UG CA S6-01 (DSC)

FOUR YEAR BACHELOR'S DEGREE COURSE

(CBCS in Accordance with NEP 2020)

Semester VI

COMPUTER GRAPHICS

COMPUTER SCIENCE

Hiranya Chandra Bhuyan School of Science and Technology

Krishna kanta Handiqui State Open University

Subject Experts

Prof. Shikhar Kumar Sarma, Department of Information Technology, Gauhati University
Prof. Hiren Kumar Deva Sarma, Department of Information Technology, Gauhati University
Dr. Sanjib Kumar Kalita, Department of Computer Science, Gauhati University

Programme Coordinators: Dr. Tapashi Kashyap Das, KKHSOU

Course Coordinators:

Dr. Tapashi Kashyap Das KKHSOU
Dr. Sruti Sruba Bharali, KKHSOU

Contributors

UNITS	Author
1	Dr.Tapashi Kashyap Das, KKHSOU
2, 3, 4, 5, 6, 8,11,12	Ms. Jonalee Barman Kakaty, NERIM
7, 13	Ms. Daisy Kalita, USTM
9	Dr. Atowar Ul Islam, USTM
10	Ms. Gitimoni Talukdar, Royal Global University
14,15	Mr. Arabinda Saikia, KKHSOU

Editorial Team

Content editing: Dr. Sanjib Kumar Kalita, Gauhati University (Unit 2 to Unit 15)
Dr. Sruti Sruba Bharali, KKHSOU (Unit 1)

Format editing: Dr. Tapashi Kashyap Das, KKHSOU

Print: June, 2026 ***ISBN:978-93-89123-30-2***

© Krishna Kanta Handiqui State Open University

This Self Learning Material (SLM) of the Krishna Kanta Handiqui State Open University is made available under a Creative Commons Attribution-Non Commercial-Share Alike 4.0 License (international): http://creativecommons.org/licenses/by-nc-sa/4.0/

Printed and published by Registrar on behalf of the Krishna Kanta Handiqui State Open University.

Headquarters: Patgaon, Rani Gate, Guwahati-781017
City Office: NH 37, Resham Nagar, Khanapara, Guwahati 781022;
Web: www.kkhsou.in

DETAILEE SYLLABUS

Computer Graphics

Unit		Pages
Unit 1:	**Introduction to Computer Graphics** Definition, Application Areas of Computer Graphics (CAD, CAM, Education and Training, Entertainment, Image Processing, Computer Art etc.), Graphical User Interfaces (GUI).	**7-21**
Unit 2:	**Graphics Systems** Cathode Ray Tubes, Random Scan Displays, Raster Scan Displays, Color CRT Monitors, Flat-Panel Displays (Plasma-Panels, Liquid Crystal Display (LCD), Electroluminescent displays), Graphics Software (GKS, PHIGS)	**22-29**
Unit 3:	**Line Drawing Algorithms** Points and Lines, Line Drawing Algorithms (DDA Algorithm, Bresenham's Line Algorithm)	**30-48**
Unit 4:	**Circle and Ellipse Drawing Algorithms** Circle Drawing Algorithms, Ellipse Drawing Algorithms	**49-59**
Unit 5:	**Filled Area Algorithms** Filling (Scan-Line Polygon Filling, Inside Outside Tests, Boundary-fill and Flood-fill Algorithm)	**60-68**
Unit 6:	**Transformations** Basic 2-D Transformations (Rotation, Reflection, Shearing, Scaling), Homogeneous Coordinate Representation, Translation, 3-D Transformations	**69-82**
Unit 7:	**2-D Viewing** 2-D Viewing: The Viewing Pipeline, Viewing Coordinate Reference Frame, Window to View-port Coordinate Transformation, Viewing FunctionsP (Cascading Style Sheet Positioning) and JSSS (JavaScript Assisted Style Sheet), Layers of Netscape, The ID Attribute, DHTML Events.	**83-95**
Unit 8:	**Clipping Algorithms** Line and Polygon Clipping Algorithms (Cohen-Sutherland and Cyrus-beck Line Clipping Algorithms, Sutherland- Hodgeman Polygon Clipping Algorithm)	**96-112**

Unit 9: 3-D Object Representation **7-21**

3-D Object Representation: Polygon Surfaces, Quadric Surfaces, Spline Representation, Basic Illumination Models, Polygon Rendering Methods

Unit 10: Bezier and B-spline Curves **22-29**

Hermite Curve, Bezier Curve and B-spline Curves, Bezier and B-spline Surfaces

Unit 11: Projections **30-48**

Projection Classification, Parallel Projections, Perspective Projections (One Point, Two Point)

Unit 12: 3-D Geometric Transformations **49-59**

Translation, Rotation, Scaling, Reflection and Shear Transformations, Composite Transformations

Unit 13: 3-D Viewing **60-68**

Viewing Pipeline, Viewing Coordinates, View Volume and General Projection Transforms and Clipping

Unit 14: Basics in Animation **69-82**

Animation Basic, Computer Animation, Principles of Animation, Types of Animation, Animation Software and Hardware, Common Terms in Animation

Unit 15: Animation Designing **83-95**

Design of Animation Sequence, Morphing, Application of Computer Animation, Future of Animation.

COURSE INTRODUCTION: COMPUTER GRAPHICS

Computer Graphics is an important branch of computer science that deals with the creation, storage, manipulation, and display of pictures and graphical objects using computers. It plays a major role in modern technology and is widely used in areas such as computer-aided design, education, entertainment, multimedia, gaming, image processing, and animation. Graphics also make interaction with computers easier and more attractive through graphical user interfaces (GUI).

This course has been designed for undergraduate learners to provide a basic understanding of computer graphics concepts, techniques, and applications. The course introduces learners to different graphics systems, line and circle drawing algorithms, filling methods, transformations, and viewing techniques used in two-dimensional and three-dimensional graphics.

The course also covers important topics such as clipping algorithms, 3-D object representation, Bezier and B-spline curves, projections, geometric transformations, and 3-D viewing. In addition, learners will gain an introduction to computer animation, animation principles, morphing techniques, and applications of animation in different fields.

The Self Learning Material (SLM) is written in a simple and learner-friendly manner. Each unit includes explanations, examples, illustrations, exercises, and self-assessment questions to help learners understand the concepts easily and study independently. After completing this course, learners will be able to understand the basic principles of computer graphics and apply various graphical techniques and algorithms in real-world applications.

UNIT 1: INTRODUCTION TO COMPUTER GRAPHICS

UNIT STRUCTURE

1.1 Learning Objectives
1.2 Introduction
1.3 Introduction to Computer Graphics
1.4 Application Areas of Computer Graphics
1.5 Graphical User Interface
1.6 Let Us Sum Up
1.7 Further Reading
1.8 Answers to Check Your Progress
1.9 Model Questions

1.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- explain the basic concepts of Computer Graphics.
- list the different application areas of Computer Graphics.
- describe the features and role of Graphical User Interface (GUI).

1.2 INTRODUCTION

In the modern world, graphics are used everywhere, such as in mobile phones, computer games, movies, websites, medical imaging, engineering design, and educational software. Earlier, computers were mainly text-based and difficult for common users to operate. With the development of graphics technology, computers became more interactive and visually attractive. Instead of typing complex commands, we now navigate effortlessly using everyday visual tools like icons, windows, menus, and images, making technology accessible to everyone.

In this unit, we will look at the basics of Computer Graphics, explore how they are used across different industries, and see how Graphical User Interfaces (GUIs) make our daily digital experiences smooth and effortless.

1.3 INTRODUCTION TO COMPUTER GRAPHICS

Computer Graphics is the branch of Computer Science that deals with the generation, manipulation, and storage of images and objects using digital computers. In simple words, computer graphics refers to creating images, drawings, charts, animations, and visual effects using computer systems. It allows users to communicate with computers in a visual form instead of only using text. It includes both two-dimensional (2D) and three-dimensional (3D) graphics. The basic components of computer graphics are:

- Hardware devices such as monitor, keyboard, mouse, and graphics card.

- Software programs used for creating and editing graphics.
- Algorithms and mathematical techniques used for image generation.

Types of Computer Graphics:

At a fundamental level, all digital graphics are categorized into two primary types based on how they are generated, processed, and stored by the computer's memory and display systems. They are:

- Raster (Bitmap) Graphics
- Vector Graphics

Raster Graphics: Raster graphics, commonly known as bitmaps, are digital images composed of a finite grid of individual *pixels* (picture elements). Each pixel is a tiny square that holds specific color and intensity data, defined by a specific number of bits (color depth). When viewed as a whole, this grid of pixels forms a complete image.

Digital photos and scanned images are common examples of raster graphics. Almost every image, banner, we see online uses raster formats like JPEG, PNG, or GIF.

Vector Graphics: Unlike raster graphics, vector graphics do not use pixels. Instead, they are mathematically defined. A vector image is a collection of geometric primitives - such as points, lines, curves, shapes, and polygons - which are calculated using mathematical equations and coordinates on a two-dimensional(2D) or three-dimensional (3D) plane.

For example, instead of storing millions of pixels to display a circle of a particular color, a vector file simply stores the mathematical formula: Center coordinates (x, y), Radius r, Border thickness t, and Fill color c.

Vector graphics can be enlarged or reduced in size without affecting their quality or clarity. Vector graphics can be seen everywhere in company logos and branding, ensuring they look crisp on a tiny business card or a massive billboard. We also rely on them for technical CAD (Computer-Aided Design) blueprints and the everyday fonts we type with so text never gets blurry when we zoom in. In our design work, we usually save these files in formats like SVG, AI, or EPS. The concept of raster and vector graphics will be discussed in detail in the next unit.

1.4 APPLICATION AREAS OF COMPUTER GRAPHICS

Computer Graphics is used in many areas of modern life because visual representation makes information easier to understand, attractive, and interactive. Instead of reading long textual information, users can quickly understand ideas through pictures, charts, animations, and graphical designs. Due to this reason, Computer Graphics has become an essential part of education, business, entertainment, science, engineering, medicine, and many other fields.

In this section, we will discuss some important application areas of Computer Graphics.

1. Graphical User Interfaces (GUI)

We interact with computer graphics every single second we look at a screen. Operating systems like Windows, macOS, Android, and iOS use GUI to create visual desktops, folders, and applications. This makes technology accessible to everyone, not just programmers.

2. Entertainment and Gaming

The entertainment industry is one of the largest users of Computer Graphics. Graphics are used in movies, cartoons, television programs, music videos, and video games. Modern films use special visual effects and 3D animation to create realistic scenes. Computer Graphics makes entertainment more attractive, interactive, and realistic.

For example, animated movies such as Frozen and Toy Story are created using computer graphics. Video games like racing games and action games use 3D graphics to provide realistic gaming experiences. Some of the uses in entertainment are:

- Animated movies
- Visual effects in films
- Video games
- Virtual reality applications
- Cartoon production

3. Computer-Aided Design (CAD)

In engineering, architecture, and manufacturing, we use computer graphics to create precise digital blueprints and 3D models. Before building a real car, an airplane, or a skyscraper, engineers design and test it virtually using CAD software. This saves time, cuts down on material waste, and prevents costly mistakes. Popular CAD software includes AutoCAD and SolidWorks.

4. Computer-Aided Manufacturing (CAM)

Computer-Aided Manufacturing (CAM) refers to the use of computer systems and software to assist, control, and automate manufacturing processes. Computer graphics plays a crucial role in CAM by providing visual tools for designing, analyzing, simulating, and producing industrial products. CAM systems use graphical representations of objects and components to guide manufacturing operations with high precision and efficiency.

In a typical manufacturing environment, a product is first designed using Computer-Aided Design (CAD) software. The graphical model created in the CAD system is

then transferred to the CAM system, which generates instructions for manufacturing equipment such as Computer Numerical Control (CNC) machines, robotic arms, and automated assembly systems. Through graphical visualization, engineers can inspect product designs, identify potential errors, and optimize manufacturing procedures before actual production begins.

Computer graphics enables the creation of accurate two-dimensional (2D) drawings and three-dimensional (3D) models of products. These models help manufacturers visualize the shape, dimensions, and assembly of components. The integration of CAD and CAM through computer graphics has significantly transformed modern manufacturing industries. It has reduced production time, improved accuracy, enhanced productivity, and lowered manufacturing costs. Today, CAM applications are widely used in automobile manufacturing, aerospace engineering, electronics production, industrial robotics, and rapid prototyping.

Thus, computer graphics serves as a vital component of Computer-Aided Manufacturing by providing powerful visualization, simulation, and automation capabilities that make modern manufacturing more efficient, reliable, and cost-effective.

5. Medical Imaging

Medical imaging is one of the most important applications of computer graphics in healthcare. Modern imaging technologies such as Magnetic Resonance Imaging (MRI), Computed Tomography (CT) scans, Ultrasound, and X-rays generate large amounts of raw data from the human body. Computer graphics techniques transform this data into clear and detailed 2D and 3D visual representations.

These visual models help doctors examine internal organs, bones, tissues, and blood vessels without the need for invasive procedures. Three-dimensional images provide a better understanding of complex anatomical structures, enabling accurate diagnosis and effective treatment planning. Computer graphics is used to assist surgeons in visualizing surgical procedures before performing them, thereby reducing risks and improving patient outcomes.

6. Education and Training

Computer Graphics has significantly enhanced the field of education and training by making learning more interactive, engaging, and effective. Graphical elements such as animations, illustrations, diagrams, simulations, and videos help learners understand complex concepts more easily than traditional text-based methods. Visual learning improves comprehension, increases student interest, and enhances memory retention.

In classrooms, teachers use computer-generated graphics to explain difficult topics in subjects such as science, mathematics, engineering, and medicine. For example, animations can demonstrate the movement of planets in the solar system, the structure and functioning of the human body, or the steps involved in chemical reactions. Interac-

tive graphical content allows students to explore concepts at their own pace and gain a deeper understanding of the subject matter.

Computer Graphics is also widely used in professional training. Flight simulators provide realistic training environments for pilots without the risks and costs associated with actual flights. Similarly, virtual laboratories enable students and researchers to perform experiments safely and repeatedly in a computer-generated environment.

Educational games and simulations make learning enjoyable while developing problem-solving and decision-making skills.

- Some important applications of Computer Graphics in education and training include:
- Smart classrooms
- E-learning platforms
- Educational games
- Interactive simulations
- Virtual laboratories

Thus, Computer Graphics has transformed education and training by providing visually rich, interactive, and immersive learning experiences.

7. Image Processing

Image processing involves improving or analyzing images in obtaining clearer and more useful visual information. Computer Graphics is widely used in medical imaging, satellite imaging, and photo editing applications. For example, doctors use MRI scans, CT scans, and X-ray images to diagnose diseases. Mobile applications and photo editing software use graphics techniques to improve image quality and add effects. Some of the applications of Image Processing are:

- Medical diagnosis
- Face recognition systems
- Security surveillance
- Satellite image analysis
- Photo editing software

8. Computer Art

Computer Graphics has opened new opportunities for creativity and digital art. Artists and designers use Computer Graphics to create digital artwork, paintings, logos, advertisements, and illustrations. Graphic design software provides various tools for drawing, coloring, and editing images.

For example, advertising companies create posters and banners using graphics software such as Adobe Photoshop and CorelDRAW. Digital artists also create paintings

and illustrations for books, websites, and social media. Some applications of Computer Art are:

- Logo design
- Digital painting
- Poster and banner design
- Web designing
- Advertisement creation

9. Scientific Visualization

Scientists use Computer Graphics to represent complex scientific data in visual form. Graphs, charts, and 3D models help researchers understand large amounts of information easily. For example, weather forecasting systems use graphics to show temperature changes, rainfall, and storm movement. Scientists also use graphics in space research and molecular modeling. Some of the applications are:

- Weather forecasting
- Space research
- Medical research
- Geological studies
- Data analysis

10. Simulation and Virtual Reality

Computer graphics is widely used in **simulation** and **virtual reality (VR)** to create realistic and interactive environments.

Simulation is the process of imitating real-world situations using computer-generated models. It is commonly used in flight simulators for training pilot, driving simulators, medical training, and military exercises. Simulations help users practice and learn safely without facing real-world risks.

Virtual Reality (VR) creates a computer-generated three-dimensional environment that users can explore and interact with. Using devices such as VR headsets and controllers, users can experience a sense of being present in a virtual world. Applications of VR include gaming, education, healthcare, architecture, tourism, and engineering. It helps users visualize complex objects and environments in an immersive way.

1.5 GRAPHICAL USER INTERFACE

Computer Graphics plays an important role in creating **Graphical User Interfaces** (**GUI**). GUI allows users to interact with computers through graphical elements such as windows, icons, menus, and buttons instead of typing commands.

GUI makes computer systems easy to learn and use. Instead of typing complex commands, users can perform tasks by clicking graphical objects using a mouse or touch screen. Examples of GUI-based operating systems are: Microsoft Windows, Linux, macOS, Android. The main features of GUI are:

- User-friendly interface
- Easy navigation
- Attractive appearance
- Supports multitasking

Elements of GUI

- *Windows*: A window is a rectangular area on the screen used to display applications or documents.
- *Icons*: Icons are small graphical symbols representing files, folders, or programs.
- *Menus*: Menus provide lists of commands or options.
- *Pointer*: The pointer is used to select objects on the screen using a mouse or touchpad.
- *Buttons*: Buttons allow users to perform actions such as saving or closing files.

Advantages of GUI: The absolute best thing about a Graphical User Interface (GUI) is just how simple and natural it makes using technology. We don't need to learn how to write code or memorize complicated commands just to use a computer. Instead, a GUI gives us a beautiful, friendly screen where everything is easy to understand.

Here are the main reasons why GUIs are so helpful in our daily lives:

- **It is user-friendly:** Because a GUI uses everyday visual symbols- like a little picture of a "folder" to hold our files- anyone can pick up a smartphone or laptop and figure it out almost instantly.
- **It looks attractive:** Instead using blank screen with text, we get a colorful interface with icons and menus that are pleasing to look at.
- **It supports multitasking:** We can easily do more than one thing at a time, like listening to music while typing a document or browsing a website.
- **Easy to use without memorizing code/commands:** We can simply point, click, or tap on what we want instead of typing out long, confusing commands.

Disadvantages of GUI: The primary drawback of a GUI is that it demands a lot of system resources. Generating thousands of colors, keeping track of moving windows, rendering smooth animations, and constantly refreshing the pixels on a high-resolution display requires a significant amount of computer memory (RAM) and processing power (CPU/GPU). A text-based interface uses almost zero processing power, whereas a modern, beautiful GUI requires robust hardware just to keep the screen running smoothly.

The basic difference between Graphical User Interface (GUI) and Command Line Interface (CLI) are:

Graphical User Interface (GUI)	Command Line Interface (CLI)
i. Uses graphical elements	i. Uses text commands
ii. Easy for beginners	ii. Requires command knowledge
iii.User-friendly	iii. Faster for programmers/experts
iv.Uses mouse and icons	iv. Uses keyboard commands



Figure.1.1(a): Views of Graphical User Interfaces (GUI) in a laptop and a mobile phone

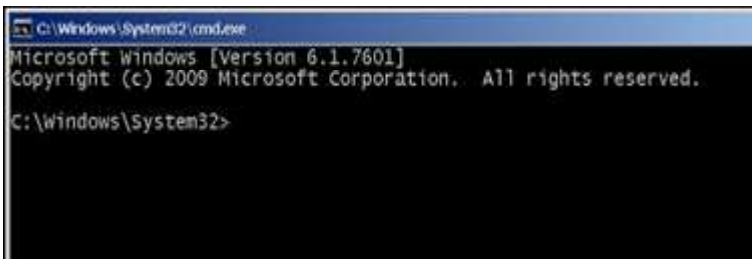


Figure.1.1(b): Views of Command Line Interface (CLI)

CHECK YOUR PROGRESS

Q1.What is Computer Graphics? Name the two types of computer graphics.
Q2.What are pixels?
Q3.What is CAD?
Q4.Mention two application areas of Computer Graphics.
Q5.How is computer graphics useful in education?
Q6.What is GUI? Name any two GUI operating systems.

1.6 LET US SUM UP

The important points discussed in this unit are summarized as follows:

- Computer Graphics is a branch of Computer Science that deals with the creation, manipulation, storage, and display of images using computers.

- It enables visual communication through images, drawings, charts, animations, and graphical objects.
- Two main types of Computer Graphics:
  - Raster (Bitmap) Graphics - composed of pixels; examples: JPEG, PNG, GIF.
  - Vector Graphics - based on mathematical equations; examples: SVG, AI, EPS.
- Raster graphics lose quality when enlarged, whereas vector graphics can be scaled without losing clarity.
- Major application areas of Computer Graphics:
  - Graphical User Interfaces
  - Entertainment and Gaming
  - Computer-Aided Design (CAD)
  - Medical Imaging
  - Education and Training
  - Image Processing
  - Computer Art
  - Scientific Visualization
  - Simulation and Virtual Reality
- GUI (Graphical User Interface) allows users to interact with computers through windows, icons, menus, buttons, and pointers.
- Common GUI-based operating systems include Microsoft Windows, macOS, Linux, and Android.
- Advantages of GUI:
  - User-friendly
  - Attractive appearance
  - Supports multitasking
  - No need to memorize commands
- Disadvantages of GUI:
  - Requires more memory and processing power
  - More resource-intensive than CLI
- GUI and CLI differ in terms of interaction methods, ease of use, and system resource requirements.

Understanding these concepts provides the foundation for studying advanced topics in computer graphics.

1.7 FURTHER READING

1. Hearn, D., & Baker, M. P. (2004). *Computer graphics* (2nd ed.). Pearson Education.
2. Maurya, R. K. (2021). *Computer graphics with multimedia* (3rd ed.). Wiley India.

✓ 1.8 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q.No. 1: Computer Graphics is the art and science of creating, manipulating, and storing pictures, images, and visual data using a digital computer. It transforms abstract numbers and computer code into visual representations that humans can easily see and understand. The two primary types of computer graphics are Raster (Bitmap) Graphics and Vector Graphics.

Ans. to Q.No. 2: A pixel (short for "picture element") is the smallest individual dot or square of color that makes up a digital image on a screen. Think of them like tiny mosaic tiles; when millions of pixels are arranged in a grid and viewed together from a distance, they form a cohesive, detailed picture. Each pixel contains specific data about its color and brightness.

Ans. to Q.No. 3: CAD stands for Computer-Aided Design. It is a specialized area of computer graphics that uses software to create precise, detailed 2D blueprints and 3D digital models of physical objects. Architects, engineers, and designers rely on CAD to plan, test, and virtually build everything from skyscrapers and airplanes to electronic circuits before they are actually manufactured.

Ans. to Q.No. 4: Two major application areas of Computer Graphics are:

- Entertainment: Used to build immersive 3D worlds in video games and to create stunning visual effects (CGI) in movies.
- Medical Imaging: Used to convert raw data from MRI, CT scans, and X-rays into clear 3D digital models, allowing doctors to look inside the human body safely.

Ans. to Q.No. 5: Computer graphics have completely transformed the way we teach and learn by making education visual, interactive, and engaging. Everyone learns differently, and trying to understand complex, abstract concepts purely by reading static text in a book can be incredibly difficult. Computer graphics bridge this gap by bringing lessons to life.

Instead of just reading about how a volcano erupts or how a human heart pumps blood, students can watch colorful, accurate animations that show the process step-by-step. In science classes, graphics allow students to interact with 3D models of molecules or solar systems, zooming in and rotating them to see how they work. Furthermore, high-end graphics power training simulators, allowing student pilots to practice flying airplanes or medical students to practice surgeries in a safe, virtual environment where mistakes don't cost real lives.

Ans. to Q.No. 6: A GUI (Graphical User Interface) is a visual system that allows us to interact with a computer using easy-to-understand visual elements like icons, windows,

menus, and pointers instead of typing text-based commands. Two common examples of GUI operating systems are Microsoft Windows and Apple macOS.

1.9 MODEL QUESTIONS

1. What is Computer Graphics? Mention the types of computer graphics
2. Write a short note on:
   a) Computer-Aided Design (CAD).
   b) Computer-Aided Manufacturing (CAM)
3. How is Computer Graphics used in scientific visualization?
4. What is GUI? Describe the main elements of a GUI.
5. Describe the various application areas of Computer Graphics with examples.
6. Explain the role of Computer Graphics in entertainment, gaming, and animation.
7. Describe the importance of Computer Graphics in education and training systems.
8. Describe how Computer Graphics is used in medical imaging and healthcare.
9. Compare GUI and CLI highlighting their advantages and disadvantages.
10 Discuss the role of Computer Graphics in CAD, CAM, scientific visualization, and simulation systems.

UNIT 2: GRAPHICS SYSTEMS

UNIT STRUCTURE

2.1 Learning Objectives
2.2 Introduction
2.3 Cathode Ray Tubes
    2.3.1 Random Scan Displays
    2.3.2 Raster Scan Displays
2.4 Color CRT Monitors
2.5 Flat Panel Displays
    2.5.1 Plasma Panels
    2.5.2 Liquid Crystal Displays
    2.5.3 Electroluminescent Displays
2.6 Graphics Software
2.7 Color Models
2.8 Let Us Sum Up
2.9 Further Reading
2.10 Answers to Check Your Progress
2.11 Model Questions

2.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- learn how the CRT monitors works and its types
- learn the different types of Flat-panel displays used in computers.
- learn about different color models.

2.2 INTRODUCTION

In the previous unit we have learnt some basic of Computer Graphics. We have also learnt the application areas of computer graphics and about Graphical User Interface(GUI).

In this unit, we will discuss about the display techniques used in the computer to display images. We will also introduce how colored pictures are generated in computer. Graphics devices can be devided into two major areas: (i) the display systems where the graphics are presented and (ii) the hardware components with which the user writes necessary instruction to generate the graphics.

2.3 CATHODE RAY TUBES

The Cathod Ray Tube (CRT) is a specialized vacuum tube that produces images when an electron beam strikes a phosphorescent screen. Previously most video monitors operates with CRT technology.

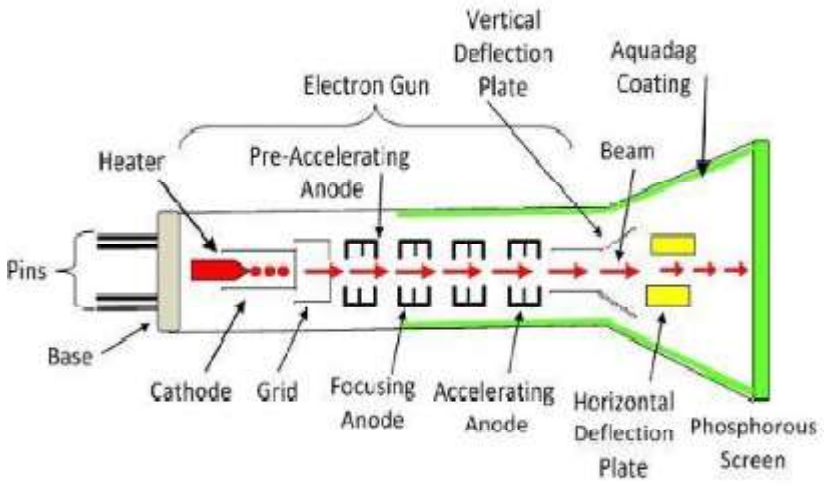


Figure. 2.1: Basic design of a CRT

In this section we will discuss the operation of a CRT. An electron beam emitted by an electron gun, passes through a focusing and deflection systems that direct the beam towards specified positions on the phosphor coated screen. The phosphor then emits light at the positions where the beam has strike. A property of phosphors is that the light emitted fades very rapidly. This property is called persistence. ***Persistence*** is defined as the time it takes the emitted light from the screen to decay to one-tenth of its original intensity.

So to maintain the picture on the screen without flicker, the picture is redrawn repeatedly by quickly directing the electron beam back over the same points. This is called ***Refresh CRT***. The lower persistence phosphors require higher refresh rates to keep the screen picture.

A cathode ray tube consists of several basic components. The electron gun generates an electron beam. The electron gun in a CRT is a heated cathode and a control grid that fits over the cathode. The anodes accelerate the electrons. The deflecting coils produce a very low frequency electromagnetic field that allows for constant adjustment of the direction of the electron beam. There are two sets of deflecting coils: horizontal and vertical. The electron beam when strikes the phosphor coated screen produces a bright spot on the screen. The intensity of the beam can be controlled by setting voltage levels on the control grid. A smaller negative voltage decreases the number of electrons passing through the small hole at the end of the control grid. A higher negative voltage repels the electrons and stops them from passing through.

The maximum number of points that can be displayed on the screen without overlap on the CRT is called the ***Resolution*** and is usually given as the number of horizontal points versus the number of vertical points. These points are called ***pixels***. For example, a monitor can have a resolution of 1024 × 768 pixels. The number of pixels determines the resolution of an image.

Aspect ratio is the ratio of the vertical points to the horizontal points to produce equal length lines in both directions on the screen.

2.3.1 Random Scan Displays

In Random Scan Display, the electron beam is directed only to those parts of the screen where the picture has to be drawn. It draws the picture one line at a time. So it is also called vector, stroke writing or calligraphic displays.

The following figure shows how the Random Scan Display works. The component lines of the picture can be drawn in any specified order.

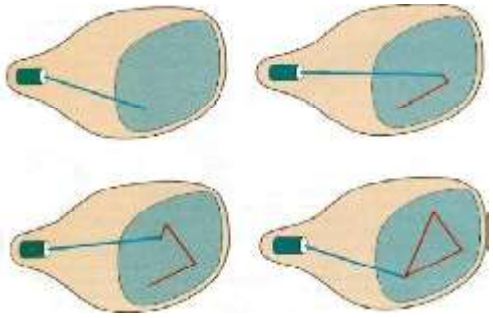


Figure.2.2: Lines are drawn in any order in a random scan system

The picture definition, in random scan displays is stored as a set of line drawing commands in a memory area called the refresh display file. The refresh rate depends on the number of lines in the picture and generally the lines of the pictures are drawn 30 to 60 times per second.

The advantages of Random Scan Displays are:

- Very high resolution as the picture definition is stored as a set of line commands and not as a set of intensity values.
- Produces smooth line drawings as the electron beam directly follows the line path
- Requires little memory.

The disadvantages are:

- Cannot draw complex pictures or realistic scenes.
- Cost is high.

A pen plotter and a laser light show are also random displays. A simple architecture of the random display systems is shown below:

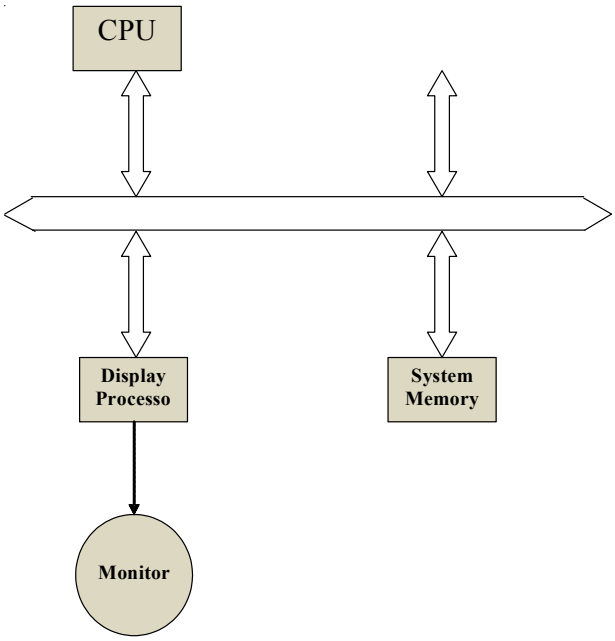


Figure.2.3: Architecture of a Random display system

2.3.2 Raster Scan Displays

In Raster Scan display, the electron beam sweeps the screen from top to bottom one row at a time in a regular pattern. The television uses this type of technology.

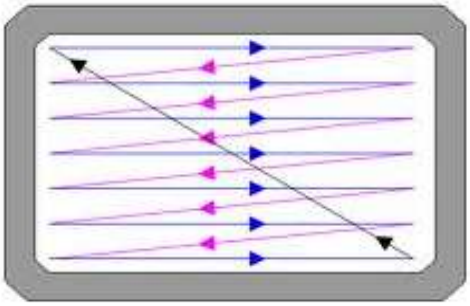


Figure.2.4: Raster Scan system

Unlike random scan systems which were a line drawing system, the raster scan CRT is a point plotting device. The electron beam is turned on and off to create a collection of bright spots. The picture definition is stored as a set of pixel intensity values in an area of memory called the *frame buffer*. Frame buffer is also called *bit map* or *refresh buffer*. The rows in a raster display are called *scan lines*. Figure 2.5 shows how the beam sweeps across each scan line.

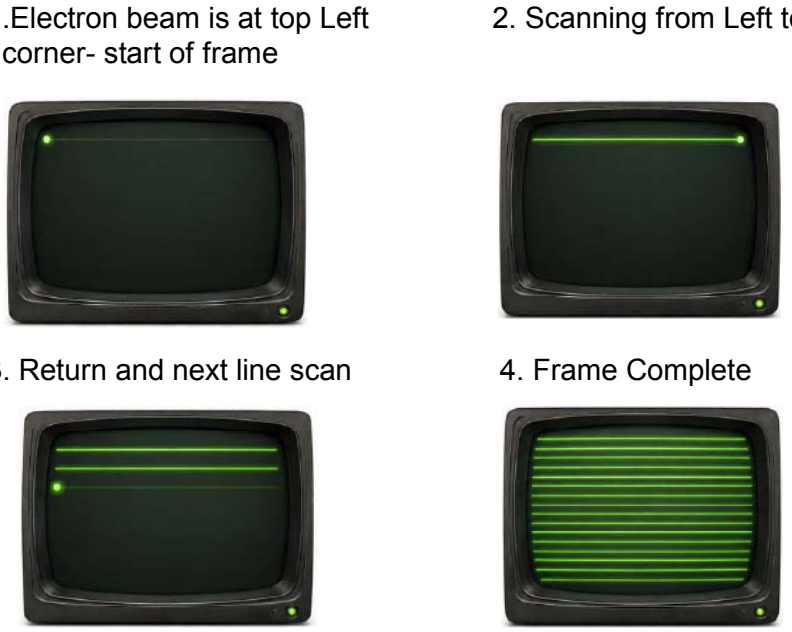


Figure 2.5: Raster scan

The raster scan displays are refreshed at a rate of 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 scan the next line. The return to the left side of the screen, after refreshing each scan line, is called the ***horizontal retrace*** of the electron beam. At the end of each frame, the beam returns to the top left corner of the screen to begin the next frame. This is called the ***vertical retrace.***

On some raster systems, each frame is displayed in two passes using an interlaced refresh process. In the first pass, the beam sweeps across every other scan line from top to bottom. Then the vertical retrace sweeps the remaining lines. A simple architecture of the Raster Scan systems is shown in figure 2.6.

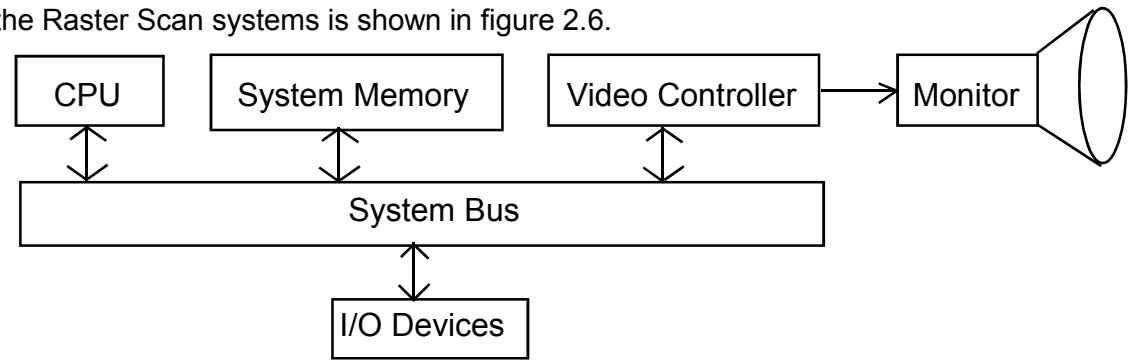


Figure 2.6: Architecture of a raster display system

The advantages of Raster Scan displays are:

- Realistic scenes can be drawn.
- Decreasing memory costs has made raster systems popular.

The disadvantages are:

- They have less resolution
- The lines produced are zig-zag as the plotted values are discrete.

2.4 COLOR CRT MONITORS

A color CRT contains a million of red, blue and green phosphors that emits different colored lights by their combination when strike on the screen. Red, green and blue are called primary colors. Any other color can be generated by mixing these three primary color with different intensity.

There are two basic techniques for producing colors displays:

a) Beam Penetration method (in Random Scan CRTs)
b) Shadow-mask method (in Raster Scan CRTs)

a) Beam Penetration Method: The inside of the CRT screen is coated with two layers of phosphors, usually the outer layer is red and the inner layer is green. The displayed color depends on how far the electron beam has been able to penetrate the phosphor layers.

A beam of slow moving electrons can excite only the outer red layer. A fast moving electron beam can penetrate the outer layer and excite the inner green layer. If the beam speed is not very fast or not very slow, then it displays two colors– yellow and orange.

The disadvantage of this method is:

- It can produce only four colors
- Picture quality is not so good.

b) Shadow-mask Method: The raster scan display systems use the shadow mask method to display color as they can display a broad range of colors compared to the beam penetration method. Each pixel position has three colored phosphor– red, green and blue. This type of CRT has three electron guns, one for each color dot. To create a crisp picture on the CRT screen, it is necessary that the beam for each color strikes only the correct dot intended. This is done by using a fine metal mesh called a shadow mask just behind the phosphor coated screen. The shadow mask contains a series of holes aligned with the phosphor dot pattern. The electron beam strikes the screen phosphors through this mask. When the three beams pass through a hole in the shadow mask, they activate a dot triangle, which appears as a small color spot on the screen. The phosphor dots in the triangles are arranged so that each electron beam can activate only its corresponding color dot when it passes through the shadow mask.

Color variations can be obtained by varying the intensity levels of the three beams. The displayed color depends on the amount of excitation of the red, blue and green phosphors.

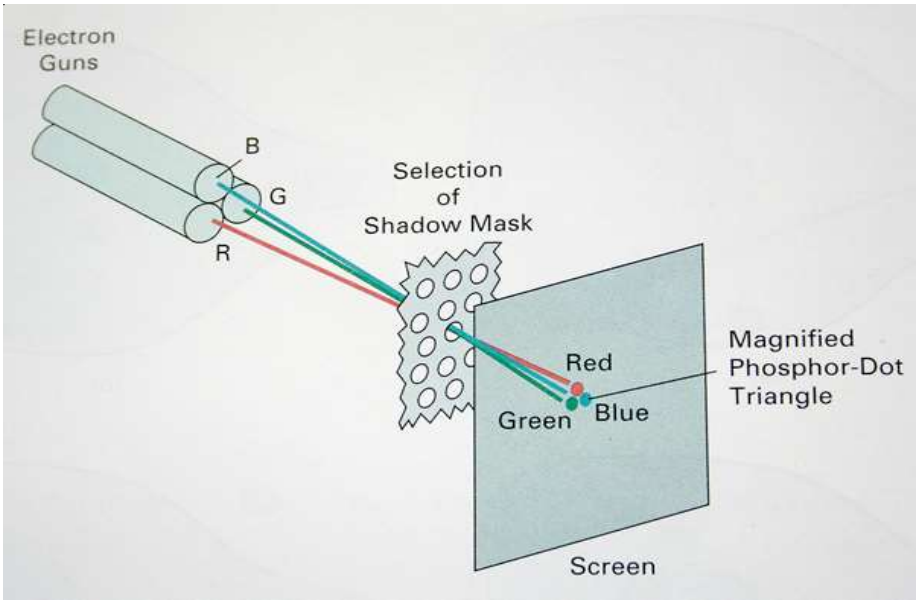


Figure. 2.7: Operation of a Shadow-Mask CRT

So, by combining different intensities from the three electron guns different colors can be obtained. For example,

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

2.5 FLAT PANEL DISPLAYS

The flat panel displays are also called Flat Screens. The technology in flat panel displays are much different from the CRTs. They are much lighter and thinner and are usually less than 4 inches thick. They also consumes much less power than the traditional CRTs. Flat-panel displays can be divided into two categories:

a) Emissive Displays and b) Non-emissive Displays

The emissive displays converts electrical energy into light. Examples of emissive displays are Plasma panels, LEDs (Light Emitting Diodes), thin-film electroluminescent displays etc.

Non-emissive displays use optical effects to convert sunlight or light from other

sources into graphical patterns. LCD (Liquid Crystal Displays) is an example of non-emissive display. Flat panel screens have a precise matrix of rows and columns based on the highest resolution supported.



