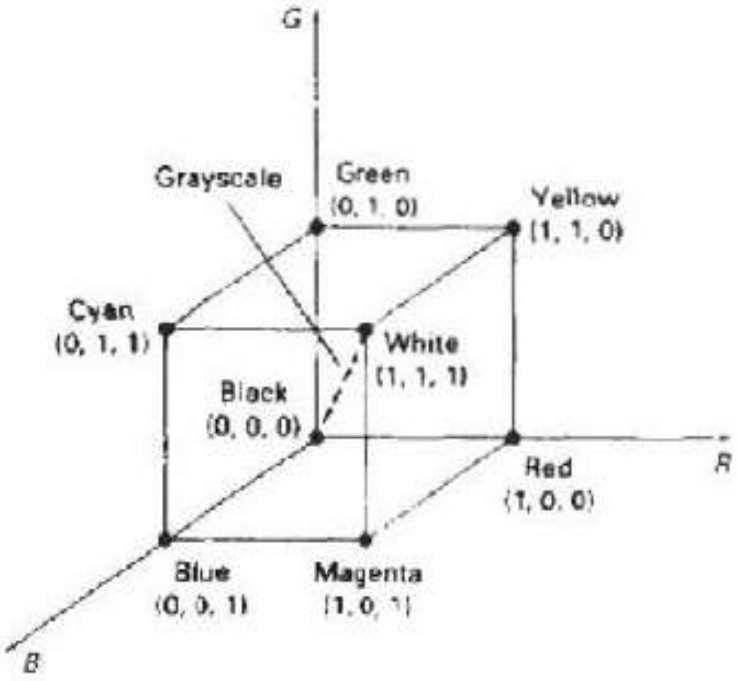


Figure15.1: RGB Color Model

Main diagonal in 3D cube represents grayscale. The value of triple for black color is (0,0,0),whit white color is (1,1,1).For more colors we need to add the two or more colors, for example, yellow color can be obtained by adding red with green.

As discussed, RGB is used in computer monitors /Displays variations for each of the additive color of red, green and blue. Therefore, there are 16,77,216 possible colors (256Rx256Gx256B) can be represented by varying the intensities of R, G, and B. The intensity of each of the red, green, blue components are represented on the scale of 0 to 255.

15.4 CMY color Model

The CMY color model stands for Cyan, Magenta and Yellow, which are the components of red, green and blue respectively. The colors in CMY are called "Subtractive Primaries", which means colors are produced by removing primaries not adding it. This model is generally used in printing systems.

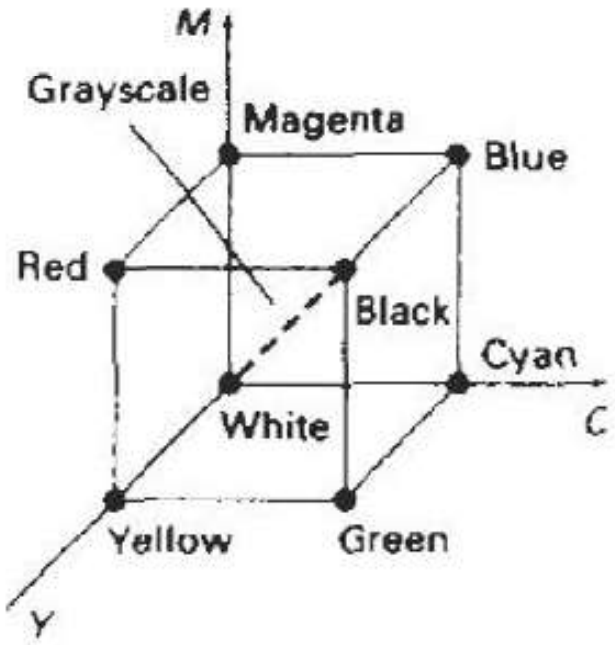


Figure 15.2: CMY Color Model

The CMY model is subtractive model appropriate to absorption of color, for example due to pigments in paints, whereas RGB model ask what is added to black to get a particular color, the CMY model ask what is subtracted from white. In this model, the primaries are Cyan, Magenta and Yellow, with Red, Green and Blue as secondary colors.

In CMY model points (1, 1, 1) represent black and (0, 0, 0) as white, which is just opposite to RGB model. The relation between RGB color model and CMY color model is given as

$$\begin{pmatrix} C \\ M \\ Y \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

15.5 YIQ Color Model

This is the most common system used by the US commercial color television (PAL is the most common system used in other countries). The YIQ model is based on the concept of CIE XYZ model.

In YIQ model, Y is same as in XYZ model. Luminescence (Brightness) information is contained in the Y parameter while the chromaticity information (hue and purity) is contained in I and Q parameters. Since Y is luminescence thus black and white TV has only Y signals.

The conversion from RGB to YIQ is given by

$$\begin{pmatrix} Y \\ I \\ Q \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{pmatrix} . \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

We can also convert RGB to YIQ by using

$$\begin{pmatrix} R \\ Y \\ B \end{pmatrix} = \begin{pmatrix} 1.000 & 0.956 & 0.620 \\ 1.000 & -0.272 & -0.64 \\ 1.000 & -1.108 & 1.705 \end{pmatrix} . \begin{pmatrix} Y \\ I \\ Q \end{pmatrix}$$

15.6 HSV Color Model

HSV stands for Hue, Saturation and Value. This color model describes color (hue or tint) in terms of their shade (saturation or amount of grey) and brightness value. The HSV color wheel is depicted as a cone or cylinder which is derived from RGB cube.

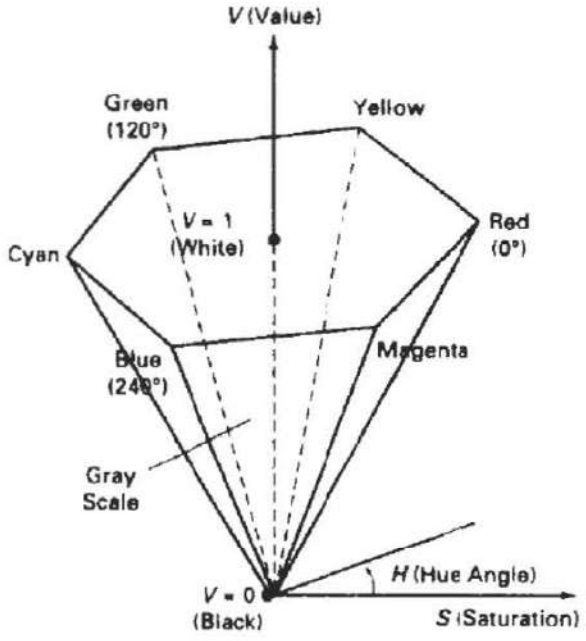


Figure 15.3: HSV Hexagon

- ***Hue*** is the name or pure value of the color such as red, green, yellow etc. It is expressed as a number from 0 to 360 degrees representing hues of red

(which start at 0), yellow (starting at 60), green (starting at 120), cyan (starting at 180), blue (starting at 240) and magenta (starting at 300).

- ***Saturation*** is the purity of the color and the amount of pure color. It varies from white to pure color. It is measured in percent from 0 to 100.
- ***Value*** (or brightness) works in conjunction with saturation and describes the brightness or intensity of the color from zero percent to 100 percent.

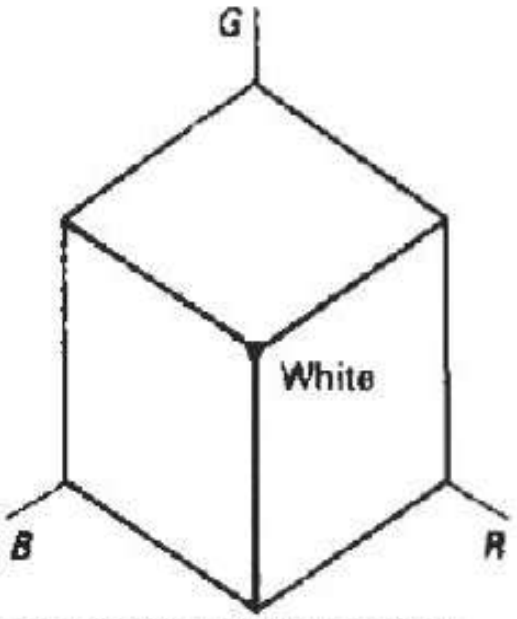


Figure 15.4(a): RGB Color Cube

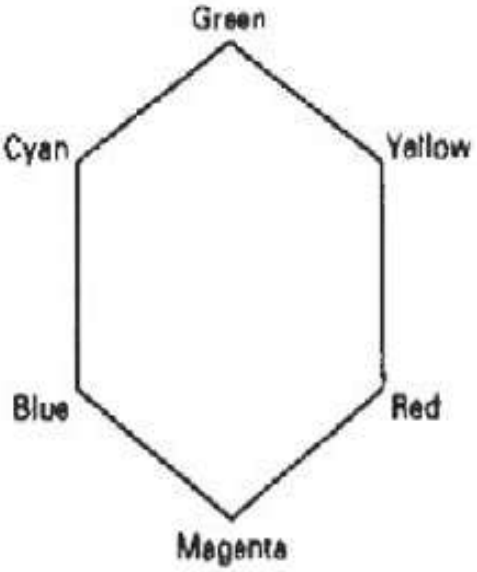


Figure15.4(b): Color Hexagon

All color models treated so far are hardware oriented. The Hue-Saturation-Value model is oriented towards the user/artist. The allowed coordinates fill a six-sided pyramid the 3 top faces of the color cube as base. Note that at the same height colors of different perceived brightness are positioned. Value is given by the height; saturation is coded in the distance from the axes and hue by the position on the boundary.

Comparison among RGB,HSV,CMY and YIQ Color Model:

Acronym	Intended use	Axis 1	Axis 2	Axis 3
RGB	Device	Intensity of	Intensity of	Intensity of

	specific color specification	red gun	green gun	blue gun
HSV	Color Mixing	Hue	Saturation	Value
CMY	Color Printing	Cyan	Magenta	Yellow
YIQ	North American Broadcast TV	Luminance	Blue-green/orange	Yellow-green/magenta

15.7 Color Selection and Applications

As general rules, the use of smaller number of colors produces a more pleasing display than a large number of colors.

Tints and Shades blend better than pure hues.

For background gray or the complement of one of the background color are best.

15.8 Summary

- Visible light is narrow frequency distributions within the electromagnetic spectrum.
- Light can be described either with wavelength or frequency
- Color model uses an abstract mathematical model to describe the color by using two to four primary colors.
- There are several color models in computer graphics i.e. RGB, CMY, YIQ, HSV
- RGB uses red, green, blue colors as primary colors and generally used in Video Monitor Display
- CMY uses cyan, magenta and yellow colors as primary colors and are generally used in printers.
- CMY is subtractive model whereas RGB is additive model.

- Other models like YIQ, HSV are based on specification of luminance and purity values.
- Since no model specified with finite set of color parameters which is capable of describing all possible colors .Hence a hypothetical color has been adopted as standard for defining all colors combination is called as XYZ color model.
- We should avoid displaying adjacent color that differs widely of harmonious color combination.
- We should limit display to a small number of color combination s formed with tints and shades rather than pure hues.

15.9 Glossary

Color Gaunt: The range of colors that can be described by the color model called ***color gaunt.***

Primary Color: The two or three colors used to describing other colors are referred to as ***Primary Color***

15.10 Exercise

Q.1. _____uses an abstract mathematical model to describe the color by using two to four primary colors.

(A) Polygon filling (B) Color Model

(C) Aliasing (D) None of these

Q.2. HSV stands for

(A) Hue, Saturation and Value (B) Red, Green and Blue

(C) Cyan, Magenta and Yellow (D) None of these

Q.3. Which color is best suited for Video Monitor Display?

(A) RGB color model

(B) YIQ color model

(C) HSV color model

(D) None of these

Q.4. Which color model is best suited for printing system?

(A) RGB color model

(B) YIQ color model

(C) HSV color model

(D) None of these

Q.5. What is Color Model? Explain various types of Color Model

Q.6. Explain RGB Color Model.

Q.7. How to convert RGB to CMY model? Explain.

15.11 Answers to Exercise

Ans.1: B

Ans.2: A

Ans.3: A

Ans.4: B

References and Suggested Readings

1. J. Foley, A. Van Dam, S. Feiner, J. Hughes: Computer Graphics- Principles and Practice, Pearson.

2. Hearn and Baker: Computer Graphics, PHI.

3. Additional programming examples and information on PHIGS primitive can be found in Howard, et al. 1991

4. Filled-Area_Primitives_I-Computer_Graphics-Lecture_Notes.pdf

UNIT-16
Computer Animation

Structure of the Unit

16.0 Objective

In this chapter, we shall focus on the following topics

- Design of Animation Sequences
- General Computer Animation Functions
- Raster Animations

- Computer Animation Languages
- Various Animation Tools

16.1 Introduction

Just what is computer animation? For decades, animation has been a trade that rested exclusively within the hands of the entertainment industry; the course of action needed an excellent deal of time, manpower, and complex equipment to achieve.

Animation means giving life to any object in computer graphics. It has the power of adding power and feelings into the most seemingly non-living objects. Computer-assisted animation and computer-generated animation are two categories of computer animation. It can be presented via film or video.

An object seen by human eye remains chemically mapped on the eye's retina for a brief time after viewing. This makes it possible for a series of images that are changed very rapidly to blend together into illusion of movement.

Animation is used in Visualization to show the time-dependent behavior of complex systems. A major part of animation is motion control. Early systems did not have the computational power to allow for animation preview and interactive control. Also, many early animators were computer scientists rather than artists. Thus, scripting systems were developed. These systems were used as a computer high-level language where the animator wrote a script (program) to control the animation.

Later systems have allowed for different types of motion control. One way to classify animation techniques is by the level of abstraction in the motion control techniques. A low-level system requires the animator to precisely specify each detail of motion, whereas a high-level system would allow them to use more general or abstract methods. For example, to move a simple rigid object such as a cube, requires six degrees of freedom (numbers) per frame.

Types of Animation

The two main categories are:

1. Computer- Assisted Animation
2. Computer generated Animation

Computer Assisted Animation: It is basically to 2D and 2 ½ D systems that computerize the traditional animation process. Interpolation between key shapes is the only use of computer in this type of animation.

Computer Generated Animation: It is basically concerned with motion control of the objects. The motion specification for computer generated animation is divided into two categories:

- Low-Level Techniques
- High-Level Techniques

Low-Level Techniques: Low-Level Techniques aid the animator in precisely specifying motion. These techniques consist of techniques, such as shape interpolation algorithms (in-betweening), which help the animator fill in the details of the motion once enough information about the motion has been specified by the animator. When using low-level techniques, the animator usually has a fairly specific idea of the exact motion that he or she wants.

High-Level Techniques: High-level techniques are typically algorithms or models used to generate a motion using a set of rules or constraints. The animator sets up the rules of the model, or chooses an appropriate algorithm, and selects initial values or boundary values. The system is then set into motion, so to speak, and the motion of the objects is controlled by the algorithm or model. The model-based/algorithmic approaches often rely on fairly sophisticated computation, such as physically based motion control.

16.2 Design of Animation Sequences

Animation sequences are designed with the following Steps:

Storyboard layout

It is the outline of the action. It defines the motion sequence as a set of basic events that are to take place in specific order. Such an ordered set of event gives the motion sequences. Depending on the type of animation to be produced, the storyboard could consist of a set of rough sketches or it could be a list of the basic ideas for the motion.

Figure 16.1

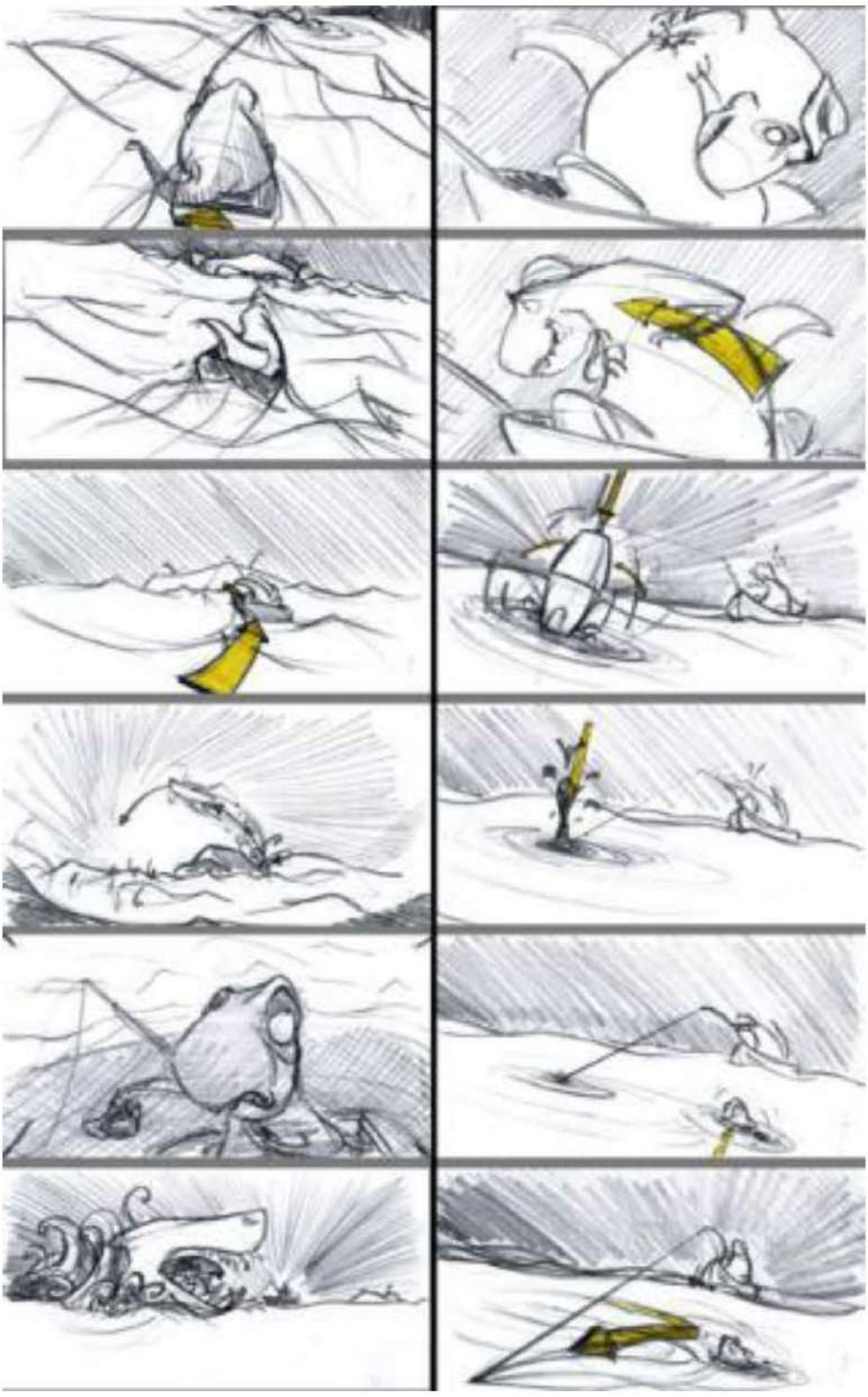


Figure 16.2

Object definitions

An object definition is given for each participant in the action. Objects can be defined in terms of basic shapes such as polygons or splines. The associated movements of each object are specified along with the shape.

Keyframe specifications

A keyframe is detailed drawing of the scene at a certain time in the animation sequence. Within each keyframe, each object is positioned according to the time for that frame. Some keyframes are chosen at extreme positions in the action; others are spaced so that the time interval between keyframes is not too much.

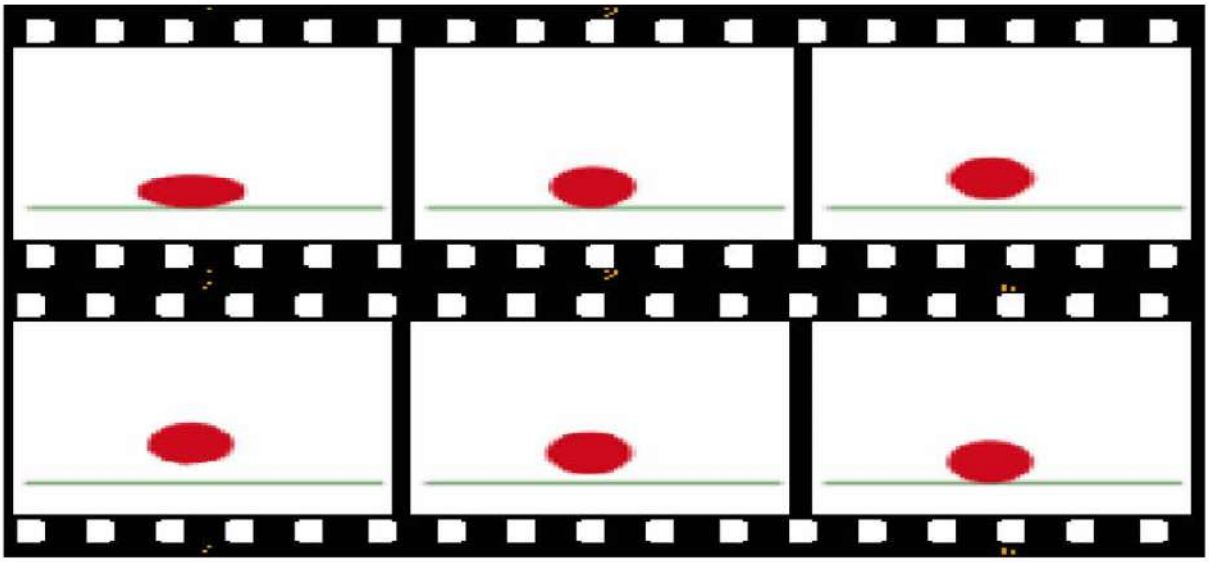


Figure 16.3

Generation of in-between frames

In-betweens are the intermediate frames between the keyframes. The number of in-between frames needed is determined by the media to be used to display the animation. Film requires 24 frames per second and graphics terminals are refreshed at the rate of 30 to 60 frames per seconds.