Figure 2.8: Flat Panel vs CRT

2.5.1 Plasma Panels

The plasma panels are also called **gas-discharge displays**. They are called plasma displays because the pixels on the screen rely on plasma cells or florescent lamps. The plasma panel produces images on the screen pixel by pixel. The above figure shows the construction of a plasma panel. It consists of two glass plates. The region between the glass plates is filled with neon based gas and is sealed. Long electrodes called the display electrode and address electrode are also put between the glass plates.

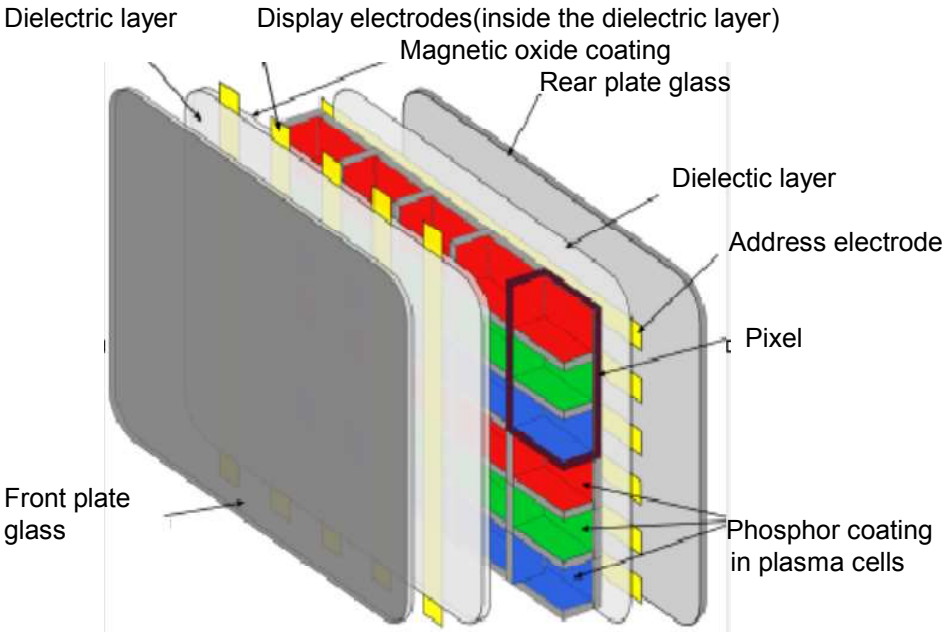


Figure. 1.9: Plasma Panels

By applying voltage between the electrodes, the gas is made to behave as if they are tiny cells which are independent of each other. These cells are called plasma. These cells are made to glow by applying a very high voltage across it by means of the electrodes. Refreshing is not required as the pixel glow can be sustained by maintaining a high frequency alternating voltage across the cells. Each pixel consists of three different colored phosphors - red, green and blue.

Advantages:

- Produces a flicker free steady image.
- Refreshing is not required.
- Slim profile and less bulky than CRTs.
- Better and more accurate color reproduction.
- Images do not suffer from degradation at high angles

Disadvantages:

- Costlier than CRTs.
- Susceptible to reflection glare in bright rooms.

1.8.2 Liquid Crystal Displays (LCD)

The term liquid crystal refers to the fact that these compounds have a crystalline arrangement of molecules, yet they flow like a liquid.

The basic structure of a LCD can be thought of as two glass plates each containing a light polarizer sandwiching a layer of nematic (thread-like) liquid crystal. The polarized light passes through the liquid crystals and gets twisted following the direction in which the molecules are arranged. It will then pass through the other polarizer and reflected

back to the viewer.

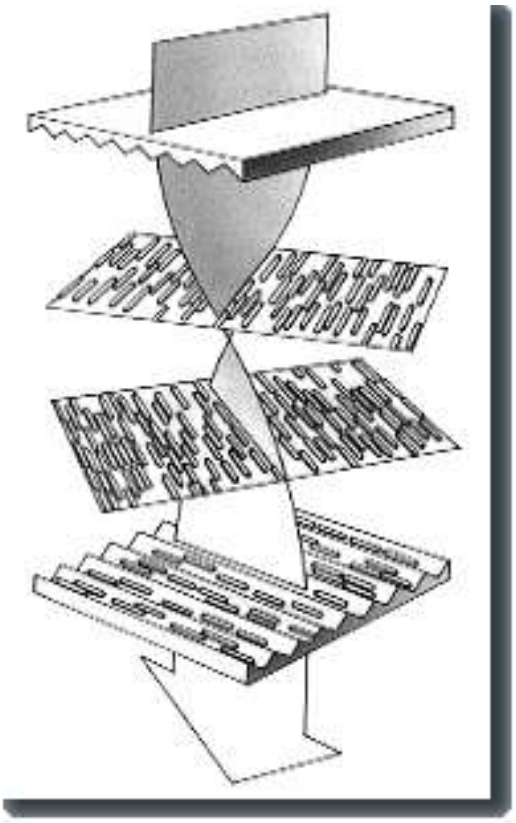


Figure 2.10: Structure of Liquid Crystal DisplaysI (LCD)

On applying voltage, the molecules rearrange themselves vertically and the light is not twisted. So it will not pass through the other polarizer and the pixel will thus be turned off. Thus when no voltage is applied to the liquid crystals, light passes. Through and when a voltage is applied the light gets blocked and the screen appears black.

Advantages:

- They are thin and compact.
- Energy efficient. Consumes much less power.
- Produces very bright images.
- Completely flat screen.

Disadvantages:

- Restricted viewing angles.
- Contrast is lower than CRTs.
- More expensive than CRTs.

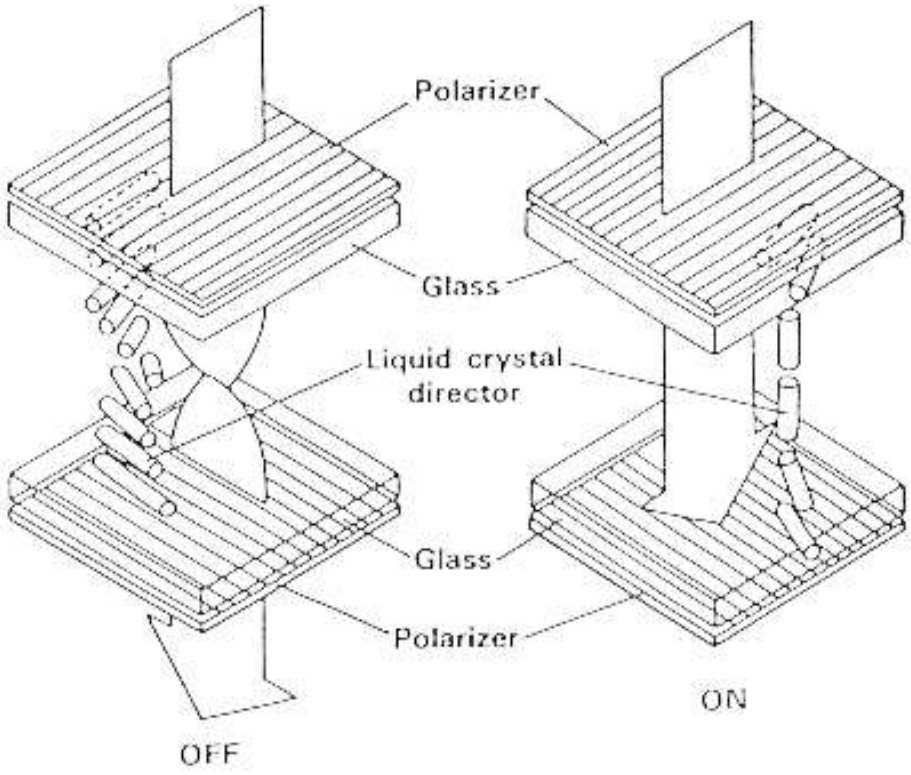


Figure 2.11: Liquid Crystal Display

2.5.3 Electroluminescent Displays

Electroluminescent (EL) Displays produce light when an alternating current (AC) voltage is applied across an electroluminescent phosphor layer. The basic structure of an EL display consists of a thin phosphor layer sandwiched between a transparent front electrode and a back electrode. When AC voltage is applied between these electrodes, the phosphor material emits visible light, which is viewed through the transparent front panel.Luminescence refers to the emission of visible radiant energy due to causes other than heat. Electroluminescence is a specific type of luminescence in which light is generated by a material exposed to an electric field without significant thermal energy production.

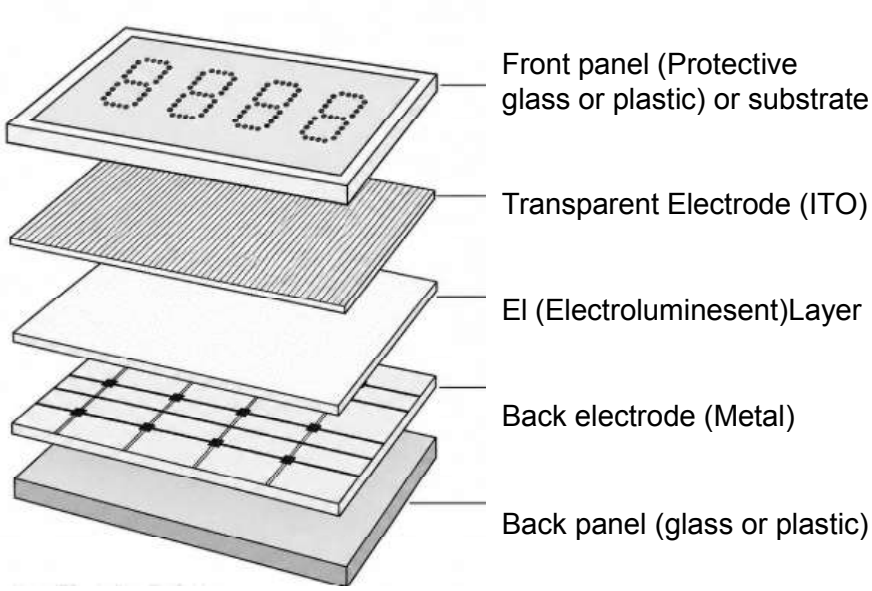


Figure 2.12: Monochrome EL Display Structure

The construction of an Electroluminescent Display (ELD) is similar to that of a plasma panel display. It consists of a transparent front layer, known as the substrate, coated on its rear surface with a transparent electrically conductive film that forms the front electrode. Above this electrode is a layer of electroluminescent phosphor material placed between two insulating layers. A conductive film deposited over the upper insulating layer acts as the back electrode. When an AC electric field is applied across the electrodes, the phosphor layer emits light, producing the display image. Monochrome EL displays generate light in a single color and are known for their high brightness, wide viewing angle, fast response time, and long operational life

2.6 GRAPHICS SOFTWARE

A graphics software is a software that is used to create, edit and manage computer graphics. The graphics may be any picture, web graphics, logos, headings or any other kind of digital images. There are two main types of graphics software:

- pixel-based image editors or Raster graphics editor
- vector-based image editors or Vector graphics editor

Raster graphics editors: This is the most popular type of graphics software. The images created and edited in this type of software are called *bitmaps* because they are made up of a set of pixels. The pixel based editors are used to edit the digital photos and scanned images as these are always pixel-based. Very realistic pictures can be created in these type of editors. They are resolution dependent. The images are saved in the computer in any of the following bitmap formats: jpeg, gif, png and tiff.

Some raster graphics editors are: *Adobe Photoshop, Microsoft Paint, Paint Shop Pro,*

*Krita, Adobe Photoshop Elenments,*etc.

Vector graphics editors: They allow to create images using lines and curves. They can be easily edited by moving the points, adjusting curves and changing the objects color. They are resolution independent. These softwares are not used for creating realistic pictures rather used for creating graphic designs, logos, cartoons etc. The files of vector graphics format are: SVG(Scsble vector grahics) ,AI(Adobe Illustrator Artwork), PDF (Portable Document format), EPS (Encapsulated postscript) WMF (Windows Metafile), etc.

Some vector graphics editors are: *Adobe Illustrator, CorelDRAW, Macromedia Freehand, Affinity Designer, LibreOffice Draw,* etc.

2.7 COLOR MODELS

A color model is a system for creating a range of colors using the primary colors. There are two types of color models : additive and subtractive. Additive color models uses light to display the colors and the subtractive color model uses printing inks. The two most common color models are:

- RGB Color Model,
- CMYK Color Model.

RGB Color Model: Computer uses the RGB color model to display colored images. The color displayed results from the light transmitted from the electron guns. The combination of the three colors: red, blue and green produces the white color. The overlapping of the red, blue and green colors produces the subtractive colors (cyan, magenta and yellow).

The RGB color model can produce a broad range of colors. This model is device-dependent. The computer, TV, camera, mobile phones all uses the RGB color model.

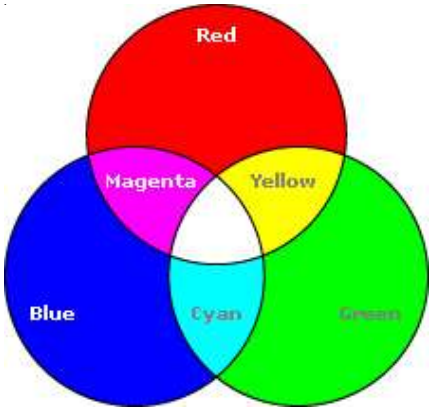


Figure 2.13 : RGB color model

CMYK Color Model : This model uses the subtractive primary colors to produce different colors. It is used for printing and therefore uses ink to display color. The combination of the three colors produces the Black color.

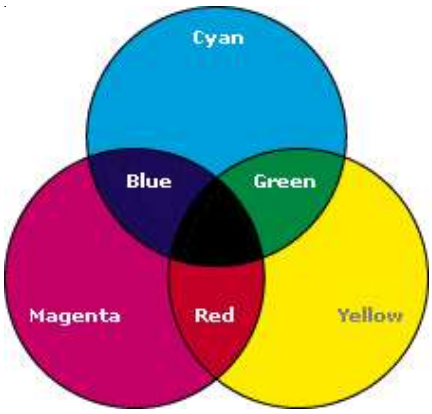


Figure 2.14: CMYK color model

The overlapping of the subtractive colors produces the additive colors that are Red, Green and Blue colors. The colors produced are not as bright as the colors produced in the RGB model.

CHECK YOUR PROGRESS

Q1: Indicate whether the following statements are True or False:

a) The CRT is a tube that produces images when an electron beam strikes a phosphorescent screen.

b) Persistence is a property of phosphors that allows the emitted light maintain its glow on the screen.

c) The components of an electron gun in a CRT is a heated cathode and a control grid that fits over the cathode.

d) Resolution of a monitor is independent of the number of pixels that can be displayed on the screen.

e) Aspect ratio is the ratio of the vertical points to the horizontal points to produce equal length lines in both directions on the screen.

f) Random scan displays is also called calligraphic displays.

g) Random Scan displays are very useful in creating realistic scenes.

h) Raster Scan CRTs is a point plotting output device.

i) Horizontal retrace in raster scan displays is the return of the electron beam to the left side of the screen, after refreshing each scan line.

j) The beam penetration method in color CRTs can produce a range of variety

colors.

Q2: Fill in the Blanks :

a) The Flat panel displays are also called .....
b) The two categories of flat panel displays are ..... and .....
c) The plasma panels are also called .....
d) the region between the two glass plates in plasma panels is filled with a ..... gas.
e) The polarized light in an LCD passes through the liquid crystals and gets twisted following the direction in which the ..... are arranged.
f) ..... is the production of visible light by a substance exposed to an electric field without thermal energy generation.
g) The images created and edited in ..... are called bitmaps because they are made up of a grid of pixels.
h) ..... are resolution independent.
i) the two common color models are ..... and .....
j) The overlapping of the red, blue and green colors in the RGB color model produces the subtractive colors .....

2.8 LET US SUM UP

- The Cathode Ray Tube (CRT) is a type of vacuum tube which displays images when the electron beam through electron guns are strikes on the phosphor coated screen.
- The picture is redrawn repeatedly by quickly directing the electron beam back over the same points so as to maintain the picture on the screen. This is called **Refresh CRT**.
- The electron beam when strikes the phosphor coated screen produces a bright spot on the screen.
- The maximum number of points that can be displayed on the screen without overlap on the CRT is called the **Resolution** and is usually given as the number of horizontal points versus the number of vertical points.
- **Aspect ratio** is the ratio of the vertical points to the horizontal points to produce equal length lines in both directions on the screen.
- In Random Scan display, the electron beam is directed only to those parts of the screen where the picture has to be drawn. It draws the picture one line at a time.
- In Raster scan display, the electron beam sweeps the screen from top to bottom one row at a time in a regular pattern.
- There are two basic techniques for producing color displays Beam Penetration method and Shadow- mask method.
- Flat Panel displays are lighter, thinner and requires less power than CRTs.

- The Plasma Panel consists of two glass plates and the region between the plates is filled with a neon-based gas.
- The electroluminescent displays are created by sandwiching a layer of electroluminescent material between two layers of conductors.
- There are two graphics software – raster graphics editor and vector graphics editor.
- RGB and CMYK are the two most commonly used color models.

2.9 FURTHER READING

1) Hearn, Donald and Baker, M. Pauline. *Computer Graphics*. Prentice-Hall.
2) Shirley, Peter; Marschner, Steve. (3rd Edition). *Fundamentals of Computer Graphics*.

2.10 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: a) True, b) False, c) True, d) False, e) True, f) True, g) False, h) True, i) True, j) False

Ans. to Q. No. 2: a) flat screens; b) emissive displays, non-emissive displays; c) gas-discharge displays; d) Neon-based; e) Crystal molecules; f) Electroluminescent; g) raster graphics editors; h) Vector graphic editors; i) RGB model, CMYK model; j) cyan, magenta and yellow.

2.11 MODEL QUESTIONS

Q1. Explain the working of Cathode Ray Tube (CRT) technology. Discuss the concepts of persistence, resolution, aspect ratio, and refresh rate.

Q2.Compare Raster Scan Displays and Random Scan Displays with suitable diagrams, advantages, and disadvantages.

Q3.Describe the architecture and working of a Raster Scan Display System with a neat diagram.

Q4.Discuss Color CRT Monitor technology. Explain the Beam Penetration Method and Shadow-Mask Method for color generation.

Q5.What are Flat Panel Displays? Explain the construction, working, advantages, and disadvantages of Plasma Panel Displays, LCDs, and Electroluminescent Displays (ELDs).

Q6.Differentiate between Emissive and Non-Emissive Displays with suitable examples.

Q7.Explain the structure and working principle of Liquid Crystal Displays (LCDs). Mention their advantages and disadvantages.

Q8.Explain the construction and working of Plasma Panel Displays. Discuss their features,

advantages, and limitations.

Q9.Discuss Graphics Software in Computer Graphics. Differentiate between Raster Graphics Editors and Vector Graphics Editors with suitable examples and file formats.

1Q10.Explain the RGB Color Model and CMYK Color Model. Compare them and discuss their applications.

Q11.Discuss the role of primary colors in Computer Graphics and explain how different colors are generated in CRT monitors.

Q12.Write short notes on the following:

i. Electroluminescent (EL) Displays

ii. Raster Graphics Editors

iii. Vector Graphics Editors

iv. Flat Panel Displays

*** ***** ***

UNIT 3: LINE DRAWING ALGORITHMS

UNIT STRUCTURE

3.1 Learning Objectives
3.2 Introduction
3.3 Points and Lines
3.4 Line Drawing Algorithms
    3.4.1 DDA (Digital Differential Analyser)
    3.4.2 Bresenham's Algorithm
3.5 Let Us Sum Up
3.6 Further Reading
3.7 Answers to Check Your Progress
3.8 Model Questions

3.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- learn cartesian coordinate system
- learn how the different line drawing algorithms work.

3.2 INTRODUCTION

In the previous unit we have learnt about the different video display devices. The raster scan and random scan display techniques are also elaborated. We have also come across the different color models.

Computer graphics comprises of pictures or images, animations, videos, films created using computers. It refers to computer-generated image with the help of graphical hardware and software. Computer graphics is responsible for displaying art and image data effectively and meaningfully to the consumer. Points and lines are the simplest components of pictures. Others include circles, ellipses, surfaces, polygons and also somewhere character strings.

In this unit we will start with some raster graphics algorithms that will draw straight lines or curves on the display unit. We will also learn how to fill the boundaries and the areas in and out of polygon.

3.3 POINTS AND LINES

The most commonly used display method in a Cathode Ray Tube (CRT) is the Raster Scan Display. It is used for realistic display of scenes containing shading and colour pattern. Each point on the screen is referred to as a *pixel,* shortened for 'picture

element'. The picture definition is stored in a memory area called the *refresh buffer* or the frame buffer. The screen positions or we can say the pixel positions are referenced in *Cartesian Coordinates*. The coordinate origin is defined at the lower left screen corner with the positive values of x- coordinate increasing to the right of screen and positive values of y- coordinate increasing from bottom to the top of screen. Points and lines are the fundamental in drawing a picture on the screen. A point is simply a pair of x, y coordinates (x,y) on the display and is graphically represented as a dot. Plotting a point on the screen is nothing but illuminating a phosphor dot on the screen. However a line is an infinite series of points. It connects two points. They can be straight or curved and continuous or broken. In plotting a line, the starting coordinates and the endpoint coordinates of the line segment should be specified.

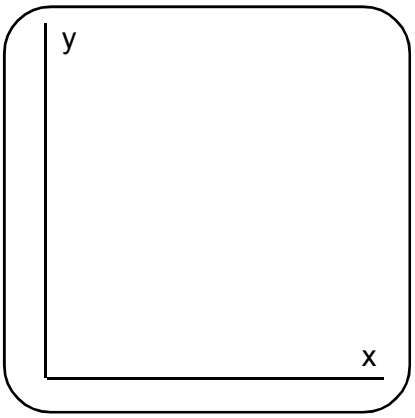


Figure 3.1: Cartesian coordinate

3.4 LINE DRAWING ALGORITHMS

Before going into the details of line drawing algorithms, we have to know some definitions.

Rasterization: It is the process of determining which pixels provides the best approximation to a line on the CRT screen.

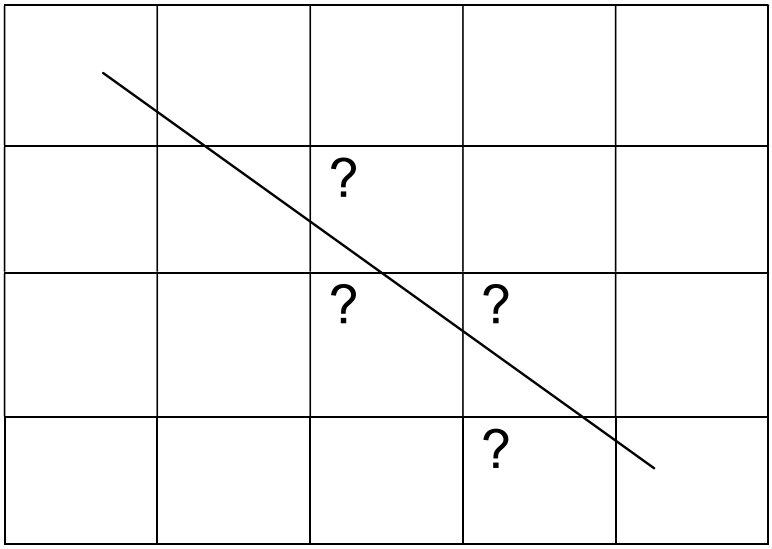


Figure 3.2: Example of one pixel in two coordinate

Scan Conversion: The generation of the picture in scan line order combined with rasterization is called Scan Conversion.

It is always important that the straight lines must appear straight and they should have constant brightness along them.

The goal of any line drawing algorithm is to construct the best possible approximation of a line. Following are the some of the qualities that are often considered while drawing the line on the raster display:

- Continuous appearance of the line
- Uniform thickness and brightness
- Are the pixels nearest to the ideal line turned on or not
- The line generation speed.

3.4.1 Digital Differential Analyser (DDA) Algorithm

A straight line may be defined by two endpoints and an equation. In figure 3.3 the two endpoints are described by (x_1, y_1) and (x_2, y_2). The equation of the line is used to describe the (x, y) coordinates of all the points that lie between the two endpoints. Using the equation of a straight line, $y = mx + b$ where $m = \triangle y/\triangle x$ and b = the y intercept. We can find the values of y by incrementing x from $x = x_1$ to $x = x_2$. By scan converting these calculated x, y values, we represent the line as a sequence of pixels.

The differential equation of a straight line is given by

dy/dx = constant

or, $\triangle y / \triangle x = y_2 - y_1 / x_2 - x_1$

= constant

where (x_1, y_1) and (x_2, y_2) are the end points of a line and are assumed to be not equal.

Now, $\triangle y = m\triangle x$, where m is the slope of the line (3.1)

Similarly, $\triangle x = \triangle y / m$ (3.2)

Equations 3.1 and 3.2 helps in determining the deflection voltages in analog devices.

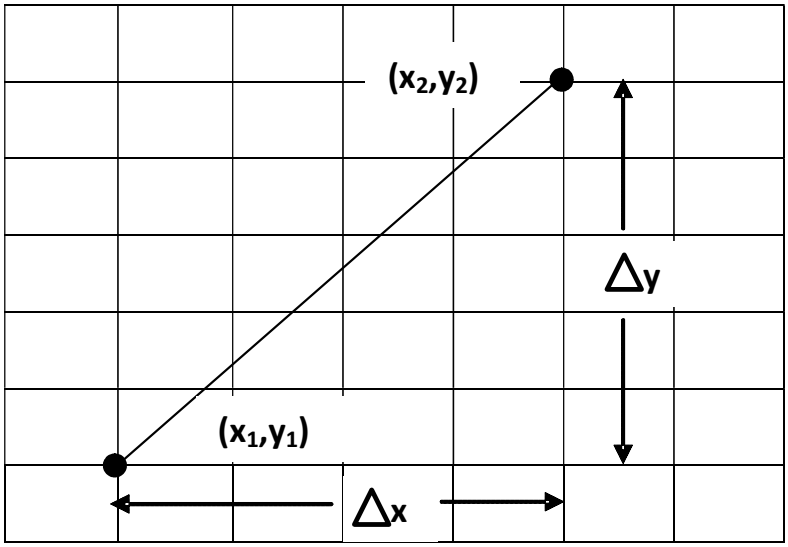


Figure 3.3: Straight line

The DDA Algorithm is a scan conversion algorithm and is based on calculating either $\triangle x$ or $\triangle y$ with the help of equations 3.1 and 3.2.

In raster systems, the lines are made of pixels one after another and the pixel separation forms a constraint in the step sizes in the vertical and horizontal directions. Therefore, sampling of a line at discrete positions and then determine the nearest pixel to the line at each sampled position becomes necessary.

Sampling is done at one coordinate and the corresponding integer values nearest the line path is determined for the other coordinate.

Considering a line with a positive slope, i.e. m<=1, we will sample at unit x intervals ($\triangle x = 1$) and the calculate each of the successive y coordinate values.

$\triangle y / \triangle x = m$

$(y_{k+1} – y_k) / x_{k+1} – x_k) = m$, where k = 1, 2, 3, …, …, …

$y_{k+1} – y_k = m\triangle x$

$y_{k+1} – y_k = m$ (as $\triangle x = 1$)

$y_{k+1} = y_k + m$ (3.3)

The value of k starts at 1 and will go on increasing till it reaches the endpoint. Also, the value of m can be any real number, so the value of y which is calculated must be rounded to the nearest integer value.

This above calculation was for lines with positive slopes but with lines with negative slopes i.e. when m > 1, we sample at unit y intervals ($\triangle y = 1$) and calculate the corresponding integer x values as:

$$x_{k+1} = x_k + 1/m \quad (3.4)$$

If the starting point is at the right side, then we may either have–

$\triangle x = -1$, and

$$y_{k+1} = y_k - m \quad (3.5)$$

or, $\triangle y = -1$, and

$$x_{k+1} = x_k - 1/m \quad (3.6)$$

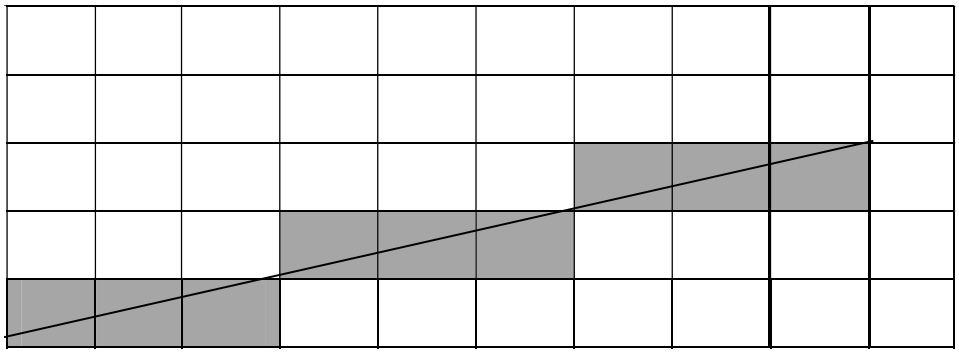


Figure 3.4: Pixels generated by DDA Algorithm

The function for the DDA Algorithm is given as follows:

```
linedda( int x1, int y1, int x2, int y2, int color)
{
    int dx, dy, step, k, roundx, roundy;
    float x, y,xincr, yincr;
    dx = x2-x1;
    dy = y2-y1;
    if(abs(dx) > abs(dy))
        step = abs(dx);
    else
        step = abs(dy);
    xincr = dx / step;
    yincr = dy / step;
    x = x1;
    y = y1;
    roundx = round(x);
    roundy = round(y);
    putpixel(roundx,roundy,color);
```

```
    for( k=1;k<=step;k++
    {
        x = x + xincr;
        y = y + yincr;
        roundx = round(x);
        roundy = round(y);
        putpixel(roundx,roundy,color);
    }
}
```

The DDA algorithm runs rather slowly because it requires real arithmetic (floating point operations).

3.4.2 Bresenham's Line Algorithm

The Bresenham's algorithm was developed by Jack Bresenham in 1965 at IBM. It is another incremental scan conversion algorithm and the advantage is that it uses only integer arithmetic thereby runs significantly faster as integer arithmetic has the advantage of speed and precision. Working with floating point values requires more time and memory and such values would need to be rounded to integers anyway.

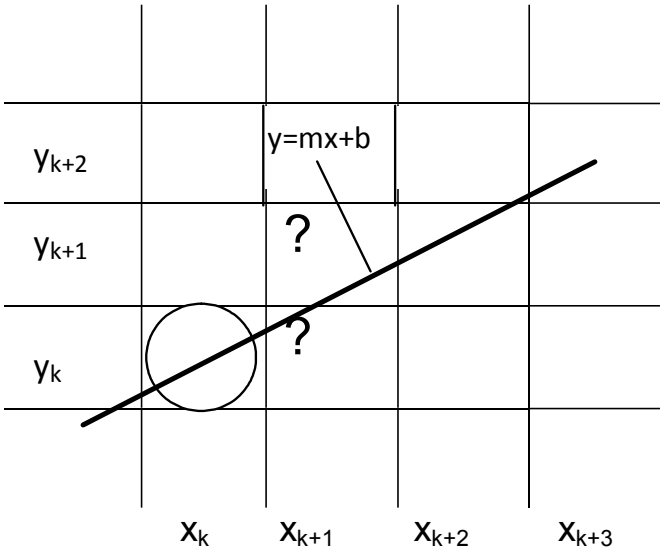


Figure 2.5: Scan line

Let us consider drawing a line with a positive slope, i.e., m<1. The pixel positions along the line are determined by sampling at unit x intervals.

Starting from the left end point of the given line we move across the x-axis

in unit intervals and go on plotting the pixel whose scan line y-axis value is closest to the path of the line.

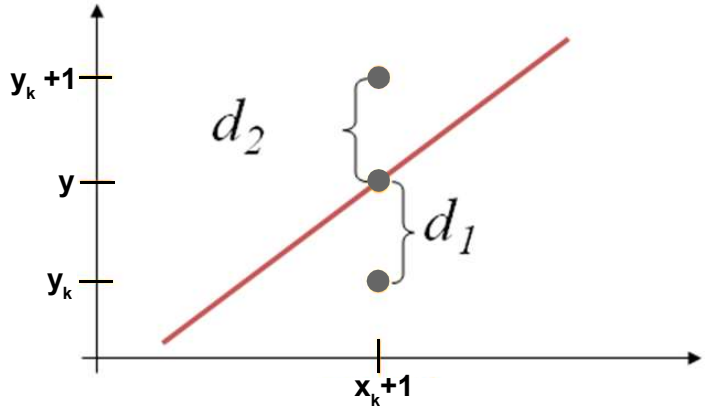


Figure 2.6:

Having plotted a pixel at (x_k, y_k), the line drawing algorithm has a limited range of options as to where to plot the next pixel in the coordinate system.

- Whether to plot the point at (x_k+1, y_k), or
- Whether to plot the point at (x_k+1, y_k+1).

So the algorithm has to decide between such two possibilities at each step.

The above figure shows the situation in which the algorithm finds itself in plotting the k^{th} pixel.

At sample position x_k+1 the vertical separations from the mathematical line are labeled as d_1 and d_2.

The y-coordinate on the mathematical line at x_k+1 is as follows:

$$y = m(x_k+1) + b$$

So, d_1 and d_2 are as given below:

$$d_1 = y - y_k = m(x_k+1)+b - y_k$$

and $d_2 = (y_k+1) - y = y_k+1 - m(x_k+1) - b$

We can use these to decide which pixel is more closer to the mathematical line and this simple decision is based on the difference between the two pixel positions.

$$d_1 - d_2 = 2m(x_k+1) - 2y_k + 2b - 1$$

Now, substituting m with $\triangle y / \triangle x$ and multiplying both sides by $\triangle x$:

$$x(d_1 - d_2) = \triangle x(2\triangle y / \triangle x(x_k+1) - 2y_k + 2b - 1)$$

$$= 2\triangle y.x_k - 2\triangle x.y_k + 2\triangle y + \triangle x(2b-1)$$

$$= 2\triangle y.x_k - 2\triangle x.y_k + c$$

where c is a constant and $c = 2\triangle y + \triangle x(2b-1)$

A decision parameter p_k for the k^{th} step is given by:

$p_k = \triangle x(d_1 - d_2) = 2\triangle y.x_k - 2\triangle x.y_k + c$

The sign of the decision parameter p_k is same as that of $d_1 - d_2$.

If p_k is negative then we choose the lower pixel, otherwise we choose the upper pixel.

At step k+1, the decision parameter p_{k+1} is as:

$p_{k+1} = 2\triangle y.x_{k+1} - 2\triangle x.y_{k+1} + c$

Substracting p_k from p_{k+1} we get the following:

$p_{k+1} - p_k = 2\triangle y(x_{k+1} - x_k) - 2\triangle x(y_{k+1} - y_k)$

But x_{k+1} and x_k+1 are same and is equals to 1, so we get,

$p_{k+1} = p_k + 2\triangle y - 2\triangle x(y_{k+1} - y_k)$

where $y_{k+1} - y_k$ is either 0 or 1 depending on the sign of p_k.

This iteration is repeated until the end point of the mathematical line is reached.

The Bresenham's Line Drawing Algorithm for m<1 is given as follows:

1) Input the two line end points, storing the left end point in (x_0,y_0).
2) Plot the point (x_0, y_0).
3) Calculate the constants $\triangle x$, $\triangle y$, $2\triangle y$ and $(2\triangle y - 2\triangle x)$ and get the first value for the decision parameter as: $p_0 = 2\triangle y - \triangle x$.
4) At each x_k along the line, starting at k = 0, perform the following tests.
   If $p_k < 0$ then the next point to plot is (x_k+1, y_k) and $p_{k+1} = p_k + 2\triangle y$.
   Otherwise the next point to plot is:
   (x_k+1, y_k+1) and $p_{k+1} = p_k + 2\triangle y - 2\triangle x$
5) Repeat the step number 4, $\triangle x$ times.

CHECK YOUR PROGRESS

Q.1: Fill in the blanks:

a) The process of determining which pixels provides the best approximation to a line on the CRT screen is called .....

b) The definition of the picture is stored in a memory area called the .....

c) The two things that has to be specified in plotting a line are ..... and .....

Q.2: What are the qualities that are considered while drawing a line on the screen?

.....

.....

.....

.....

.....

Q.3: Suppose a point is plotted at position (x_k, y_k) on the screen coordinate system. What are the possibilities for a line drawing algorithm to plot the next point on the screen?

.....

.....

.....

.....

.....

Q.4: What are the advantages of the Bresenham's Line drawing algorithm over the Digital Differential Analyser (DDA) algorithm.

.....

.....

.....

.....

.....

.....

.....

3.5 LET US SUM UP

- A pixel is a point (x, y) on the display screen and a line is an infinite series of such points.
- Rasterization is the process of determining which pixels provides the best approximation to a line on the CRT screen.
- Scan Conversion is the generation of the picture in scan line order combined with rasterization.
- Two different line drawing algorithms discussed above are DDA (Digital Differential Analyser) and Bresenham's Line drawing algorithms.
- The DDA algorithm works slow as it involves floating point operations. On the other hand, Bresenham's algorithm is faster in terms of speed and precision as it involves only integer arithmetic.

3.6 FURTHER READING

1) Hearn, Donald and Baker, M. Pauline. *Computer Graphics*. Prentice-Hall
2) Newman and Sproull. (1979). *Principles of Interactive Computer Graphics*. McGraw-Hill.
3) Plastock and Kalley. (1986). *Computer Graphics, Schaum's Outlines*. McGraw-Hill.

3.7 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: a) rasterization; b) frame buffer; c) starting coordinates, endpoint coordinates.

Ans. to Q. No. 2: The following are some of the qualities that are often considered while drawing the line on the raster display.

i) Continuous appearance of the line
ii) Uniform thickness and brightness
iii) Are the pixels nearest the ideal line turned on or not
iv) The line generation speed.

Ans. to Q. No. 3: Having plotted a pixel at (x_k, y_k), the line drawing algorithms has a limited range of options as to where to plot the next pixel in the coordinate system. They are:

i) Whether to plot the point at (x_k+1, y_k), or
ii) Whether to plot the point at (x_k+1, y_k+1).

So the algorithm has to decide between such two possibilities at each step.

Ans. to Q. No. 4: The Bresenham's Line drawing algorithm has the following advantages:

i) It is a fast incremental algorithm
ii) It uses only integer calculations

On comparing the Bresenham's algorithm to the DDA algorithm, the DDA algorithm has the following drawbacks:

i) Accumulation of round-off errors can make the pixelated line drift away from what was intended
ii) The rounding operations and floating point arithmetic are time-consuming.

3.8 MODEL QUESTIONS

Q.1: Write the DDA algorithm and the Bresenham's line drawing algorithm. Compare both these two algorithm.

Q.2: The endpoints of a given line are (0, 0) and (6, 18). Compute each value of y as x steps from 0 to 6 and plot the result.

Q.3: What are the steps required to plot a line using the slop method.

Q.4: What are the steps required to plot a line whose slope is between 0 and 45^o using Bresenham's method.

*** ***** ***

UNIT 4: CIRCLE AND ELLIPSE DRAWING ALGORITHMS

UNIT STRUCTURE

4.1 Learning Objectives
4.2 Introduction
4.3 Circle Generating Algorithms
    4.3.1 Mid-Point Circle Algorithm
4.4 Ellipse Generating Algorithms
    4.4.1 Mid-Point Ellipse Algorithm
4.5 Let Us Sum Up
4.6 Further Reading
4.7 Answers to Check Your Progress
4.8 Model Questions

4.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- learn how to draw circle and ellipse using Computer Graphics
- learn about circle generating algorithm
- learn about ellipse generating algorithms.

4.2 INTRODUCTION

In the previous unit we have discussed few line drawing algorithms. We have learnt how to draw lines using Bresenham's line drawing algorithm and DDA.

In this unit we will start with circle drawing algorithms that will draw circle and ellipse on the display unit. A circle is a symmetrical figure. Any circle generating algorithm can take the advantage of the circle symmetry to plot eight points for each value that the algorithm calculate. Eight-way symmetry is used by reflecting each calculated point around each 45^o axis. For example, if point 1 is calculated with a circle drawing algorithm, then seven more points could be found by reflection as shown in figure 4.1.

4.3 CIRCLE GENERATING ALGORITHM

A circle is defined as a set of points that are at a given distance r from a centre position (x_c, y_c).

The equation for a circle is: $x^2 + y^2 = r^2$

where, r is the radius of the circle.