Time intervals for the motion are set up so there are from 3 to 5 in-between for each pair of keyframes. Depending on the speed of the motion, some keyframes can be duplicated. For a 1 min film sequence with no duplication, 1440 frames are needed. Other required tasks are:

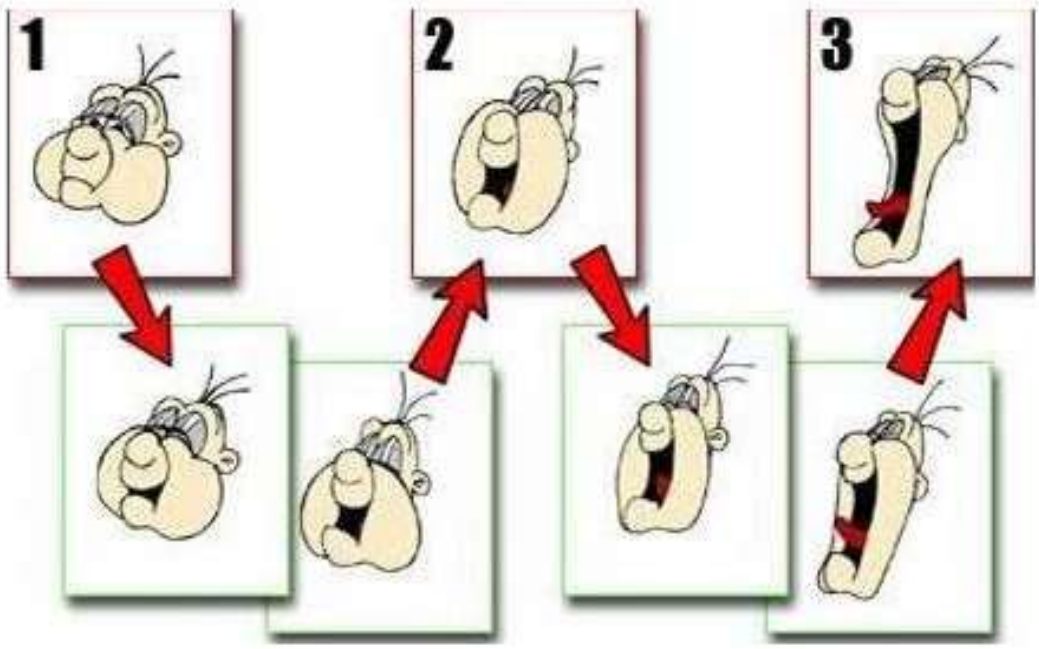


Figure 16.4

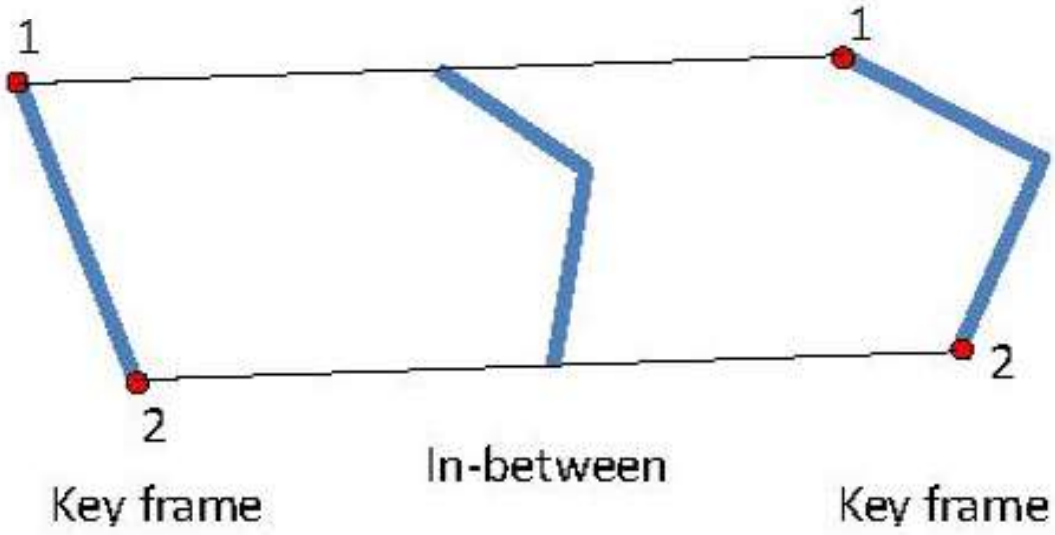


Figure 16.5

- Motion verification
- Editing
- Production and synchronization of a soundtrack.

16.3 General Computer-Animation Functions

Computer animation can be created with computer and animation software. Some of animation software are: WaveFront, Amorphium, Art of illusion, 3D studio max, Adobe Flash and many more. These software provide basic functions to create animation and to process the individual object. The functions are:

- Object manipulation and rendering – To store and manage the object database (object shapes and associated parameters are stored and updated in the database), Object motion generation (2-D or 3-D transformations) and Object rendering.
- Camera motions generation – Zooming, Panning (rotating horizontally or vertically), Tilting.
- Another function to identify visible surfaces.

16.4 Raster Animations

Raster animation is the most basic type of computer animation. It involves creating an image, and then using a computer to put that image in motion. To generate real-time animation we have execute sequence of raster operations. For example we can rotate an object like square box, as shown in the Figure 16.6. In each iteration the box is rotated by some angle. The resulted position of the box after rotation

generates new frame. By continuing such we can generate number of frames and the effect is rotation of box.

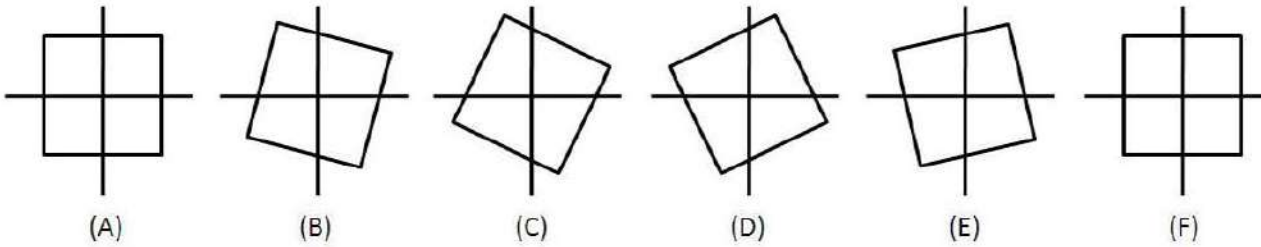


Figure 16.6

On raster systems, real-time animation in limited applications can be generated using raster operations. Sequence of raster operations can also be executed to produce real-time animation of either 2D or 3D objects.

In color displays, 24-bits per pixel are commonly used, where 8-bits represent 256 levels for each color. Here, it is necessary to read 24-bits for each pixel from frame buffer. This is very time-consuming. To avoid this video controller uses look-up-table (LUT) to store many entries of pixel values is RGB format. This look-up-table is commonly known as color-table. With this facility, now it is necessary to read index to the color-table from the frame buffer for each pixel. This index specifies one of the entries in the color-table. The specified entry in the color-table is then used to control the intensity or color of the CRT.

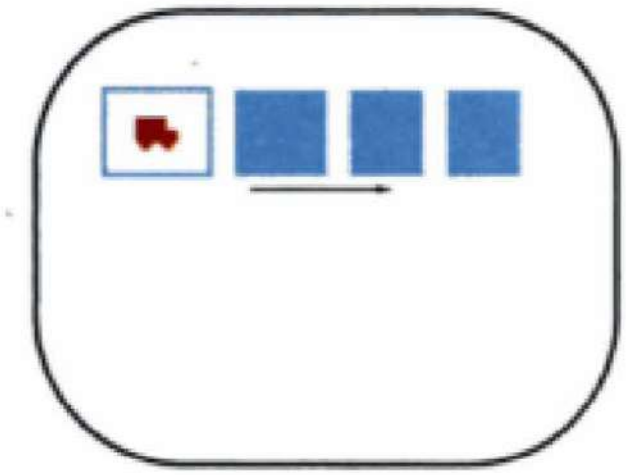


Figure 16.7

Usually, color –table has 256 entries. Therefore, the index to the color-table has 8-bits, and hence for each pixel frame buffer has to store 8-bits per pixel instead of 24-bits.

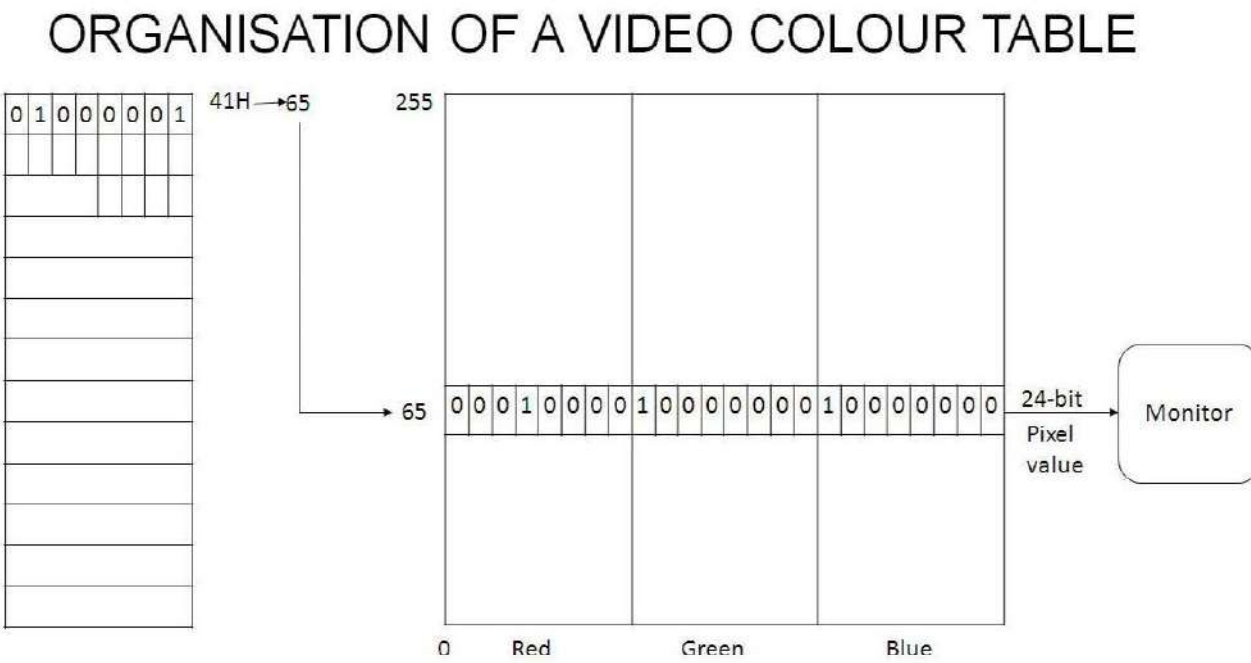
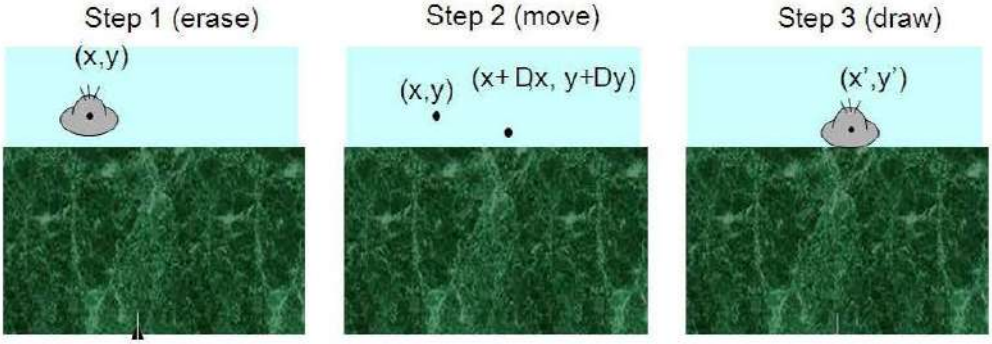


Figure 16.8

It is possible to animate object along two-dimensional motion paths using the color-table. It can be achieved by predefining the object at successive positions along the motion path, and setting the successive blocks of pixel values to color table entries. Initially, the pixels at the first position of the object are made ON, and the pixels at the other positions of the object are set to the background color. The animation is then accomplished by changing the color-table values so that the object is ON at successively position along the animation path. Each time the preceding pixels at the preceding position are set to the background color.

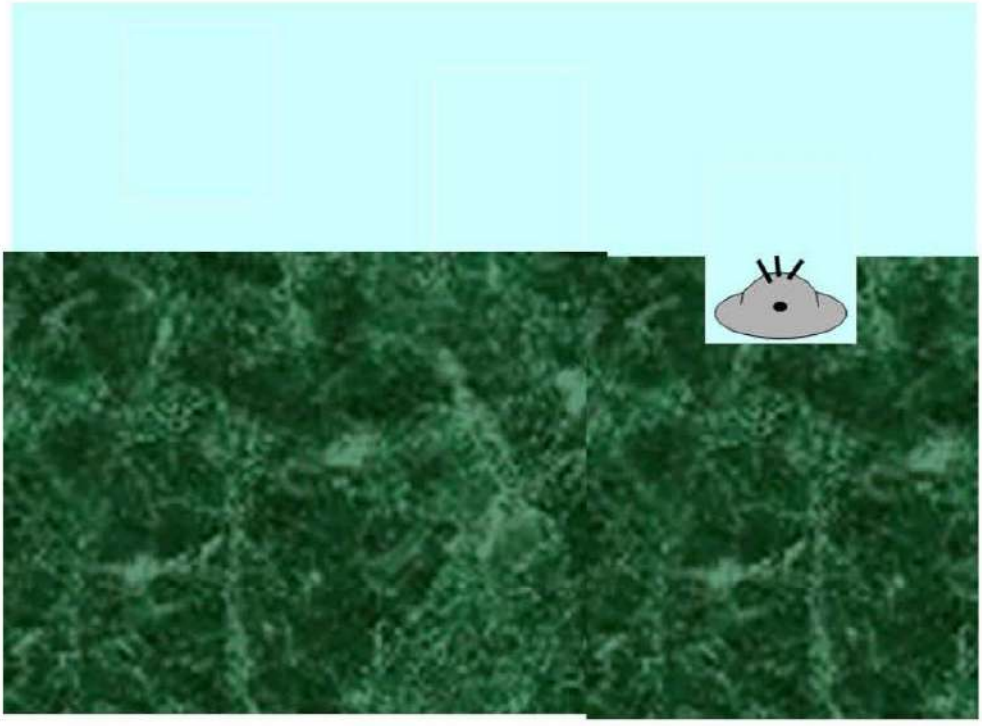
Example:

Ship is redrawn in background color



x' = x +Dx
y' = y +Dy

Move ship



16.5 Computer-Animation Languages

Design and control of animation sequences are handled with a set of animation routines. A general-purpose language, such as C, C++, LISP, Pascal, or FORTRAN, is often used to program the animation functions, but several specialized animation languages have been developed. Animation functions include a graphics editor, a keyframe generator, an in-between generator, and standard graphics routines. The graphics editor allows us to design and modify object shapes, using spline surfaces, constructive solid-geometry methods, or other representation schemes.

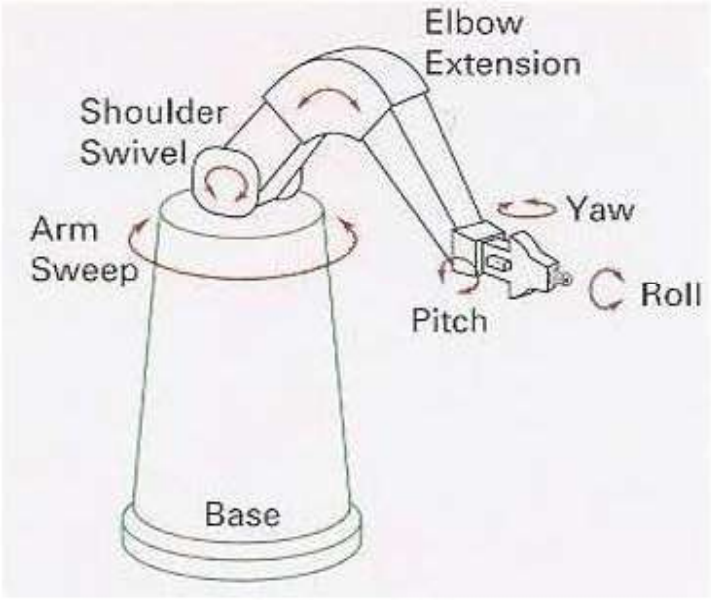


Figure 16.9

A typical task in an animation specification is scene description. This includes the positioning of objects and light sources, defining the photometric parameters (light-source intensities and surface-illumination properties), and setting the camera parameters (position, orientation and lens characteristics). Another standard function is action specification. This involves the layout of motion paths for the objects and camera. And we need the usual graphics routines: viewing and perspective transformation, geometric transformations to generate object

movements as a function of acceleration or kinematic path specifications, visible-surface identification, and the surface-rendering operations.

Keyframe systems are specialized animation languages designed simply to generate the in-betweens from the user specified keyframes. Usually, each object in the scene is defined as a set of rigid bodies connected at the joints and with a limited number of degrees of freedom.

In the above figure, the single-arm robot has six degree of freedom, which are called arm sweep, shoulder swivel. Elbow extension, pitch, yaw and roll. We can extend the number of degree of freedom for this robot arm to nine by allowing three-dimensional translations for the base (in the below figure). If we allow base rotations, the robot arm can have a total of 12 degree of freedom. The human body, in comparison, has over 200 degree of freedom.

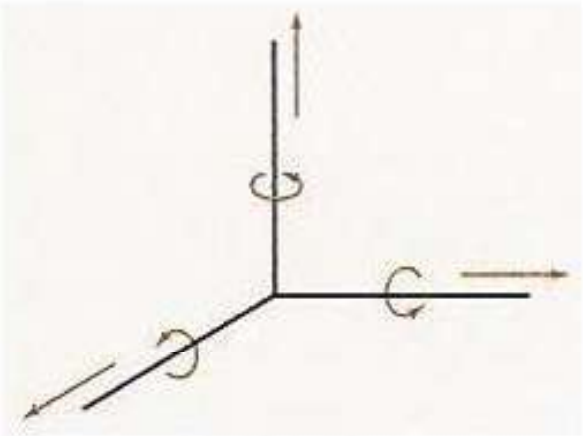


Figure 16.10

Parameterized systems allow object motion characteristics to be specified as part of the object definitions. The adjustable parameters control such object characteristics as degrees of freedom motion limitations and allowable shape changes.

Scripting systems allow object specifications and animation sequences to be defined with a user input script. From the script, a library of various objects and motions can be constructed.

Keyframe Systems

We generate each set of in-betweens from the specification of two (or more) keyframes. Motion paths can be given with a kinematic description as a set spline curves, or the motions can be physically based by specifying the forces acting on the objects to be animated.

For complex scenes, we can separate the frames into individual components or objects called cels (celluloid transparencies), an acronym from cartoon animation. Given the animation paths, we can interpolate the positions of individual objects between any two times.

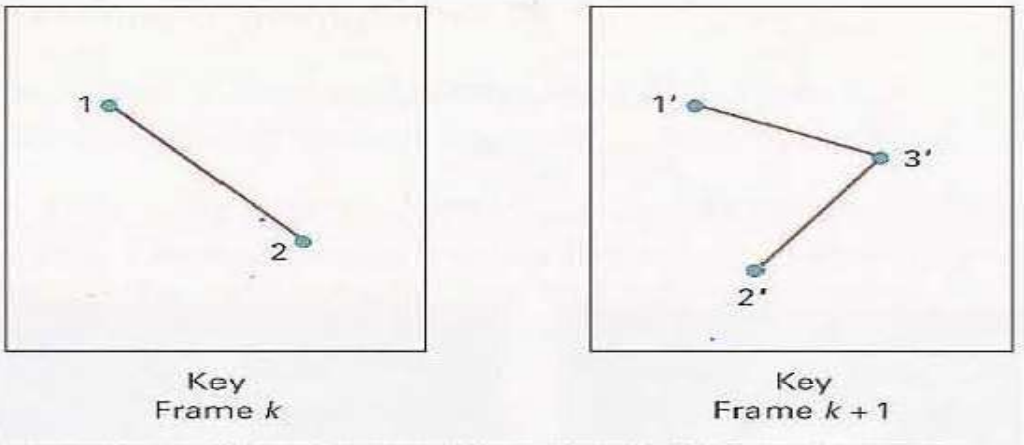
With complex object transformations, the shapes of objects may change over time. Examples are clothes, facial features, magnified detail, evolving shapes, exploding or disintegrating objects, and transforming one object into another object. If all surfaces are described with polygon meshes, then the number of edges per polygon can change from one to the next. Thus, the total number of line segments can be different frames.

Morphing

Transformation of object shapes from one form to another is called morphing, which is a shortened form of metamorphosis. Morphing methods can be applied to any motion or transition involving a change in shape.

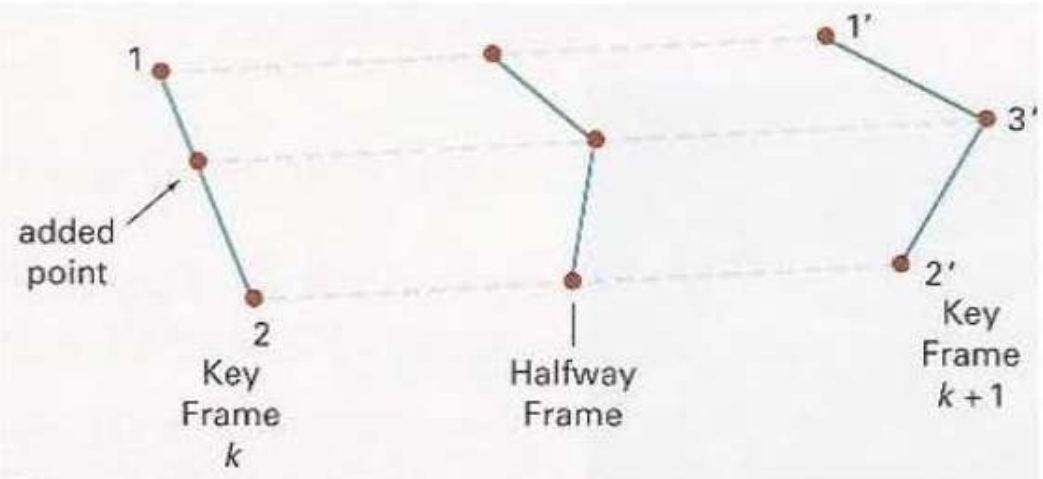
Given two keyframes for an object transformation, we first adjust the object specification in one of the frames so that the number of polygon edges (or the number of vertices) is the same for the two frames. This pre-processing step illustrated in Figure 16.11 a straight-line in keyframe k is transformed into two line segments in keyframe k + 1. Since keyframe k + 1 has an extra vertex, we add a vertex between vertices 1 and 2 in keyframe k to balance the number of vertices (and edges) in the two keyframes. Using linear interpolation to generate the in-betweens, we transition the added vertex in keyframe k into vertex 3' along the straight-line path shown in figure 16.12. An example of a triangle linearly

expanding into a quadrilateral is given in figure 16.13. Figure 16.14 and figure 16.15 show examples of morphing in television advertising.