From the above equation we can calculate the y-coordinate values as: $y = \pm\sqrt{r^2 - x^2}$

These equations can be used to calculate the points on a circle circumference at unit x intervals.

However, this method has some disadvantages:

- The resulting circle has large gaps where the slope approaches the vertices.
- It involves considerable calculations.

So a more efficient and accurate solution is required.

The computations required in generating a circle can be reduced by considering the symmetry of the circles. The shape of the circle is similar in each quadrant. The circle section in the first and second quadrants are symmetric with respect to the y-axis. Circle sections in the third and fourth quadrants can be obtained from sections in the first and second quadrants by considering the symmetry about the x-axis. There is also symmetry between octants. Circles thus have eight way symmetry.

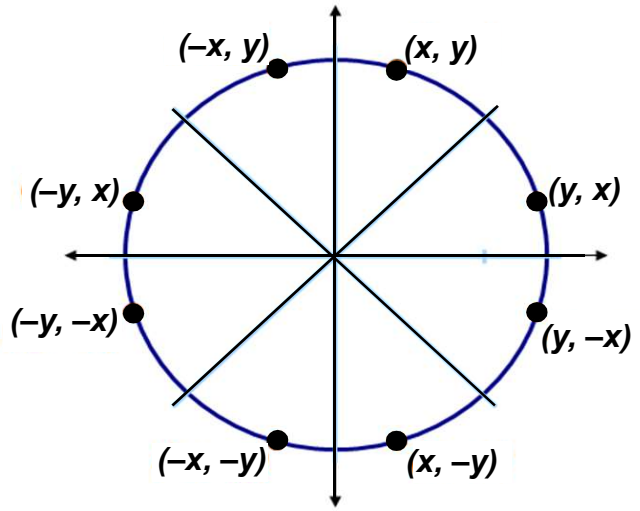


Figure 4.1 : Symmetry of a circle

This symmetry conditions are illustrated in the figure 4.1 where calculation of a circle point at (x, y) in one octant yields the circle points shown for the seven other octants.

The more efficient circle algorithms are based on incremental calculations of decision parameters as in the case with drawing of lines.

4.3.1 Mid-Point Circle Algorithm

In the Mid-Point circle algorithm, the eight-way symmetry of circles is used. In this algorithm, only the points for the top right eighth of a circle are calculated and then use symmetry to obtain the rest of the circle circumference points.

As in the case of line drawing algorithms, we sample at unit intervals and determine the closest pixel to the circle path.

Let us define a circle function, f_{circle} as: $f_{circ}(x, y) = x^2 + y^2 - r^2$

where (x,y) is any point on the boundary of the circle with radius r.

The above equation evaluates as follows:

$$f_{circ}(x, y) \begin{cases} < 0, & \text{if } (x, y) \text{ is inside the circle boundary} \\ = 0, & \text{if } (x, y) \text{ is on the circle boundary} \\ > 0, & \text{if } (x, y) \text{ is outside the circle boundary} \end{cases}$$

If the point (x,y) is in the circle interior, then the circle function is negative and if it lies outside the circle boundary, then the circle function is positive.

By evaluating this function at the mid-point between the candidate pixels we can make the decision. So this circle function is the decision parameter in the mid-point circle algorithm.

Suppose the pixel at (x_k, y_k) has been plotted. So the next step is to choose a point that is nearest to the actual circle path. Here the two possibilities are: $(x_k + 1, y_k)$ and $(x_k + 1, y_k - 1)$.

The decision parameter can then be defined as:

$$\begin{aligned} p_k &= f_{circ}(x_k + 1, y_k - ½) \\ &= (x_k + 1)^2 + (y_k - ½)^2 - r^2 \end{aligned}$$

If $p_k < 0$ then the mid-point is inside the circle and the pixel at y_k is closer to the circle. Otherwise the mid-point is outside and y_k–1 is closer.

The successive decision parameters are obtained using incremental calculations. Now,

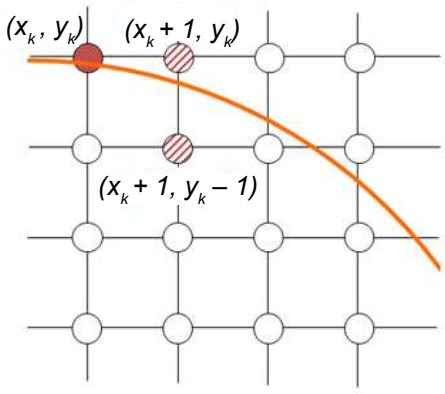


Figure 2.8: Circle scan conversion

$p_{K+1} = f_{circ}(x_{k+1} + 1, y_{k+1} - ½)$

$= [(x_k + 1) + 1]^2 + (y_{k+1} - ½)^2 - r^2$

Or, $p_{k+1} = p_k + 2(x_k + 1) + (y^2_{k+1} - y^2_k) - y_{k+1} - y_k) + 1$

where y_{k+1} is either y_k or y_k–1 depending on the sign of p_k

The first decision variable is given as: $p_0 = f_{circ}(1, r - ½)$

$= 1 + (r - ½)^2 - r^2$

$= {}^5/_4 - r$

If $p_k < 0$ then the next decision parameter is as: $p_{k+1} = p_k + 2x_{k+1} + 1$

Otherwise, $p_{k+1} = p_k + 2x_{k+1} + 1 - 2y_k + 1$

The **Mid-Point Circle Algorithm** is as follows:

1) Input the radius r and circle center (x_c, y_c) and obtain the first point on the circumference of a circle centered on the origin as: $(x_0, y_0) = (0, r)$
2) Calculate the initial value of the decision parameter as:
   $p_0 = {}^5/_4 - r$
3) Starting with k = 0, at each position x_k, 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 $p_{k+1} = p_k + 2x_{k+1} + 1 - 2y_{k+1}$
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) to plot the coordinate values .
6) Repeat the steps 3 through 5 until x>=y.

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

4.4 ELLIPSE GENERATING ALGORITHM

An ellipse is defined as a set of points such that the sum of the distances from two fixed points is a constant.

That is, $d_1 + d_2$ = constant

Let us assume that the foci are at $F_1(x_1, y_1)$ and at $F_2(x_2, y_2)$.

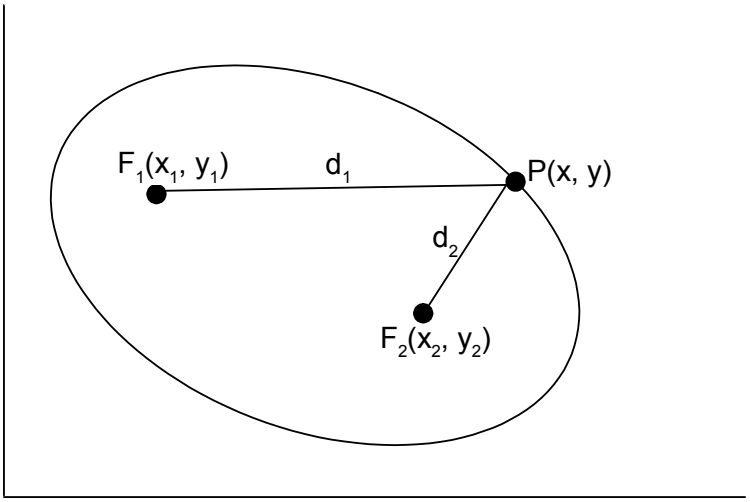


Figure 4.3: Ellipse generated about foci F_1 and F_2

By equating squares, isolating the radical and again equating squares, we will get an equation equivalent to the basic conic form:

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

where the parameters A, B, C, D, E and F depend on where the ellipse is centered and the lengths of its major and minor axes.

Suppose that the major and minor axes are parallel to the x and y axes using r_x for semi-major axis and r_y for semi minor axis, we get an ellipse equation having center as (x_c, y_c)–

$$\left\{ \left[\frac{x - x_c}{r_x}\right]^2 + \left[\frac{y - y_c}{r_y}\right]^2 = 1 \right.$$

4.4.1 Mid-Point Ellipse Algorithm

Mid-Point ellipse algorithm is a method for drawing ellipses in computer graphics. It is a modification of the Bresenham's algorithm and the advantage of this method is that only addition operations are done thereby leading to fast implementation.

Let us consider a quarter of an ellipse as given in the figure 4.4. The curve is divided into two regions.

Circle and Ellipse Drawing Algorithms

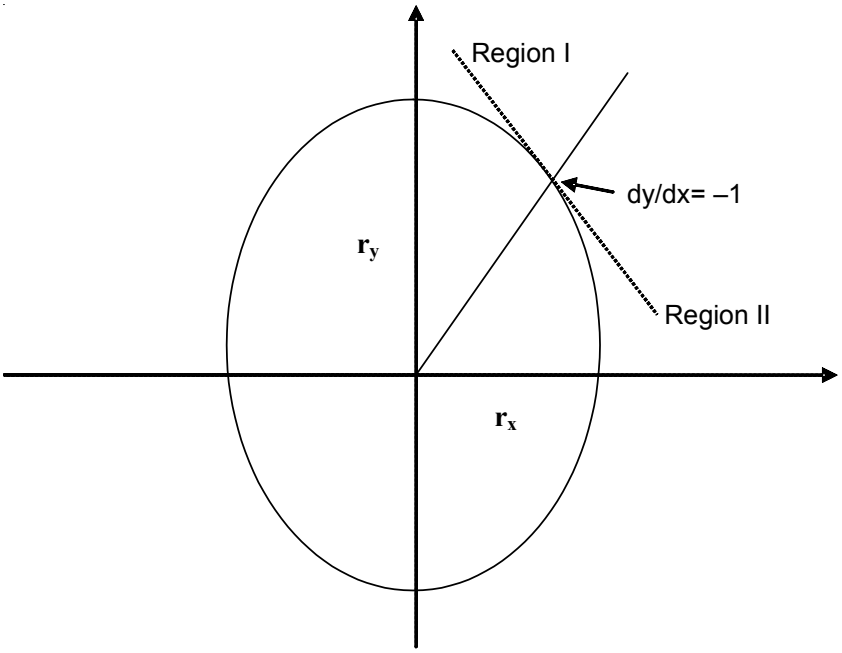


Figure 4.4: Ellipse processing regions

This algorithm is applied to the first quarter in two parts where $r_x < r_y$. The ellipse centre is assumed to be at the origin. The coordinates of the points in the other quarters can be obtained using the ellipse symmetry while implementation. For a pixel (x, y) in the first quarter, the corresponding points in the other three quarters are (x, –y), (–x, y) and (–x, –y) respectively. If the centre is at (x_c, y_C), all calculated coordinate (x, y) should be adjusted by adding the offset (x_C, y_C). The process is by taking unit steps in x-direction where the slope is less than 1 and by taking unit steps in y where the slope is greater than 1.

For region I, we can proceed from $(0, r_y)$ with unit steps in x-direction. For region II we can proceed from $(r_x, 0)$ with unit steps in negative y-direction.

An ellipse function with $(x_c, y_c) = (0, 0)$ is defined as:

$$f_{ellipse}(x,y) = r_y^2x^2 + r_x^2y^2 - r_x^2r_y^2$$

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

This ellipse function is the decision parameter in the mid-point ellipse algorithm. At each sampling position, the next pixel along the ellipse path is selected according to the decision parameter's sign that is evaluated at the mid-point between two candidate pixels. The ellipse symmetry is shown in the figure 4.5.

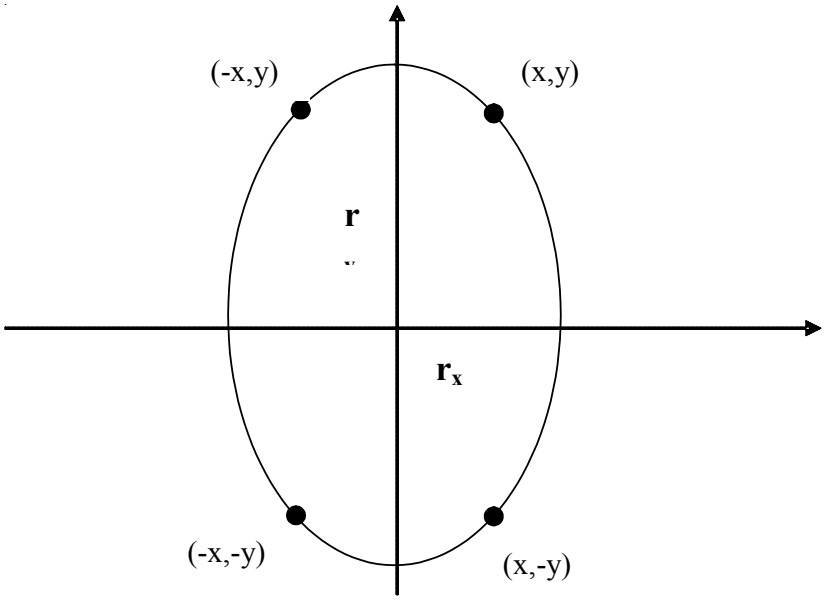


Figure 4.5: Symmetry of an ellipse

For drawing the ellipse we start at the point $(0,r_y)$ and take unit steps in the x-direction until the boundary between region I and region II is reached. Then we move unit steps in the y-direction over the remaining curve in the first quarter. At each step the value of the ellipse slope is calculated.

The ellipse slope is calculated as–

$$\frac{d_y}{d_x} = \frac{2r_y^2x}{2r_x^2y}$$

At the boundary between region I and region II, $d_y/d_x = -1$, so we get,

$$2r_y^2x = 2r_x^2y$$

Whenever $2r_y^2x >= 2r_x^2y$, we move out of region I.

$$r^2_y x^2 + r^2_x y^2 - r^2_x r^2_y = 0$$

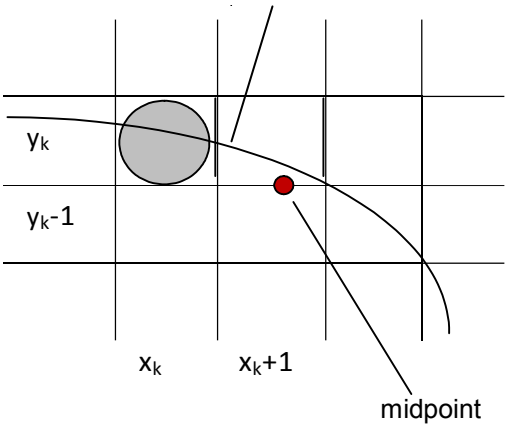


Figure 4.6: Midpoint between candidate pixels at x_k+1 along ellipse path

Suppose the pixel at position (x_k, y_k) has been plotted. We now have to decide which pixel to plot next along the ellipse path. The choices are:

(x_k+1, y_k) and $(x_k+1, y_k–1)$

The next position is determined by evaluating the decision parameter at the midpoint. The midpoint here is:

$(x_k+1 + x_k+1)/2$ and $(y_k + y_k – 1)/2$

$= (x_k+1, y_k–½)$

Now, the decision parameter at this midpoint can be defined as–

$p1_k = f_{ellipse}(x_k+1,y_k–½)$

$= r^2_y(x_k+1)^2 + r^2_x(y_k-½)^2 – r^2_x r^2_y$

If $p1_k < 0$, then the mid-point is inside the ellipse and the pixel at y_k is closer to the ellipse boundary. Otherwise the mid-point is outside or on the ellipse boundary and the pixel $y_k–1$ is closer.

The decision parameter for the next point p_{k+1} at position

$(x_k + 2,y_{k+1} – ½)$ is then evaluated.

As $x_k + 2 = x_{k+1}+1$ we can write as:

$p1_{k+1} = f_{ellipse}(x_{k+1}+1, y_{k+1}–½)$

$= r_y^2[(x_k+1) + 1]^2 + r_x^2[y_{k+1}–½]^2 – r_x^2 r_y^2$

$= p1_k + 2r_y^2(x_k+1) + r_y^2 + r_x^2[(y_{k+1} – ½)^2 – (y_k–½)]^2$

Where y_{k+1} is either y_k or $y_k–1$ depending on the sign of $p1_k$.

For drawing the ellipse the decision parameters are then incremented by the following amounts–

If $p1_k < 0$ then it is incremented by $2r^2_y x_{k+1} + r^2_y$

Otherwise if $p1_k >= 0$ then it is incremented by $2r^2_y x_{k+1} + r_y^2 – 2r^2_x y_{k+1}$

After all the ellipse points has been generated in this way, we shift each pixel to be centered at (x_c,y_c) by $x = x + x_c$; $y = y + y_c$

The **Mid-point Ellipse Algorithm** is as given below:

1) Input the radius 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 I as: $p1_0 = r^2_y – r^2_x r_y + ¼ r^2x$
3) At each x_k position in region I, starting at k = 0 the following tests are performed. If $p1_k < 0$ the next point is (x_{k+1}, y_k). the decision parameter is then, $p1_{k+1} = p1_k +$

$2r^2_y x_{k+1} + r^2_y$, else the next point is at

(x_k+1, y_k-1) and $p1_{k+1} = p1_k + 2r^2_y x_{k+1} - 2r^2_x y_{k+1} + r^2_y$

where, $2r_y^2 x_{k+1} = 2r^2_y x_k + 2r^2_y$ and

$2r^2_x y_{k+1} = 2r^2_x y_k - 2r^2_x$

4) Calculate the first value of the decision parameter in the region II using the last point of region I.

   $p2_0 = r^2_y(x_0+½)^2 + r^2_x(y_0-1)^2 - r^2_x r^2_y$

5) In region II at each y_k perform the following tests.

   If $p2_k > 0$, the next point in the ellipse path is (x_k, y_k-1) and

   $p2_{k+1} = p2_k - 2r^2_x y_{k+1} + r^2_x$; else the next point is at (x_k+1, y_k-1).

   $p2_{k+1} = p2_k + 2r^2_y x_{k+1} - 2r^2_x y_{k+1} + r^2_x$

6) Determine the symmetry points in the other three quadrants.

7) Shift each of the pixel position (x, y) on the elliptical path centered on (x_c, y_c) to plot the coordinate values $x = x + x_c$, $y = y + y_c$

8) Repeat the steps for region I until $2r^2_y x >= 2r^2_x y$.

CHECK YOUR PROGRESS

Q.1: Why is it not a brilliant solution to write a simple circle drawing algorithm by solving the equation for y at unit x-intervals using $y = \pm\sqrt{r^2 - x^2}$?

.....

.....

.....

.....

.....

Q.2: Fill in the blanks:

a) Circles have a ..... way symmetry and the ellipses have a ..... way symmetry.

b) In the mid-point circle algorithm, only the points for the ..... of a circle are calculated and then uses its ..... to plot the rest of the circle circumference points.

c) If the circle function returns a negative value then we can say that a point (x, y) is in the ..... and if the point lies outside the circle boundary, then the circle function returns a ..... value.

d) If the pixel at (x_k, y_k) has been plotted then the next step is to choose a point that is nearest to the actual circle path. The two possible points are ..... and .....

e) When the ellipse function returns 0 it means that the point (x, y) lies on the .....

f) The mid-point circle and ellipse algorithms considers the centre to be at the origin (0, 0). If the actual centre is at (x_c, y_C), then all the calculated coordinates (x, y) should be adjusted by adding the offset .....

4.5 LET US SUM UP

- A circle is a symmetrical figure. Circles have eight way symmetry.
- The equation for a circle is: $x^2 + y^2 = r^2$, where, r is the radius of the circle.
- The eight-way symmetry of circles is used in Mid-Point circle algorithm.
- An ellipse is defined as a set of points such that the sum of the distances from two fixed points is a constant.
- Mid-Point ellipse algorithm is a modification of the Bresenham's algorithm and the advantage of this method is that only addition operations are done thereby leading to fast implementation.
- In case of circle and ellipse drawing algorithms, their symmetry is taken into consideration.

4.6 FURTHER READING

1) Hearn, Donald and Baker, M. Pauline. *Computer Graphics*. Prentice-Hall.

2) Newman and Sproull. (1979). *Principles of Interactive Computer Graphics*. McGraw-Hill.

3) Plastock and Kalley. (1986). *Computer Graphics, Schaum's Outlines*. McGraw-Hill.

4.7 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: Using the given equation for y and writing an algorithm for drawing circles is not a brilliant idea because

- Firstly, the resulting circle has large gaps where the slope approaches the vertical
- Secondly, the calculations are not very efficient. Basically the square operations and the square root operations.

Ans. to Q. No. 2: a) 8, 4; b) top right eighth, symmetry; c) circle interior, positive; d) (x_k+1, y_k) and (x_k+1, y_k-1); e) ellipse boundary; f) (x_C, y_C).

4.8 MODEL QUESTIONS

Q.1: Write the Mid-Point circle algorithm.

Q.2: Describe the Ellipse generating procedure.

Q.3: Write the equation of circle and ellipse.

*** ***** ***

UNIT 5: FILLED AREA ALGORITHMS

UNIT STRUCTURE

5.1 Learning Objectives
5.2 Introduction
5.3 Filling Polygons
  5.3.1 Scan-converting Polygon Filling
  5.3.2 Inside Outside Tests
  5.3.3 Boundary Fill Algorithm
  5.3.4 Area Fill Algorithm
  5.3.5 Flood Fill Algorithm
5.4 Let us Sum Up
5.5 Further Reading
5.6 Answers to Check Your Progress
5.7 Model Questions

5.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- learn how to fill a region
- learn about different filling techniques
- describe different polygon filling algorithms.

5.2 INTRODUCTION

In the previous unit we have learnt about the circle and ellipse drawing algorithms. In this unit we will learn filling process. Filling is the process of coloring a definite area or region. Regions may be described at the pixel or geometric level. In the first case the region is called interior defined and the collection of algorithms used for filling such a region. The other type of region is called aboundary defined region. For both the region different algorithms are used.

5.3 FILLING POLYGONS

In raster scan systems, there are two approaches in filling polygons.

First Approach:

1) **Polygon Scan Conversion:** Rasterize a polygon scan line by scan line, determining which pixels to fill on each of the line (figure 5.1).

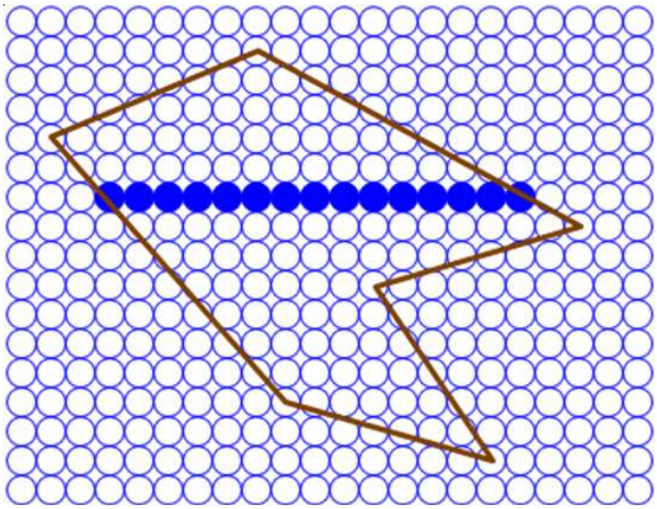


Figure 5.1: Filling using line scan conversion

Second Approach:

2) **Polygon Fill:** Select a pixel inside the polygon and grow in outward direction until the whole polygon is filled. This is depicted in figure 5.2.

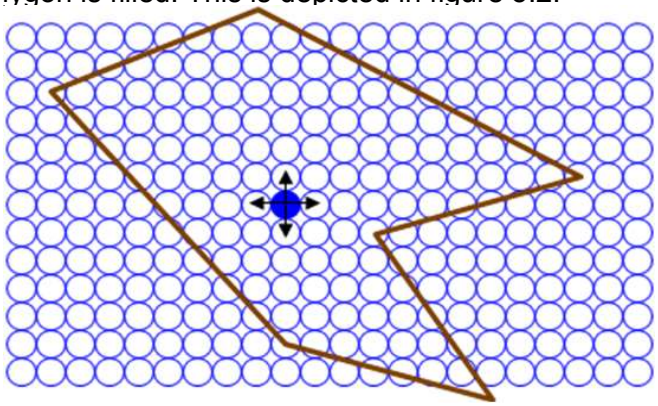


Figure 5.2: Filling using pixel enhancing

5.3.1 Scan-Converting Polygon Filling

Figure 5.3 illustrates the scan-line procedure for filling of polygon areas.

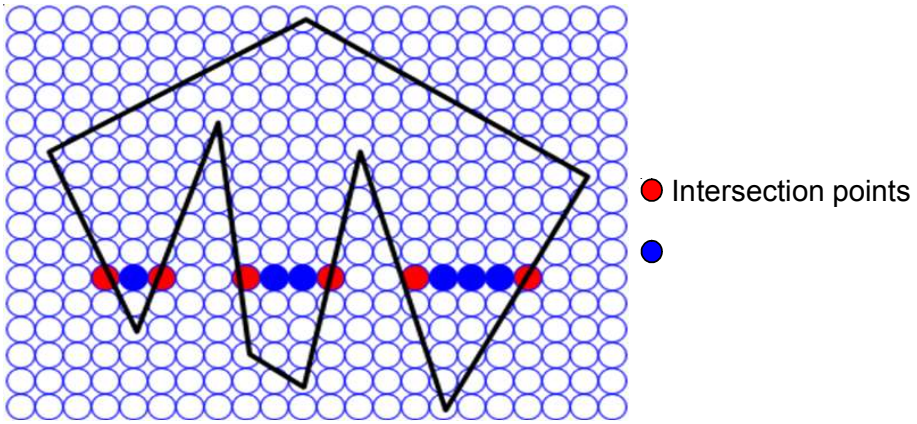


Figure 5.3: Scan line converting filling

The scan conversion algorithm works as follows:

A) Process each of the scan line
   - For each scan line crossing a polygon, find out the intersections of the scan line with all the polygon edges.
   - Sort the intersections from left to right, that is, sort by the x coordinate.
   - The corresponding frame buffer positions between each intersection pair are set to the specified color

B) Some scan line intersections at the polygon vertices require special handling.
   - A scan line passing through a vertex intersects two polygon edges at that position, so two points area added to the list of intersection points.

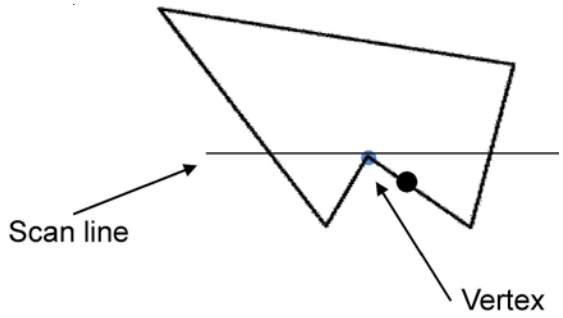


Figure 5.4 : Scan conversion special case

   - When a scan line passes through a vertex, it is checked whether the edges meeting at that vertex lie on the same side or on the opposite side to that vertex.

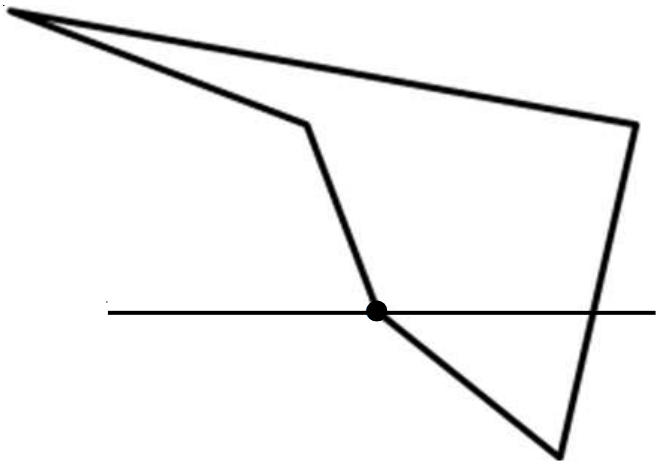


Figure 5.5:

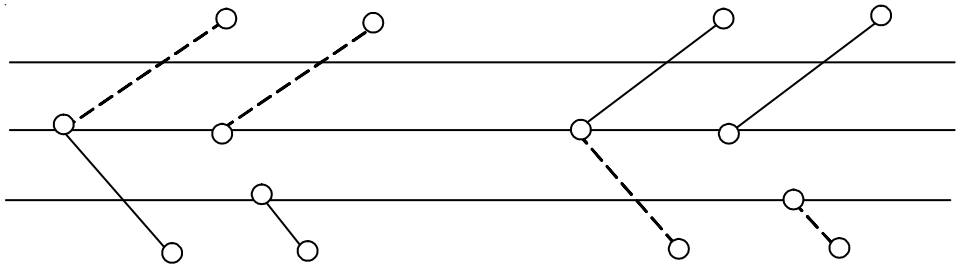
   - These vertices can be identified by tracing around the polygon boundary either in clock-wise or anti-clockwise order and then observing the relative changes in the vertex y coordinates as we move from one edge to the next.

a) If the y coordinate values either decreases or increases only, then the two edges are on the opposite sides. So a single point is added to the list.

b) If the y coordinate values first decreases and then increases (or vice versa), then the two edges are on the same side. So two points are added to the list.

- An alternative solution to (a) is to shorten some polygon edges to split those vertices.

Proceeding from bottom to top, when the y values are increasing, the y value of upper end of current edge (lower edge) is decreased by 1.



Bottom to top **Top to Bottom**

Figure 5.6:

Proceeding from top to bottom, when the y values are decreasing, the y value of upper end of next edge is decreased by 1.

5.3.2 Inside Outside Tests

To check whether a point lies inside or outside a polygon, there are two rules.

1) ODD-EVEN RULE (Odd parity rule or Even parity rule):

a) A line is drawn from any position P to a distant point outside coordinate extents

b) Number of edge crossings along that line is counted.

c) If the number of edge crossed is odd, then P is an interior point. Otherwise P is an exterior point.

d) For the vertices, the line path chosen should not intersect any vertex of the polygon.

2) NON-ZERO WINDING NUMBER RULE: It counts the number of times the polygon edges wind around a particular point in anti-clockwise direction. This count is called *winding number*.

a) Initialize winding number to 0. Imagine a line drawn from any position P to a distant point beyond coordinate extents. The line must not pass through

vertices.

b) We will trace anti-clockwise the edges of the polygons. As we move along the line from P to the distant point, for every time an edge crosses the line from left to right, we add 1 to the winding number. For every time it crosses from right to left, we substract 1 from the winding number.

c) If the net value of the winding number is non-zero, P is an interior point. Else it is an exterior point.

This rule is more effective than the Odd-Even rule.

5.3.3 Boundary Fill Algorithm

A second approach in filling the area of polygons is to use a filling technique, rather than scan-conversion. In this technique, we have to select a point inside the polygon and then fill the neighboring pixels until the polygon boundary is reached. The boundary is specified in a particular color. The boundary fill algorithm then proceeds outwards pixel by pixel until that boundary color is reached. If the fill color and boundary color is same then a solid color region is displayed.

The **Boundary Fill Algorithm** is as follows :

1) Draw the polygon boundary.
2) Accept as input an interior point (x,y), the fill color and the boundary color.
3) Starting at the interior point, the algorithm tests the neighboring pixels to determine whether they are of the boundary color. If it is not the fill or the boundary color, then,
   - Set this pixel to the fill color
   - Propagate to the pixel's neighbors and continue until all pixels upto the boundary color have been tested.

There are two most frequently used approaches for propagating to neighboring points.

1) Four neighboring points are tested. This method is called 4-connected.

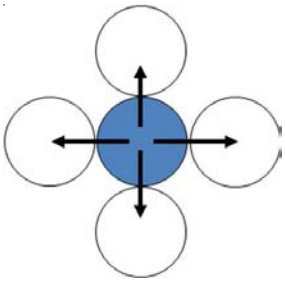


Figure 5.7: Four connected neighbours

2) Eight neighboring points are tested. This method is called 8-connected.

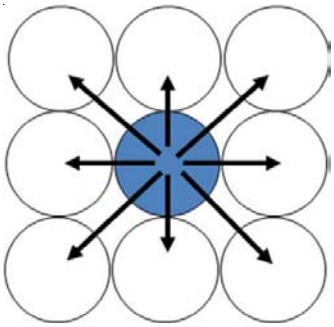


Figure 5.8: Eight connected neighbours

Pseudocode for 4-connected–

```
Procedure boundaryFill4(x,y,fill,boundary : integer)
var current : integer
begin current = getpixel(x,y);// color of the current
pixel
if((current<>boundary) and (current<>fill)) then
    begin
        setpixel(x,y,fill);
        boundaryFill4(x+1,y,fill,boundary);
        boundaryFill4(x-1,y,fill,boundary);
        boundaryFill4(x,y+1,fill,boundary);
        boundaryFill4(x,y-1,fill,boundary);
end     // if statement
end     // end of the procedure
```

For the 8-connected, recursively call the procedure for 8 times.

The 8-connected method is used in more complex figures. The 4-connected algorithm fills the polygons partially, but 8-connected algorithm is better.

The fill algorithm has some problems in case of 4-connected,

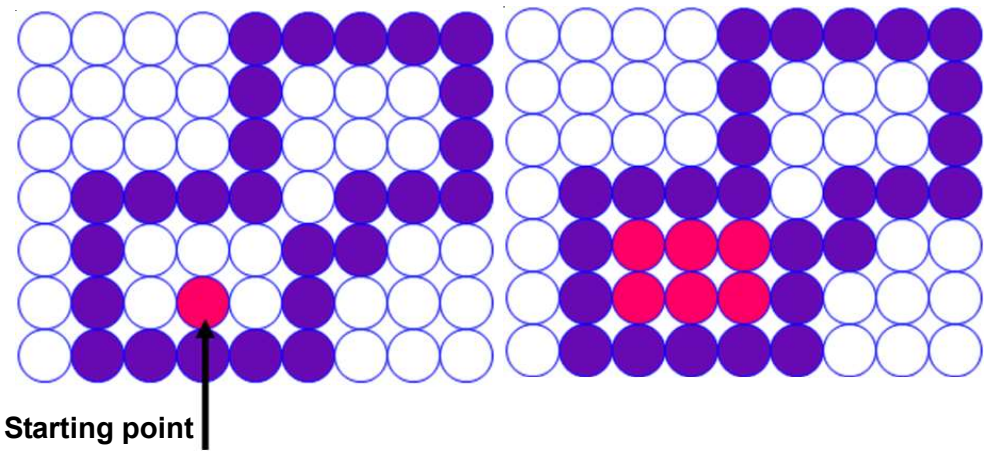


Figure 5.9: Filling using 4-connected

Similarly, the 8-connected can leak over to other polygon.

5.3.5 Flood Fill Algorithm

In the flood filling process, the user will generally privide an initial pixel called the seed. Taking the seed as initial point the algorithm will search each of the sorrunding eight connected pixels to determine whether the extent has been reached. This process is repeated until all pixels inside the region have been processed.

The algorithm is as follows:

1) Select an interior point (x,y) in the polygon.
2) Re-assign all the pixel values currently assigned a given interior color to the specified fill color.
3) If the interior pixels have more than one interior color, then first assign all the pixels to have the same interior color.
4) Using either 4-connected or 8-connected approach, propagate through the pixel positions until all the interior pixels have been repainted with the particular fill color.

The pseudocode for the Flood Fill Algorithm is as follows:

```
Procedure floodFill4(x,y,fillcolor,oldcolor : integer)
begin
    if  getpixel(x,y) = oldcolor then
    begin
        setpixel(x,y,fillcolor)
        floodFill4(x+1,y,fillcolor,oldcolor);
        floodFill4(x-1,y,fillcolor,oldcolor);
        floodFill4(x,y+1,fillcolor,oldcolor);
        floodFill4(x,y-1,fillcolor,oldcolor);
    end
    end
```

CHECK YOUR PROGRESS

Q.1: State whether the following are True or False:

a) Scan conversion approach is not used for filling areas of polygon.

b) In the scan-converting polygon filling approach, the number of intersecting points of the edges are counted.

c) If the edges lie on the same side of the vertex through which the scan-line passes, then 2 points are added to the list of intersecting points.

d) If the edges lie on the opposite sides of the vertex, then proceeding from bottom to top, if the y values is increasing then the y value of the upper end of the current edge is decreased by 1.
e) In the Odd-Even rule, the line drawn from a point P to a distant point should always intersect the vertices of the polygon.
f) In the Fill algorithms, 5-connected and 9-connected methods are used to propagate to the neighboring points.

5.4 LET US SUM UP

- There are two methods of polygon filling : Polygon Scan Conversion and Polygon Filling
- To test whether a point lies inside or outside the polygon region there are two tests: Odd-Even Rule and the Non-Zero winding number rule.
- In Boundary fill algorithm, a point inside the polygon is selected and then the neighboring pixels are filled with the fill color until the polygon boundary is reached.
- In Flood fill algorithm, first the pixels inside the polygon are set to the same interior color and then these pixels are assigned the specified fill color.

5.5 FURTHER READING

1) Hearn, Donald and Baker, M. Pauline. *Computer Graphics*. Prentice-Hall.
2) Newman and Sproull. (1979). *Principles of Interactive Computer Graphics*. McGraw-Hill.
3) Plastock and Kalley. (1986). *Computer Graphics, Schaum's Outlines*. McGraw-Hill.

5.6 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: a) False; b) True; c) True; d) True; e) False; f) False

5.7 MODEL QUESTIONS

Q.1: Discuss the scan-line procedure for filling of polygon areas.

Q.2: What are the rules for checking whether a point lies inside or outside a polygon boundary?

Q.3: Discuss the Boundary fill and Flood-fill algorithms.

*** ***** ***

UNIT 6: 2-D TRANSFORMATIONS

UNIT STRUCTURE

6.1 Learning Objectives
6.2 Introduction
6.3 2-D Transformation
    6.3.1 Translation
    6.3.2 Rotation
    6.3.3 Scaling
    6.3.4 Reflection
    6.3.5 Shear
6.4 Homogeneous Coordinate Representation
6.5 Let Us Sum Up
6.6 Further Reading
6.7 Answers to Check Your Progress
6.8 Model Questions

6.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- learn about different transformations like translation, rotation, scaling, reflection etc.
- learn how the transformations are done in graphics
- define 2D and 3D transformations using homogeneous coordinate system
- learn about projections in computer graphics.

6.2 INTRODUCTION

In the previous unit we have discussed the different polygon filling algorithms. We now know how to use boundary fill algorithm and what the flood fill algorithm does.

In computer graphics, animations are done on the screen using the different transformation techniques. Graphics is now used almost everywhere and it is rapidly growing. The two dimensional (2D) and three-dimensional (3D) graphics are used extensively by the interactive softwares in business and education.

In this unit, we will introduce you to the transformation techniques of the pictures and graphs. Also we will be discussing how to project the objects of a scene onto another coordinate view plane.

6.3 2D-TRANSFORMATION

A transformation is an operation on a point (x, y) that maps the point's coordinate into a new coordinate (x_1, y_1). Transformations are important in computer graphics.

The basic geometric transformations are:

- Translation
- Rotation
- Scaling

Other transformations are:

- Reflection
- Shear

Let us now discuss how these transformations are applied to reposition and re-size the 2D-objects.

6.3.1 Translation

Repositioning an object along a straight line path from one coordinate location to another coordinate location is called *translation transformation*.

A point (x, y) is translated to a new location (x', y') by adding the translation parameters t_x and t_y to the original coordinates x and y.

$$x' = x + t_x$$

$$y' = y + t_y$$

The distance pair (t_x, t_y) is called transformation vector or shift vector.

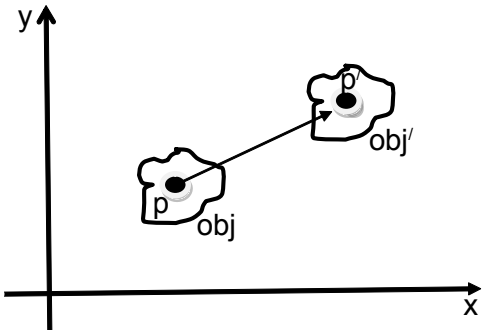


Figure 6.1: Translation

In matrix form, $P' = P + T$

where, $P' = \begin{bmatrix} x' \\ y' \end{bmatrix}$, $P = \begin{bmatrix} x \\ y \end{bmatrix}$, $T = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$

- Translation is a rigid body transformation which moves the objects without deformation.

- To translate a straight line, the translation is first applied to the two endpoints of the line. Then a line is redrawn between the two new endpoints.
- In case of curves or polygons also similar method is used.

3.3.2 Rotation

To reposition an object along a circular path in the xy plane, the 2D-rotation is used. In rotation, the object is rotated through Φ^0 about the origin.

There is a rotation angle θ and a point (x_r, y_r) called the rotation point or the pivot point about which the object is to be rotated.