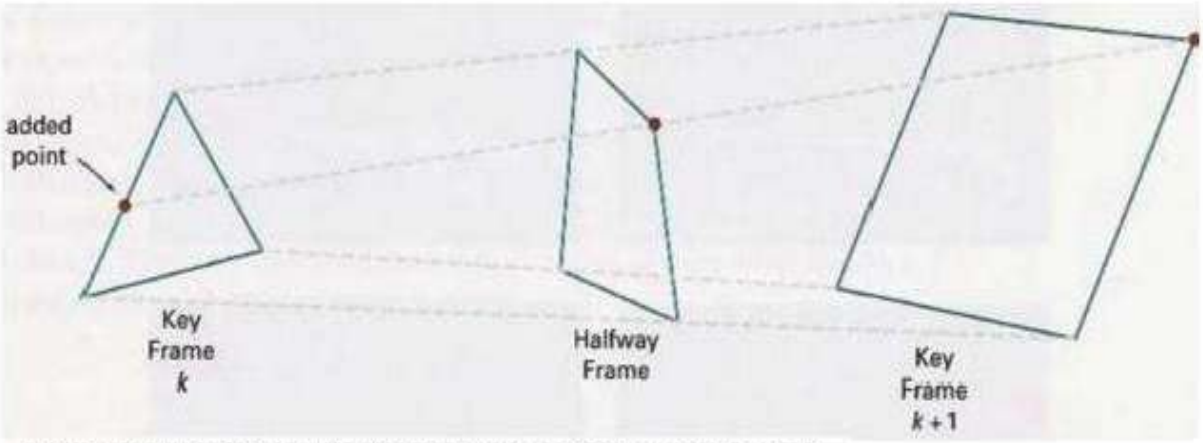
An edge with vertex positions 1 and 2 in key frame k evolves into two connected edges in key frame $k + 1$.

Figure 16.11



Linear interpolation for transforming a line segment in key frame k into two connected line segments in key frame $k + 1$.

Figure 16.12



Linear interpolation for transforming a triangle into a quadrilateral.

Figure 16.13

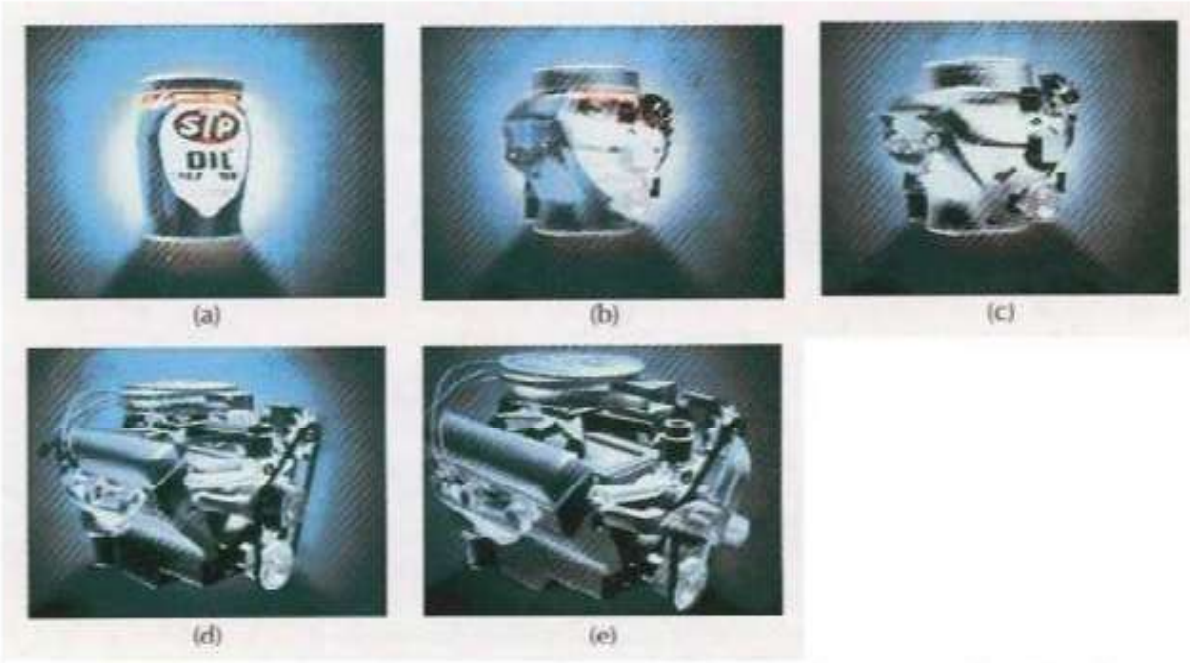


Figure 16.14: Transformation of an STP oil can into an engine block.

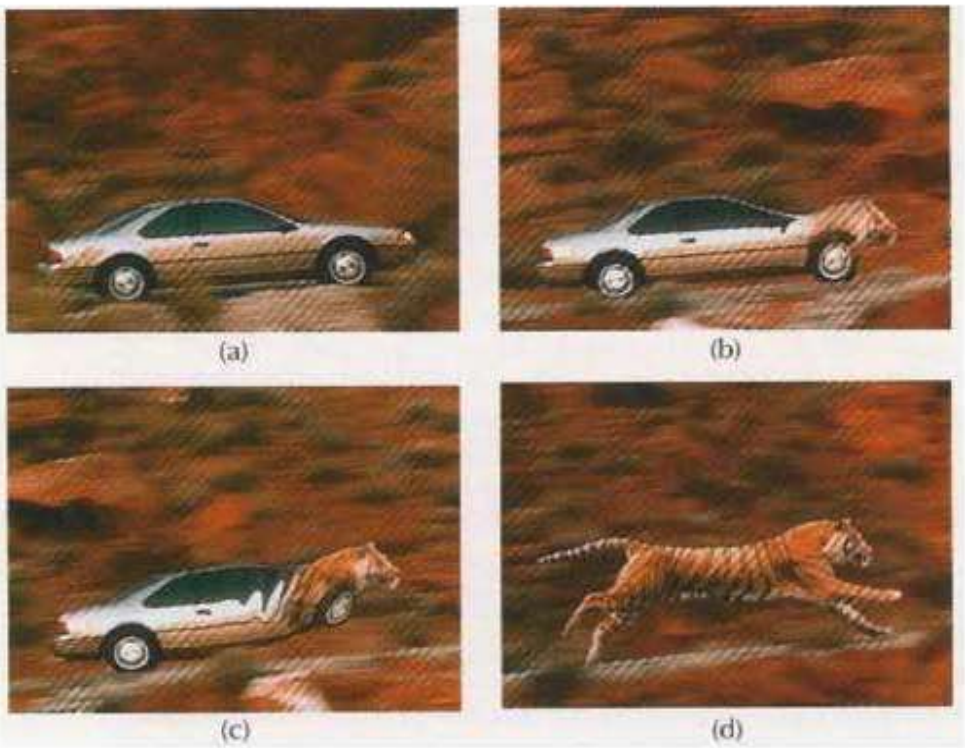


Figure 16.15: Transformation of an moving automobile into a running tiger.

The preprocessing is accomplished by

1. Dividing N_e edges of $keyframe_{min}$ into $N_s + 1$ section.
2. Dividing the remaining lines of $keyframe_{min}$ into N_s sections.

For example, if $L_k = 15$ and $L_{k+l} = 11$, we divide 4 lines of $keyframe_{k+l}$ into 2 sections each. The remaining lines of $keyframe_{k+l}$ are left intact.

If the vector counts in equalized parameters V_k and V_{k+l} are used to denote the number of vertices in the two consecutive frames.

In this case we define

$$V_{max} = \max(V_k, V_{k+l}), \qquad V_{min} = \min(V_k, V_{k+l})$$

and

$$N_{ls} = (V_{max} - l) \bmod (V_{min} - l)$$

$$N_p = \text{int}\left((V_{max} - 1)/(V_{min} - 1)\right)$$

Preprocessing using vertex count is performed by

1. Adding N_p points to N_{ls} line section of keyframe_{min}.
2. Adding N_p-1 points to the remaining edges of keyframe_{min}

Simulating Accelerations

Curve-fitting techniques are often used to specify the animation paths between keyframes. Given the vertex positions at the keyframes, we can fit the positions with linear or nonlinear paths. Figure illustrates a nonlinear fit of keyframe positions. This determines the trajectories for the in-betweens. To simulate accelerations, we can adjust the time spacing for the in-betweens.

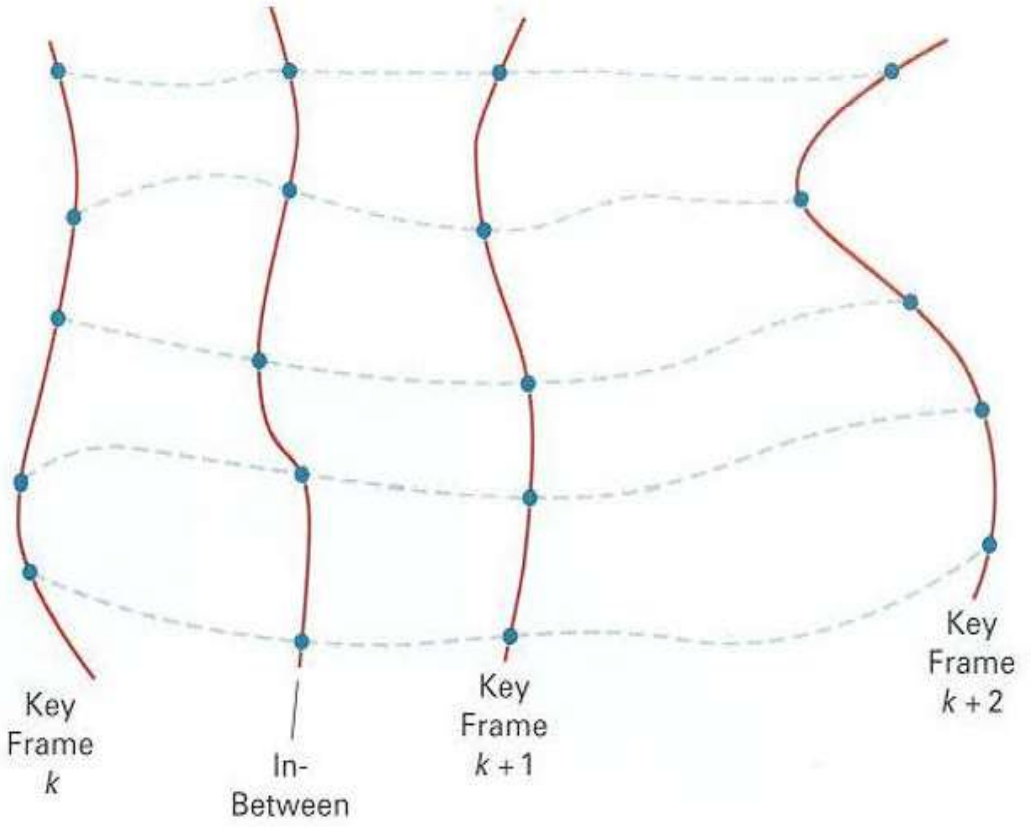


Figure 16.16

For constant speed (zero acceleration), we use equal-interval time spacing for the in-betweens. Suppose we want n in-betweens for keyframes at times t_1 and t_2.

The time interval between keyframes is then divided into n + 1 subintervals, yielding an in-between spacing of

$$\Delta = (t2-t1) / (n+1)$$

We can calculate the time for any in-between as

$$tB_j = t_1 + j \Delta t, \quad j = 1,2, \ldots\ldots n$$

and determine the values for coordinate positions, colour, and other physical parameters.

16.6 Various Animation Tools

Animations are graphic scenes played back sequentially and rapidly. These tools adopt an object-oriented approach to animation. For the implementation of animation we need two types of tools:

Hardware Tools

Software Tools

Hardware Tools

Hardware comes in many shapes, size and capabilities. Some hardware's are specialized to do only certain tasks. Other kind of hardware does a variety of things. The most common hardware used for computer animation are as follows:-

SGI (Silicon Graphics Inc.)

SGI platforms are one of the most widely used hardware platform in professional or broadcast quality computer animation production. SGI computer are very fast, give excellent results and operate using UNIX OS. For Example: Indy, Indigo 2 Extreme, Onyx machines. These machines produce animations and Onyx can even do complex calculations.

PC (Personal Computers) and Laptops

PC's and Laptops are very versatile and mobile. They are favourite of various users because of the following features it has: Combination of flexibility and power, less expensive, Provide good quality for their price. They can be used to do animations.

Mac (Macintosh)

Mac's were originally designed to be graphic and desktop publishing machines. Mac's did not become widely known. Many people consider Mac's slow and inefficient but that is not true. Mac is pretty useful for small scale companies wishing to do nice looking applications. Many software companies are using Mac for producing graphic application software.

Amiga

Originally owned by Commodore. Amiga Computes hold a good position in animation field.

Software Tools

Our hardware is dead-piece of machine without good software. The some of the most popular software's packages used by the companies, schools and individuals all round the globe are:

FlipBook (DigiCel)

For creating 2D animation of any kind, Flash and FlipBook should be the very first two programs. DigiCel's FlipBook animation software does it all, from scanning to digital painting to mattes to lighting and any other novice or veteran trick that we might required. For 2D animator who wants minimal computer involvement, then FlipBook must be great tool for computer animation.

Flash (Adobe)

Flash (Adobe) is used to make web animations. Flash is the Web standard for vector graphics and animation. With Flash, we can quickly liven up our Web pages with interesting animated graphics and text effects. Flash also allows us to add sophisticated interactivity to our site without complicated scripting.

Flash is designed to make our Web animations download and begin playing quickly. Because it creates vector-based graphics, Flash can generate files which download in much less time than bitmapped image formats such as GIFs and JPEGs. In addition, Flash's streaming format means that Web movies can begin to play while they are still downloading.

Blender (The Blender Foundation)

We need an "easy to learn and use" equivalent for 3D animation software? If so, then we should start here. Not only is it small and not processor demanding, but it's also free and comes with a Web community that offers hundreds of free classes and tutorials on its site. It's also been used to make award winning short films and has plenty of advanced features that are worth checking out.

It works as an open-sourced, community development program where people from around the world contribute to its success. Blender is a rendering\animation\game development open-sourced freeware program maintained by the Blender Foundation. The Blender Foundation is an independent organization (a Dutch "stichting"), acting as a non-profit public benefit corporation Poser (Smith Micro Software).

3ds Max (Autodesk)

Modelling in 3D is similar to sculpting. Many different techniques can be used to create the objects in our scene. Autodesk 3ds Max is the premier software package for 3D modelling, texturing, and animation, and it has many features specifically designed to assist artists, architects, engineers, and designers in various disciplines in the realization of their projects.

Autodesk 3ds Max 2013 provides users with cutting-edge rendering technology, easy-to-use materials, improved interoperability with other related design and CAD software, enhancements to modelling and animation tools, and better viewport interactivity.

It includes a Software Development Kit (SDK), which is used to develop plug-ins that give the program additional functionality. The Lighting Analysis tool is used to help meet the Leadership in Energy and Environmental Design (LEED) 8.1 certification standards.

Autodesk 3ds Max 2013 provides users with cutting-edge rendering technology, easy-to-use materials, improved interoperability with other related design and CAD software, enhancements to modelling and animation tools, and better viewport interactivity.

Maya (Autodesk)

Maya, is a 3D computer graphics software that runs on Windows, macOS and Linux, originally developed by Alias Systems Corporation (formerly Alias|Wavefront) and currently owned and developed by Autodesk, Inc. It is used to create interactive 3D applications, including video games, animated film, TV series, or visual effects. Originally a next-generation animation product based on code from The Advanced Visualizer by Wavefront Technologies, PowerAnimator by Alias Research, Inc., and Alias Sketch!. The code was ported to IRIX and animation features were added; the porting project codename was Maya.

Autodesk Maya is an industry leading 3D animation software application that enables video professionals who work with animated film, television programs, visual effects, and video games to create highly professional three-dimensional (3D) cinematic animations. Prior to two-dimensional (2D) and 3D animation software, manual hand animation tools such as drawing paper and pencils, erasers, paints and brushes, light tables, and transparencies only offered a subset of what can now be done with programs such as Maya.

Cinema 4D (Maxon)

There are plenty of other animation software packages out there that compete with one another, but the one that stands out e as gaining the most momentum in the past five years is Maxon's Cinema 4D. CINEMA 4D is a 3D modelling, animation, motion graphic and rendering application developed by MAXON Computer GmbH in Germany. It is capable of procedural and polygonal modelling, animating,

lighting, texturing, rendering, and common features found in 3D modelling applications. It may not be as rampant in the 3D gaming industry, but it's seen an abundance of use in the film industry for dozens of high budget box office hits, and because its popularity is newer by comparison.

16.7 Self Learning Exercise

Q. 1 An illusion of motion that is created by displaying a series of images over a period of time is known as-

a) Graphics

b) Animation

c) Multimedia

d) Frames

Q. 2 The technique of transition of one shape into other shape is known as-

a) Multimedia

b) Animation

c) Morphing

d) Raster Image

Q. 3 Frame is -

a) A part of a view window

b) A Cell

c) A complete screen full image of animation sequence

d) Tweening

Q. 4 Which of the following SGI Machines can be used to produce animation -

a) Indy

b) Indies

c) India

d) Amiga

16.8 Summary

Animation means giving life to any object in computer graphics. A computer animation sequence can be set up by specifying the storyboard, the object definitions, and the keyframes. The storyboard is an outline of the action, and the keyframes define the details of the object motions for selected position in the animation. Once the keyframes have been established, a sequence of in-betweens can be generated to construct a smooth motion from one keyframe to the next. A computer animation can involve motions specifications for the objects in a scene as well as motion paths for a camera that moves through the scene. In Raster System, we can generate real time animation. For the implementation of animation we need two types of tools like Hardware Tools & Software Tools.

.

16.9 Glossary

Frame- An animation frame is a single photographic image in a movie.

Frame rate- The frame rate is the speed at which the frames are played.

Scene- A shot in a movie or show. A sequence is composed of several scenes

in-between- The drawings that exist between the key poses. These are drawn to create fluid transitions between poses.

Keyframe- A keyframe is a computer-generated position at a specific moment (frame) on a given trajectory.

Morphing-A feature for creating computer-generated drawings between a source drawing and a destination drawing.

Storyboard- A visual plan of all the scenes and shots in an animation.

16.10 Answers to Self-Learning Exercise

Q.1 (b)

Q.2 (c)

Q.3 (c)

Q.4 (a)

16.11 Exercise

Q. 1 The layout of complete animation theme is given by- a) Object definition

b) Storyboard

c) Fast motion

d) Multimedia

Q. 2 The distance from one pixel to the next pixel is-

a) Resolution

b) Dot Pitch

c) Pixmap

d) ppi

Q. 3 Raster graphics are composed of-

a) Pixels

b) Paths

c) Palette

d) None of these

Q. 4 The transformation that produces a parallel mirror image of an object are called

a) Rotation

b) Reflection

c) Translation

d) Scaling

Q. 5 The rectangle portion of the interface window that defines where the image will actually appear are-

a) Clipping window

b) Screen coordinate system

c) View port

d) Scaling

Q. 6 Give an example for absolute locator device

a) Mouse

b) Keyboard

c) Light pen

d) Touch panel

Q. 7 Identify an relative locator device from the following

a) Light pen

b) Touch panel

c) Mouse

d) Keyboard

16.12 Answers of Exercise

Q. 1 (b)

Q. 2 (a)

Q. 3 (a)

Q. 4 (b)

Q. 5 (c)

Q. 6 (d)

Q. 7 (c)

References and Suggested Readings

1. Computer Graphics, C Version, D. Hearn & M.P. Baker, Pearson Education
2. Computer Graphics, A.P.Godse & D.A.Godse, Technical Publications
3. Computer Graphics, Rajiv Chopra, S Chand
4. Computer Graphics & Animation, M.C. Trivedi, Jaico Publications

UNIT-17
Graphical User Interfaces and Input Methods

Structure of the Unit

17.0 Objective

In this chapter, we shall focus on the following topics

- The User Dialogue Windows and Icons
- Input of Graphical Data Logical Classification of Input Devices
- Input Functions and Input Modes
- Initial Values Parameters

17.1 Introduction

Mostly systems involve extensive graphics for human-computer interface regardless of the application. General systems consist of windows, pull-down and pop-up menus, icons, and pointing devices, such as a mouse or space ball for positioning the screen cursor. Recently graphical user interfaces involve X-Windows, Macintosh, Open-Look, and Motif. These interfaces are used in a variety of applications, like word processing, spreadsheets, databases and file-management systems, presentation systems, and page-layout systems. In graphics packages, specialized interactive dialogues are designed for individual applications, such as engineering design, business graphs, architectural design, drafting and artist's paintbrush programs. An example is the X Window System interface with PHIGS.

In this chapter, we discuss the basic elements of graphical user interfaces. We also consider how dialogues in graphics packages can allow us to construct and manipulate picture components, select menu options, and assign parameter values, select and position text strings.

17.2 The User Dialogue Windows and Icons

In an application, the user's model serves the basis for the design of the dialogue. The user's model describes what the system is designed to accomplish and what graphics operations are available. It states the type of objects that can be displayed and how the objects can be manipulated.

For example, if the graphics system is to be used as a tool for architectural design, the model describes how the package can be used to construct and display views of buildings by positioning walls, doors, windows etc.

Similarly for a facility-layout system, objects could be defined as a set of furniture items like tables, Cham and the available operations would include those for positioning and removing different pieces of furniture within the facility layout. And a circuit-design program may include electrical or logic elements for objects, with positioning operations available for adding or deleting elements over the all circuit design.

All information in the user dialogue is then presented in the language of the application. In an architectural design package, all interactions The User Dialogue are described only in architectural terms, without reference to particular data structures or other concepts that may be unfamiliar to an architect. In the following sections, we include some of the general considerations in structuring a user dialogue.

Windows and Icons

Visual representations are used both for objects and actions. Objects can be manipulated in an application and actions that are to be performed on the application objects. A window system provides a window-manager interface for the user and functions for handling the display and manipulation of the windows. Some common functions for the window system are opening and closing windows, resizing windows, repositioning windows and display routines that provide interior and exterior clipping and other graphics functions. Generally, windows are displayed with sliders, buttons, and menu icons for selecting various window options.