If θ > 0 (positive angle), then the rotation takes place in anti-clockwise direction about the pivot point.

If θ < 0(negative angle), then the rotation takes place in clockwise direction about the pivot point.

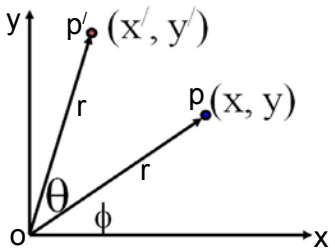


Figure 6.2: Rotation of point p

Transformation of point P when the rotation point is at the origin:

Using trigonometric identities, we get,

$x' = r\cos(\Phi+\theta)$

$= r\cos\Phi\cos\theta - r\sin\Phi\sin\theta$

$y' = r\sin(\Phi+\theta)$

$= r\cos\Phi\sin\theta + r\sin\Phi\cos\theta$

The original coordinates of the point (x,y) making an angle Φ with the x-axis are:

$x = r\cos\Phi$

$y = r\sin\Phi$

Therefore, we get, $x' = x\cos\theta - y\sin\theta$

$y' = x\sin\theta + y\cos\theta$

In matrix form, $P' = R.P$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Rotation does not change the shape or size of the objects as every point on the object is rotated through the same angle.

6.3.3 Scaling

Scaling means changing the size of an object. It is carried out by multiplying the coordinate values (x, y) of the vertex of the polygons by the scaling factors s_x and s_y. The new transformed coordinates of the object ($x^/$, $y^/$) are thus obtained as follows:

$$\mathbf{x^/ = x.s_x}$$

$$\mathbf{y^/ = y.s_y}$$

The scaling factor s_x scales the objects in the x-direction and the scaling factor s_y scales in the y-direction.

In matrix form we can write as: $P^/ = S.P$

Where, $P^/ = \begin{bmatrix} x^/ \\ y \end{bmatrix}$, $S = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$, $P = \begin{bmatrix} x \\ y \end{bmatrix}$

- Positive values of s_x and s_y increases the object's size whereas negative values decreases its size.
- When $s_x = s_y$, the scaling is uniform.
- When $s_x \neq s_y$, the scaling is differential.

The objects when are scaled are transformed and repositioned. To make the position of the object not changed after scaling, we can choose a point as the Fixed Point and scale the objects relative to that fixed point. This fixed point can be any vertex of the polygon or any other point. The fixed point is say (x_f, y_f).

The coordinate of the transformed point is then,

$$x^/ = x_f + (x - x_f)s_x$$

$$y^/ = y_f + (y - y_f)s_y$$

Scaling is not a rigid body transformation as it changes the size of the objects.



(a) Squate (b) transformed into a rectangle with $s_x = 2$ and $s_y = 1$

Figure 6.3: Scaling

6.3.4 Reflection

Reflection produces a mirror image of an object. An object is rotated at a particular angle about the axis of reflection to get the reflected image.

For 2D reflection, the object is rotated about the axis of rotation in 180^0.

A) Reflection about the x-axis (y = 0): This is obtained by keeping the x-coordinate of a point unchanged and inverting the y-coordinate.

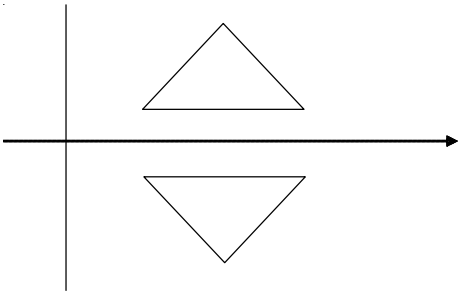


Figure 6.4: Reflection about the x-axis

The transformation matrix is as follows:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

B) Reflection about the y-axis (x = 0): This is obtained by keeping the y-coordinate of a point unchanged and inverting the x-coordinate.

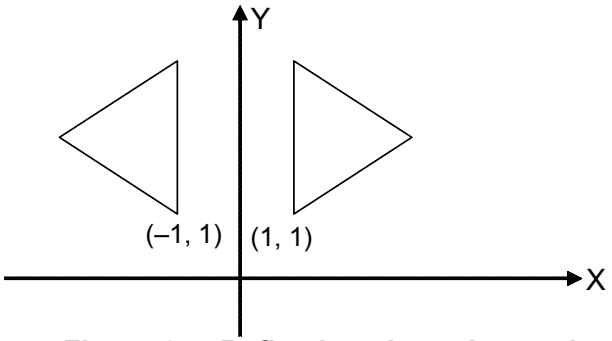


Figure 6.5: Reflection about the y-axis

The transformation matrix is as:

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

C) Reflection about an axis perpendicular to the xy-plane: This is accomplished by inverting both the x and y coordinate values.

(–1, 1)

(1, –1) X

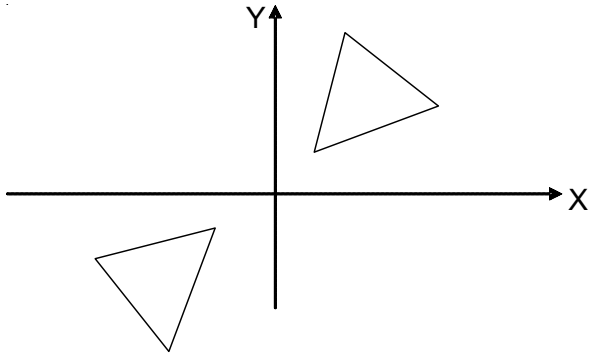


Figure 6.6: Reflection about an axis perpendicular to the xy-plane

The transformation matrix is as:

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

D) Reflection about a line where y = x: This can be obtained by interchanging the x and y coordinate values.

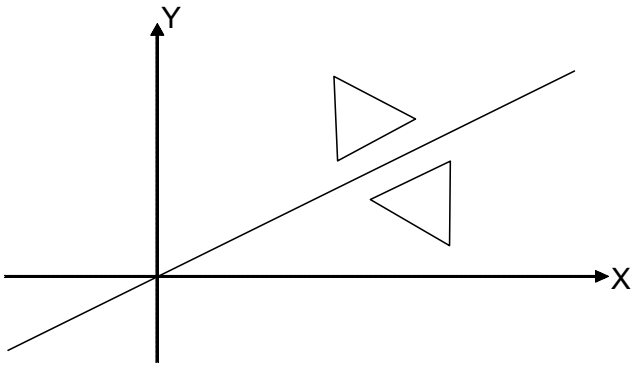


Figure 6.7: Reflection about a line where y = x

The line y = x passes through the origin (0, 0).

The transformation matrix is as:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

E) Reflection about a line where y = –x. This can be obtained by exchanging and inverting the x and y coordinate values.

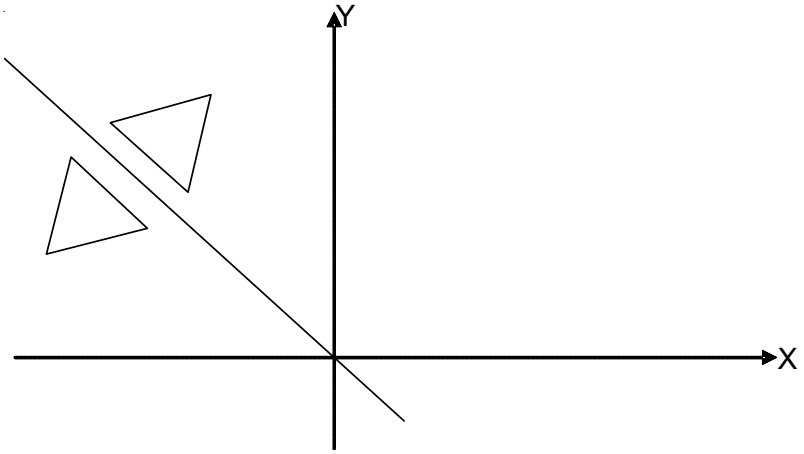


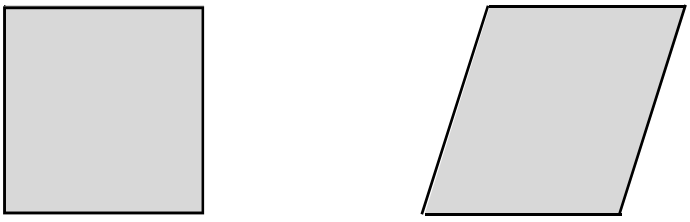
Figure 6.8: Reflection about a line where y = –x

The transformation matrix is as:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

6.3.5 Shear

This kind of transformation distorts the shape of the objects and they appears to look as if the internal layers are caused to slide over each other. The object is pulled horizontally by different amounts.



Original After Shear

Two common shear operation are those that :

a) Shift the x-coordinate values

b) Shift the y-coordinate values.

- A x-direction shear relative to the x-axis is given by the following matrix:

$$\begin{bmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Where $x' = x + sh_x y$

$y' = y$

The sh_x above is the shear parameter whose value can be any real number either positive or negative.

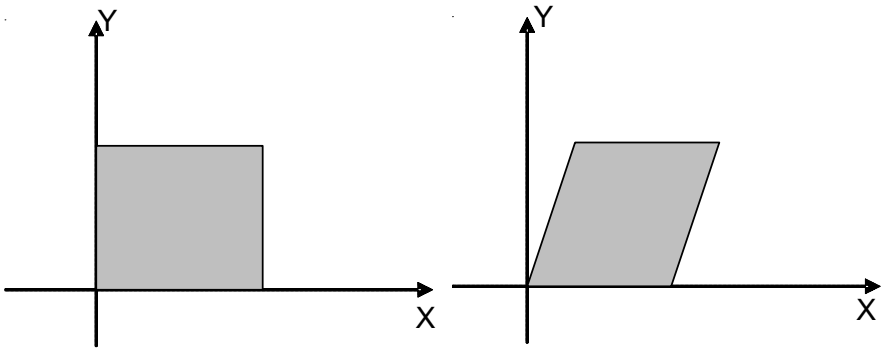


Figure 6.9: Unit Square transformed to a parallelogram

- A x-direction shear relative to other reference lines ($y = y_{ref}$) is as shown below:

$$\begin{bmatrix} 1 & sh_x & -sh_x y_{ref} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where the coordinate positions are transformed as:

$x' = x + sh_x (y - y_{ref})$

$y' = y$

Example: A square of unit size can be transformed to a parallelogram with $sh_x = 2$

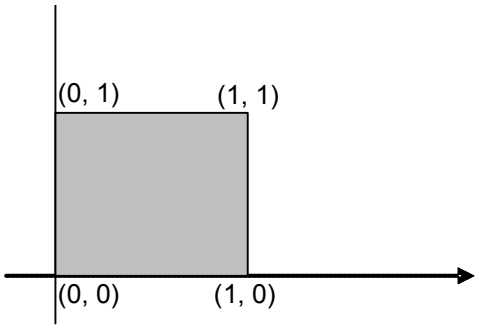


Figure 6.10:

Square: Here, the transformation of the square must be in reference to the x-axis and not to any other reference line. So in this example, $y_{ref} = 0$

Now, in case of x-direction shear relative to the x=axis, we have

$x' = x + sh_x y$

$y' = y$

Therefore, when x = 0, y = 0

$x' = 0 + 2(0) = 0$

$y' = 0$

So after shear the new coordinate value is (0, 0)

for the point (1,0) i.e. when x = 1, y = 0

$x' = 1 + 2(0) = 1$

$y' = 0$

The new coordinate value is (1, 0)

when x = 1, y = 1

$x' = 1 + 2(1) = 3$

$y' = 1$

After shear the new coordinate value is (3, 1)

Similarly for the point (0, 1)

$x' = 0 + 2(1) = 2$

$y' = 1$

After shear the new coordinate value is (2, 1)

Hence from this we can know draw the parallelogram as follows:

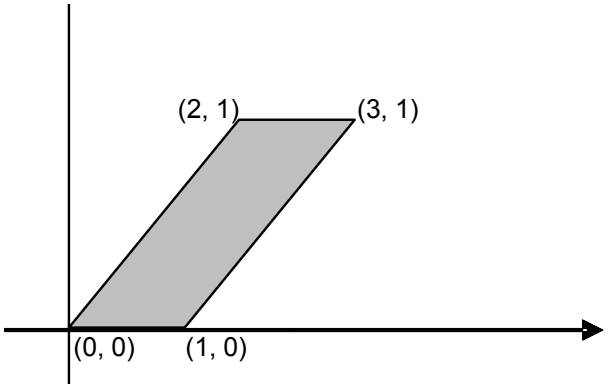


Figure 6.11: Parallelogram

- Any directional shear relative to $x = x_{ref}$ is give by the following matrix:

$$\begin{bmatrix} 1 & 0 & 0 \\ sh_y & 1 & -sh_y x_{ref} \\ 0 & 0 & 1 \end{bmatrix}$$

where, $x' = x$

$y' = sh_y(x - x_{ref}) + y$

CHECK YOUR PROGRESS

Q.1: Fill in the blanks:

a) A transformation is an operation on a point that maps the point's coordinates (x, y) to a new set of .....

b) Reflection is a .....

c) If the rotation angle θ is positive, then the rotation takes place in ..... about the pivot point.

d) For 2D reflection, the object is rotated about the axis of rotation in .....

e) Reflection about the y-axis is obtained by keeping the y-coordinate of a point ..... and inverting the .....

f) In Shear, the object is pulled ..... by different amounts so as to look like the internals layers ..... each other.

g) The common shear operations are shifting the ..... and the .....

h) The scaling factor s_x scales the objects in the ..... and the scaling factor s_y scales in the .....

i) Scaling is not a ..... transformation.

Q.2: Say True or False:

a) Rotation changes the size of objects.

b) Rotation is used to reposition an object along a circular path.

c) Reflection produces a mirror image of an object.

d) Shear changes the shape of an object and distorts it.

e) Scaling retains the original size of the objects.

f) Positive values of s_x and s_y increases the object's size whereas negative values decreases its size.

6.4 HOMOGENEOUS COORDINATE REPRESENTATION

The homogeneous coordinate system is used in projective geometry much as cartesian coordinate system is used in euclidean geometry. Homogeneous coordinates provides a method for doing calculations and proving theorems in projective geometry when used in practical applications.

This homogeneous coordinates provides a method for performing certain operations on points in euclidean space by means of matrix multiplications. They have the advantage that the coordinates of a point even those at infinity can be represented using finite coordinates. They are used in computer graphics where they allow transformations to be easily represented by a matrix.

The usual cartesian coordinates for a point consists of a list of n points where n is the dimension of the space. Now, the homogeneous coordinates for the same point requires n+1 coordinates. So we add a coordinate to the end of the list, and make it equal to 1.

Thus the two dimensional point (x, y) becomes (x, y, 1) in homogeneous coordinates system. Similarly, the three dimensional point (x, y, z) becomes (x, y, z, 1) in homogeneous coordinates.

The most common use of homogeneous coordinates is for one, two and three dimensional spaces. The final coordinate is the homogeneous parameter called "h" and it will not interfere with the normal x, y and z-coordinates.

In homogeneous coordinates, we can express all transformations as matrix multiplications. The coordinates are represented with three element column vectors and the transformation operations are written as 3x3 matrices.

For Rotation with respect to the origin,

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$P' = R(\theta) \cdot P$

The rotation parameter, $R(\theta)$ is a 3x3 matrix and its inverse is obtained by replacing θ by $-\theta$.

For Scaling,

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

The inverse of scaling is obtained by replacing s_x, s_y by $1/s_x$ and $1/s_y$

$P' = S(s_x, s_y) \cdot P$

CHECK YOUR PROGRESS

Q.3: Fill in the blanks:

a) A point (x, y) has the homogeneous coordinates .....

b) ..... is repositioning an object along a straight line path from one coordinate location to another coordinate location.

c) Translation is a rigid body transformation which moves the objects without .....

d) 2D transformations are represented by ..... matrices using homogeneous coordinates.

Q.4: Derive the transformation that rotates an object point θ^0 about the origin. Write the matrix representation for this rotation.

.....

.....

.....

.....

.....

.....

.....

6.5 LET US SUM UP

- A transformation is any operation on a point in space (x, y) that maps the point's coordinates into a new set of coordinates (x1, y1)
- The geometric transformations are *translation*, *rotation*, *scaling*, *reflection* and *shear*.
- Rotation is repositioning an object along a circular path.
- Reflection produces a mirror image of an object. For 2D reflection, the object is rotated about the axis of rotation in 180^0.
- Shear distorts the shape of the objects and it seems the object is pulled horizontally in different amounts.
- Scaling means changing the size of an object. It is carried out by multiplying the coordinate values (x, y) of the vertex of the polygons by the scaling factors s_x and s_y.
- Positive values of s_x and s_y increases the object's size whereas negative values decreases its size.
- When $s_x = s_y$, the scaling is uniform and the original shape of the object is maintained.
- 2D transformations are represented by 3x3 matrices using homogeneous coordinates.

6.6 FURTHER READING

1) Hearn, Donald and Baker, M. Pauline. *Computer Graphics*. Prentice-Hall
2) Newman and Sproull. (1979). *Principles of Interactive Computer Graphics*. McGraw-Hill.
3) Plastock and Kalley. (1986). *Computer Graphics, Schaum's Outlines*. McGraw-Hill.

6.7 ANSWER TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: a) coordinates (x1, y1); b) geometric transformation; c) anti-clockwise direction; d) 180^0; e) unchanged, x-coordinate; f) horizontally, slides over; g) x-coordinate values, y-coordinate values; h) x-direction, y-direction; i) rigid body

Ans. to Q. No. 2: a) False; b) True; c) True; d) True; e) False; f) True

Ans. to Q. No. 3: a) (x, y, 1); b) Translation; c) Deformation; d) 3 × 3;
e) 4 × 4;

Ans. to Q. No. 4: From figure 6.11 the definition of the trigonometric functions sin and cos can be written as:

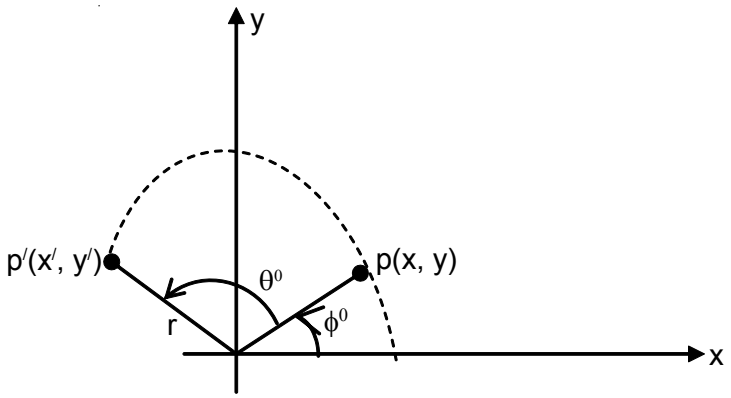


Figure 6.12: Point p is rotated θ^0

$x' = r\cos(\theta + \phi)$ $y' = r\sin(\theta + \phi)$

and $x = r\cos\phi$ $y = r\sin\phi$

Using trigononometric identities, we obtain,

$r\cos(\theta + \phi) = r(\cos\theta\cos\phi - \sin\theta\sin\phi) = x\cos\theta - y\sin\theta$

and $r\sin(\theta + \phi) = r(\sin\theta\cos\phi + \cos\theta\sin\phi) = x\sin\theta + y\cos\theta$

or $x' = x\cos\theta - y\sin\theta$ $y' = x\sin\theta + y\cos\theta$

Writing, $P' = \begin{pmatrix} x' \\ y' \end{pmatrix}$, $P = \begin{pmatrix} x \\ y \end{pmatrix}$ and $R_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$

We can write now $P/ = R_\theta . P$

6.8 MODEL QUESTIONS

Q.1: Discuss the 2D planar geometric transformations.

Q.2: What is 2D reflection? Discuss the x-axis and y-axis reflection.

Q.3: Write the matrix representations of the 2D transformation.

*** ***** ***

UNIT 7: 2-D VIEWING

UNIT STRUCTURE

7.1 Learning Objectives
7.2 Introduction
7.3 2-D Viewing Pipeline
7.4 Viewing Coordinate Reference Frame
7.5 Window-To-Viewport Coordinates Transformation
7.6 Two-Dimensional Viewing Functions
7.7 Let Us Sum Up
7.8 Further Reading
7.9 Answer to Check Your Progress
7.10 Model Questions

7.1 LEARNING OBJECTTIVES

After going through this unit, you will be able to:

- give the definition of window and viewport
- learn about viewing pipelines
- describe viewport transformation
- learn about world coordinates and normalized coordinates
- learn about 2-D viewing functions.

7.2 INTRODUCTION

When we consider the mechanism to display views on an output device, graphics packages allows a user to specify an image to be displayed and where that part of the image to be displayed on the device. For two dimensional displays a user can choose a particular subarea of the whole image. A user can select single area or multiple areas for simultaneous display or for an animated panning sequence across the scene. The image parts within the selected areas are mapped onto specified areas of the device coordinates. When multiple view area are selected, these area can be placed in separate display locations where some areas would be inserted into other, larger display areas. Transformation from the real world to device coordinates involve translation, scaling and rotation operations and some procedures which delete the parts of the picture which are outside the limits of display device.

Displaying an image of a picture involves mapping the coordinates of the points and lines that form the picture into the appropriate coordinates on the device or computer

where the image to be displayed. This is done through the use of coordinate transformations known as ***viewing transformations***.

7.3 2D VIEWING PIPELINE

A world coordinate area that is selected for display is called window and an area of display device where window is mapped is called *viewport*. The window defines 'what' is to be displayed and viewport defines 'where' to be displayed. Normally, we consider window and viewport both are rectangular in shape, with standard position and rectangular edges are parallel to the coordinate axes. Polygon, circles are considered as shape of the window or viewport in some special applications. In general, the mapping of the world-coordinate scene to device coordinates is referred to as a **viewport transformation.** Two-dimensional viewing transformation is simply referred to as **window-to-viewport transformation or windowing transformation.** Figure 7.1 shows the World coordinates and viewport.

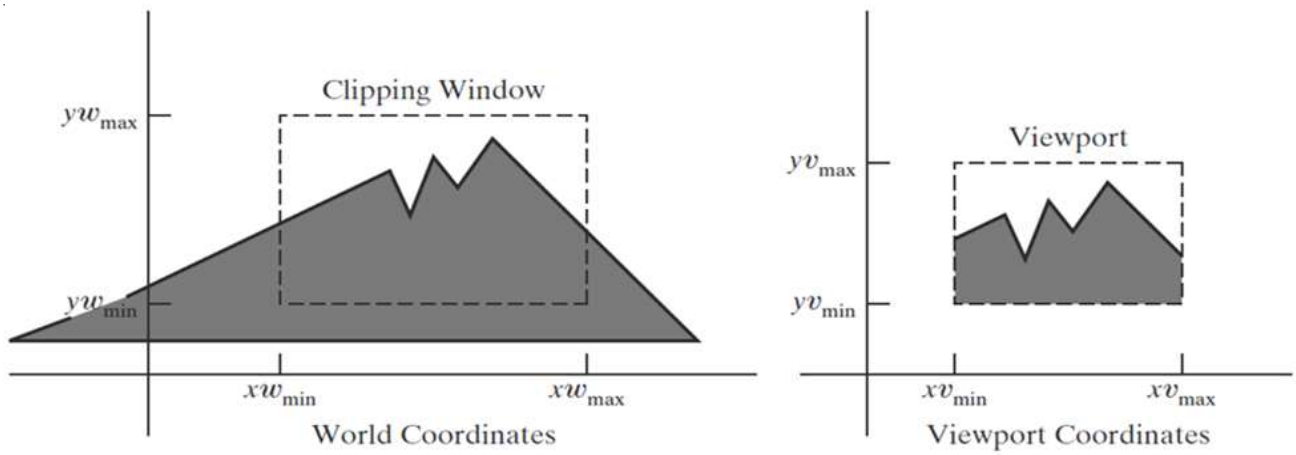


Figure 7.1: World coordinates and viewport

Viewing transformation carry out the several steps. First we construct world coordinate scene using output primitives and attributes and next we can set up a two dimensional coordinate system in the world coordinate plane to obtain the orientation for the window. The viewing reference frame is used to provide a method for setting up arbitrary orientation for rectangular windows. When the reference frame is established, we can transform the object in world coordinates to viewing coordinates. Then we define a viewport in normalized coordinates(in the range from 0 to 1) and mapped the viewport coordinates of the scene to normalized coordinates. And in the final step, the parts of the picture lie outside the viewport are clipped. Then the viewport contents are transferred to the device coordinates.

By changing the position of the viewport, we can display the object at different position of display device and if we varying the size of viewport, we can change the

proportion and size of the displayed object. By successive mapping the different sized window on the fixed sized viewport, we can achieve the zooming effect.

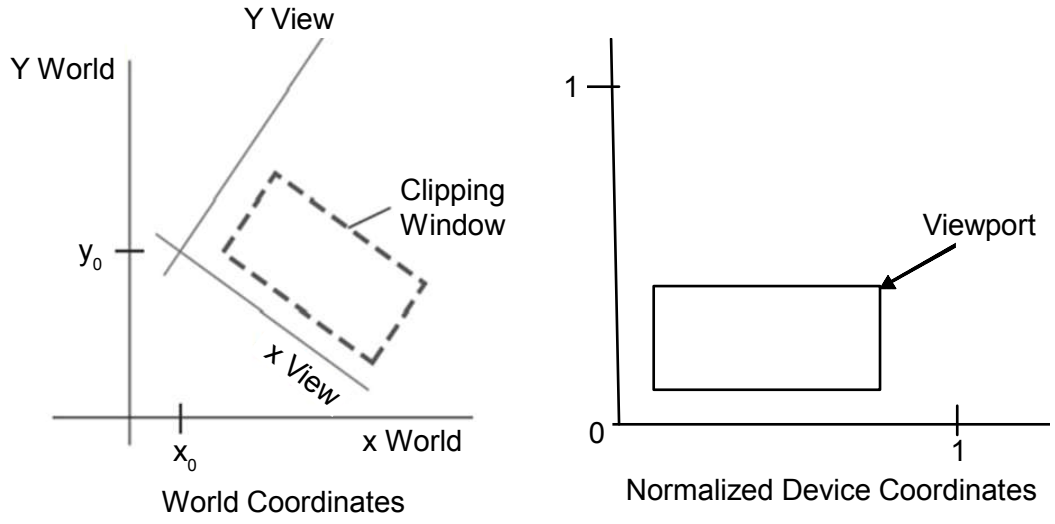


Figure 7.2: Normalized device coordinates

The term viewing pipeline describes a series of transformations, which are passed by data position to end up as image data displayed on a device. The two-dimensional viewing pipeline describes this process for two-dimensional data:

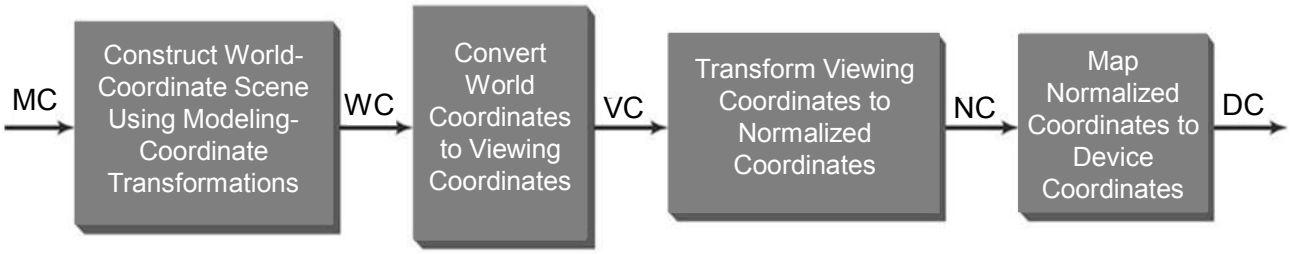


Figure 7.5: Two-dimensional viewing pipeline

The coordinates which describes the position of individual objects (models) are created are called model (or object) coordinates. When more objects are assembled into a scene, they are described by world coordinates. After transformation into the coordinate system of the viewer they become viewing coordinates. Their projection onto a window yields device-independent normalized coordinates. Finally, after mapping those normalized coordinates to a specific device, we get device coordinates.

CHECK YOUR PROGRESS

Q.1: The process of selecting and viewing the picture with different views is called .....

a) Windowing b) clipping

c) both (a) and (b) d) projecting

Q.2: In two dimensional viewing, we have .....?

a) 2D window, 3D viewport c) 2D window, 2D viewport

b) 3D window, 2D viewport d) 3D window, 3D viewport

Q.3: Define viewport and window.

.....

.....

.....

.....

Q.4: What is windowing transformation?

.....

.....

.....

.....

7.4 VIEWING COORDINATE REFERENCE FRAME

It is a reference frame which specifies the world coordinate window. First we select a viewing coordinate origin at some world position P=(x_0,y_0). then we establish the rotation, or orientation, of this reference frame. For this we specify a world vector V that defines the viewing y_v direction. Vector V called as **view up vector**.

We can calculate the components of unit vector v = (v_x, v_y) and u = (u_x, u_y) for the viewing y_v and x_v axes respectively, for the given V. These unit vectors make the first and second rows of the rotation matrix R which is aligns the viewing x_v, y_v axes with the world x_w, y_w axes.

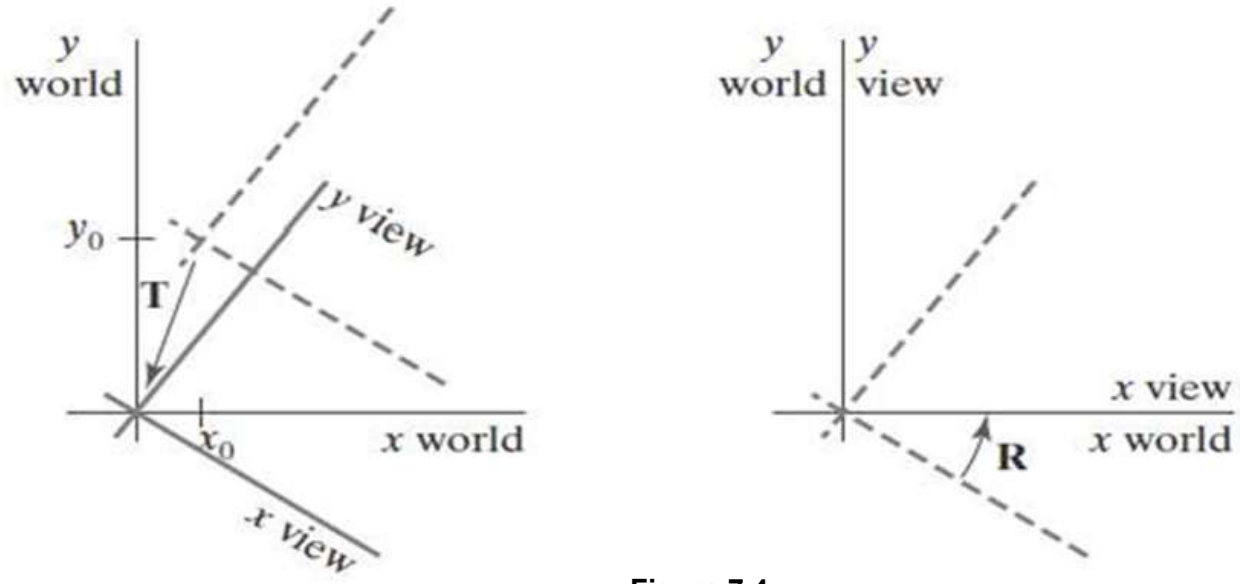


Figure 7.4:

Using two steps composite transformation, we obtain the matrix for converting world coordinate position to viewing coordinates

1) Translate the viewing origin to world origin.
2) Rotate to align two coordinate reference frames..

So composite transformation is:

$$\mathbf{M_{wc,\,vc} = R.T} \qquad (7\text{-}1)$$

Where T is the Translation matrix that takes the viewing origin point P_0 to the world origin and R is the rotation matrix that align the axes of both reference frame.

7.5 WINDOW-VIEWPORT COORDINATE TRANSFORMATION

A window-viewport transformation describes the mapping of a (rectangular) window in one coordinate system into another (rectangular) window in another coordinate system. This transformation is defined by the section of the original image that is transformed (clipping window), the location of the resulting window (viewport), and how the window is translated, scaled or rotated. The following derivation shows how easy this transformation generally is (i.e., without rotation):

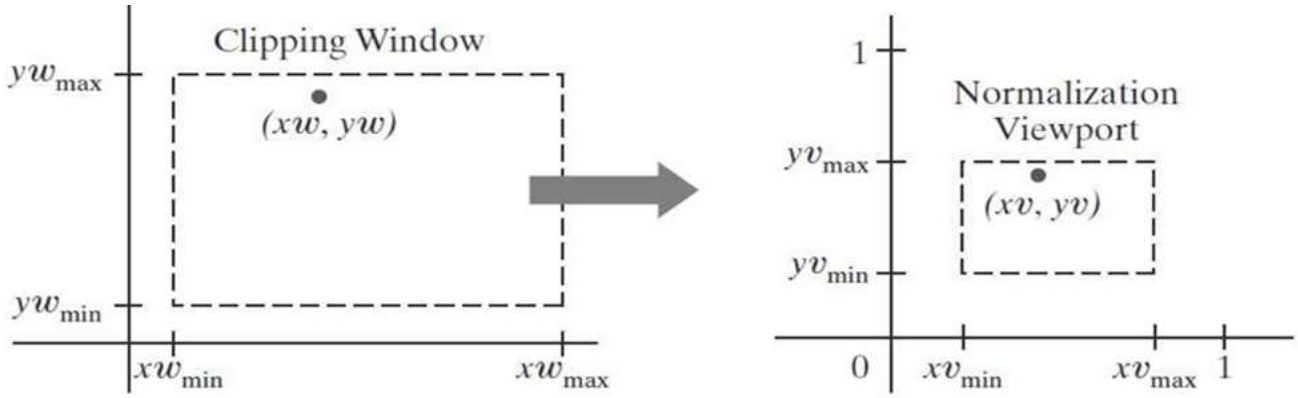


Figure 7.5: Normalized viewport

$$\frac{xv - xv_{min}}{xv_{max} - xv_{min}} = \frac{xw - xw_{min}}{xw_{max} - xw_{min}}$$

$$\frac{yv - yv_{min}}{yv_{max} - yv_{min}} = \frac{yw - yw_{min}}{yw_{max} - yw_{min}} \qquad (7\text{-}2)$$

By solving this expression for the viewport position (xv,yv), we have,

$$xv = s_x xw + t_x$$

$$yv = s_y yw + t_y \qquad (7\text{-}3)$$

Where scaling factors are:

$$s_x = \frac{xv_{max} - xv_{min}}{xw_{max} - xw_{min}}$$

$$s_y = \frac{yv_{max} - yv_{min}}{yw_{max} - yw_{min}} \qquad (7\text{-}4)$$

And translation factors are:

$$t_x = \frac{xw_{max}xv_{min} - xw_{min}xv_{max}}{xw_{max} - xw_{min}}$$

$$t_y = \frac{yw_{max}yv_{min} - yw_{min}yv_{max}}{yw_{max} - yw_{min}} \qquad (7\text{-}5)$$

Equation 7-3 can also be derived with a set of transformations that converts the window area into the viewport area. This conversion is performed with the following sequence of transformations:

1) Perform the scaling transformation using a fixed point position of (xw_{min}, yw_{min}) that scale the window area to the size of the viewport.
2) Translate the scaled window area to the position of the viewport.

If the scaling factors are equal (sx = sy), then the relative proportion of objects are maintained. Otherwise world objects will be constructed in either x or y direction when displayed on output device.

7.6 TWO-DIMENSIONAL VIEWING FUNCTIONS

We define here a viewing reference system in a PHIGS application program with the following function:

1) evaluateViewOrientation Matrix(x0, y0, xv, yv, error, viewMatrix)

it calculate viewmatrix for a world-to-viewport transformation matrix

where, x0, y0 are the coordinates of the viewing origin,

xv, yv are the world coordinate position for view up vector

an integer error code is generated if the input parameters are in error otherwise it produce the output.

2) evaluateViewMappingMatrix(xwmin, xwmax, ywmin, ywmax, xvmin, xvmax, yvmin, yvmax,error, viewMappingMatrix)

it sets the elements of the window-to-viewport mapping matrix:

where, xwmin, xwmax, ywmin, ywmax are the parameters for the window limits in viewing coordinates;

xvmin, xvmax, yvmin, yvmax aresets the limits of viewport with normalized coordinate position.

3) ***set ViewRepresentation (ws, viewIndex, viewMatrix, viewMappinMatrix, xclipmin, xclipmax, yclipmin, yclipmax, clipxy)***

with this function, we can store combinations of viewing and window-to viewport mappings for various workstations in a viewing table.

Where parameter ws assigned for workstation, parameter viewIndex sets an integer identifier for this particular window-viewport pair. viewMatrix and viewMappingMatrix are concatenated and referenced by viewIndex. Parameter clipxy is assigned to either the value noclip or the value clip. It allows us to turn off the clipping if we want to see the scene outside the viewport. Parameters xclipmin, xclipmax, yclipmin, yclipmax are define the additional clipping limits otherwise viewport boundaries define the clipping limits.

4) ***setViewIndex (viewIndex)***

it select a particular set of options from the viewing table. The viewIndex selection is then applied to subsequently specified output primitives and associated attributes and generates a display on each of the active workstation.

5) At the final stage, we apply a workstation transformation by selecting a workstation window-viewport pair:

setWorkstationwindow(ws, xwsWindmin, xwsWindmax, ywsWindmin, ywsWindmax)

setWorkstationViewport(ws, xwsVPortmin, xwsVPortmax, ywsVPortmin,ywsVPortmax)

where parameter ws gives the workstation number, window coordinate extents are specified in normalized space (form 0 to 1), and the viewport limits are in integer device coordinates.

CHECK YOUR PROGRESS

Q.5: What is view up vector?

.....

.....

.....

.....

Q.6: How can we get normalized viewport?

.....

.....

.....

.....

Q.7: What is the use of setViewIndex() function?

.....

.....

.....

.....

7.7 LET US SUM UP

- The world coordinate system (WCS) is a right handed Cartesian coordinate system where we describe the picture that is to be displayed.
- The normalized device coordinate system is a right handed coordinate system in which the display area of the virtual display device correspond to the unit (1 × 1) square whose lower left hand corner is at the origin of the coordinate system.
- Normalization transformation maps world coordinates to normalized device coordinates.
- Workstation transformation maps normalized device coordinates to physical device coordinates.
- Different types of PHIGS functions used in 2D viewing.

7.8 FURTHER READING

1) Dam, Andries Van; John, F. Hughes; Foley, James D.; Feiner, Steven K. (2002); *Computer Graphics Principles and Practice in C*. Pearson Education.
2) Hearn, Donald; Baker, M. Pauline. (2nd Edition). *Computer Graphics*. Prentice-Hall.

7.9 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: (a)

Ans. to Q. No. 2: (b)

Ans. to Q. No. 3: A world coordinate area that is selected for display is called window and an area of display device where window is mapped is called viewport. The window defines 'what' to be displayed and viewport defines 'where' to be displayed.

Ans. to Q. No. 4: The mapping of the world-coordinate scene to device coordinates is referred to as a viewport transformation. Two-dimensional viewing transformation is simply referred to as window-to-viewport transformation or windowing transformation.

Ans. to Q. No. 5: To establish a reference frame first we select a world coordinate let p = (x, y), then we establish the rotation, or orientation of this reference frame. For this we specify a world vector V that defines the viewing y_v direction. Vector V called as **view up vector**

Ans. to Q. No. 6: This transformation is defined by the section of the original image that is transformed (clipping window), the location of the resulting window (viewport), and how the window is translated, scaled or rotated. If the range of viewport define from 0 to 1, then its called as normalized viewport.

Ans. to Q. No. 7: it select a particular set of options from the viewing table. The viewIndex selection is then applied to subsequently specified output primitives and associated attributes and generates a display on each of the active workstation.

7.10 MODEL QUESTION

Q.1: Describe the Transformations used in 2-D viewing ?

Q.2: What is normalized coordinates?

Q.3: Define window and viewport. What are the use of these?

Q.4: Explain window-to-viewport transformation.

Q.5: Write a function to combine window-to-viewport and viewing for different workstation.

Q.6: Describe the process to get normalized coordinates from world coordinates.

Q.7: Write a procedure to implement the evaluateViewMappingMatrix function to calculates the elements of a matrix for performing the window-to-viewport transformation.

*** ***** ***

UNIT 8: CLIPPING ALGORITHMS

UNIT STRUCTURE

8.1 Learning Objectives
8.2 Introduction
8.3 Clipping
8.4 Line Clipping
    8.4.1 Cohen-Sutherland Algorithm
    8.4.2 Cyrus-Beck Line Clipping Algorithm
8.5 Polygon Clipping
    8.5.1 Sutherland-Hodgeman Algorithm
8.6 Let Us Sum Up
8.7 Further Reading
8.8 Answers to Check Your Progress
8.9 Model Questions

8.1 LEARNING OBJECTIVES

After going through this unit, you will be able to

- learn about clipping
- describe polygon clipping
- learn how the line clipping algorithms works
- describe polygon clipping algorithms.

8.2 INTRODUCTION

In the previous unit, we have learnt about the two-dimensional viewing which is a transformation process of real world objects into position point. The viewing functions have also been discussed.

In this unit, we will describe line clipping in computer graphics. Here we will find how line clipping and polygon clipping are done with their algorithms.

8.3 CLIPPING

The word ‘Clipping’ refers to the removal of objects, lines, or line segments that are outside the viewing pane. Comninos [2006] describes clipping as follows

'Clipping' is a process that subdivides elements of a picture to be displayed into its visible and invisible parts, thus allowing us to discard the invisible parts of the picture. In 2-D, the clipping process can be applied to a variety of graphics primitives. Clipping can be performed with respect to a clipping boundary, which may be a convex or concave polygonal boundary.

The specified region of the scene is called a clip window or simply window. Internal clipping removes parts of a picture outside the window and external clipping removes those parts that are inside a region.

Clipping is used to extract parts of a defined scene for viewing, antialiasing lines or object boundaries, identifying visible surfaces in three-dimensional views and many other applications.

Depending on the type of application, the clip window can be a general polygon or a window that have curved boundaries. Here we will be considering only rectangular shaped windows.

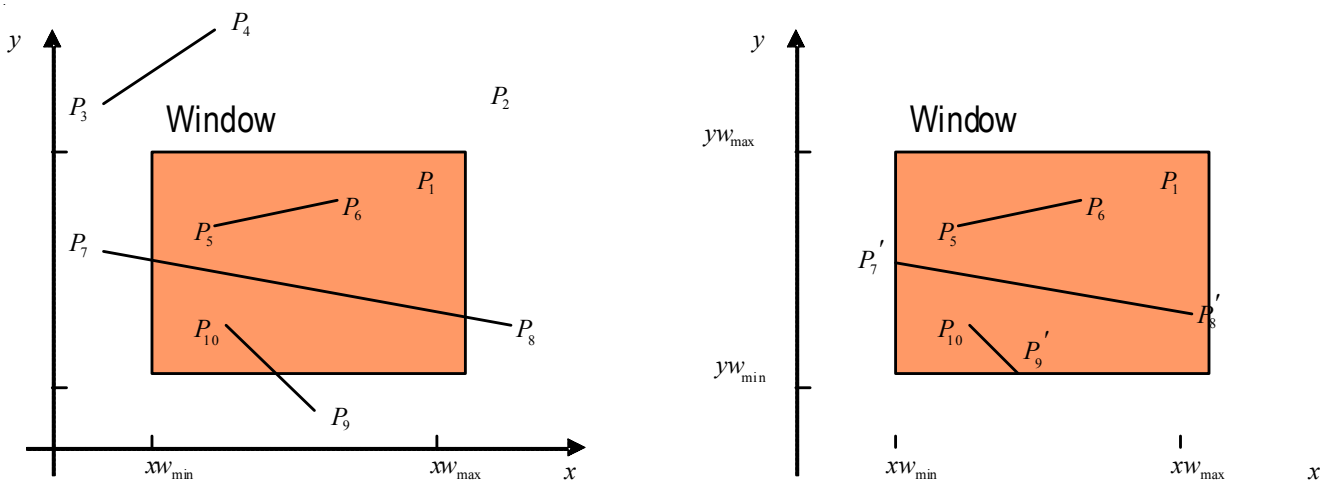
Clipping algorithms can be applied in world coordinates so that the contents of the window interior are mapped to device coordinates.

In the following sections we will discuss clipping that are applied to the following primitive types:

1) Lines
2) Polygon

8.4 LINE CLIPPING

Line clipping removes portions of line segments outside the clip window. To clip a line against a clip window, only the end-points are tested rather than testing all the points on the line. Figure 8.1 shows the line clipping against a clip window.



(a) Before Clipping **(b) After Clipping**

Figure 8.1: Line clipping

A line clipping algorithm has to test the following before clipping a line or any part of the line.

a) Whether the line lies completely inside the window.
b) Whether the line is completely outside the window
c) If the line lies partially inside and partially outside the specified window.

Lines that are not intersecting any border of the clipping window are either completely outside or completely inside the window boundary. The intersecting lines have one segment of the line inside and the rest outside.

The clipping region is commonly called as clipping window, or if the window is a standard rectangle, it is called as clipping rectangle. There are many line clipping algorithms. The Cohen-Sutherland Line clipping algorithm is generally used for line clipping.

8.4.1 Cohen-Sutherland Algorithm

This algorithm is one of the simple and most widely used algorithm. This algorithm works in two phases:

a) It identifies those lines which are intersecting the window boundary
b) Perform the clipping operation.

Let P(x,y) be a line. Say (x_s,y_s) and (x_e, y_e) be the line end-points.

A line is said to be completely inside if the following condition holds:

Xw_{min} d" x_s d" xw_{max}

Xw_{min} d" x_e d" xw_{max}

and yw_{min} d" y_s d" yw_{max}

yw_{min} d" y_e d" yw_{max}

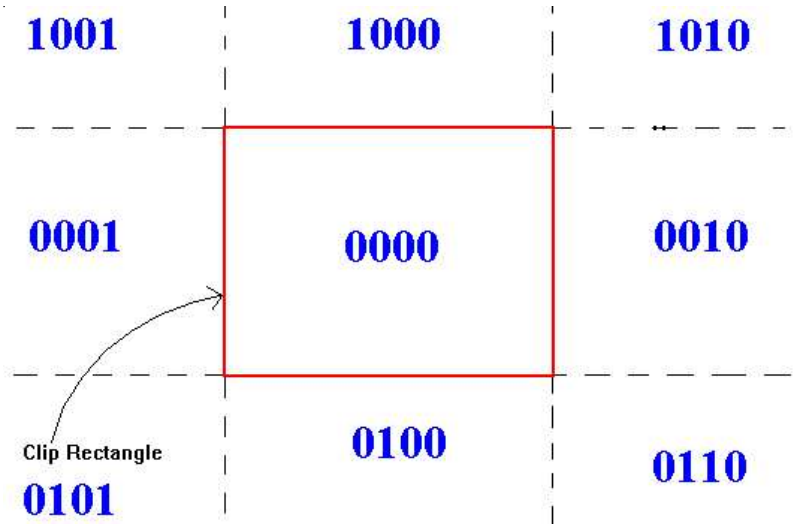
A line is said to be completely outside if the following condition holds:

$X_s < xw_{min}$ and $x_e < xw_{min}$

Or, $X_s > xw_{max}$ and $x_e > xw_{max}$

The algorithm assigns an 8 digit binary code to each of the two end-points of the line. This code is called the ***region code***. The region code identifies the location of the point relative to the boundaries of the clipping window. Each bit position in the region code is used to indicate one of the four relative coordinate positions of the point with respect to the clip window: to the left, right, top or bottom.

The figure below shows the binary codes that are assigned to the end-points of a line.



yw_{min} xw_{min} xw_{max}

Figure 8.2: The binary region codes assigned to the line end-points

This algorithm has extended the window boundaries to define 9 regions:

Top-left, top-center, top-right, center-left, center, center-right, bottom-left, bottom-center, bottom-right.

These 9 regions are uniquely identified by the given region code. The order– left, right, bottom, top **(LRBT)** will be used for the 8-bit region code.

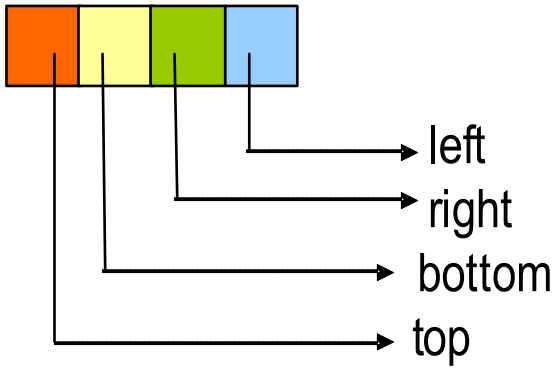


Figure 8.3: Region code order

A value of 1 in any bit position indicates that the point lies in that position; otherwise that bit position is set to a value of 0. For any point **P(x,y):**

- Left (first) bit is set to 1 if P lies to left of window, that is, when $x < xw_{min}$
- Right (second) bit is set to 1 if P lies to right of window, that is, when $x > xw_{-max}$
- Bottom (third) bit is set to 1 if P lies below window, that is, when $y < yw_{min}$
- Top (fourth) bit set is set to 1 if P lies above window, that is, when $y > yw_{max}$

If the point P lies within the clip window, then all the bit positions are set to a value 0, that is, 0000.

For a point that lies to the top-left corner of the clip window, the region code is 1001.

Thus the region codes for the line end-points are established. Using this region code it is now easier to find out which of the lines are inside the clip window and which are not.

- If the region code for both the end-points of the line is 0000, then the line is completely inside the clip window and is accepted.
- If the region codes for both the end-points has a 1 in the same bit position, then the line is completely outside the clip window and it is trivially rejected. For example, for the line where one end-point is having the region code as 1001 and the other end-point is having the code 0101.
- A logical AND operation can also be performed on the two region codes of the two end-points of the line to check for clipping the lines.

The lines which are not completely inside or completely outside the clip window are tested for intersection with the window boundaries. Figure 8.4 shows some lines and the clip window.

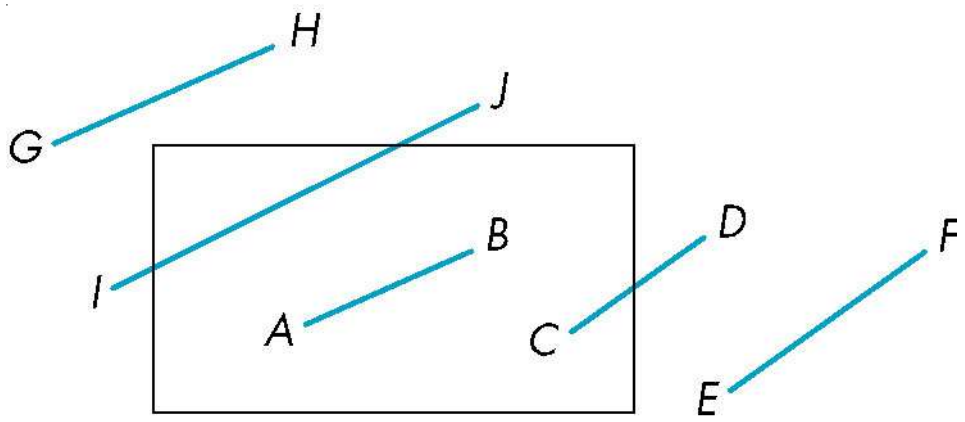


Figure 8.4: Clip window and some line segments

Let us consider the five cases above where we have to perform line clipping.

AB: The region code for endpoint A is 0000.

The region code for B is 0000.

So the line AB is accepted as both the region codes are 0000.

CD: C has code 0000.

D has 0010.

The line CD thus neither lie completely inside nor completely outside as one of the end-point lie inside the window and the other lie to the right of the window (as the right bit is 1). The line thus has to be shortened by intersecting with one of the window edge. Compute the code of intersection that is the new

end-point of the shortened line segment and then check the end-points (C and new one for D) again.

The location of the bit 1 determines which edge of the window to intersect. *If there were two 1's in the region codes of the end-points then two intersections might have to be done.*

EF: The end-point E has the code 0010

F has the code 0010.

As there is a 1 in the same bit position of the two end-points of the line, the line EF lies completely outside the window and it is thus discarded.

GH: The region code for point G is 1001

Code for the point H is 1000.

As there is a 1 bit in the same position of G and H, the line is completely outside and is discarded.

IJ: Region code for I is 0001

Region code for J is 1000.

As there is a 1 bit in the left bit position of I, it means that it lies to the left of the clip window. Again as a 1 exists in the top bit position of J so it lies to the top of the window. This means that the line IJ might have intersected the window in two different positions. The intersection point of I with the window say I' is first calculated. So the new line segment is now $I'J$. Then the intersection point of J with the window is calculated, say we have J'. So, $I'J'$ is the new line segment. The region codes for I' and J' is now checked. For I' it is 0000 and for J' it is also 0000. so the line $I'J'$ is accepted and lies inside the clip window.

8.4.2 Cyrus-beck Line Clipping Algorithm

The Cyrus-Beck line clipping algorithm is more efficient than the Cohen-Sutherland algorithm which uses repetitive clipping. It uses parametric line representation and simple dot products.

The algorithm explained above assumes that the clipping window is a regular rectangle. Cyrus and Beck have developed a generalized line clipping algorithm. It is applicable to an arbitrary convex region. It uses a parametric equation of a line segment to find the intersection points of a line with the clipping edges. This algorithm generally clips only once or twice unlike the Cohen Sutherland algorithm where the lines are clipped about four times. The algorithm is applicable to an arbitrary convex region. A parametric equation of a line segment is used in this algorithm to find the intersection points of a line with the clipping edges.

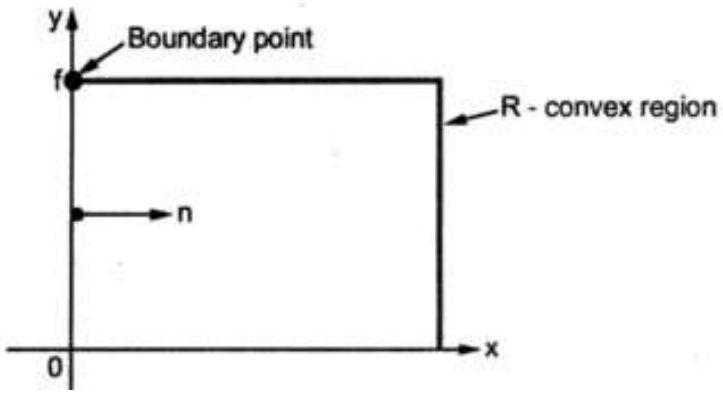


Figure 8.5: Cyrus-Beek line clipping

The parametric equation of a line segment from point P_1 to P_2 is:

$P(t) = P_1 + (P_2 – P_1)t$; $0 \leq t \leq 1$

Where **t** is a parameter, t = 0 at P_1 and t = 1 at P_2

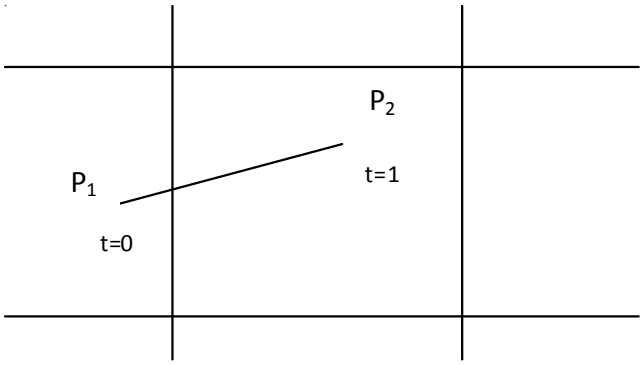


Figure 8.6: Cyrus beck algorithm with line segment p_1p_2

The value of the dot product is calculated so as to find the intersection point with the clipping window. Let $\mathbf{P}_E$ be a point on the clipping plane *E*.

Calculate: if < 0, vector pointed towards interior;

if = 0, vector pointed parallel to plane containing P;

if > 0, vector pointed away from interior.

Here, **n** stands for normal of the current clipping plane (pointed away from interior).

By this we select the point of intersection of line and clipping window where (dot product is 0) and hence clip the line.

CHECK YOUR PROGRESS

Q.1: Fill in the blanks:

a) ..... refers to the removal of objects, lines, or line segments that are outside the viewing pane.

b) Clipping can be performed with respect to a clipping boundary, which

may be a ..... or ..... polygonal boundary.

c) Internal clipping removes parts of a picture ..... the window and external clipping removes those parts that are ..... a region.

d) To clip a line against a clip window, only the ..... are tested rather than testing all the points on the line.

e) The region code in the Cohen- Sutherland line clipping algorithm identifies the location of the point relative to the ..... of the clipping window.

f) The ..... line clipping algorithm is more efficient than the Cohen-Sutherland algorithm which uses repetitive clipping.

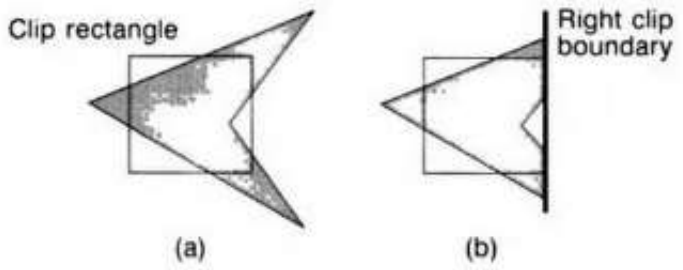
8.5 POLYGON CLIPPING

In the previous sections we have seen the line clipping algorithms. In this section we will go through the polygon clipping algorithm. Although the polygon is a collection of lines, the polygon clipping algorithms and the line clipping algorithms are different. If a line algorithm is used to clip a closed polygon, the original closed polygon becomes one or more open polygons or discrete lines.

In case of polygon clipping, 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. Sometimes, clipping one polygon may result in many polygons.

8.5.1 Sutherland-Hodgman Algorithm

One of the earliest polygon clipping algorithms is the Sutherland-Hodgeman algorithm. It is based on clipping the entire polygon against an edge of the window, then clipping the new polygon against the next edge of the window and so on until the polygon has been clipped against all of the four edges. In other words, it uses a divide-and-conquer strategy.



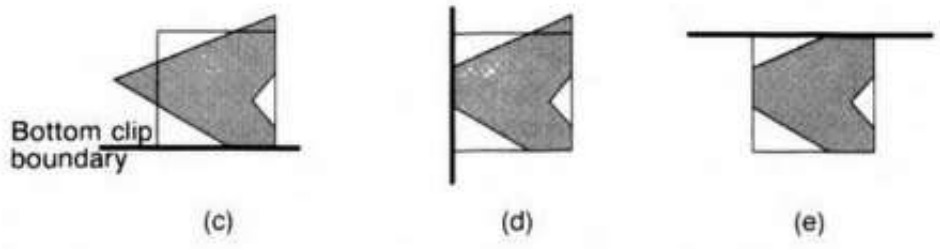


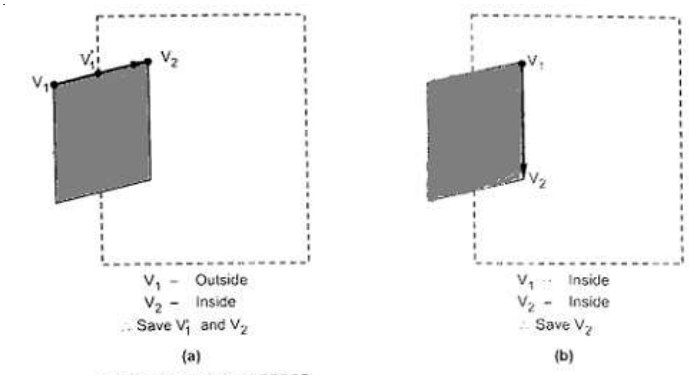
Figure 8.7:

The figure above shows polygon clipping, edge by edge. Figure (a) shows before clipping. The figure (b) shows clipping on the right, (c) Clip on bottom, (d) Clip on left, (e) Clip on the top and the polygon is fully clipped.

All the polygon vertices are processed against each clip rectangle boundary in turn. Beginning with the original polygon, it is clipped. At each step a new set of polygon vertices is generated and passed on to the next window boundary clipper, as shown in the figure above. This is the fundamental idea that is used in this algorithm.

The output of this algorithm is a list of polygon vertices all of which are on the visible side of a clipping plane. This is achieved by processing two vertices of each edge of the polygon around the clip boundary. This results in four possible relationships between the edges and clip boundary.

a) If the first vertex of the edge is outside the window boundary and the second vertex of the edge is inside then the intersection point of the polygon edge with the window boundary and the second vertex are added to the output vertex list.
b) If both vertices of the edge are inside the window boundary, only the second vertex is added to the output vertex list.
c) If the first vertex of the edge is inside the window boundary and the second vertex of the edge is outside, only the edge intersection with the window boundary is added to the output vertex list.
d) If both vertices of the edge are outside the window boundary, nothing is added to the output list.



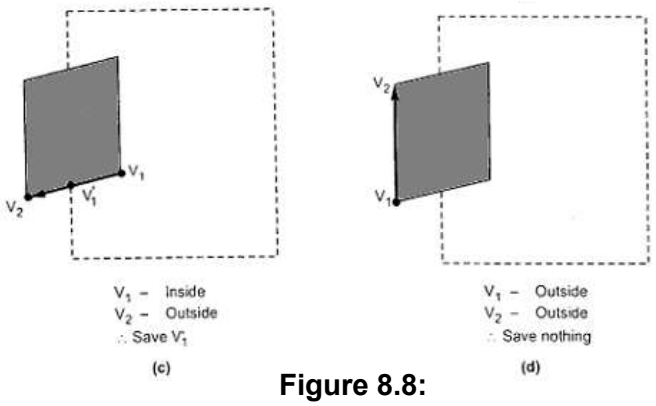


Figure 8.8:

CHECK YOUR PROGRESS

Q.2: Fill in the blanks:

a) ..... uses a divide-and-conquer strategy.

b) All the ..... vertices are processed against each clip rectangle boundary in turn.

c) The output of the Sutherland–Hodgman algorithm is a list of polygon vertices all of which are on the visible side of a .....

d) If both vertices of the edge are inside the window boundary, only the ..... is added to the output vertex list.

e) If both vertices of the edge are outside the window boundary, ... is added to the output list.

8.6 LET US SUM UP

- 'Clipping' refers to the removal of parts of a picture that are either inside or outside a specified region.
- Internal clipping removes parts of a picture outside the window; external clipping removes those parts that are inside a region.
- A point P(x,y) is inside the clip window if the point lies between the window boundaries. That is,
  xw_{min} d" x d" xw_{max}
  yw_{min} d" y d" yw_{max}
- A line clipping algorithm has to test the following before clipping a line or any part of the line.
  - Whether the line lies completely inside the window.
  - Whether the line is completely outside the window.
- The algorithm assigns an 8 digit binary code to each of the two end-points of the line.

This code is called the ***region code***. The region code identifies the location of the point relative to the boundaries of the clipping window.

- The Cyrus-Beck line clipping algorithm is more efficient than the Cohen-Sutherland algorithm which uses repetitive clipping
- Cyrus-Beck line clipping algorithm generally clips only once or twice unlike the Cohen Sutherland algorithm where the lines are clipped about four times.
- Although the polygon is a collection of lines, the polygon clipping algorithms and the line clipping algorithms are different.
- One of the earliest polygon clipping algorithms is the Sutherland-Hodgeman algorithm.
- The output of Sutherland-Hodgeman polygon clipping algorithm is a list of polygon vertices all of which are on the visible side of a clipping plane.

8.7 FURTHER READING

1) Hearn, D. & Baker, M. P. (1986). *Computer Graphics*. Prentice-Hall.
2) Plastock, R. A. & Kalley, G. (1986). *Schaum's Outline of Theory and Problems of Computer Graphics*. McGraw-Hill.

8.8 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: a) Clipping; b) Convex, concave; c) Outside, inside; d) end-points; e) boundaries; f) Cyrus-Beck

Ans. to Q. No. 2: a) Sutherland-Hodgman algorithm b) polygon; c) clipping plane; d) second vertex; e) nothing

8.9 MODEL QUESTIONS

Q.1: Define clipping. What is line and polygon clipping?

Q.2: Discuss Line clipping. Explain the Cohen-Sutherland Algorithm.

Q.3: Discuss Cyrus-beck line clipping algorithm.

Q.4: Explain the Sutherland–Hodgman algorithm for polygon clipping.

*** ***** ***

UNIT 9: 3-D OBJECT REPRESENTATION

UNIT STRUCTURE

9.1 Learning Objectives
9.2 Introduction
9.3 Polygon Surfaces
9.4 Polygon Tables
9.5 Plane Equations
9.6 Polygon Meshes
9.7 Spline Representations
9.8 Basic Illumination Models
  9.8.1 Ambient Light
  9.8.2 Diffuse Reflection
  9.8.3 Specular Reflection and the Phong Model
9.9 Polygon Rendering Methods
  9.9.1 Constant Intensity Shading
  9.9.2 Gouraud Shading
  9.9.3 Phong Shading
9.10 Let Us Sum Up
9.11 Further Reading
9.12 Answers to Check Your Progress
9.13 Model Questions

9.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- deduce equation of plane and explain the needs of planes in graphics
- describe various surface representation scheme
- describe cubic polynomial equation and their need in object representation
- explain the concept of surface revolution.

9.2 INTRODUCTION

In the previous unit we have acquainted with the concept of clipping along with various line and polygon clipping algorithms.

In this unit we will discuss transformations in three dimensions (3D). The most commonly used boundary representation for a 3D graphics object is a set of surface polygons. In most cases, the mathematics of linear transformations is easy to extend

from two dimensions to three, but the discussion here demonstrates that certain transformations such as rotations, are more complex in three dimensions because there are more directions about which to rotate and because the terms clockwise and counterclockwise no longer apply. We start with a short discussion of coordinate systems in three dimensions.

Coordinate System in Three Dimensions: In two dimensions, there is only one Cartesian coordinate system, with two perpendicular axes labeled *x* and *y.* A coordinate system in three dimensions consists of three perpendicular axes labeled *x*, *y*, and *z*, but there are two such systems, a ***left-handed*** and a ***right-handed*** and they are different [Figure 9.1(a)].

A right-handed coordinate system is constructed by the following rule. Align your right thumb with the positive *x* axis and your right index finger with the positive *y* axis. Your right middle finger will then point in the direction of positive *z*.

The rule for a left-handed system uses the left hand in a similar manner. It is also possible to define a left-handed coordinate system as the mirror image (reflection) of a right-handed one. Notice that one coordinate system cannot be transformed into the other by translating or rotating it. The difference between left-handed and right-handed coordinate system becomes important when a three-dimensional object is projected on a two-dimensional screen. We assume that the screen is positioned at the *xy* plane with its origin (i.e., its bottom left corner) at the origin of the three-dimensional system. We also assume that the object to be projected is located on the positive side of the *z* axis and the viewer is located on the negative side, looking at the projection of the image on the screen. Figure 9.1(b) shows that in a left-handed three-dimensional coordinate system, the directions of the positive *x* and *y* axes on the screen coincide with those of the three-dimensional *x* and *y* axes. In a right handed system as shown in figure 9.1(c), though, the two-dimensional *x* axis (on the screen) and the three-dimensional *x* axis point in opposite directions.

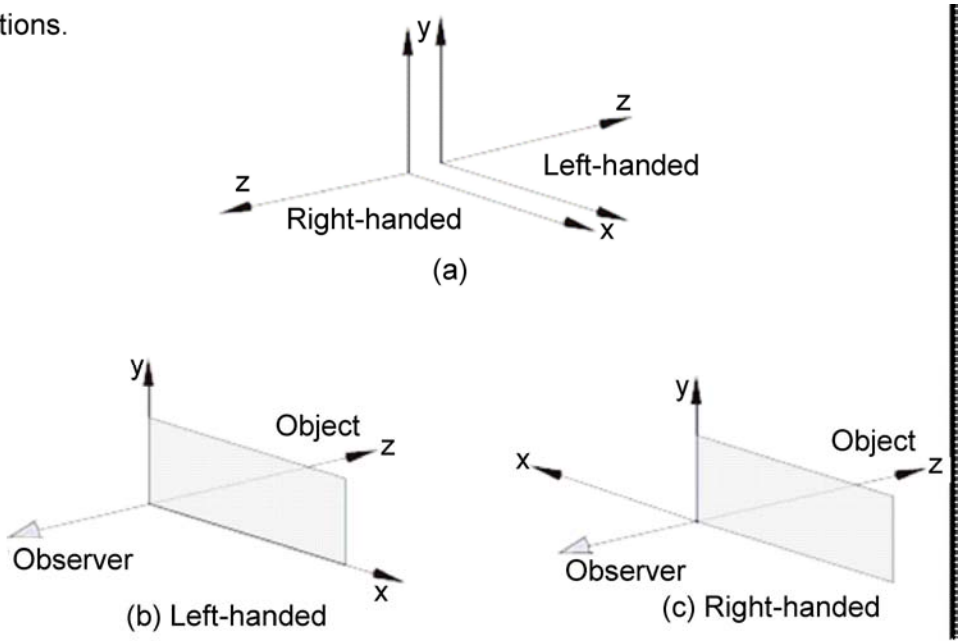


Figure 9.1: 3D Representation

Graphics scenes can contain many different kinds of objects such as flowers, clouds, rocks, water, bricks, wood paneling, rubber, paper, marble, steel, glass, plastic, and cloth, just to mention a few. So, it is probably not too surprising that there is no one method that we can use to describe objects that will include all characteristics of these different materials.

9.3 POLYGON SURFACES

The most commonly used boundary representation for a three-dimensional graphics object is a set of surface polygons that enclose the object interior. Many graphics systems store all object descriptions as sets of surface polygons. Polygon surface is the most common representation for 3D graphics object.

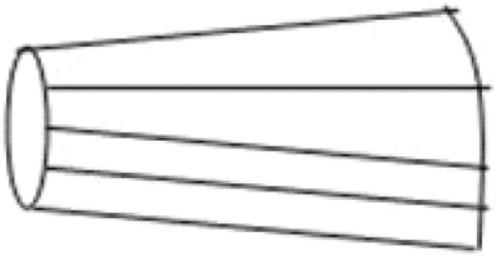


Figure 9.2: A 3D Object Representation by polygons

The polygon surface are common in design and solid-modeling applications, since wire frame display can be done quickly to give general indication of surface structure. Then realistic scenes are produced by interpolating shading patterns across polygon surface to illuminate.

9.4 POLYGON TABLES

Representation of vertex coordinates, edges and other property of polygon into table form is called ***polygon table***. Polygon data tables can be organized into two groups: *geometric table* and *attributes table*. Geometric table contains vertex coordinate and the other parameter which specify geometry of polygon. Attribute table stores other information like color, transparency etc. Convenient way to represent geometric table into three different table namely vertex table, edge table, and polygon table.

Vertex Table	
V1:	X1, Y1, Z1
V2:	X2, Y2, Z2
V3:	X3, Y3, Z3

Edge Table	
E1:	V1, V2
E2:	V2, V3
E3:	V3, V2

Polygon Surface Table	
S1:	E1, E2, E3
S2:	E3, E4, E5, E6

V4:	X4, Y4, Z4
V5:	X5, Y5, Z5

E4:	V3, V4
E5:	V4, V5
E6:	V5, V2

Figure 9.2: Geometric Data Table representation of a polygon

Vertex table stores each vertex of the polygon. *Edge table* stores each edge with the two endpoint vertex pointers back to vertex table. *Polygon table* stores each surface of polygon with edge pointer for the each edge of the surface. These three tables representation stores each vertex one time only and similarly each edge is also one time. So it will avoid representation of storing common vertex and edge so it is memory efficient method.

Another method to represent with two table vertex table and polygon table but it is inefficient as it will store common edge multiple times. Since tables have many entries for large number of polygon we need to check for consistency as it may be possible that errors may occurs during input. For dealing with that we add extra information into tables. For example, figure 9.3 shows edge table of above example with information of the surface in which particular edge is present.

E1:	V1, V2, S1
E2:	V2, V3, S1
E3:	V3, V1, S1, S2
E4:	V3, V4, S2
E5:	V4, V5, S2
E6:	V5, V1, S2

Figure 9.3: Edge table of above example with extra information as surface pointer

Now, if any surface entry in polygon table will find edge in edge table it will verify whether this edge is of particular surface's edge or not if not it will detect errors and may be corrected if sufficient information is added.

9.5 PLANE EQUATIONS

For producing display of 3D objects we must process the input data representation for the object through several procedures. For this processing we sometime need to find orientation and it can be obtained by vertex coordinate values and the equation of polygon plane. Equation of plane is given by,

$$Ax + By + Cz + D = 0$$

where, x, y, and z are any point on the plane and A, B, C, D are constants by solving three plane equation for three non collinear points. Let us solve simultaneous equation for ratio A/D, B/D, and C/D as follows:

$$\frac{A}{D}x_1 + \frac{B}{D}y_1 + \frac{C}{D}z_1 = -1$$

$$\frac{A}{D}x_2 + \frac{B}{D}y_2 + \frac{C}{D}z_2 = -1$$

$$\frac{A}{D}x_3 + \frac{B}{D}y_3 + \frac{C}{D}z_3 = -1$$

Solving by determinant

$$A = \begin{vmatrix} 1 & y_1 & z_1 \\ 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \end{vmatrix} \quad B = \begin{vmatrix} x_1 & 1 & z_1 \\ x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \end{vmatrix} \quad C = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} \quad D = -\begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix}$$

By expanding a determinant we get:

$$A = y_1(z_2 - z_3) + y_2(z_3 - z_1) + y_3(z_1 - z_2)$$

$$B = z_1(x_2 - x_3) + z_2(x_3 - x_1) + z_3(x_1 - x_2)$$

$$C = x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2)$$

$$D = -x_1(y_2z_3 - y_3z_2) + x_2(y_3z_1 - y_1z_3) - x_3(y_1z_2 - y_2z_1)$$

This values of A, B, C, D are then store in polygon data structure with other polygon data.

Orientation of plane is described with normal vector to the plane.

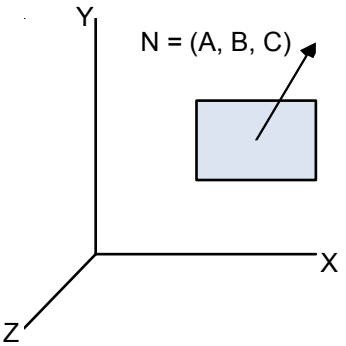


Figure 9.4: The vector N normal to the surface

Here, N = (A, B, C) where A, B, C are the plane coefficient. When we are dealing with the polygon surfaces that enclose object interior we define the side of the faces towards object interior is as inside face and outward side as outside face. We can calculate normal vector N for any particular surface by cross product of two vectors in counter clockwise direction in right handed system then.

N = (V2 – V1)(V3 – V1)

Now N gives values of A, B, C for that plane and D can be obtained by putting these values in plane equation for one of the vertices and solving for D.

Using plane equation in vector form we can obtain D as:

N . P = –D

Plane equation is also used to find position of any point compare to plane surface as follows:

If Ax + By + Cz + $\neq$ 0 the point (x, y, z) is not on that plane.

If Ax + By + Cz + < 0 the point (x, y, z) is inside the surface.

If Ax + By + Cz + D > 0 the point (x, y, z) is outside the surface.

These equations are valid for right handed system provided plane parameter A, B, C, and D were calculated using vertices selected in a counter clockwise order when viewing the surface in an outside to inside direction.

9.6 POLYGON MESHES

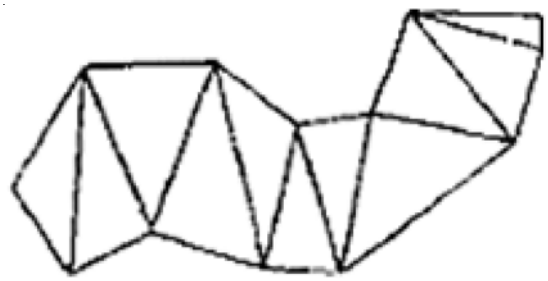


Figure 9.5: A triangle strip formed with 11 triangle connecting 13 vertices

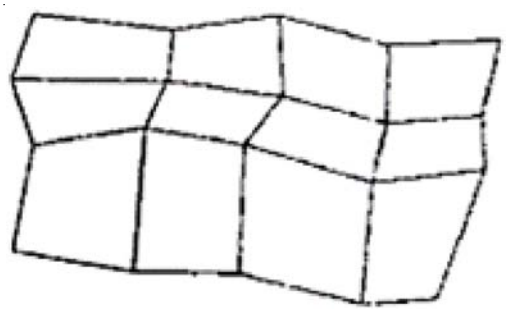


Figure 9.6: A quadrilateral mesh containing 12 quadrilaterals constructed from a 5 by 4 input vertex array

Polygon mesh is a collection of edges, vertices and faces that defines the shape of the polyhedral object in 3D computer graphics and solid modeling. An edge can be shared by two or more polygons and vertex is shared by at least two edges.

Polygon mesh is represented in following ways:

- Explicit representation
- Pointer to vertex list
- Pointer to edge list

Explicit Representation: In explicit representation each polygon stores all the vertices in order in the memory as,

$$P = (((x_1, y_1, z_1), (x_2, y_2, z_2)), ..., (x_m, y_m, z_m), (x_n, y_n, z_n)))$$

It process fast but requires more memory for storing.

Pointer to Vertex List: In this method, each vertex stores in vertex list and then polygon contains pointer to the required vertex.

$$V = ((x_1, y_1, z_1), (x_2, y_2, z_2)), ..., (x_n, y_n, z_n))$$

And now polygon of vertices 3, 4, 5 is represented as:

$$P = ((v_3, v_4), (v_4, v_5), (v_5, v_3)).$$

It is considerably space saving but common edges is difficult to find.

Pointer to Edge List: In this polygon have pointers to the edge list and edge list have pointer to vertex list for each edge two vertex pointer is required which points in vertex list.

$$V = ((x_1, y_1, z_1), (x_2, y_2, z_2)), ..., (x_n, y_n, z_n))$$

$$E = ((v_1, v_2), (v_2, v_3), ..., (v_n, v_m))$$

$$P = (E_1, E_2, E_3)$$

This approach is more memory efficient and easy to find common edges.

9.7 SPLINE REPRESENTATIONS

Spline is flexible strip used to produce a smooth curve through a designated set of points. Several small weights are attached to spline to hold in particular position. Spline curve is a curve drawn with this method. The term spline curve now referred to any composite curve formed with polynomial sections satisfying specified continuity condition at the boundary of the pieces. A spline surface can be described with two sets of orthogonal spline curves. There are different Spline representation methods. Some of them are described below.

Interpolation and Approximation Splines: We specify spline curve by giving a set of coordinate positions called *control point*s. This indicates the general shape of the curve. When curve section passes through each control point, the curve is said to interpolate the set of control points and that spline is known as ***Interpolation Spline***.





Figure 9.7: Interpolation spline **Figure 9.8: Approximation spline**

When curve section follows general control point path without necessarily passing through any control point, the resulting curve is said to approximate the set of control points and that curve is known as ***approximation spline***. Spline curve can be modified by selecting different control point position. We can apply transformation on the curve according to need like translation scaling etc.

The convex polygon boundary that encloses a set of control points is called **convex hull**.

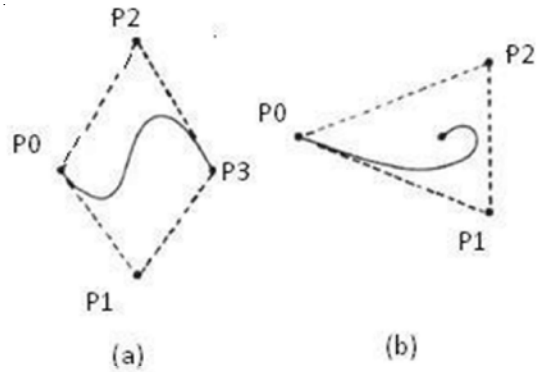


Figure 9.9: Convex hull shapes for two sets of control points

A poly line connecting the sequence of control points for an approximation spline is usually displayed to remind a designer of the control point ordering. This set of connected line segment is often referred as control graph of the curve.

Control graph is also referred as control polygon or characteristic polygon.

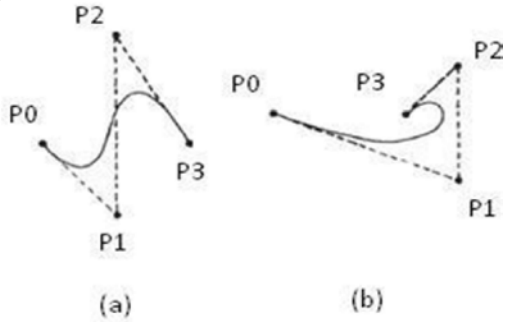


Figure 9.10: Control-graph shapes for two different sets of control points

Cubic Spline Interpolation Methods: Cubic splines are mostly used for representing path of moving object or existing object shape or drawing.