Some general system, such as X Windows is capable of supporting multiple window managers so that different window styles can be accommodated, each with its own window manager. Then window managers can be designed for specific applications.

Icons representing objects such as furniture items and circuit elements are often referred to as application icons. The icons representing actions, such as rotate, magnify, scale, clip, and paste are called control icons, or command icons.

Interface Design Aspects

1. **Accommodating multiple Skill Levels**

Interactive graphical interfaces provide several methods for selecting actions.

For example, options could be selected by pointing at an icon and clicking different mouse buttons, or by accessing pull-down or pop-up menus, or by typing keyboard commands. This allows a package to accommodate users that have different skill levels.

For a less experienced user, simple point and click operations are easiest operations of an application packages. The inexperienced user only focus on application work rather than remembering details of interface. While an experienced user typically wants speed that means less prompts and more inputs from keyboard or with multiple mouse button click. Actions are selected using function keys or always used by a set of shortcut keys.

2. Consistency

An important design consideration in an Interface is consistency. For example, a particular icon shape should always have a single meaning rather than serving to represent different actions or objects depending on the context. Another examples of consistency are always placing menus in the same positions so that a user does not require to a particular option. Similarly a particular combination of keyboard keys for the same action with color coding so that the same color does not have different meaning in different situations.

3. Minimization memorization

Operations in an interface should also be structured so that they are easy to understand and remember. Complicated, inconsistent and abbreviated command format create confusion and reduce the effectiveness of the use of packages. One key or button used for all delete operations is easier to remember than a number of different keys for different types of delete operations.

Icons and window systems also follow minimizing memorization. Different kinds of information can he separated into different windows, so that we do not have to focus on memorization when different information displays overlap. We can simply retain the multiple information on the screen in different windows.

Icons are used to reduce memorizing by displaying easily recognizable shapes for various objects and actions.

4. Back up and Error Handling

Back up or aborting is another feature of interfacing. Backup can be provided in many forms. A standard undo key or command is used to cancel a single operation. Sometimes a system can be backed up through several operations, allow to reset the system to some specified point. Using extensive backup capabilities, all inputs could be saved so that we can back up and "replay" any part of a session. Once we have deleted the trash in the desktop, we cannot recover the deleted files. In this case, the interface would ask to verify the delete operation before proceeding. Good diagnostics and error messages are designed to help determine the cause of an error. Additionally, interfaces attempt to minimize error possibilities by anticipating certain actions that could lead to an error.

5. Feedback

Without feedback, we might begin to wonder what the system is doing and whether the input should be given again. When each input is received, the system normally provides some response. An object is highlighted, an icon appears, or a message is displayed. If processing of any operation cannot be completed within a few seconds, several feedback messages might be displayed to keep us informed of the progress of the system.

With function keys, feedback can be given as an audible click or by lighting up the key that has been pressed. Audio feedback has the advantage that it does not use up screen space, when messages are displayed on the screen, a fixed message area can be used so that we always know where to look for messages.

A typical raster feedback technique is to invert pixel intensities, particularly when making menu selections. Other feedback methods include highlighting, blinking, and color changes.

17.3 Input of Graphical Data Logical Classification of Input

Graphics programs support several kinds of input data. Any picture specifications need values for coordinate positions, values for the character-string parameters, scalar values for the transformation parameters, values specifying menu options, and values for identification of picture parts. To make graphics packages independent of the particular hardware devices used, input functions can be structured according to the data description to be handled by each Function.

The various kinds of input data are discussed in the following six logical device classifications used by PHIGS and GKS:

- LOCATOR - a device for specifying a coordinate position (x, y)
- STROKE - a device for specifying a series of coordinate positions
- STRING - a device for specifying text input
- VALUATOR - a device for specifying scalar value
- CHOICE - a device for selecting menu options
- PICK-a device for selecting picture components

Each of the six logical input device classification can be implemented with any hardware devices. But some hardware devices are more convenient for certain kinds of data than others. Like a device is pointed at a screen position is more convenient for entering coordinate data than keyboard.

1. Locator Devices

A standard method for interactive selection of a coordinate point is by positioning the screen cursor. We can do this with a mouse, joystick, trackball, space ball, thumbwheels, dials, a digitizer stylus or hand cursor device. When the screen cursor is at the desired location, a button is activated to store the coordinates of that screen point.

Keyboards can be used for locator input in several ways. The keyboard usually has four cursor-control keys that move the screen cursor up, down, left, and right. With an additional four keys, we can move the cursor diagonally as well. Rapid cursor movement is accomplished by holding down the selected cursor key. Alternatively, a joystick, joydisk, trackball, or thumbwheels can be mounted on the keyboard for relative cursor movement. Light pens have also been used to input coordinate positions, light pens operate by detecting light emitted from the screen phosphors, some nonzero intensity level must be present at the coordinate position to be selected. A light pen can be used as a locator by creating a small Light pattern for the pen to detect. The pattern is moved around the screen until it finds the light pen.

2. Stroke Devices

This class of logical devices is used to input a sequence of coordinate positions. The set of input points is often used to display line sections. Many physical devices used for generating locator input that can be used as stroke devices. The continuous movement of a mouse, trackball, joystick, or tablet hand cursor is translated into a series of input coordinate values. The graphics tablet is one of the more common stroke devices.

Button activation can be used to place the tablet into "continuous" mode. As the cursor is moved across the tablet surface screen, a stream of coordinate values is generated. This process is used in paintbrush systems that allow artists to draw scenes on the screen. While in engineering systems layouts can be traced and digitized for storage.

3. String Devices

Keyboard is the primary physical device used for string input. Input character strings are used for picture or graph labels. Other physical devices can be used for generating character patterns in a "text-writing" mode. For this input, individual characters are drawn on the screen with a stroke or locator-type device.

A pattern-recognition program then interprets the characters using a stored dictionary of predefined patterns.

4. Valuator Devices

This logical class of devices is used in graphic systems to input scalar values.

Valuators are used for setting various graphics parameter, such as rotation angle and scale factors, and for setting physical parameters associated with a particular application (temperature settings, voltage levels, stress factors, etc.).

A typical physical device used to provide valuator input is a set of control dials. Floating-point numbers within any predefined range are input by rotating the dials. Dial rotations in one direction increase the numeric input value, and opposite rotations decrease the numeric value. Rotary potentiometers convert dial rotation into a corresponding voltage. This voltage is then translated into a real number within a defined scalar range, such as -10.5 to 25.5.

Any keyboard with a set of numeric keys can be used as a valuator device. A user simply types the numbers directly in floating-point format, although this is a slower method than using dials or slide potentiometers. Joystick, trackball, tablets, and other interactive devices can be treated as valuator input by interpreting pressure or movement of the device relative to a scalar range. For example, one direction of movement i.e. left to right, increasing scalar values can be input. Movement in the opposite direction decreases the scalar input value.

Another technique for valuator input is to display sliders, buttons, rotating scales, and menus on the video monitor. Locator input from a mouse, joystick, and space ball device is used to select a coordinate position on the display, and the screen coordinate position is then converted to a numeric input value. Then the selected position on a scale can be marked with some symbol.

5. Choice Devices

A choice device is defined as one that enters a selection from a list (menu) of alternatives. Commonly used choice devices are a set of buttons, a cursor

positioning device, such as a mouse, trackball, or keyboard cursor keys. A function keyboard, or "button box", is often used to enter menu selections. Usually, each button is programmable, so that its function can be altered according to applications. Single-purpose buttons have fixed, predefined functions. Programmable function keys and fixed function buttons are often included with other standard keys on a keyboard.

Cursor control devices are used for screen selection of listed menu options. When a coordinate position (x, y) is selected, it is compared to the coordinate extents of each listed menu item. A menu item with vertical and horizontal boundaries at the coordinate values i.e. x_{min}, x_{max}, y_{min} and y_{max} is selected if the input coordinates (x, y) satisfy the inequalities

$$x_{min} \leq x \leq x_{max} \qquad y_{min} \leq y \leq y_{max}$$

A touch panel is used for larger menus with a few options displayed at a time. With a cursor-control device, such as a mouse, a selected screen position is compared to the area occupied by each menu choice.

Alternate methods for choice input is keyboard and voice entry. A standard keyboard can be used to type in commands or menu options. For method of choice input, some abbreviated format is useful. Menu listings can be numbered or given short identifying names. Similar codings can be used with voice-input systems, and implemented when the number of options is small (20 or less).

6. Pick Devices

Pick devices are used to select parts of a scene that are to be transformed or edited in later on. For example with a mouse or joystick, we can position the cursor over the primitives in a displayed structure and press the selection button. The position of the cursor is then recorded. First, the cursor position is compared to the coordinate extents of the various structures in the scene. If the bounding rectangle of a structure contains the cursor coordinates, the picked structure has been identified. But if two or more structure areas contain the cursor coordinates, further checks are necessary. If the cursor coordinates are determined to be inside the

coordinate extents of only one line, we have identified the picked object. Otherwise, we require additional checks to find out the closest line to the cursor position.

First method is to determine the closest line to the cursor position is to calculate the distance squared from the cursor coordinates (x, y) to each line segment whose bounding rectangle contains the cursor position. For a line with endpoints

(x_1, y_1) and (x_2, y_2), distance squared from (x, y) to the line is calculated as

$$d^2 = \frac{[\Delta x(y - y_1) - \Delta y(x - x_1)]^2}{\Delta x^2 + \Delta y^2}$$

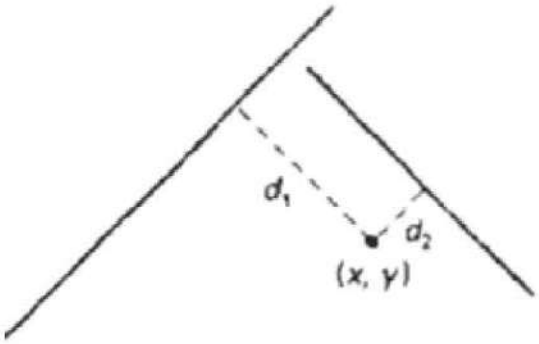


Figure 17.1: Distance to line segment from pick position.

Where $\Delta x = x_2 - x_1$ and $\Delta y = y_2 - y_1$.

Another method for the closest line to the cursor position is to specify the size of a pick window. The cursor coordinates are centered on this window and the candidate lines are clipped to the window, by making the pick window small enough, we can ensure that a single line will cross the window. To avoid for calculation we highlight the candidate structures and allow the user to resolve the pick ambiguity. One way for this is to highlight the structures that overlap the cursor position one by one. The user then signals when the desired structure is highlighted.

A second button is used to stop the process when the desired structure is highlighted. An additional button is used to help identify the structure. The first button can initiate a rapid successive highlighting of structures. A second button call again be used to stop the process, and a third button can be used to back up. Finally, we use a keyboard to type in structure names i.e. straightforward, but less interactive, pick-selection method. Descriptive name is used to help the user in the pick process.

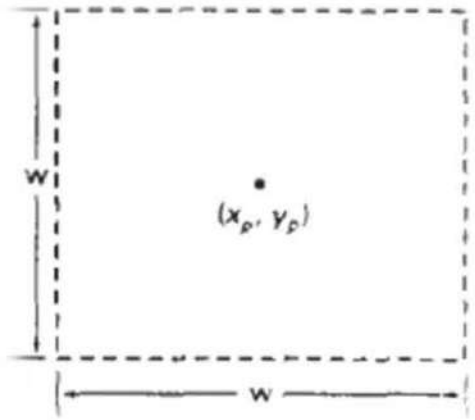


Figure 17.2: A pick window centered on pick coordinates, for resolving object overlapping.