Sometimes it also used for design the object shapes. Cubic spline gives reasonable computation on as compared to higher order spline and more stable compare to lower order polynomial spline. So it is often used for modeling curve shape.

Cubic interpolation splines obtained by fitting the input points with piecewise cubic polynomial curve that passes through every control point.

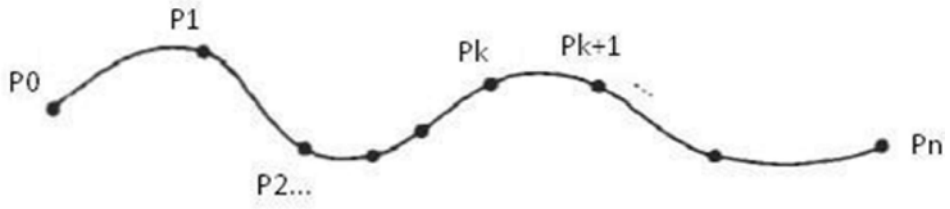


Figure 9.11: A piecewise continuous cubic spline interpolation of n+1 control points

$p_k = (x_k, y_k, z_k)$ where, k = 0, 1, 2, 3, ..., n

Parametric cubic polynomial for this curve is given by,

$(u) = a_x u^3 + b_x u^2 + c_x u + d_x$

$(u) = a_y u^3 + b_y u^2 + c_y u + d_y$

$(u) = a_z u^3 + b_z u^2 + c_z u + d_z$

where $(0 \leq u \leq 1)$

For above equation we need to determine for constant a, b, c and d the polynomial representation for each of n curve section.

This is obtained by settling proper boundary condition at the joints.

Natural Cubic Spline: Natural cubic spline is a mathematical representation of the original drafting spline. We consider that curve is in c^2 continuity means first and second parametric derivatives of adjacent curve section are same at control point. For the "n+1" control point we have n curve section and 4n polynomial constants to find. For all interior control points we have four boundary conditions. The two curve section on either side of control point must have same first and second order derivative at the control points and each curve passes through that control points.We get other two condition as p_0 (first control points) starting and (last control point) is end point of the curve.We still required two conditions for obtaining coefficient values.

One approach is to setup second derivative at p_0 and p_n to be 0. Another approach is to add one extra dummy point at each end. i.e., we add p_{-1} and p_{n+1} then all original control points are interior and we get 4n boundary condition. Although it is mathematical model it has major disadvantage which is with change in the control point entire curve is

changed. So it is not allowed for local control and we cannot modify part of the curve.

9.8 BASIC ILLUMINATION MODELS

These models give simple and fast method for calculating the intensities of light for various reflections.

Basic Illumination Models: An illumination is a formula in variable associated to the surface properties and light condition to calculate the intensity of light reflected from a point on a surface. Based on standard lighting in a scene, following basic models are:

9.8.1 Ambient Light

A simple way to model indirect reflection. All surfaces in all positions and orientations are illuminated equally. We can describe it as follows:

- Ambient light is usually a scene-based intensity of non directional light.
- It affects every object in that scene.
- We can then incorporate this into our illumination equation,
  $I = I_a k_a$
- I_a is the ambient light intensity. The scene-based value that remains the same for all surfaces in the scene.
- K_a is the ambient reflection coefficient, a material based property that determines how much ambient light is actually reflected.
- Material based properties are the properties that characterise one surface from another.
- This equation allows individual surface properties to “reflect” a level of ambient light.

9.8.2 Diffuse Reflection

The diffused shading is produced by dull, smooth objects. When some intensity of light falls on object surface and that surface reflect light in all the direction in equal amount, then the resulting reflection is called **diffuse reflection**.

- Diffuse reflections are constant over each surface independent of our viewing direction.
- Amount of reflected light is dependent on the parameter Kd, the **diffuse reflection coefficient** or **diffuse reflectivity.**
- Kd is assign value in between 0 and 1 depending on reflecting property. Shiny surface reflect more light so Kd is assign larger value while dull surface assign small value.

- If surface is exposed to only ambient light we calculate ambient diffuse reflection as: $I_{ambdiff} = K_d I_a$

  where, I_a the ambient light falls on the surface.

Practically, most of times each object is illuminated by one light source.

We assume that the diffuse reflection from source are scattered with equal intensity in all directions, independent of the viewing direction such a surface are sometimes referred as **ideal diffuse reflector** or **lambertian reflector**.

This is modelled by **lambert's cosine law.** this law states that the radiant energy from any small surface area dA in any direction ϕ_n relative to surface normal is proportional to $\cos\phi_n$.

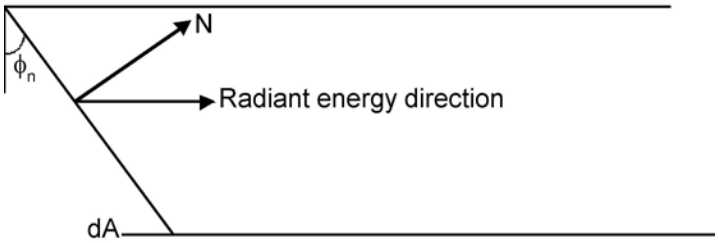


Figure 9.12: Radiant energy from a surface area dA in direction ϕ_n relative to the surface normal direction

The reflected light intensity does not depend on viewing direction so for lambertian reflection, the intensity of light is same in all viewing direction. Even though there is equal light distribution in all direction from perfect reflector the brightness of a surface does depend on the orientation of the surface relative to light source.

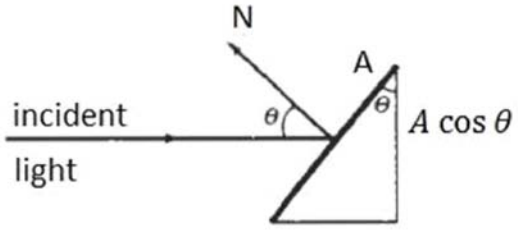


Figure 9.13: An illuminated area projected perpendicular to the path of the incoming light rays

If we denote the **angle of incidence** between the incoming light and surface normal as θ, then the projected area of a surface patch perpendicular to the light direction is proportional to $\cos\theta$.

If I_l is the intensity of the point light source, then the diffuse reflection equation for a point on the surface can be written as:

$$I_{l,diff} = K_d I_{l}\cos\theta$$

Surface is illuminated by a point source only if the angle of incidence is in the range 0^0 to 90^0 other than this value of θ light source is behind the surface.

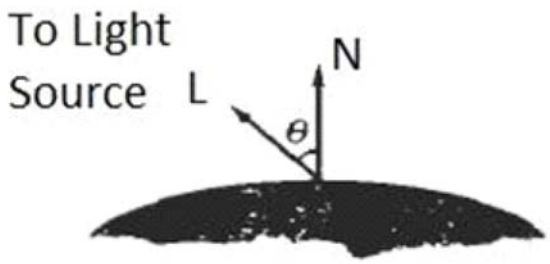


Figure 9.14: Angle of incidence θ between the unit light-source direction vector L and the unit surface normal N

As shown in figure, **N** is the unit normal vector to surface and **L** is the unit vector in the direction of light source, then we can take dot product of this two as:

$$N \cdot L = \cos\theta$$

and

$$I_{l,diff} = K_d I_l (N.L)$$

Now, in practical ambient light and light source both are present and so total diffuse reflection is given by:

$$I_{l,diff} = K_d I_l (N.L)$$

Here K_d is ambient reflection coefficient.

9.8.3 Specular Reflection and the Phong Model

When we look at an illuminated shiny surface, such as polished metal we see a highlight, or bright spot, at certain viewing directions. This phenomenon is called **specular reflection,** which is the result of total, or near total reflection of the incident light in a concentrated region around the specular reflection angle.

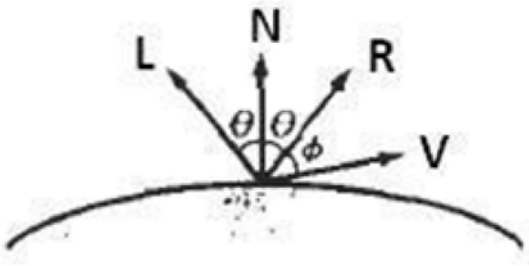


Figure 9.15: Specular reflection angle equals angle of incidence θ

Figure 9.15 shows specular reflection direction at a point on the illuminated

surface. The specular reflection angle equals the angle of the incident light. Here we use R as unit vector in direction of reflection, L is the unit vector point towards light vector, N is unit normal vector and V is unit vector in viewing direction.

Objects other than ideal reflectors exhibits specular reflection over a finite range of viewing positions around vector R. Shiny surface have a narrow specular reflection range and dull surface have wide specular reflection range. By **phong specular reflection model** or simply **phong model** sets the intensity of specular reflection proportional to $\cos^{ns}\phi$. Angle ϕ varies in between 0^0 to 90^0. Values assigned to **specular reflection parameter** (ns) is determined by the type of surface that we want to display. A shiny surface assign ns values large nearly 100 and dull surface assign small nearly 1. Intensity of specular reflection depends on the material properties of the surface and the angle of incidence as well as **specular reflection coefficient, (θ) for each surfaces.**

Then specular reflection is given by:

$$I_{spec} = w(\theta) I_{l} \cos^{ns}\phi$$

Where, I_l is the intensity of light source and ϕ is angle between viewing direction V and specular reflection direction R. Since, ϕ is angle between two unit vector V and R we can put $\cos\phi = V \cdot R$.

And also for many surfaces (θ) is constant so we take specular reflection constant as K_s so equation becomes.

$$I_{spec} = K_s I_l (V.R)^{ns}$$

Vector r is calculated in terms of vector L and N as shown in figure 9.16.

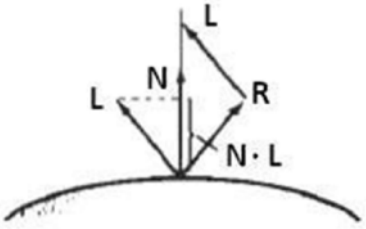


Figure 9.16: Calculation of vector R by considering projection onto the direction of the normal vector N

$$R + = (2N \cdot L)$$

$$= (2N \cdot L) - L$$

Somewhat simplified phong model is to calculate between half way vectors H and use product of H and N instead of V and R.

Here H is calculated as followes:

$$H = \frac{L + V}{|L + V|}$$

9.9 POLYGON RENDERING METHODS

Each polygon can be rendered with a single intensity, or the intensity can be obtained at each point of the surface using an interpolation scheme. Rendering means giving proper intensity at each point in a graphical object to make it look like real world object. Different rendering methods are:

- Constant Intensity Shading
- Gouraud Shading
- Phong Shading

Before going to the explanation of different rendering method, let us remind about shading. Shading is referred as the implementation of illumination model at the pixel points or polygon surfaces of the graphics objects.

9.9.1 Constant Intensity Shading

Constant Intensity Shading is also called Flat Shading. It is a fast and simple method. A single intensity is calculated for each polygon (in the polygon-mesh). All points in the surface of polygon are displayed with the same intensity value.

For getting good results through constant intensity shading:

i) Object is a polyhedron.
ii) Light sources illuminating the object are sufficiently far from the surface.
iii) The viewing position is sufficiently far from the surface.

9.9.2 Gouraud Shading

Intensity interpolation scheme developed by Gouraud. It represents a polygon surface by linearly interpolating intensity across the polygon surface. Intensity values for each polygon are matched with adjacent polygon along common edges. Eliminates the intensity discontinuity that can occur in flat shading.

In Gourd shading, polygon surface is represented by the following calculations:

i) Determine the average unit vector at each polygon vertex.
ii) Apply an illumination model to each vertex to calculate the vertex intensity.
iii) Linearly interpolate the vertex intensities over the polygon surface.

Advantages: It removes intensity discontinuities associated with flat shading.

Disadvantages: Highlights on the surface are sometimes displayed in irregular shapes. Linear intensity interpolation can cause intensity stripe called Mach bands,

to appear on the surface.

9.9.3 Phong Shading

It is normal vector interpolation shading. It's developed by Phong Bui Tuong. The displays more realistic highlights. It interpolate normal vectors and then apply illumination

Advantage: Intensity calculations produce more accurate results. Displays more realistic highlights on the surface.

Disadvantage: Requires more calculations.

CHECK YOUR PROGRESS

Q.1: A three dimensional graphics has

a) Two axes b) Three axes

c) Both (a) & (b) d) None of these

Q.2: ..... as the most commonly used boundary presentation for a 3-D graphics object.

a) Data polygon b) Surface polygon

c) System polygon d) None of these

Q.3: A three dimensional object can also be represented using .....

a) Method b) Equation

c) Point d) None of these

Q.4: An ..... can be considered as an extension of spherical surface

a) Bezier b) Ellipsoid

c) Shearing d) None of these

Q.5: ..... curve is one of the sp line approximation methods.

a) Bezier b) Ellipsoid

c) Shearing d) None of these

9.10 LET US SUM UP

- A polyline is a chain of connected line segments
- If we require that the interpolating piecewise polynomial be joined as smoothly as possible at the data points, the resulting curve is called a spline. Therefore, a spline of degree m has continuous derivatives up to order m-1 at the data point.
- Cubic splines are mostly used for representing path of moving object or existing object shape or drawing. Sometimes it also used for design the object shapes.Cubic spline

gives reasonable computation on as compared to higher order spline and more stable compared to lower order polynomial spline. So it is often used for modeling curve shape.

- Cubic interpolation splines obtained by fitting the input points with piecewise cubic polynomial curve that passes through every control point.

9.11 FURTHER READING

1) Agarwal, Udit. (2011). *Computer Graphics*. Katson Books.
2) Hearn, D. & Baker, M. P. (1986). *Computer Graphics*. Prentice-Hall.

9.12 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: (b) **Ans. to Q. No. 2:** (b)

Ans. to Q. No. 3: (b) **Ans. to Q. No. 4:** (a)

Ans. to Q. No. 5: (a)

9.13 MODEL QUESTIONS

Q.1: What is Polygon mesh?

Q.2: What is surface patch?

Q.3: What are the advantages of rendering polygons by scan line method?

Q.4: What are the advantages of rendering by patch splitting?

Q.5: What is a spline?

Q.6: Differentiate between interpolation spline and approximation spline.

Q.7: What are the steps involved in 3D transformation?

Q.8: Explain polygon surface technique and polygon table.

Q.9: Explain different Splin Representation technique.

Q.10: Explain different Basic Illumination Technique.

Q.11: Briefly describe different Polygon Rendering Technique.

*** ***** ***

UNIT 10: BEZIER AND B-SPLINE CURVES

UNIT STRUCTURE

10.1 Learning Objectives
10.2 Introduction
10.3 Hermite Curve
10.4 Bezier Curve
10.5 B-spline Curve
10.6 Bezier Surfaces and B-spline Surfaces
10.7 Let Us Sum Up
10.8 Further Reading
10.9 Answers to Check Your Progress
10.10 Model Questions

10.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- learn about the importance of Bezier and B-spline curves
- describe Hermite curves
- learn about Bezier and B-spline surfaces.

10.2 INTRODUCTION

In the previous unit we have discussed various three dimensional (3D) object representations.

In the field of mathematics and computer graphics, the common way to display smooth curve has been Bezier curve and B-spline curve. In this unit we will describe various concepts associated with Bezier and B-spline curves.

10.3 HERMITE CURVE

Hermite curve is a curve where two end points are specified. All of its control points are interpolated and convex hull property is not found in this curve.

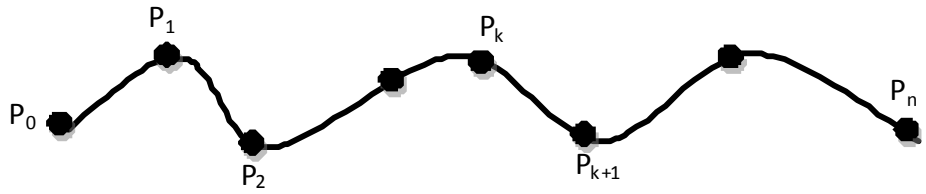


Figure 10.1: Continuous cubic-spline interpolation of n+1 control

Figure 10.1 shows the natural cubic spline. If we have n+1 control points to fit as shown in Figure 10.1 then we have n curve segments. Although natural cubic splines are a mathematical model for the drafting spline, they have a major disadvantage. If the position of any one control point is altered, the entire curve is effected.

A Hermite spline(developed by Charles Hermite) interpolating piecewise cubic polynomial with a specific tangent at each control point. Unlike natural cubic splines, Hermite spline can be adjusted locally because each curve segment is only dependent on its endpoint constraints.

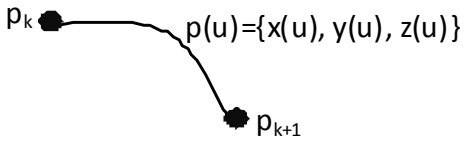


Figure 10.2: Hermite curve segment between control points p_k and p_{k+1}

If p(u) represents a parametric cubic point function for the curve segment between control points p_k and p_{k+1} as shown in figure 10.2, then the boundary conditions that define this Hermite curve segment are

$$P(0) = p_k$$
$$P(1) = p_{k+1} \qquad (10.1)$$
$$p'(0) = D\ p_k$$
$$p'(1) = D\ p_{k+1}$$

Where Dp_k and Dp_{k+1} specifies the values for the parametric derivatives (slop of the curve) at control points p_k and p_{k+1} respectively. Thus, the parametric cubic polynomial that is to be fitted between the pair of control points p_k and p_{k+1} for this Hermite-curve segments will be:

$$p(u) = au^3 + bu^2 + cu + d \quad 0 \le u \le 1 \qquad (10.2)$$

Here, a, b, c and d are four polynomial co-efficients and the x component of p is:

$$x(u) = a_x u^3 + b_x u^2 + c_x u + d_x \qquad (10.3)$$

Equation 10.2 can be represented in matrix form as follows:

$$p(u) = [u^3 u^2\ 1\ 0].\begin{bmatrix} a \\ b \\ c \\ c \end{bmatrix} \qquad (10.4)$$

and the derivative of the point function can be expressed as:

$$p'(u) = [3u^2 2u\ 1\ 0].\begin{bmatrix} a \\ b \\ c \\ c \end{bmatrix} \qquad (10.5)$$

Substituting endpoint values 0 and 1 for parameter u into equation 10.4 and 10.5, we can express the Hermite boundary conditions as in equation 10.1 in the matrix form:

$$\begin{bmatrix} p_k \\ p_{k+1} \\ Dp_k \\ Dp_{k+1} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \qquad (10.6)$$

Solving equation (10.6) for the polynomial coefficients a,b,c and d, we have:

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} p_k \\ p_{k+1} \\ Dp_k \\ Dp_{k+1} \end{bmatrix}$$

$$= \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} p_k \\ p_{k+1} \\ Dp_k \\ Dp_{k+1} \end{bmatrix}$$

$$= M_H \begin{bmatrix} p_k \\ p_{k+1} \\ Dp_k \\ Dp_{k+1} \end{bmatrix}$$

Where M_H is the Hermite matrix, is the inverse of the boundary constraint matrix. Thus equation 10.3 can be expressed in terms of boundary condition as:

$$p(u) = [u^3 u^2\ u\ 0].M_H \begin{bmatrix} p_k \\ p_{k+1} \\ Dp_k \\ Dp_{k+1} \end{bmatrix} \qquad (10.7)$$

Now, the Hermite blending function can be expressed by carrying out the matrix multiplications in equation 10.7 and collecting coefficients for the boundary constraints to obtain the polynomial form:

$$p(u) = p_k(2u^3 - 3u^2 + 1) + p_{k+1}(-2u^3 + 3u^2) + Dp_k(u^3 - 2u^2 + u) + Dp_{k+1}(u^3 - u^2)$$

$$= p_k H_0(u) + p_{k+1} H_1(u) + Dp_k H_2(u) + Dp_{k+1} H_3(u) \qquad (10.8)$$

The polynomials $H_k(u)$ for k = 0, 1, 2, 3 are called the blending functions because they blend the boundary constraint values. Hermite polynomials can be useful for some specific digitizing applications.

CHECK YOUR PROGRESS

Q.1: What is Hermite curve?

.....

.....

.....

.....

Q.2: Explain the process of calculation of x co-ordinate and y co-ordinate of a point on the Hermite curve.

.....

.....

.....

.....

.....

.....

.....

.....

10.4 BEZIER CURVE

A Bezier curve can use any number of control points. It is not necessary to specify tangent vector at any of the control points. The control points are blended by a set of

characteristic polynomial approximating functions called Bezier blending functions to result a Bezier curve segment.

If there are n+1 control points P_0, P_1, ..., P_n then mathematically Bezier curve segment that will fit to these n+1 points, is defined by:

$$\mathbf{P(u) = \sum_{i=0}^{n} P_i B_{i,n}(u) \quad where\ 0 <= u <= 1}$$

Here $\mathbf{B_{i,n}(u)}$ are referred to as Bernstein Basis functions or Bezier blending functions defined as:

$\mathbf{B_{i,n}(u) = C(n, i)u^i(1-u)^{n-i}}$ where **C(n, i)** is the binomial coefficient.

$\mathbf{C(n, i) = n! / (n-i)! i!}$

The equation can be expanded to get,

$$\mathbf{P(u) = P_0B_{0,n}(u) + P_1B_{1,n}(u) + ... + P_{n-1}B_{n-1,n}(u) + P_nB_{n,n}(u)}$$
$$\mathbf{= P_0\{C(n, 0)u^0(1–u)^n\} + P_1\{C(n, 1)u^1(1–u)^{n-1}\} + ...}$$
$$\mathbf{+ P_{n-1}\{C(n, n–1)u^{n–1}(1–u)\} + P_n\{C(n, n)u^n(1–u)^0\}}$$

The above equation highlights two important facts regarding Bezier curves which are as follows:

a) The degree of Bezier curve segment and thus the polynomial blending functions is one less than the number of control points.

b) Only the first control point P_0 and the last control point P_n is passed by the Bezier curve as P(u = 0) = P_0 and P(u = 1) = P_1.

CHECK YOUR PROGRESS

Q.3: What is Bezier curve?

.

.....

.....

.....

.....

.....

Q.4: How can Bezier curve be represented mathematically?

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

10.5 B-SPLINE CURVE

Bezier curves can be generalized to form B-spline curve. A special set of blending functions are given by it which provides local control of the curve shape as opposed to global control. In B-spline curve, without increasing the degree of the curve, control points can be added.

If P(u) be the position vectors along the curve of the function of the parameter 'u', we can define the B-spline curve as:

$$P(u) = \sum_{k=0}^{n} P_k B_{k,d}(u),\ u_{min} <= u <= u_{max} \text{ and } 2 <= d <= n+1$$

where P_k are an input set of control points.

B-spline parameters determines the range of parameter 'u' . The B-spline blending function $B_{k,d}$ is of degree d-1 where 'd' can be an integer value that can lie between 2 and number of control points n+1.

Blending functions for B-spline curves are defined by the Cox-deBoor recursion formulas:

$B_{k,1}(u) = 1$ if $u_k <= u <= u_{k+1}$ or 0, otherwise

$$B_{k,d}(u) = ((u-u_k) / (u_{k+d-1} - u_k))\, B_{k,d-1}(u) + ((u_{k+d} - u) / (u_{k+d} - u_{k+1}))\, B_{k+1,d}(u)$$

where d subintervals define each blending function of the total range of u. The selected set of subinterval end-points u_j is called knot vector.

CHECK YOUR PROGRESS

Q.5: Describe B-spline curve along with its blending function.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

10.6 BEZIER SURFACES AND B-SPLINE SURFACES

An object surface can be designed by two sets of Bezier curves which are orthogonal to each other with specification of the input mesh of control points. The Bezier surface is formed by the Cartesian product of Bezier blending functions.

$$\mathbf{P(u,v) = \sum_{j=0}^{m}\sum_{k=0}^{n} P_{j,k} B_{j,m}(v)\, B_{k,n}(u)}$$

where the location (m+1) is specified by $P_{j,k}$ by n+1 control points.

The above equation is the parametric vector equation of Bezier surface.

The vector point function of B-spline surface is given by the Cartesian product of B-spline blending functions in the form as:

$$\mathbf{P(u,v) = \sum_{k1=0}^{n_1}\sum_{k2=0}^{n_2} P_{k1,k2}\, B_{k1,d1}(u)\, B_{k2,d2}(v)}$$

where the location (n_1+1) is specified by the vector values of $P_{k1,k2}$ by (n_2+1) control points.

CHECK YOUR PROGRESS

Q.6: Write the parametric vector function of Bezier surface.

.....

.....

.....

.....

Q.7: Write the parametric vector function of B-Spline surface.

.....

.....

.....

.....

10.7 LET US SUM UP

- In the field of mathematics and computer graphics, the common way to display smooth curve has been Bezier curve and B-spline curve.
- Hermite curve is quite similar to Bezier curve where two end points are specified and another two points for the determination of tangents and these end-points. All of its control points are interpolated and convex hull property is not found in this curve.
- A Bezier curve can use any number of control points. It is not necessary to specify tangent vector at any of the control points. The control points are blended by a set of characteristic polynomial approximating functions called Bezier blending functions to result a Bezier curve segment.
- Bezier curves can be generalized to form B-spline curve. A special set of blending functions are given by it which provides local control of the curve shape as opposed to global control. In B-spline curve without increasing the degree of the curve control points can be added.

10.8 FURTHER READING

1) Agarwal, Udit. (2011). *Computer Graphics*. Katson Books.
2) Hearn, D. & Baker, M. P. (1986). *Computer Graphics*. Prentice-Hall.

10.9 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: Hermite curve is quite similar to Bezier curve where two end points are specified and another two points for the determination of tangents and these end-points. All of its control points are interpolated and convex hull property is not found in this curve.

Ans. to Q. No. 2: Hermite curve is quite similar to Bezier curve where two end points are specified and another two points for the determination of tangents and these end-points. All of its control points are interpolated and convex hull property is not found in this curve.

These curves require four control points and a cubic polynomial defined in terms of 't'. The control points q_0 and q_1 define the position of the curve at t = 0 and t = 1 respectively and its derivatives by q_0' and q_1'. If only the x co-ordinate of the points along the curve is considered, then the cubic polynomial can be defined as:

$$x(t) = a_3t^3 + a_2t^2 + a_1t + a_0$$

The values of a_0, a_1, a_2 and a_3 corresponding to a given set of control points are based on the above constraints that the curve passes through q_0 with derivative

q_0' and q_1 with derivative q_1'. The x components of each q namely x_0, $x2_0$, x_1 and $x2_1$ can be used to find the x co-ordinate of the curve. The y co-ordinate can be calculated in the similar manner. The polynomial can be rearranged in the following form:

$$x(t) = (2t^3-3t^2+1)x_0 + (t^3-2t^2+t)x2_0 + (-2t^3+3t^2)x_1 + (t^3-t^2)x2_1$$

A set of basis functions can be obtained in terms of t and the four control points can be blended by these to evaluate the final co-ordinate. x_0 corresponds to the position at t = 0 and x_1 corresponds to the position at t = 1.

Ans. to Q.No. 3: A Bezier curve can use any number of control points. It is not necessary to specify tangent vector at any of the control points. The control points are blended by a set of characteristic polynomial approximating functions called Bezier blending functions to result a Bezier curve segment.

Ans. to Q. No. 4: If there are n+1 control points P_0, P_1, ..., P_n then mathematically Bezier curve segment that will fit to these n+1 points, is defined by:

$$P(u) = \sum_{i=0}^{n} P_i B_{i,n}(u) \quad \text{where } 0 <= u <= 1$$

Here $B_{i,n}(u)$ are referred to as Bernstein Basis functions or Bezier blending functions defined as:

$B_{i,n}(u) = C(n, i)u^i(1-u)^{n-i}$ where **C(n, i)** is the binomial coefficient.

$C(n, i) = n! / (n-i)!\, i!$

The equation can be expanded to get,

$$P(u) = P_0B_{0,n}(u) + P_1B_{1,n}(u) + \ldots + P_{n-1}B_{n-1,n}(u) + P_nB_{n,n}(u)$$
$$= P_0\{C(n, 0)u^0(1-u)^n\} + P_1\{C(n, 1)u^1(1-u)^{n-1}\} + \ldots$$
$$+ P_{n-1}\{C(n, n-1)u^{n-1}(1-u)\} + P_n\{C(n, n)u^n(1-u)^0\}$$

Ans. to Q. No. 5: If P(u) be the position vectors along the curve of the function of the parameter 'u', we can define the B-spline curve as:

$$P(u) = \sum_{k=0}^{n} P_k B_{k,d}(u),\ u_{min} <= u <= u_{max} \text{ and } 2 <= d <= n+1$$

where P_k are an input set of control points.

B-spline parameters determines the range of parameter 'u'. The B-spline blending function $B_{k,d}$ is of degree d-1 where 'd' can be an integer value that can lie between 2 and number of control points n+1.

Blending functions for B-spline curves are defined by the Cox-deBoor recursion formulas:

$B_{k,1}(u) = 1$ if $u_k <= u <= u_{k+1}$ or 0, otherwise

$B_{k,d}(u) = ((u-u_k) / (u_{k+d-1} - u_k)) B_{k,d-1}(u) + ((u_{k+d} - u) / (u_{k+d} - u_{k+1})) B_{k+1,d}(u)$

where d subintervals define each blending function of the total range of u. The selected set of subinterval end-points u_j is called knot vector.

Ans. to Q. No. 6: The Bezier surface is formed by the Cartesian product of Bezier blending functions.

$$P(u,v) = \sum_{j=0}^{m} \sum_{k=0}^{n} P_{j,k} B_{j,m}(v) B_{k,n}(u)$$

where the location (m+1) is specified by $P_{j,k}$ by n+1 control points.

The above equation is the parametric vector equation of Bezier surface.

Ans to Q. No. 7: The vector point function of B-spline surface is given by the Cartesian product of B-spline blending functions in the form as:

$$P(u,v) = \sum_{k1=0}^{n_1} \sum_{k2=0}^{n_2} P_{k1,k2} B_{k1,d1}(u) B_{k2,d2}(v)$$

where the location (n_1+1) is specified by the vector values of $P_{k1,k2}$ by (n_2+1) control points.

10.10 MODEL QUESTIONS

Q.1: Explain the properties of Bezier curve.

Q.2: Explain the properties of B-Spline curve.

Q.3: How can B-Spline curve be represented mathematically?

Q.4: How can you differentiate between Bezier curve and B-Spline curve?

Q.5: Explain Bezier surface by giving its parametric vector function.

Q.6: Explain B-Spline surface by giving its parametric vector function.

Q.7: What are Bernstein basis functions?

Q.8: The Bezier blending function is given by:

$B_{i,n}(u) = C(n, i) u^i (1-u)^{n-i}$ where **C(n, i)** is the binomial coefficient and is represented as: $C(n, i) = n! / (n-i)! i!$

Generate an expression for the blending functions of a cubic Bezier curve.

*** ***** ***

UNIT 11: PROJECTIONS

UNIT STRUCTURE

11.1 Learning Objectives
11.2 Introduction
11.3 Planar Geometric Projections
    11.3.1 Parallel Projections
    11.3.2 Perspective Projections
11.4 Let Us Sum Up
11.5 Further Reading
11.6 Answers to Check Your Progress
11.7 Model Questions

11.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- learn about projection
- describe the types of projections
- define Parallel and Perspective projection.

11.2 INTRODUCTION

We have already learnt polygon surfaces, quadratic surfaces and basic illumination models in 3D object representation along with two different types of curves in the previous units.

This unit concentrates on projections. Projection is a way of converting the object in N dimensional system to N-1 dimensions. For example, the objects in 3D can be converted to 2D objects with the help of projections. In this unit we will discuss the types of projections.

11.3 PLANAR GEOMETRIC PROJECTIONS

The projection of a 3D object is defined by straight projection rays, called the projectors, emanating from a center of projection, passing through each point of the object, and intersecting a projection plane to form the projection.

The projection of a line is itself a line. So only the endpoints of the line actually needs to be projected. In this section we will deal with projections which are known as

planar geometric projections. These projections are always onto a plane rather than any curve surface and they uses straight projectors. There are other types of projections which are *non-planar* and *non-geometric*.

The planar geometric projections or simply projections can be divided into two basic classes:

a) **Parallel projections**

b) **Perspective projections**

This classification is done based on the relation of the center of projection to the projection plane.

If the distance from the one to the other is finite, then the projection is ***perspective***; if the distance is infinite then the projection is ***parallel***. The center of projection, being a point has the homogeneous coordinate in the form (x, y, z, 1).

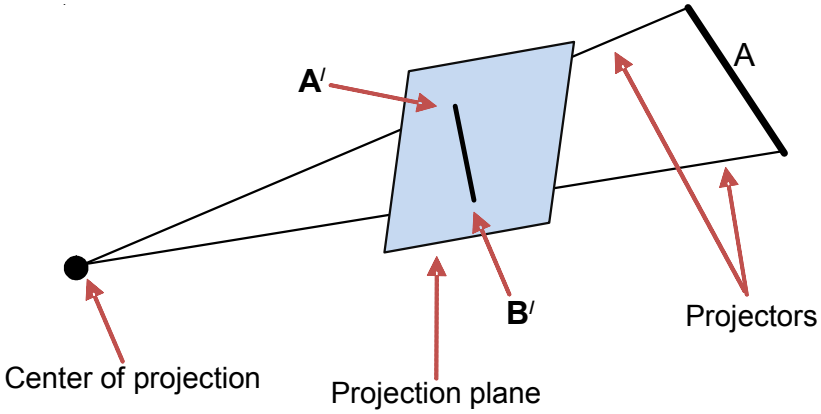


Figure 11.1: Perspective projection of an object to the projection plane/view plane

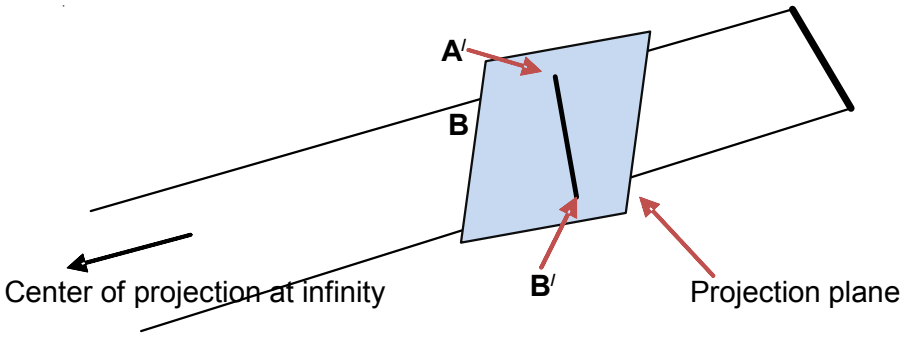


Figure 11.2: Parallel projection of an object to the projection plane/view plane

11.3.1 Parallel Projections

In parallel projection as the center of projection is at infinity the projectors

are parallel to each other. Parallel projection can be specified with a projection vector that defines the direction of projection. The parallel projection is a less realistic view.

Parallel projections are again divided into two types:

a) Orthographic parallel projection
b) Oblique parallel projection.

When the projection is perpendicular to the projection plane, it is an orthographic parallel projection. Otherwise, it is an oblique parallel projection.



(a) Orthographic Projection (b) Oblique Projection

The orthographic projections are mostly used to produce the front, side and top views of an object and so are often used in engineering drawings as the distances and angles can be measured from them.

They can also be used to display more than one side of an object. Such views are called axonometric orthographic projections. The isometric projection is a commonly used axonometric projection. An isometric projection is generated by aligning the projection plane so that it intersects each coordinate axis in which the object is defined, called the ***principal axis***, at the same distance from the origin.

Oblique parallel projection is obtained by projecting points along parallel lines that are not perpendicular to the projection plane.

11.3.2 Perspective Projections

To obtain a perspective projection of a 3D object, the points along the projectors that meet at the center of projection are transformed.

The visual effect of the perspective projection is similar to that of the human vision system and that of the photographic system. The projection view is very much realistic.

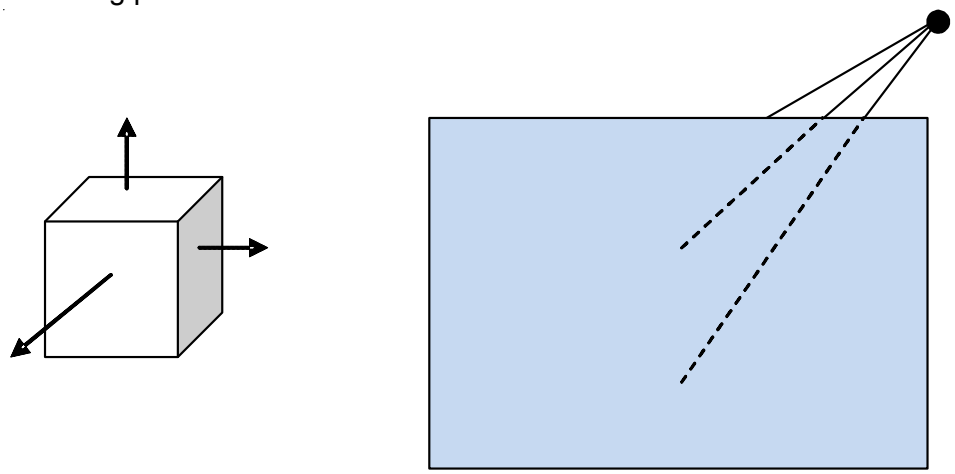
The size of an object's projection varies inversely with the distance of that object from the center of projection. This is known as ***perspective foreshortening***. So the exact shape, measurements and the distance and angles cannot be measured from the perspective projection.

The point at which a set of projected parallel lines appear to converge is called a ***vanishing point***. In 3D, the parallel lines meet only at infinity, so the

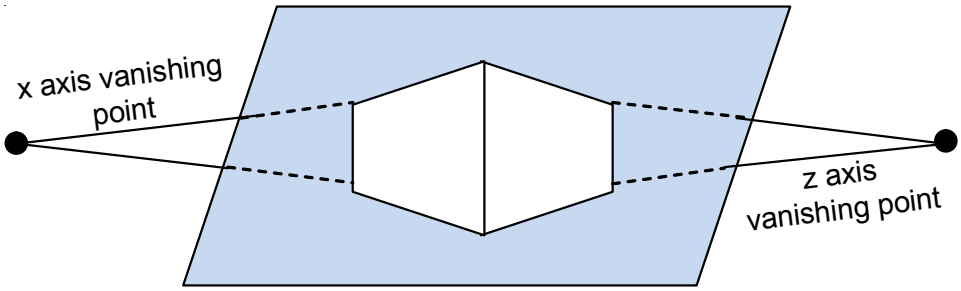
vanishing point is at infinity. There may be a number of vanishing points in an image or a picture. It depends on how many set of parallel lines exists in that picture.

In case of 3D, if the set of parallel lines is parallel to any one of the three principal axes, then the vanishing point is called the *axis vanishing point*. As there are three principal axes, there can be at most three axis vanishing points. For example, if the projection plane cuts only the z-axis (that is, it is perpendicular to it), only the z-axis has a principal vanishing point as the lines which are parallel to the x and y axis are also parallel to the projection plane and so have no vanishing point.

Perspective projections are classified depending on the number of principal vanishing points.



(a) Coordinate description (b) One-point Perspective Projection



(c) Two-point Perspective Projection

The figure (b) above shows a one-point perspective projection of a cube. From the figure it is clear that it is a one-point projection because the lines parallel to the x and y-axes do not converge; only the lines which are parallel to the z-axis do converge. The next figure (c) above shows the construction of a two-point perspective projection. Here the projection plane cuts the x and the z-axis. The

lines parallel to the y-axis do not converge. Three point projections are very less frequently used.

CHECK YOUR PROGRESS

Q.1: State True or False:

a) Projection is a way of converting the object in N dimentional system to N-1 dimensions.

b) The center of projection is called projector.

c) Planar geometric projections are the ones which are always onto a plane.

d) Perspective projection is the only class of projection.

e) If the distance from the center of projection to the projection plane is infinite, then the projection is called parallel projection.

f) Projective projections are more realistic than the parallel projections.

g) When the projection is perpendicular to the projection plane, it is an oblique parallel projection.

h) The orthographic projections which are used to display more than one side of an object are called axonometric projections.

i) Perspective foreshortening means the size of an object's projection varies directly with the distance of that object from the center of projection.

j) One-point projection and two-point projections are the two types of perspective projections.

11.4 LET US SUM UP

- Projections can be divided into two basic classes : parallel projections and perspective projections.
- Parallel projections has less realistic view.
- The size of an object's projection varies inversely with the distance of that object from the center of projection. This is known as *perspective foreshortening*.

11.5 FURTHER READING

1) Foley, James D. *Computer Graphics: Principles and Practice*.

2) Hearn, D. & Baker, M. P. (1986). *Computer Graphics*. Prentice-Hall.

3) Plastock and Kalley. (1986). Computer Graphics, Schaum's Outlines. McGraw-Hill.

✓ 11.6 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: a) true; b) false; c) true; d) False; e) True; f) True; g) false; h) true; i) false; j) true

? 11.7 MODEL QUESTIONS

Q.1: Define projection.

Q.2: Describe the types of projections.

Q.3: Give the mathematical description of perspective projections.

Q.4: What is Perspective foreshortening?

*** ***** ***

UNIT 12: 3-D GEOMETRIC TRANSFORMATIONS

UNIT STRUCTURE

12.1 Learning Objectives

12.2 Introduction

12.3 3D Transformations

12.3.1 Rotations in 3D

12.4 Let Us Sum Up

12.5 Further Reading

12.6 Answers to Check Your Progress

12.7 Model Questions

12.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- learn about different transformations
- learn how 3D transformations are done in Graphics
- define 3D transformations using homogeneous coordinate system.

12.2 INTRODUCTION

Previous unit introduces us projections and its classifications. Before going on to 3D-transformation, let us learn what 3D Computer Graphics is.

3D Graphics are representation of the objects displayed in the three dimensions, that is, length, breadth and height. The 3D graphics are often referred to as *3D models*. The difference between a 2D and a 3D graphics is essentially the difference between a painting and a sculpture in general. The 3D Computer Graphics can be used for real-time 3D viewing in animations, videos, training, simulations, architectural visualization etc.

The 3D transformations are almost same as the 2D transformations that we have disussed earlier in the previous block. The difference is that the 2D transformations were represented by 3 × 3 matrices but in case of 3D transformations 4 × 4 matrices are used.

All 2D transformations can be extended to 3D transformations by including the z-axis coordinates. 2D transformations are represented by 3 × 3 matrices using homogeneous coordinates. Similarly, 3D transformations can be represented by 4 × 4 matrices using homogeneous coordinates. Rotations in 3D are more complicated than the 2D rotations.

- **Translation:** A point P(x, y, z) in 3D coordinate system is translated to $P^{/}(x, y, z)$ as:

  $x^{/} = x + t_x$

  $y^{/} = y + t_y$

  $z^{/} = z + t_z$

  where t_x, t_y and t_z are the translation distances and can have any real value. In matrix form,

$$\begin{bmatrix} x^{/} \\ y^{/} \\ z^{/} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- **Scaling:** Scaling with respect to the coordinate origin can be written as:

  $x^{/} = x.s_x$

  $y^{/} = y.s_y$

  $z^{/} = z.s_z$

  where s_x, s_y ,s_z are the scaling parameters and can be any positive value.

In Matrix form, $\begin{bmatrix} x^{/} \\ y^{/} \\ z^{/} \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$

Scaling changes the size of the object and repositions it with respect to the coordinate origin. If the parameters values are not same then the relative dimensions are also changed. If the parameters values are same, that is in case of uniform scaling the original shape is retained.

- **Reflection:** 3D reflections are performed with respect to a reflection axis or a reflection plane. Reflection around a reflection axis is equivalent to 180^0 rotation about that axis. Reflection in a plane converts a right handed coordinate system into a left handed coordinate system. For example reflection in the xy-plane is given as follows:

$$RF = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- **Shearing:** In 3D transformation, the z-axis is taken into consideration to produce the distorted or sheared objects.

  Shearing with respect to the z-axis is given as:

$$SH_z = \begin{bmatrix} 1 & 0 & s_x & 0 \\ 0 & 1 & s_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

  s_x and s_y can have any real value.

12.3.1 Rotations in 3D

Rotation: Unlike 2D rotations, where all transformations are carried out in the xy plane, a 3D rotation can be specified around any line in the space. By convention, positive rotation angles produces anticlockwise rotations and negative angles produces clockwise rotations.

- **Coordinate Axes Rotations:** In 3D, a point can be rotated about any one of the three axes:
- **Rotation about the z-axis:** The 3D rotation equations are:

$$\left.\begin{aligned} x' &= x\cos\theta - y\sin\theta \\ y' &= x\sin\theta + y\cos\theta \\ z' &= z \end{aligned}\right\} \longrightarrow \textbf{(A)}$$

  where θ is the rotation angle. In homogeneous coordinate representation, we have,

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

  The transformation equations for the x and y axes can be obtained by a cyclic permutation of x, y and z. That is, in the above equation (A) replace all x by y, all y by z and all z by x.

- **Rotation about the x-axis:** The 3D rotation equations are:

$$y' = y\cos\theta - z\sin\theta$$

$$z' = y\sin\theta + z\cos\theta$$

$$x' = x$$

In matrix form, $\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$

- **Rotation about the y-axis:** The 3D rotation equations are:

  $z' = z\cos\theta - x\sin\theta$

  $x' = z\sin\theta + x\cos\theta$

  $y' = y$

In matrix form, $\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$

The inverse rotation matrix is formed by replacing θ by $-\theta$.

General 3D Rotation: A rotation matrix for any axis that does not coincide with a coordinate axis can be set up as a composite transformation involving combination of translations and the coordinate-axes rotations.

1) Translate the object so that the rotation axis passes through the coordinate origin.

2) Rotate the object so that the axis rotation coincides with one of the coordinate axes.

3) Perform the specified rotation about that coordinate axis.

4) Apply inverse rotation axis back to its original orientation.

5) Apply the inverse translation to bring the rotation axis back to its original position.

If the object is to be rotated about an axis which is parallel to any one of the coordinate axis then rotation is done as follows:

- Translate the object so that the rotation axis coincide with the parallel coordinate axis.
- Rotate the object about that axis.
- Translate back the object to its original position.

CHECK YOUR PROGRESS

Q.1: Fill in the blanks:

a) A point (x, y) has the homogeneous coordinates .....

b) ..... is repositioning an object along a straight line path from

one coordinate location to another coordinate location.

c) Translation is a rigid body transformation which moves the objects without .....

d) 2D transformations are represented by ..... matrices using homogeneous coordinates

e) 3D transformations can be represented by ..... matrices

f) In 3D transformation, the ..... axis is taken into consideration.

g) In uniform scaling, all the parameters have the ..... value.

h) Reflection around a reflection axis is equivalent to ..... about that axis.

i) Reflection in a plane converts a right handed coordinate system into a ..... coordinate system.

j) In 3D transformation, rotation can be specified around any ..... in the space.

Q.2: Say True or False:

a) Projection is a way of converting the object in N dimentional system to N-1 dimensions.

b) The center of projection is called projector.

c) Planar geometric projections are the ones which are always onto a plane.

d) Perspective projection is the only class of projection.

e) If the distance from the center of projection to the projection plane is infinite, then the projection is called parallel projection.

f) Projective projections are more realistic than the parallel projections.

g) When the projection is perpendicular to the projection plane, it is an oblique parallel projection.

h) The orthographic projections which are used to display more than one side of an object are called axonometric projections.

i) Perspective foreshortening means the size of an object's projection varies directly with the distance of that object from the center of projection.

j) One-point projection and two-point projections are the two types of perspective projections.

12.4 LET US SUM UP

- A transformation is any operation on a point in space (x, y) that maps the point's coordinates into a new set of coordinates (x1, y1).

- The geometric transformations are translation, rotation, scaling, reflection and shear.
- Rotation reposition an object along a circular path.
- Reflection produces a mirror image of an object.
- Shear distorts the shape of the objects and it seems the object is pulled horizontally in different amounts.
- Scaling means changing the size of an object. It is carried out by multiplying the coordinate values (x, y) of the vertex of the polygons by the scaling factors s_x and s_y.
- Positive values of s_x and s_y increases the object's size whereas negative values decreases its size.
- When $s_x = s_y$, the scaling is uniform and the original shape of the object is maintained.
- In homogeneous coordinate system, the 2D point (x, y) becomes (x, y, 1). Similarly, the 3D point (x, y, z) becomes (x, y, z, 1).
- Translation is repositioning an object along a straight line path from one coordinate location to another coordinate location.
- 3D transformations can be represented by 4 × 4 matrices.

12.5 FURTHER READING

1) Hearn, D. & Baker, M. P. (1986). *Computer Graphics*. Prentice-Hall.
2) Plastock and Kalley. (1986). Computer Graphics, Schaum's Outlines. McGraw-Hill.

12.6 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: a) (x, y, 1); b) Translation; c) Deformation; d) 3 × 3; e) 4 × 4; f) z – coordinate; g) same; h) 180^0 rotation; i) left-handed; j) line.

Ans. to Q. No. 2: a) true; b) false; c) true; d) False; e) True; f) True, g) false; h) true; i) false; j) true

12.7 MODEL QUESTIONS

Q.1: Discuss the 2D planar geometric transformations.

Q.2: What is 2D reflection? Discuss the x-axis and y-axis reflection.

Q.3: Write the matrix representations of the 2D and 3D transformation.

*** ***** ***

UNIT 13: 3-D VIEWING

UNIT STRUCTURE

13.1 Learning Objectives
13.2 Introduction
13.3 3-D Viewing Pipeline
13.4 Viewing Coordinates
13.5 View Volume and General Projection Transformation
13.6 Clipping
13.7 Let Us Sum Up
13.8 Further Reading
13.9 Answers to Check Your Progress
13.10 Model Questions

13.1 LEARNING OBJECTS

After going through this unit, you will be able to:

- learn about 3-D viewing pipeline
- describe viewing coordinate system
- learn to transform from World to Viewing Coordinates
- explain different types of general projection transformation
- define clipping
- normalize view volumes
- learn about viewport clipping
- learn about clipping in Homogeneous Coordinates.

13.2 INTRODUCTION

We have learnt various 3D geometric transformations in the previous unit. In this unit, we will learn about 3D viewing.

In two dimensional graphics application, viewing operations transfer positions from world coordinate plane to pixel positions in the output device plane. In two dimensional view, window and viewport both are normally consider as rectangular plane, so graphics package maps the world scene to device coordinates and clip the scene within the boundary of the viewport. In three dimensional graphics application, we have choices to how we represent the object. We can choose any spatial position like from front, from above or from the top to represent the view of the object. Three dimensional graphics techniques

and their application are fundamental to the entertainment, games, and computer-aided design industries.

13.3 THREE DIMENSIONAL VIEWING PIPELINE

The steps for computer generation of a view of 3D scene are similar to the process of taking photograph by a camera. For a snapshot, we positioned the camera at a particular point in space and then need to decide camera orientation for taking the snap. Finally when we snap the shutter, the scene is cropped to the size of window of the camera and the light from the visible surfaces is projected into the camera film.

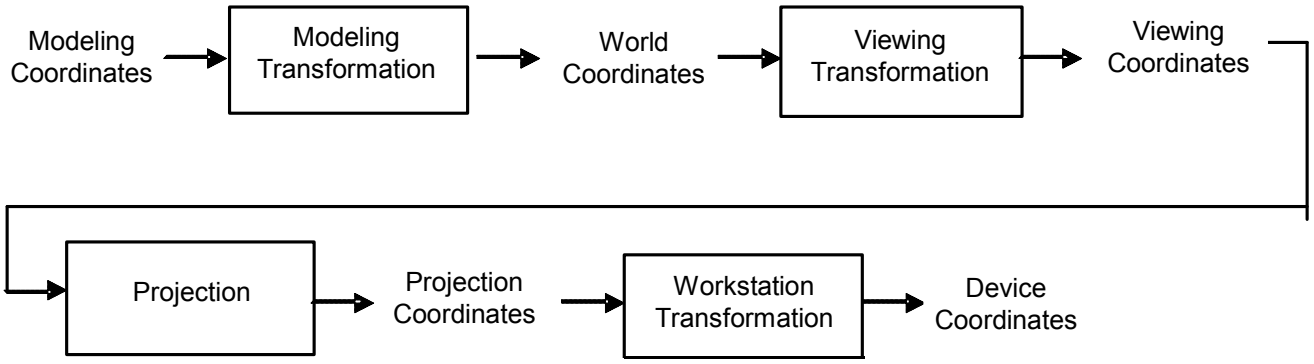


Figure 13.1: Viewing pipeline

13.4 VIEWING COORDINATES

To generate a view of an object in three dimension is similar to take a photograph of an object from different angles, at various distances and with different orientation of camera. Type and size of the camera lens determines the final appearance of the picture. These ideas are incorporated into three dimensional packages which generate the view of an object from spatial position and orientation and aperture size of the camera.

Specifying the view plane: To choose the particular view for scene, first establish the **viewing coordinate system**, also called as **view reference coordinate system.** A view plane or projection plane is set up perpendicular to viewing z_v axis. World coordinate positions in the scene are transformed to viewing coordinates, then viewing coordinates are projected onto the view plane.

To establish the viewing coordinate reference frame, we pick a world coordinate position called the viewing reference point. Next we select the positive direction for the viewing z_v axis, and the orientation of the view plane, by specifying the view plane normal vector, N. Graphic packages orient N relative to world coordinate origin. Now normal N is

directed line segment from the world origin to the selected coordinate position. Finally establish the view up vector 'V' in the positive y_v direction.

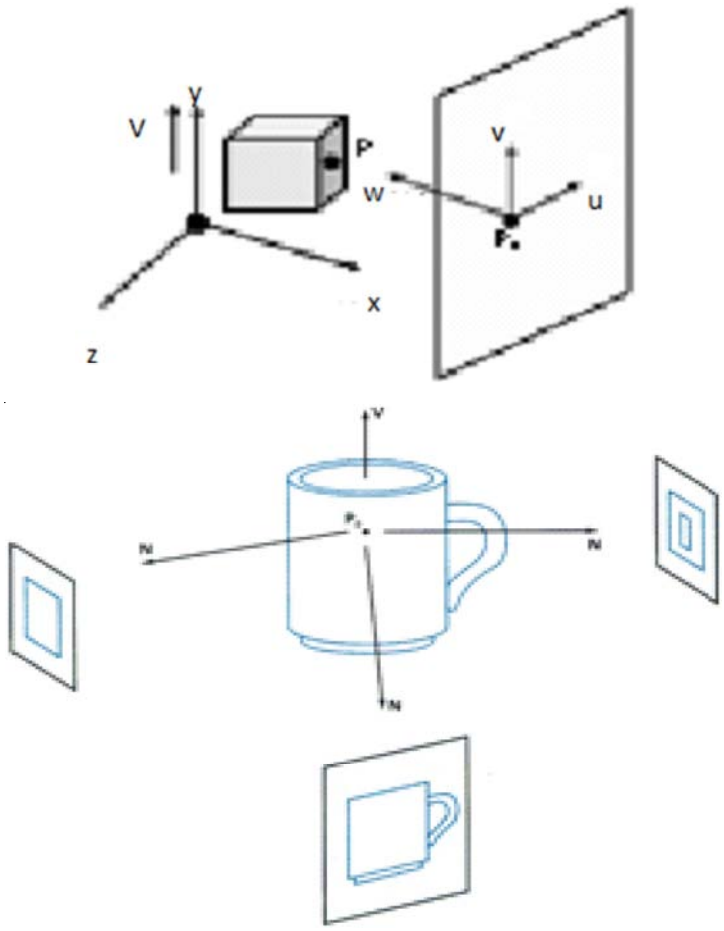


Figure 13.2: viewing scene from different direction with a fixed view reference point

Transformation from World to Viewing Coordinates: It is equivalent to a transformation that imposes the viewing reference frame onto the world frame using the basic geometrical translation and rotation operations:

1) Translate the view reference point to the origin of the world coordinate system.
2) Apply rotations to align x_v, y_v, z_v axes with world x_w, y_w, z_w axes respectively.

If the view reference point is specified at world position (x_0, y_0, z_0), this point is translated to the world origin with the matrix transformation:

$$T = \begin{bmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Another method for generating the rotation matrix is to calculate unit uvn vectors and form the composite rotation matrix directly.

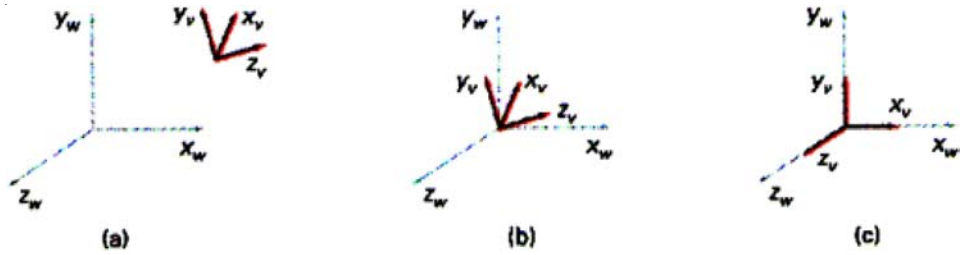


Figure 13.3: World to viewing coordinates

The vector N and V are calculated as:

$$n = \frac{N}{|N|} = (n1, n2, n3)$$

$$u = V \times \frac{N}{|V \times N|} = (n1, n2, n3) \qquad (13.1)$$

$$v = n \times u = (v1, v2, v3)$$

The composite Rotation matrix for viewing transformation is then,

$$R = \begin{bmatrix} u1 & u2 & u3 & 0 \\ v1 & v2 & v3 & 0 \\ n1 & n2 & n3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Which transforms u onto the world x_w axis, v onto the y_w axis and n onto the z_w axis.

All these transformations can be merged by multiplication into one matrix, which looks about like this:

$$M_{WC,VC} = R.T \qquad (13.2)$$

CHECK YOUR PROGRESS

Q.1: What is the basic requirement for a particular scene?

.....

Q.2: Name pocesses required for world to viewing coordinates transformation.

.....

13.5 VIEW VOLUMES AND GENERAL PROJECTION TRANSFORMATIONS

In three dimensional viewing, with the specification of the view window, we can set up a view volume. The volume of space that is actually converted into the image is called the **view volume**. Things inside the view volume come into the image; things that are not in the view volume are clipped and cannot be seen. For purposes of drawing, graphics packages applies a coordinate transform that maps the view volume onto a **cube**. The cube center is placed at the origin and extends from -1 to 1 in the x-direction, in the y-direction, and in the z-direction. The coordinate system on this cube is called as **clip coordinates**. The transformation from eye coordinates to clip coordinates is known as **projection transformation**. At this point, we haven't quite projected the 3D scene onto a 2D surface, but we can now do so simply by discarding the z-coordinate.

In the end, when things are drawn at **device coordinates**, the 2D coordinate system in which the actual drawing takes place on a physical display device such as the computer screen. Ordinarily, pixel is the unit measure in device coordinates. The drawing region is a rectangle of pixels which is as the viewport. The process which takes x and y from the clip coordinates and scales them to fit the viewport is called **viewport transformation**.

The primitive, such as a line or triangle, that is part of the scene and that might appear in the image that we want to make of the scene. The primitive goes through the following sequence:

1) The points that define the primitive are specified in object coordinates.
2) The points are first subjected to the model view transformation, which is a combination of the modeling transform that places the primitive into the world and the viewing transform that maps the primitive into eye coordinates.
3) The projection transformation is then applied to map the view volume that is visible to the viewer onto the clip coordinate cube. If the transformed primitive lies outside that cube, it will not be part of the image, and the processing stops. If part of the primitive lies inside and part outside both, the part that lies inside is processed further and other part clipped away and discarded.
4) Finally, the viewport transform is applied to produce the device coordinates that will actually be used to draw the primitive on the display device.

For a perspective transformation, far and near values must be positive numbers, and far must be greater than near. Anything that is closer to the viewer than the near distance or a distance which is greater than the far distance is discarded and does not appear in the rendered image. The volume of space that is represented in the

image is thus a “truncated pyramid” which is called as “**frustum”**. This pyramid is the view volume for a perspective projection:

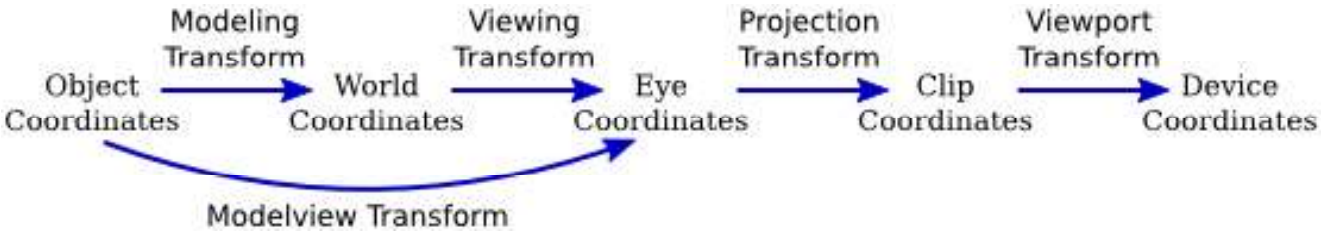


Figure 13.4: View volume

Orthographic projections are easier to understand: In an orthographic projection, z-coordinate is discarded. So the three dimensional world is projected onto a 2D image. This type of projection is unrealistic. For example, the apparent size of an object does not depend on its distance from the viewer. Objects in back of the viewer as well as in front of the viewer can be visible in the image. Orthographic projections are still useful, however, especially in interactive modeling programs where it is useful to see true sizes and angles, undistorted by perspective. Negative value is allowed for near if the clipping plane behind the viewer.

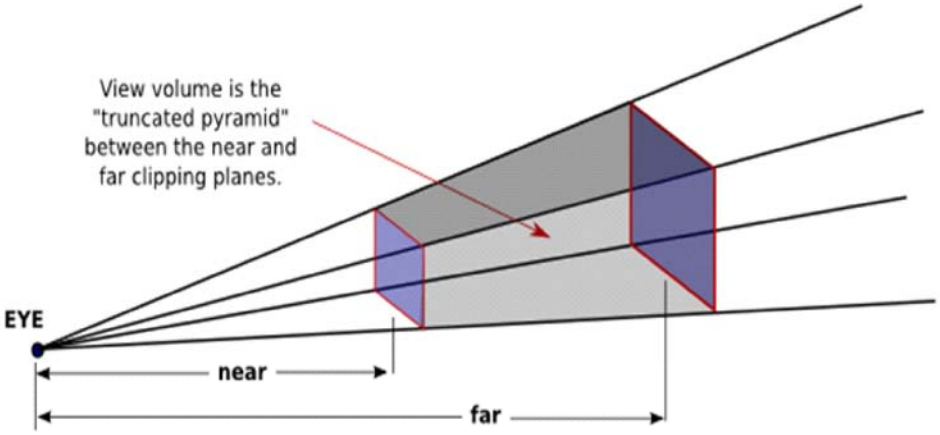


Figure 13.4: View volume

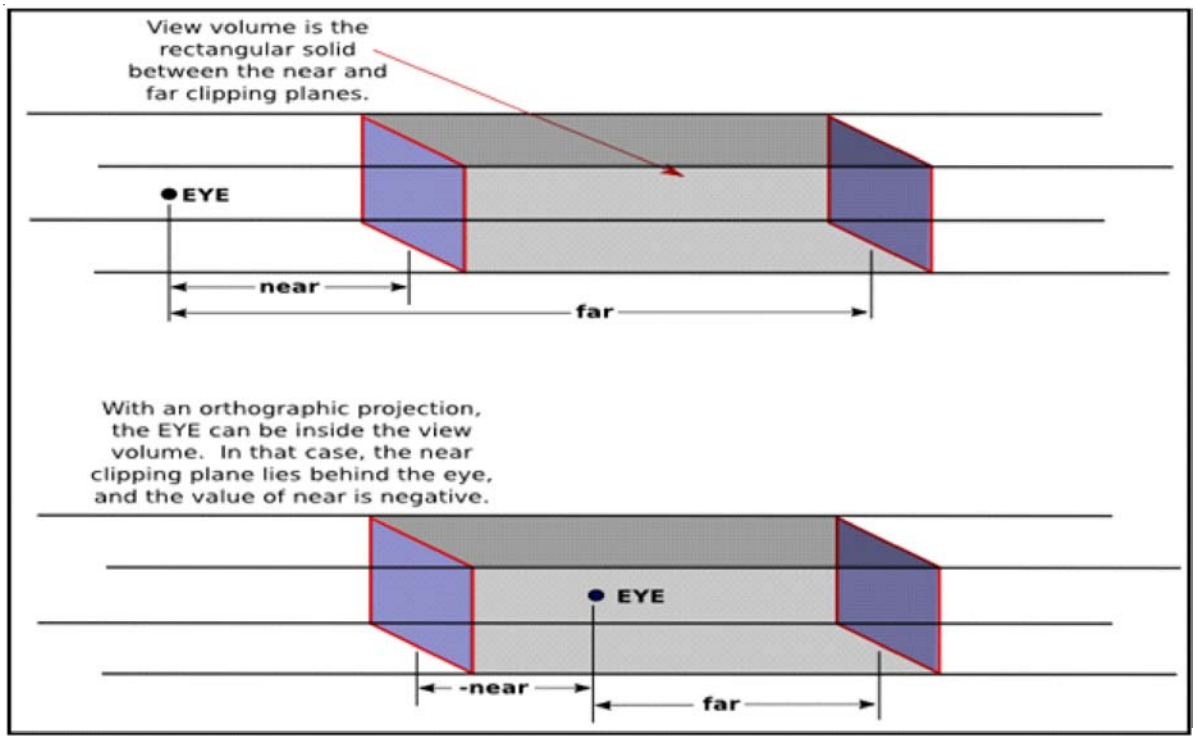


Figure 13.5: View volume parallel projection

Orthographic projections are easier to understand: In an orthographic projection, z-coordinate is discarded. So the three dimensional world is projected onto a 2D image. This type of projection is unrealistic. For example, the apparent size of an object does not depend on its distance from the viewer. Objects in back of the viewer as well as in front of the viewer can be visible in the image. Orthographic projections are still useful, however, especially in interactive modeling programs where it is useful to see true sizes and angles, undistorted by perspective. Negative value is allowed for near if the clipping plane behind the viewer.

General Parallel-Projection Transformation: In PHIGS, the direction of parallel projection is specified with a projection vector from the projection reference point to the center of the view window. We get the oblique projection transformation with shear operation that covert the view volume to rectangular parallelepiped.

Suppose the elements of the vector in the viewing coordinates are:

$$V_p = (p_x, p_y, p_z)$$

We need to determine the elements of a shear matrix that will align the projection vector V_p with the view plane normal vector N. this transformation can be expressed as:

$$V_p\grave{i} = M_{parallel}.V_p = \begin{bmatrix} 0 \\ 0 \\ p_z \\ 0 \end{bmatrix}$$

Where $M_{parallel}$ represents a z-axis shear of the form:

$$M_{parallel} = \begin{bmatrix} 1 & 0 & -p_x/p_z & 0 \\ 0 & 1 & -p_y/p_z & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

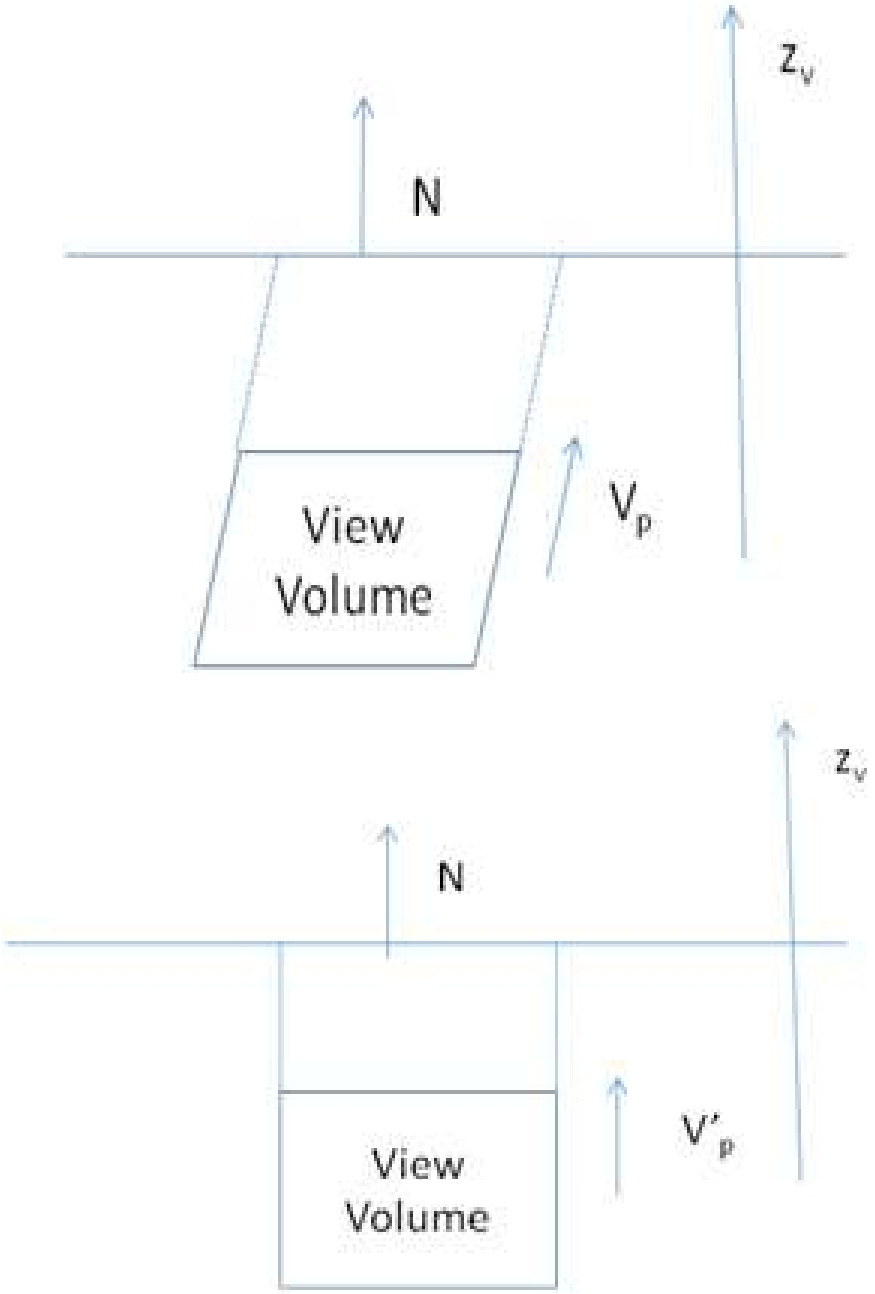


Figure 13.6: Parallel view volume

General Perspective-Projection Transformation: In perspective projection, reference point can be located at any position other than view plane and front and back

clipping planes. We can obtain the general perspective projection transformation with the following operations:

1) Shear the view volume so that centerline of the frustum is perpendicular to the view plane.
2) Scale the view volume with a scaling factor that depends on 1/z.

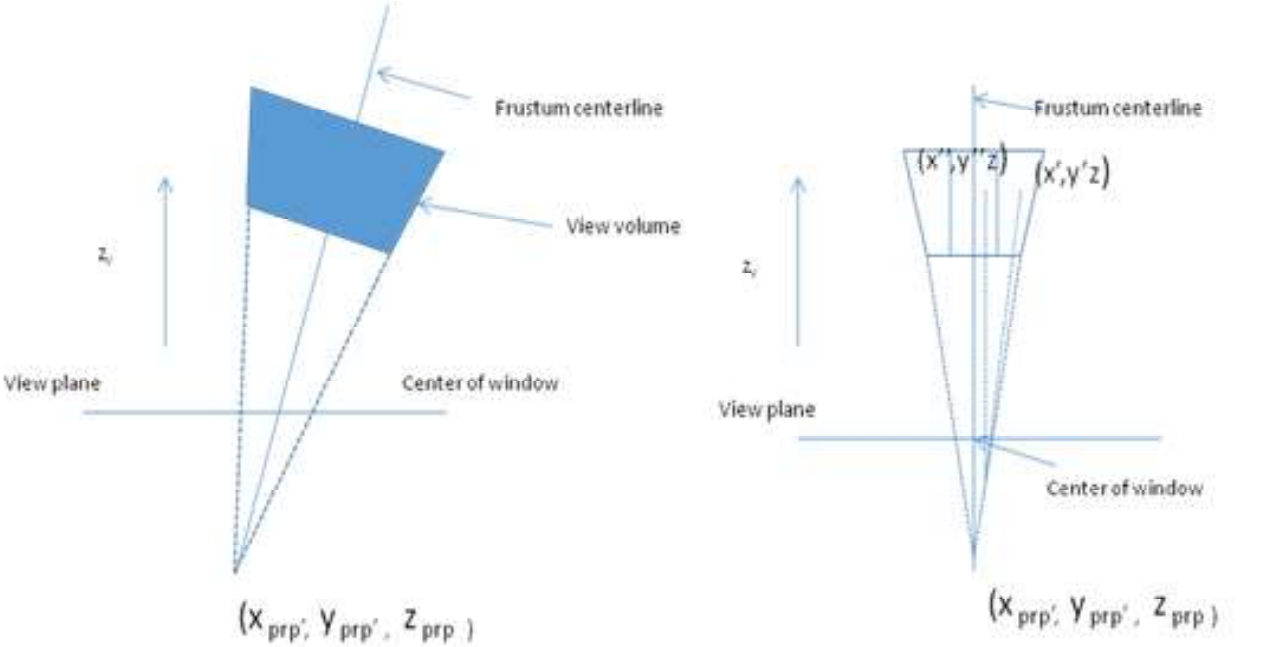


Fig: General shape for the perspective view volume with a reference point that is not on the z_v axis

Fig: shearing a general perspective view volume to center it on the projection window

Figure 13.7: Canonical view volume

The transformation with projection reference point at general position (x_{prp}, y_{prp}, z_{prp}) involves a combination z-axis shear and a translation:

$$M_{shear} = \begin{bmatrix} 1 & 0 & a & -az_{prp} \\ 0 & 1 & b & -bz_{prp} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where the shear parameters are:

$a=-(x_{prp} - (xw_{min}+xw_{max})/2)/z_{prp}$ and $b = -(y_{prp} - (yw_{min} + yw_{max})/2)/z_{prp}$

13.6 CLIPPING

An algorithm for three dimensional clipping identifies and save all surface segments within the view volume for display on the output device. All parts of the object which lies outside the view volume are discarded.

To clip a line segment against the view volume, we would need to test the relative position of the line using the view volume's boundary plane equation. The endpoint (x, y, z) of a line segment is outside the boundary plane if Ax + By + Cz + D > 0 where A, B, C and Dare the plane parameters for that boundary. Similarly the point is inside the boundary if Ax + By + Cz + D < 0 and the point is on the boundary if Ax +By + Cz + D = 0.

Normalized View Volume: At the first step, a scene is constructed by transforming the object descriptions from the modeling coordinates to world coordinates. Then, a view mapping convert the world coordinates to viewing coordinates. At the projection stage, the viewing coordinates transform to projection coordinates, which convert the view volume to the rectangular parallelepiped. Then the parallelepiped is mapped into the unit cube, a **normalized view volume** called the **normalized projection coordinates** system**.** Finally, at the workstation stage, normalized projection coordinates are converted to device coordinates for display.

The normalized view volume is a region defined by the planes:

x = 0, x = 1, y = 0, y = 1, z = 0, z = 1

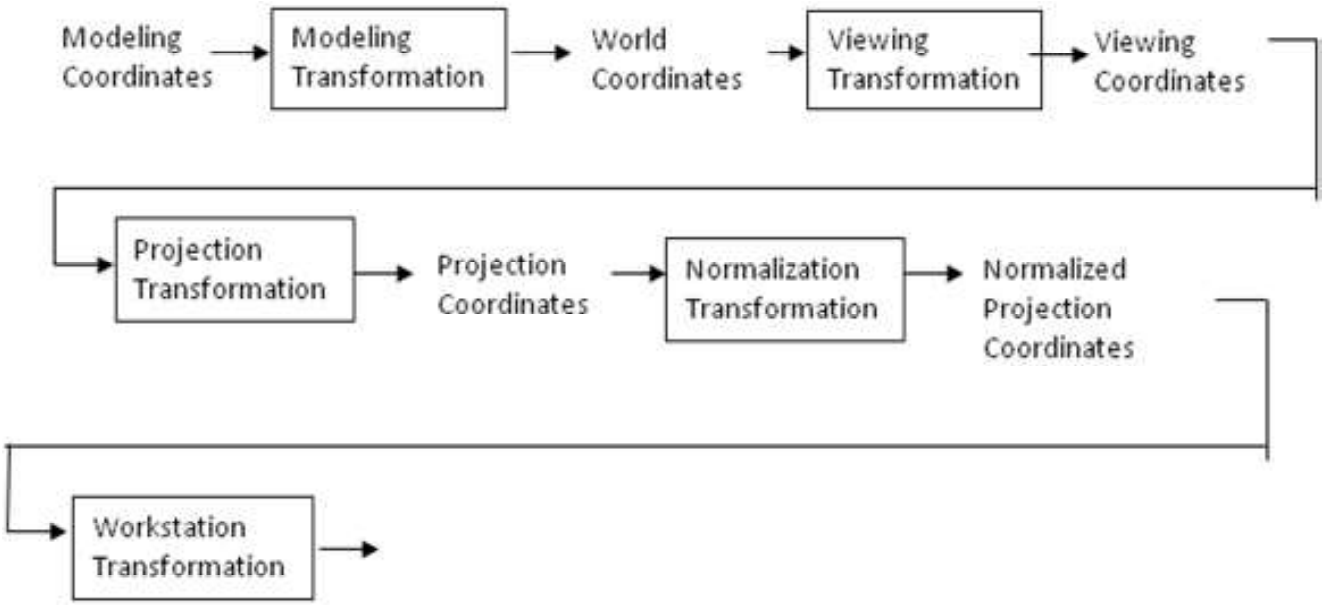


Figure 13.8:

Viewport Clipping: The window served as clipping boundary in two-dimensional space. In three dimensional spaces the concept can be extended to a clipping volume or view volume. A rectangular parallelepiped, i.e., a box, used for parallel or axonometric projection, and a truncated pyramidal volume used for perspective projections are the two common three dimensional clipping volume the following figure shows these volumes. These **volumes are six sided with sides: left, right, top, bottom, hither (near), and yon (far).**

The two-dimensional concept of region codes can be extended to three dimensions by considering 6-bit code for six sides instead of 4-bit code for four sides. Like two-dimension, we assign the bit positions in the region code from right to left as

Bit 1 = 1, if the end point is to the left of the volume.

Bit 2 = 1, if the end point is to the right of the volume.

Bit 3 = 1, if the end point is the below the volume.

Bit 4 = 1, if the end point is above the volume.

Bit 5 = 1, if the end point is in front of the volume.

Bit 6 = 1, if the end Point is behind the volume.

Otherwise, the bit is set to zero. As an example, a region code of 101000 identifies a point as above and behind the view volume, and the region code 000000 indicates a point within the view volume.

A line segment with both endpoint's region code 000000 can be immediately identified as completely within the view volume. If any endpoint of a line segment does not have a region code of 000000, we perform the logical AND operation on the two endpoint codes. If the result of this AND operation is nonzero then line is completely invisible because both endpoints are outside the view volume. On the other hand, if the result of AND operation is zero then line segment may be partially visible depending on point of intersection of the line and the clipping volume.

We have seen that determining the end point codes for a rectangular parallelepiped clipping volume is a straight forward extension of the two dimensional algorithm.

In the perspective clipping volume shown in following figure(b) requires some additional processing. A shown in the Figure (a), the line connecting the center of projection and the center of the perspective clipping volume coincides with the z-axis in a right handed co-ordinate system.