17.4 Input Functions and Input Modes

INPUT FUNCTIONS

Graphical input functions allow users to specify the following options:

- Which physical devices are to provide input within a particular logical classification (for example, a tablet used as a stroke device)
- How the graphics program and devices are to interact (input mode). Either the program or the devices can initiate data entry, or both can operate simultaneously.
- When the data are to be input and which device is to be used at that time to deliver a particular input type to the specified data variables.

Input Modes

Input to functions can be structured to operate in various input modes which specify how the program and input devices interact. Program and input devices both could be operating simultaneously, or data input are initiated by the devices. There are three input modes that are request mode, sample mode, and event mode.

Request mode, the application program initiates data entry. Input values are requested and processing is suspended until the required values are received. The program and the input devices operate alternately. Devices are put into a wait state until an input request is made, then the program waits until the data are delivered.

Sample mode, the application program and input devices operate independently. Input devices may be operating at the same time when the program is processing other data. New input values from devices are stored, replacing older input data values.

Event mode, devices initiate data input to the application program. The program and the input devices again operate concurrently, but now the input devices deliver data to an input queue, all input data are saved. When the program requires new data, it goes to the data queue. Any number of devices can be operated at the same time in sample and event modes. But only one device at a time can be providing input in request mode.

An input mode within a logical class for a particular physical device operating on a specified workstation is declared with one of six input-class functions of the form:

Set . . . Mode (ws, deviceCode, inputMode. echoFlag)

Where ws denote workstation, deviceCode is a positive integer, inputMode may be request, sample, or event. And echoFlag is assigned either the value echo or the value noecho.

TABLE 8-1
ASSIGNMENT O F INPUT-DEVICE CODES

Device Code	*Physical Device Type*
1	Keyboard
2	Graphics Tablet
3	Mouse
4	Joystick
5	Trackball
6	Button

Using the assignments in this table, we could make the following declarations:

SetLocatorMode (1, 2, sample, noecho)
SetTextMode (2, 1, request. echo)
SetPickMode (4, 3, event, echo)

Thus, the graphics tablet is declared as a locator device in sample mode on workstation 1 with no input data feedback echo, the keyboard is denoted as a text device in request mode on workstation 2 with input echo, and the mouse is declared to be a pick device in event mode on workstation 1 with input echo.

1. Request Mode

When we ask for an input in request mode, other processing is suspended until the input is received. After a device has been assigned to request mode, input requests can be made to that device using one of the six logical-class functions represented by the following:

Request . . . (ws, deviceCode, status . . . .)

A value of *ok* or *none* (the input device was activated so as to produce invalid data) is returned in parameter status, according to the validity of the input data.

For locator input, none means the coordinates were out of range. For pick input, the device could have been activated while not pointing at a structure.

Locator and Stroke Input in Request Mode:

The request functions for these two logical input classes are.

RequestLocator (ws, devCode, status, viewIndex, pt)

RequestStroke (ws, devCode, nMax, status, viewIndex, n, pts)

For locator input, pt is the world-coordinate position selected.

For stroke input, pts is a list of n coordinate positions, where nMax gives maximum number of points that can go in the input list. And viewIndex is assigned the two-dimensional view index number.

Determination of a world-coordinate position is a two-step process: (1) the physical device selects a point in device coordinates and the inverse of the workstation transformation is performed to obtain the corresponding point in normalized device coordinates. (2) Then, the inverse of the window-to-viewport mapping is carried out to get to viewing coordinates, then to world coordinates.

View index 0 has the highest priority. ViewIndex identifies the reference viewing transformation, and parameter priority is assigned either the value lower or the value higher. For example, we may change the priority of the first four viewing transformations on workstation 1, as shown in Fig. 17-3, with the sequence of functions:

SetViewTransformationInputPriority (ws, viewIndex, refViewIndex, Priority)

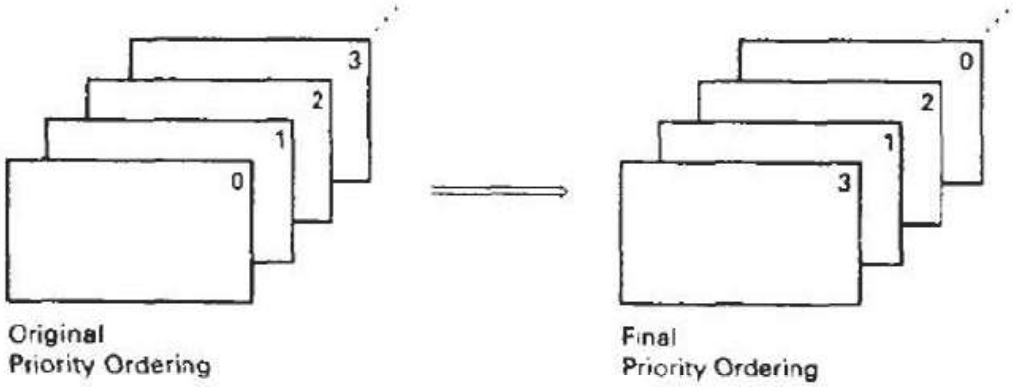


Figure 17.3: Rearranging Viewing Priorities

SetViewTransformationInputPriority (1, 3, 1, Higher)

SetViewTransformationInputPriority (1, 0, 2, Lower)

String Input in Request Mode

The request input function is:

RequestString (ws, devCode, status, nChars, str)

Parameter str is assigned an input string. nChars shows the number of characters in the string.

Valuator Input in Request Mode

A numerical value is input in request mode with function:

RequestValuator (ws, devCode, status, value)

Choice Input in Request Mode

We do a menu selection with the following request function:

RequestChoice (ws, devCode, status, itemNum)

itemNum is assigned a positive integer corresponding to the menu item selected.

Pick Input in Request Mode

We obtain a structure identifier number with the function

RequestPick (ws, devCode, maxPathDepth, status. pathDepth, pickpath)

Parameter pickpath is a list of information identifying the primitive selected. List contains the structure name, pick identifier for the primitive, and the element sequence number. Parameter pathDepth is the number of levels returned in pickpath, and maxPathDepth is the specified maximum path depth that can be included in pickpath.

2. Sample Mode

Once sample mode has been set for one or more physical devices, data input begins without waiting for program direction. If a joystick has been designated as a

Locator device in sample mode, coordinate values for the current position of the activated joystick are immediately stored.

Sampling of the current values from a physical device in this mode begins when a sample command is encountered in the application program. A locator device is sampled with following function:

Sample . . . (ws, deviceCode . . .)

Other input parameters are the same as in request mode. For example, suppose we want to translate and rotate a selected object. A final translation position for the object can be obtained with a locator, and the rotation angle can be supplied by a valuator device, as denoted by following:

SampleLocacor (ws1, dev1, viewIndex, pt)

SampleValuator (ws2. dev2, angle)

3. Event Mode

All input devices active in event mode can enter data (referred to as "events") into this single-event queue, with each device entering data values as they are generated. Data entered into the queue are identified according to logical class, workstation number, and physical-device code. An application program can be use following function for event queue:

AwaitEvent (time, ws, deviceClass, deviceCode)

Parameter time is for maximum waiting time for the application program in the queue i.e. number of seconds spent after arrival. The parameter deviceClass is assigned the value none. When time is given the value 0, the program checks the queue and immediately returns to other processing if the queue is empty. DeviceClass Codes, identifying the particular workstation and physical device that made the input.

For example, a set of points from a tablet (device code 2) on workstation 1 is input to plot a series of straight-line segments connecting the input coordinates denoted by:

GetLocator (viewIndex, pt)
SetStrokeMode (1, 2, event, noecho)

17.5 Initial Values Parameters

Quite a number of parameters can be set for input devices using the initialize function for each logical class:

Initialize . . . (ws, deviceCode, ..., pe, coordExt, dataRec)

Parameter pe is the prompt and echo type, parameter coordExt is assigned a set of four coordinate values, and parameter dataRec is a record of various control parameters.

For locator input, some values that can be assigned to the prompt and echo parameter are:

pe = 1: installation defined
pe = 2: crosshair cursor centered at current position
pe = 3: line from initial position to current position
pe = 4: rectangle defined by current and initial points

For structure picking, we have the following options:

pe = 1: highlight picked primitives
pe = 2: highlight all primitives with value of pick id
pe = 3: highlight entire structure

When an echo of the input data is requested, it is displayed within bounding rectangle specified by the four coordinates in parameter coordExt. Additional options can also be set in parameter dataRec. For example, we can set any of the following:

- size of the pick window
- minimum pick distance
- type and size of cursor display

- type of structure highlighting during pick operations
- range (min and max) for valuator input
- resolution (scale) for valuator input

17.6 Self Learning Exercise

Q.1. User can make any change on image with the use of

(A) Non interactive graphics (B) Interactive graphics

(C) Both a and b (D) None of these

Q.2. The device which allows screen positions to be selected with the touch of a finger.

(A) Touch Panel (B) Image Scanner

(C) Light Pen (D) Mouse

Q.3. A particular text index value is chosen with function

(A) setTextIndex() (B) setTextindex(ti)

(C) SetTextIndex(ti) (D) setTextIndex(ti)

Q.4. Which mode allows the input devices and program to operate concurrently?

(A) Event mode (B) Request mode

(C) Sample mode (D) none of these

17.7 Summary

We have discussed window manager interface scheme with menus and icons that allows users to open, close, reposition, and resize windows. Icons are graphical symbols that are designed for quick identification of application processes. Graphical interfaces are designed to manage consistency in user interaction with different user skill levels. In addition, interfaces are designed to minimize user memorization, to provide sufficient feedback, and to provide

adequate backup and error-handling capabilities. Then discussed six logical devices commonly use are locator, stroke, string, valuator, choice, and pick. Input functions are available in a graphics package can be defined in three input modes. Request mode places input under the control of the program. Sample mode allows the input devices and program to operate concurrently. And event mode allows input devices to initiate data entry and control processing of data.

17.8 Glossary

PHIGS: Programmer's Hierarchical Interactive Graphics System.

Open Look: It is a window manager for open windows developed by Sun Microsystem. Original window manager is X11 desktop environment.

17.9 Answers to Self-Learning Exercise

Ans.1: B **Ans.2:** A

Ans.3: D **Ans.4:** C

17.10 Exercise

Q.1. Explain design of the input function for the request mode.

Q.2. Explain design of the input function for the event mode.

Q.3. Explain design of the sample mode input function.

Q.4. Write short note on following input devices:

- Pick devices
- Locator devices
- Valuator Devices
- Choice Devices
- String Devices

Q.5. Write a short note on initial values parameters for various input devices.

Q.6. Explain the request mode with various input format with their function.

Q.7. What are different types of input modes? Explain in brief.

17.11 Answers to Exercise

All answers are discussed in chapter in details. See respective contents.

References and Suggested Readings

1. J. Foley, A. Van Dam, S. Feiner, J. Hughes: Computer Graphics- Principles and Practice, Pearson.

2. Hearn and Baker: Computer Graphics, PHI.

3. The evolution of the concept of logical for virtual input device is discussed in Wallace (1476). Input-device classifications is to be found in Newman (1068).

4. Input operations in PHIGS can be found in Hopgood and Dooe (1991), Howard etal (1491). For information on GKS input functions, see Hopgood etal. (1983).