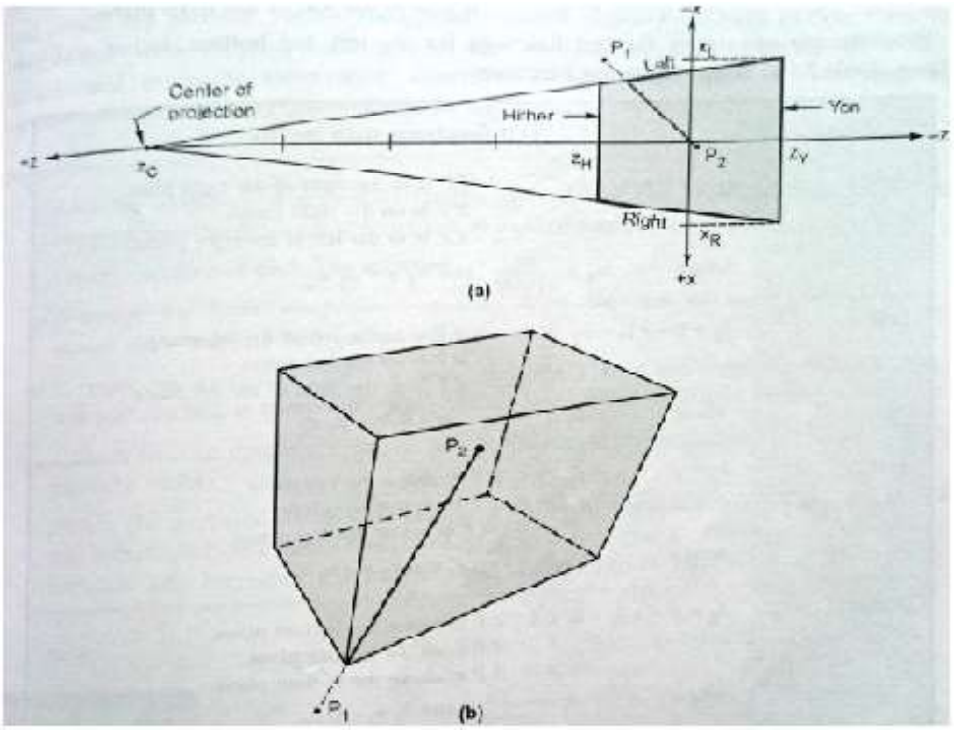


Figure 13.9: Top view of the perspective clipping volume

Figure 13.9 shows a top view of the perspective clipping volume. The equation of the line which represents the right hand plane in this view can be given as:

$$x = \frac{z - z_c}{z_y - z_c} x_R = z\alpha_1 + \alpha_2$$

Where, $\alpha_1 = \frac{x_R}{z_y - z_c}$ and $\alpha_2 = -\alpha_1 z_c$

This equation of right hand plane can be used to determine whether a point is to the right, on or to the left of the plane i.e., outside the volume, on the right hand plane, or inside the volume. Substituting the x and y coordinates of a point P into x – z α1 – α2 gives the following results.

fR = x – z α1 – α2 > θ if P is to the right of the right plane.

= θ if P is on the right plane.

< θ if P is to the left of the right plane.

CHECK YOUR PROGRESS

Q.3: The line segment from the view plane to view reference point is called

a) view distance b) view path

c) projection distance d) both a and b

Q.4: .....is most commonly used boundary representation of 3D graphics object.

a) Data polygon b) surface polygon

c) system polygon d) none of these

Q.5: What is view volume?

.....

.....

.....

Q.6: What is "frustum"?

.....

.....

Q.7: What is Region code and its Bit representation?

.....

.....

.....

.....

.....

.....

.....

13.7 LET US SUM UP

- Viewing procedure for 3-D scenes follows the general approach of 2-D viewing.
- Objects in the 3-D scenes are clipped against the view volume.
- Parallel projection can be specified with a projection vector.
- 3-D clipping rules are same with 2-D clipping.
- Some extension of 2-D clipping are made in 3-D clipping.

13.8 FURTHER READING

1) Dam, Andries Van; John, F. Hughes; Foley, James D.; Feiner, Steven K. (2002); *Computer Graphics Principles and Practice in C*. Pearson Education.

2) Hearn, D. & Baker, M. P. (1986). *Computer Graphics*. Prentice-Hall.

UNIT 14: BASICS OF ANIMATION

UNIT STRUCTURE

14.1 Learning Objectives
14.2 Introduction
14.3 Animation Basic
14.4 Computer Animation
14.5 Principles of Animation
14.6 Types of Animation
14.7 Animation Software and Hardware
14.8 Common Terms in Animation
14.9 Let Us Sum Up
14.10 Further Reading
14.11 Answers To Check Your Progress
14.12 Model Questions

14.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- explain computer animation
- learn about the principles and types of animation
- describe the types of animation
- learn the design of animation sequence.

14.2 INTRODUCTION

An old proverb says, "A picture is worth a thousand words." Even with all of our new technology, this is truer than ever before. We can see the evidence of this everywhere–including the Internet, which was initially text based only. Now you can open any web page, and you will see the images and animations popping out from every corner of the visible area. Human mind is more receptive to visual pictures and animation, than it is to dull text. In this unit, we will discuss computer animation including its principle, types, sequence of animation and its application area etc.

14.3 ANIMATION BASIC

The word animation derives from the latin words *anima* meaning life, and *animare* meaning to breathe life into. Animate means *to give new life or energy to.*

Thus, animating means moving something that cannot move on it's own. Animation is a series of still drawings that, when viewed in rapid succession, gives the impression of a moving picture. Animation is the process of creating images that appear to move. Do you know that– the motion pictures don't really move? Anyone who looks a piece of film knows that the medium is made up of many still images. From a strictly scientific standpoint, animation relies on the mechanics of the eye. When things move faster than a certain rate (between 18 and 24 times per second), a physiological phenomenon called persistence of vision comes into play and the motion tends to blur together. This happens because a single image flashed at the eye is retained by the brain longer than it is actually registered on the retina. Thus, if a second image is flashed within a certain minimum time (about 50 milliseconds), the brain still retains the last image and the two images may be combined. When a series of images is flashed in rapid succession, as is accompleshed with a movie projector, the brain blends the images together. When these images are only slightly changed one to the next, the end effect is that of continous motion. This very remarkable illusion is the perceptual foundation of film and television. You can imgine that if the eye didn't have persistence of vision, the world would appear a strange place indeed.

14.4 COMPUTER ANIMATION

Computer animation is the art of creating moving images with the use of computers. It is a subfield of computer graphics. **Or** we can say that **animation** is the rapid display of a sequence of images of 2-D(2 dimension) or 3-D (3 dimension) artwork or model positions in order to create an illusion of movement. Originally the animated objects were created by means of 2-dimensional graphics but now-a-days most of the animated objects are created by the use of 3-dimensional graphics. To create the illusion of movement, an image is displayed on the computer screen and repeatedly replaced by a new image that is similar to the previous image, but advanced slightly in the time domain (usually at a rate of 24 or 30 frames/second).

There are two basic division of computer animations - one is high tech computer animation used for films and another one is the low cost comp uter animation used in personal computer and video game area.

14.5 PRINCIPLES OF ANIMATION

The **principles of animation** were created in the early 1930s by animators at the Walt Disney Studios. These principles were used to guide production and creative discussions as well to train young animators better and faster. The principles of animation may appear very simple, even obvious. These principles became one of the foundations

of hand-drawn cartoon character animation.

They are–

- Squash and stretch
- Slow In and out
- Anticipation
- Exaggeration
- Follow through and overlapping action
- Timing
- Staging
- Straight ahead action and pose-to-pose action
- Arcs
- Secondary action
- Appeal
- Solid Drawing

- **Squash and Stretch:** This element creates the illusion of weight and volume in a character or object. The amount used depends on what the particular animation requires. This principle is based on the fact that even in reality, unless an object is completely rigid and mechanical, it will deform and change shape when moved. This method is used in all types of character animation from a bouncing ball to a simple character running. It is the most important element of animation that needs to be mastered and will be used often.

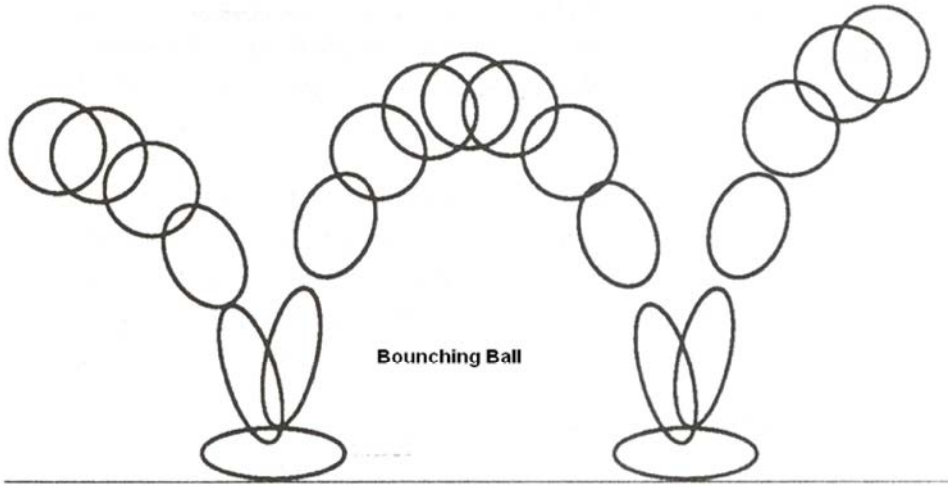


Figure 14.1: Squash and stretch in bounching ball

If you bounce a rubber ball, it squashes up as it hits the ground, and then it stretches out which is shown in the above figure. The most important rule to squash and stretch is that, no matter how squashed or stretched out a particular object gets, its volume remains constant. Squash and Stretch help to define how rigid an object is.

All objects change shape when a force is applied to them. In the case of a bouncing ball a rubber ball bounces higher and squashes more upon impact than a hard league ball. The ease with which an object squashes and stretches defines the rigidity of the material making up an object.

- **Slow in and slow out:** Most actions start slowly and end slowly. If you think of a car, it doesn't go from 0 to 60 instantaneously, nor it is likely to stop immediately from traveling at any speed. Slow in and slow out models this acceleration and deceleration. This is the idea of varying the timing and spacing between successive drawings to make actions speed up and slow down. Slow in and slow out creates a more natural and realistic motion when executed correctly. The slow in and slow out is illustrated Figure 14.2.

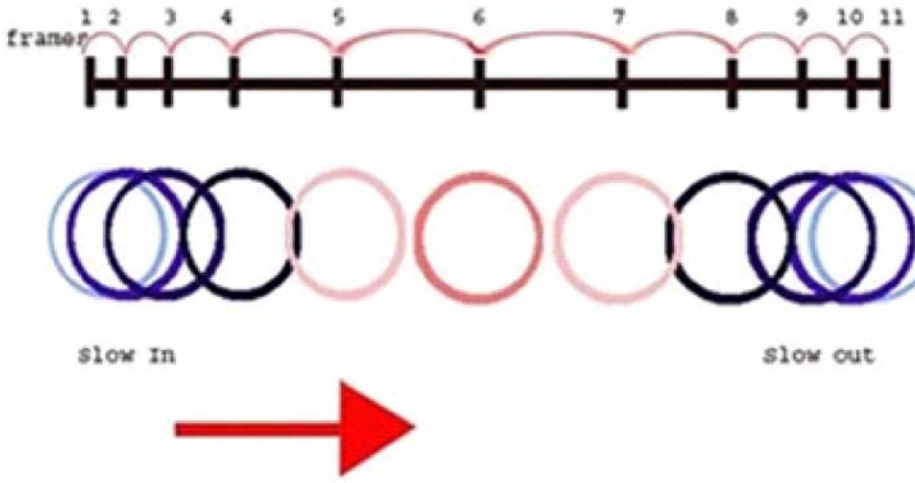


Figure 14.2: Slow in and slow out

- **Anticipation:** For good clear animation, the viewer should know what (anticipation) is about happen, what is happening (the actual action itself) and what happened (related to follow through). *Anticipation* basically draws the attention of the viewers, so that they focus on a particular area of the screen and prepare themselves for something to happen there. The classic example is of throwing a ball, where, if you are throwing a ball the anticipation will be bringing your arm back to prepare to throw the ball. Anything done in order to complete an action is anticipation. Without anticipation, you have surprise– or maybe even an unnatural looking action. In Figure 15.3, the anticipation effect is shown clearly. Notice how the arm bends before stretching out and how the hand opens (in drawing 3) in anticipation of closing around the glass in drawing 4. Also notice the balancing action of the other arm.



Figure 14.3: Anticipation effect

- **Exaggeration:** Exaggeration means the act of making something more noticeable than usual. Exaggeration can be used in animation with great results.
  - It is used to accentuate an action.
  - It should be used in a careful and balanced manner, not arbitrarily.

  The things that one can exaggerate are - *motions, poses and expressions.*
- **Follow through and overlapping action:** When a person comes to a stop after running, they do not simply stop. The laws of physics dictate that all the rest of the body will still have momentum and therefore will still be moving. Incorporating this idea into animation is known as follow through.

  When you think about animating any action of anything, you could break it down into two parts: starting and stopping. What you must bear in mind is that when a body in motion comes to a halt, not all parts stop at the same time. Look at the following figure– the girl gets ready for the jump by bending her knees and crouching see drawing 3 (which is called anticipation). She then launches into the jump, lifting her arms. At the height of the jump (drawing 8), her skirt has lifted, but has not yet reached its highest position (drawing 9, lifted by air resistance on her downward path). As she lands, her feet may stop first (drawing 10), but the rest of her body will still be moving downwards: her upper body and arms, her hair, her clothes. Each part will finally come to rest at different times. This effect, following the laws of motion, is what we call overlapping action. Your animation will appear dull and unnatural if you ignore this.



Figure 14.4: Follow through and overlapping action

- **Staging:** Staging refers the set – or setting – of the scene, and the way in which it is depicted. In animation, staging a scene is likely to involve “set design”, “lighting” and choice of camera angle and other camera techniques (close-ups, slow-motion etc.), as well as visual composition of the scene in much the same way that a photograph is composed.The actions of the character should be crystal clear for the audience to understand. This is achieved through the positioning of the character on screen and how they relate to all other visual elements including the background, other character and interactive objects.
- **Straight ahead action and pose-to-pose action:** There are two main methods of animating known as Straight ahead and Pose to pose. In straight ahead animation the animator draws or sets up objects one frame at a time in sequential order. For example, the animator draws the first frame of the animation, and then draws the second, and so on until the sequence is complete. In this way there is one drawing or image per frame that the animator has setup. This approach tends to yield a more creative and fresh look but can be difficult to time correctly, and tweak.

  Straight ahead action is used for wild, scrambling actions where spontaneity is important.

  Straight ahead is illustrated by the following figure of a walking man picking up a piece of chalk and writing on a black board.

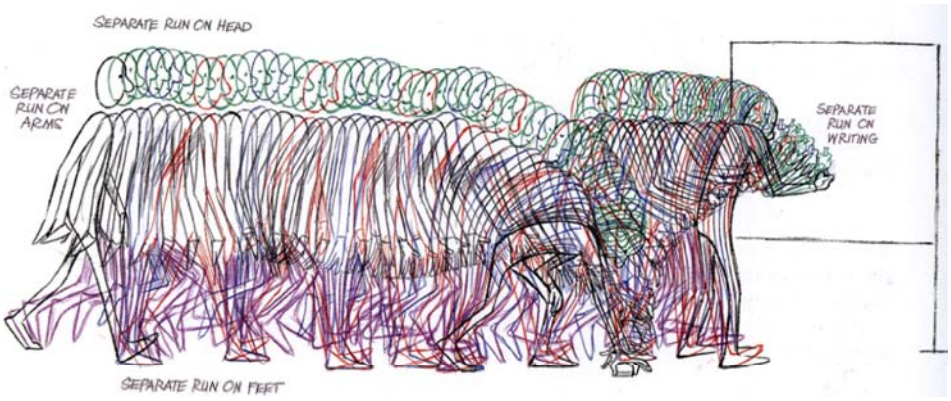


Figure 14.5: Straight ahead action

The pose to pose approach is to plan out key poses in an action, draw them as best as possible and then go back and draw in the extreme positions. It is then simply a matter of creating all the frames in between these key poses in order to create a smooth motion. The following figure illustrates the pose to pose approach–

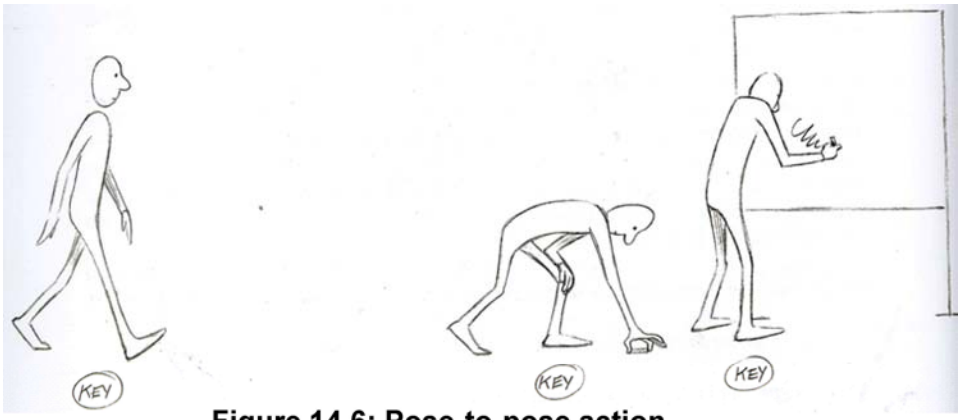


Figure 14.6: Pose-to-pose action

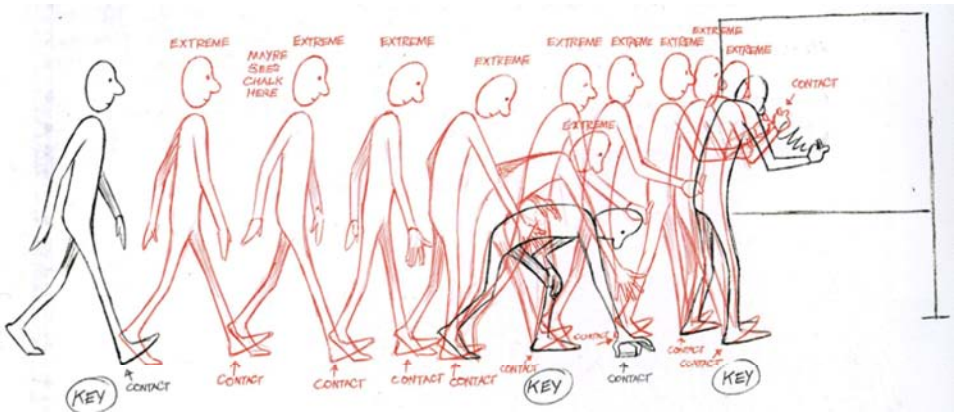


Figure 14.7: Pose-to-pose action with intermediate frame

- **Timing:** Timing refers to the positioning of the animation across time. This is a fairly natural concept used more in pose-to-pose animation. A few frames between poses means that the animation will go quickly; more and it will go slower. Timing is used to plan out what frame a pose needs to occur on. These are usually simple lines representing time with marks indicating key poses drawn on proportionally. The timing chart is then used as reference for the number of frames between poses as the animation is created.

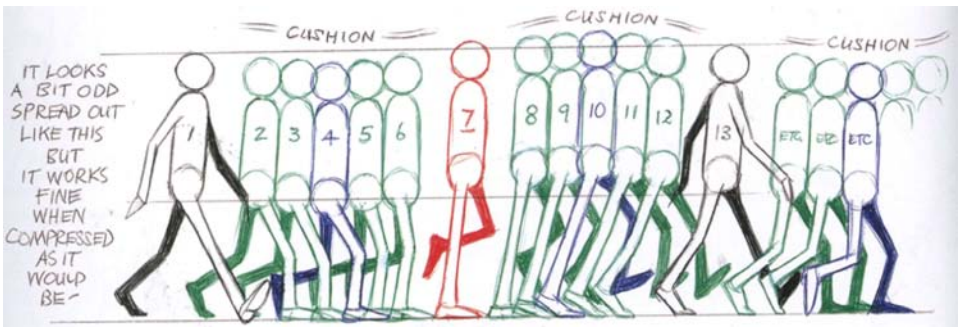


Figure 14.8: Timing in animation

- **Arcs:** Most things don't move in straight lines. If you throw a ball, it doesn't fly horizontally for a bit, then drop vertically down. Gravity acts on it all the time, pulling it in a downward arc. Arcs create a much more natural and flowing movement and are particularly important when animating the human figure or the actions of an animal. All body movements, from the swinging of an arm to the turning of a head, describe an arc through space. For body parts, if you swing your arm round, your hand doesn't describe a square shape– it follows a circle, an arc, centred on your shoulder. In the following figure, you will notice the presence of arc in the motion of a bouncing ball.

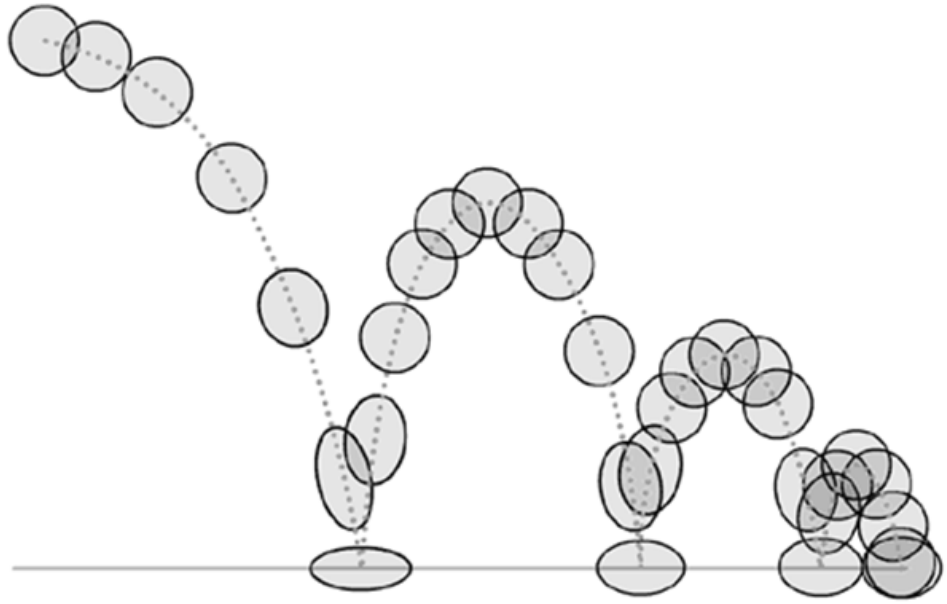


Figure 14.9: Arcs in bouncing ball

- **Secondary action:** This is an action that directly results from another action. Secondary actions are the incidental actions that make things look more realistic and may add to the anticipation of an action. For example, our ball thrower might be playing with the ball in their hands, or bouncing it from one hand to another, before actually performing the main throwing action. The movement of hair or clothing, or body twitches and facial gestures, are often likely to count as secondary action.
- **Appeal:** This is to create a character design or an action that the audience will enjoy watching. Appeal comes about through simplicity of design, clear drawing and the development of a personality that will captivate the audience. Every character, whether heroic, evil, comic or cute should have appeal.
- **Solid Drawing:** In hand drawn animation this property is probably the most obvious and simple to understand. In order for an animation to work it must be drawn well. Drawings should have weight, depth, balance and all the other basic aspects used to create the illusion of solid, three dimensional shapes on paper. Also, when put together

as an animated sequence, the drawings must work together and form the illusion of fourth-dimensional life. The fourth dimension is movement in time.

14.6 TYPES OF ANIMATION

The animations can be roughly divided into four main disciplines:

- 2D Drawn or Traditional animation
- 2D Computer generated animation
- Stop frame animation
- 3D Computer generated animation

- **2D Drawn or Traditional animation:** 2D drawn animation (also popularly known as **Cel animation**) was the first technique to develop. Developed in early 20th century, it mainly contains hand drawn drawings by the graphic artist, each slightly progressive than the previous one. When run rapidly on the screen in the sequence, it creates an illusion of motion. Traditionally, the work was done on thin transparent sheets of paper called Cel (hence, the name) and then projected onto the screen. This type of animation is more or less obsolete now as it requires lot of manual and tedious work by the artist and quality control is quite difficult. But still, even today, some animation lovers produce such type of animation purely for their pleasure. In Figure 14.10, you will notice that each character is slightly progressive than the previous one.



Figure 14.10: Animated characters in slighting progression

- **2D Computer generated animation:** This is a modern version of the traditional technique mentioned above. It still works on the basic principle of rapidly displaying a sequence of progressive images to create an illusion of movement. But in this case, the images are computer generated rather than hand drawn. Images are also created and edited using the 2D Vector graphics. Tweening, Morphing, onion skinning and Interpolated rotoscoping are the computer versions of traditional animation techniques used in 2D Animation. **Tweening** and **morphing** are two most popular techniques to auto generate images. In both these techniques, the first and last images are fed into

the computer (digitally like bitmap images) and the intermediate images are computed by using sophisticated algorithms based on user parameters. One of the most popular program using this technique is Flash animation program. The Figure 14.11 is created using 2D animation techniques. Power Point animation is an example of 2D animation.



Figure 14.11: 3D-animation

- **Stop frame animation:** This type of animation is created by physically manipulating real-world objects and photographing them one frame at a time to create the illusion of movement. There are many different types of stop-motion animations, but the most popular is the 'clay' animation. In this, real world clay models are developed and shot frame-by-frame. Stop frame animation is one of the most difficult type to achieve (smooth movements means more number of shots hence more work). This type is typically seen in children's TV serials and learning series. Figure 14.12 is created using the stop frame animation.



Figure 14.12: Stop frame animation

- **3D Computer generated animation:** This is the latest in animation techniques and technology. In 3D animation, a model is developed in highly advanced software, like: **3Dmax** or **Maya** and then animated. Big animation studios like *Pixar* are at the forefront of this technology. These are also used in movie special effects. This is the most advanced and highly complex animation technique ever conceived. There are hundreds of sub branches and techniques.

  For example, for achieving real life-like motion, actual human or animal movement can be captured (called motion capture) using special infrared cameras and then fed into computer to animate the model. Visual effects of movies are the example of 3D animation.

  There are some less frequently used animations. They are:

  - Drawing of Film animation
  - Painting on Glass animation
  - Pin screen animation
  - Sand animation

- **Drawing on Film Animation:** In drawn on film animation technique, the images are directly created on the **film** stock for the footage to be produced.
- **Paint on glass animation:** In paint on glass animation technique, slow drying oil paints are manipulated on various sheets of glass. The **animation films** are created

using paint on animation technique.

- **Pin Screen Animations:** Pin screen animation technique is solely used for animation films with textural effects. These textual effects cannot be produced with **Traditional animation**. Animators use movable pins on the whole screen. These movable pins can be moved in or out by pressing an object on the screen. These movable pins cast shadows when the screen is lit from the sides.
- **Sand Animations:** In Sand animation technique, sand is moved around on a piece of glass to create **frame-by-frame animation film**. The glass is either front lighted or backlighted.

14.7 ANIMATION SOFTWARE AND HARDWARE

Software and hardware are two of the major factors that determine the quality of computer animation that is produced. As both of these components work in conjunction with each other, it is important to make the right choice for a product that is of lasting quality and accuracy. Many graphical user interfaces (GUIs), including Mac OS, Microsoft Windows, or the X Window System, are primarily based on 2D graphical concepts.

The following are some of the 2D and 3D animation software packages used for creating animations:

- **2D Softwares:** Flash, Pencil, Ulead Gif Animator, Plastic Animation Paper etc.
- **3D Softwares:** 3D Studio Max, Maya, Lightwave, Anim8or etc.

In 2D computer graphics, the computer screen may be considered as a canvas on which an image is drawn or composed. Several techniques exist for depicting 2D graphics on a computer screen; these may be broadly categorized into raster graphics, in which a rectangular array of pixels is drawn to the screen, and vector graphics, in which images are composed of mathematical representations of lines, curves, and other geometric shapes.

Modern computer graphics card used in PC use raster techniques due to the relatively low cost of raster-based video hardware as compared with vector graphic hardware. Moreover, special purpose graphics hardware can produce real-time computer animation. Graphics workstations have built-in display processors which can typically handle tens of thousands of polygons in real-time. Professional animators of movies, television, and video sequences on computer games make photorealistic animation with high detail. They use many powerful workstation computers. Graphics workstation computers use two to four processors, and thus are a lot more powerful than a home computer, and are specialized for rendering. A large number of workstations (known as a render farm) are networked together to effectively act as a giant computer. The result is a

computer animated movie that can be completed in about one to five years.

The SGI platform is one of the most widely used platform for quality computer animation production. SGI stands for **Silicon Graphics Inc.** or **Silicon Graphics Computer Systems** or **SGCS** was a manufacturer of high-performance computing solutions, including computer hardware and software, founded in 1981 by Jim Clark and Abbey Silverstone. Its initial market was 3D graphics display terminals, but its products, strategies and market positions evolved significantly over time. SGI computers operates using the wide spread UNIX system. The computers are extremely fast and produce excellent results.

14.8 COMMON TERMS IN ANIMATION

- **Frame:** Animation is based on a principle of human vision. If you view a series of related still images in quick succession, you perceive them as continuous motion. Each individual image is referred to as a frame.
- **Key frames and in-betweens or tweens:** The main difficulty of an animator is to produce a large number of frames for creating a final animation. One minute of animation might require between 720 and 1800 separate images, depending on the quality of the animation. Creating images by hand is a big job. That's where the technique of keyframing comes in. We have come to know that most of the frames in an animation are routine, incremental changes from the previous frame directed toward some goal. Traditional animation studios realized they could increase the productivity of their master artists by having them draw only the important frames, called **key frames**. The assistants could then figure out what belonged on the frames in between the keyframes. The *in-between* frames were called **tweens**.

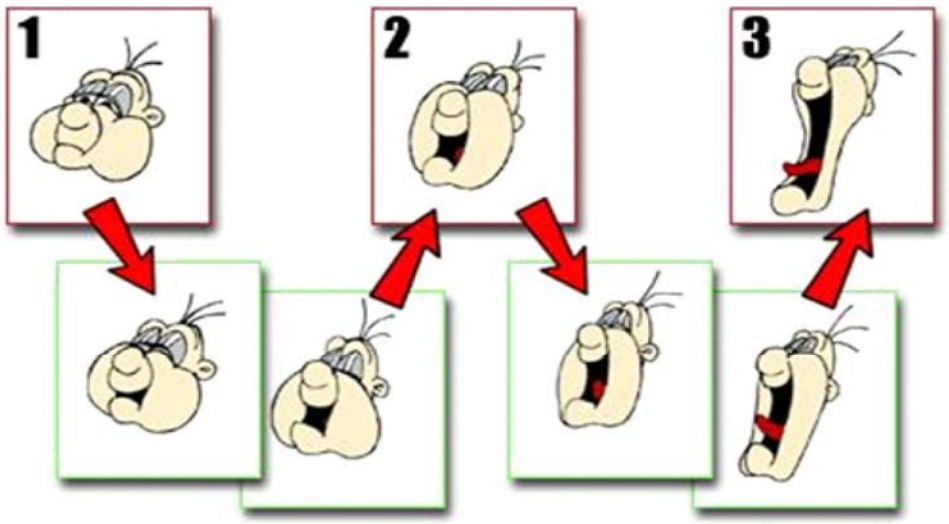


Figure 14.13: Key frames and tweens

Once all of the keyframes and tweens were drawn, the images had to be inked or rendered to produce the final images. Even today, production of a traditional animation usually requires hundreds of artists to generate the thousands of images needed. In the following figure the frames marked 1, 2, 3 are the key frames and other frames are tweens.

- **Cycles:** Animation moves in cycles, and a *cycle* is a series of frames that make up an action. For example, the cycle of walking can be made up of the following frames as shown in the figure 14.14.

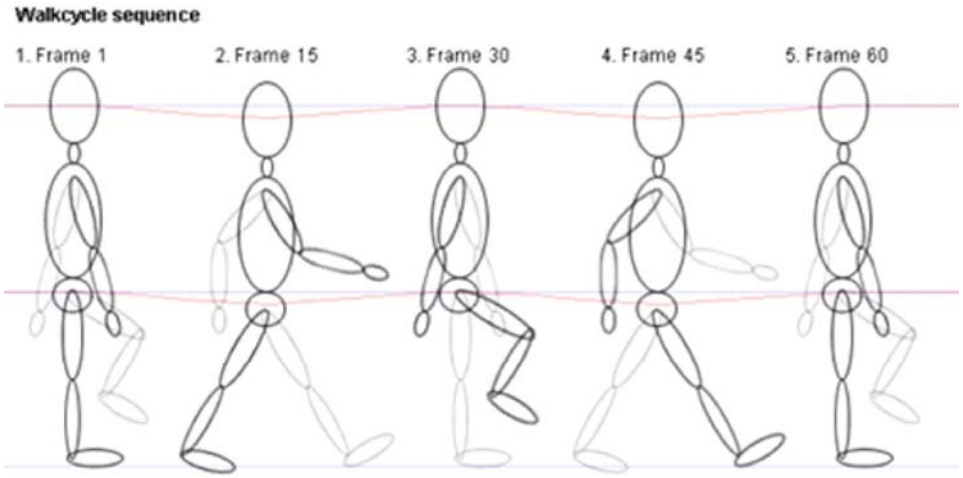


Figure 14.14: Cycles of animation

- **Frame Rates:** *Frame rates* are basically the speed at which frames or cels of animation are shown, expressed in number of frames per second of time. A normal frame rate for animated motion pictures is 24 frames per second (fps). With computer animation, frame rates look much better at about 40fps–50fps.

CHECK YOUR PROGRESS

Q.1: Fill in the blanks of the following:

a) animation relies on the mechanics of the eye whichis called .....

b) The principle ..... help to define how rigid an object is.

c) ..... basically draws the attention of the viewers.

d) ..... is an action that directly results from another action.

e) ..... was the first technique to develop animation.

f) ..... and ..... are two most popular techniques used in 2D animation.

g) Clay animation is a ..... type of animation.

14.9 LET US SUM UP

- Animation is the illusion of continuous movement of objects or camera viewpoint created by a series of still images. From a strictly scientific standpoint, animation relies on the mechanics of the eye which is called persistence of vision.
- To study various techniques and algorithms used in computer animation, it is useful to fi rst understand their relationship to the animation principles used in hand-drawn animation. The principles are *squash and stretch*, *timing, secondary action*, *slow in and slow out, arcs, follow through/overlapping action*, *exaggeration*, *appeal, anticipation, staging, solid drawing,* and *straight ahead and pose to pose.*
- The animations can be roughly divided into four main disciplines: 2D Drawn or Traditional animation, 2D Computer generated animation, Stop frame animation, 3D Computer generated animation.
- Some of the 2D and 3D animation software packages used for creating animations: Flash, Pencil, Ulead Gif Animator, Plastic Animation Paper etc are 2D animation software and 3D Studio Max, Maya, Lightwave, Anim8or etc are the 3D aniamtion softwares.

14.10 FURTHER READING

1) Kuperberg, Marica; Bowman, Martin. *A Guide to Computer Animation for TV, Games, Multimedia and Web*. Focal Press.
2) Parent, Rick; Ebert, David S. *Computer Animation Complete*. Elsevier.

14.11 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: a) persistence of vision; b) Squash and Stretch; c) Anticipation; d) secondary action; e) cel animation; f) Tweening, morphing; g) stop frame; h) Maya; i) 2D animation; j) tweens; k) cycle; l) storyboard; m) morphing

14.12 MODEL QUESTIONS

Q.1: What is animation ? Write the technique behind the animation.

Q.2: What is cel animation ? Discuss the principles of animation.

Q.3: Give two examples of animation software. Discuss the various types of animation types.

Q.4: Define the following terms:
Frames, keyframes, in-between, tweens, cycles, frame rate, morphing.

Q.5: What is computer animation ? Discuss the application of computer animation.

*** ***** ***

UNIT 15: ANIMATION DESIGNING

UNIT STRUCTURE

15.1 Learning Objectives
15.2 Introduction
15.3 Design of Animation Sequance
15.4 Morphing
15.5 Application of Computer Animation
15.6 Future of Animation
15.7 Let Us Sum Up
15.8 Further Reading
15.9 Answers To Check Your Progress
15.10 Model Questions

15.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- learn the design of animation sequence
- describe morphing technique
- explain the application of animation.

15.2 INTRODUCTION

In the previous unit, we have learnt the basics of animation including types of animation, priciples of animation etc. In this unit, we will discuss design of animation sequesnce, morphing and various applications of animation.

15.3 DESIGN OF ANIMATION SEQUENCE

The standard approach for designing any animation sequence is to have the following four steps:

- Storyboard layout
- Object definitions
- Key-frame specifications
- Generation of in-between frames

A **Storyboard** is a sequence of drawings that is used to help visualise the animation and to communicate ideas clearly. It defines the motion sequence as a set of basic events that are to take place. It is often the first opportunity for others to experience a real

impression of the animation before it enters the production stage. Depending on the type of animation to produced, the storyboard could consists of a set of rough sketches or it could be a list of basic ideas for the motion. In the following figure, you will notice the sequence of drawings for creating a final output.



Figure 15.1: Story board layout

An **object definition** is given for each object in the action. Objects can be defined in terms of basic shapes and the movements of that object is also specifiedalong with its shapes.

A **key frame** is a detailed drawing of the scene at a certain time in the animation sequence. Within each key frame, each object is positioned according to the time for that frame.

In-betweens are the intermediate frames between the key frames. The number of in-betweens needed is determined by the media to be used to display animation. Films requires 24 frames per second, and graphics terminals are refresmhed at a rate of 30 to 60 frames per second.

Interpolation and transformation are the basic techniques used in animation which are used for in-between the key frames.

15.4 MORPHING

It is an interesting technique used in computer animation (2D or 3D) where there is a transformation of object shapes from one form to another. Morphing is just a transformation of images. It is used in many animations and movies to warp a first image into a second image. The morph process transforms two images so that they have the same “shape” –the resulting images can then be cross dissolved to produce a sequence. The morphing process basically consisting with two steps: warp the two images and cross-dissolves their colours. The following figure shows the morphing of the source1

image to the source2 image using the above two steps:



Figure 15.2: Morphing example-1

Warping is the process of digitally manipulating an image such that any shapes portrayed in the image have been significantly distorted. Warping may be used for correcting image distortion as well as for creative purposes. While an image can be transformed in various ways, pure warping means that points are mapped to points without changing the colors. Warping algorithms are used for this purpose.

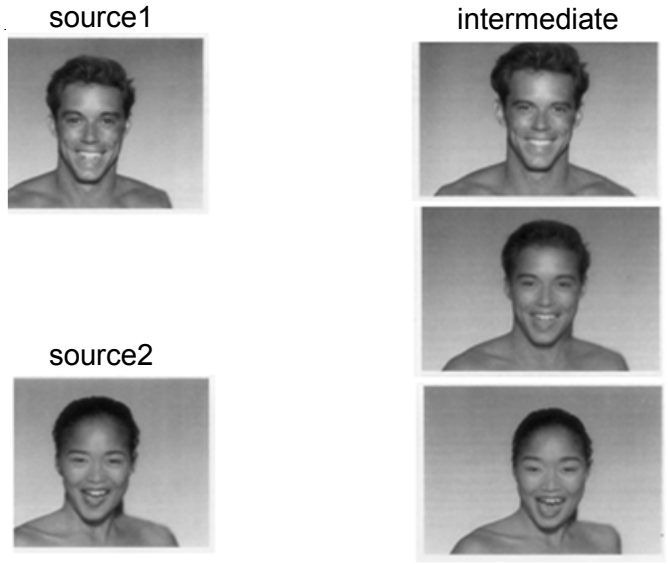


Figure 15.2: Morphing example-2

Cross dissolve simply means combining two images pixel-by-pixel. The figure 15.2 shows an another example of morphing.

15.5 APPLICATION OF COMPUTER ANIMATION

Animation is widely used in the entertainment industry (motion pictures, cartoons, video games), advertising, education, in industrial applications such as control system and heads-up displays and flight simulators for aircraft, and in scientific research.

Cartoons and Movies: The most common use of animation, and perhaps the origin of it, is cartoons. Cartoons appear all the time on television and the cinema and can be used for entertainment, **advertising**, presentations and many more applications that are only limited by the imagination of the designer. The most important factor about making cartoons on a computer is reusability and flexibility. The system that will actually do the animation needs to be such that all the actions that are going to be performed can be repeated easily. Speed here is not of real importance, as once the sequence is complete, it can be recorded on film or video, frame by frame and played back at an acceptable speed.

Movies such as "Terminator Two", "Jurassic Park", and "Abyss" use computer animation. Companies such as ***Pixar*** use computer animation to make three dimensional fully computer animated movies such as "Toy Story", and "A Bugs Life". Computer animation is also used in creating three dimensional video games and also used in advertising.Computer animation is all around us sometimes more than you might think.

Simulations: Many times it is much cheaper to train people to use certain machines on a virtual environment (i.e., on a computer simulation), than to actually train them on the machines themselves. Simulations of all types that use animation are supposed to respond to real-time stimuli, and hence the events that will take place are non-deterministic. The response to real-time stimuli requires a fast response and the non-determinism, requires a fast system to deal with it. This means that speed is the most important factor in simulation systems. For example, in flight simulation a pilot uses it to practice flying in situations that would be dangerous in real life.

Scientific Visualisation: The scientific application of computer graphics and especially of animation have come to be known under the group called scientific visualization, which does not merely contain the application of graphics to science and engineering but involves other disciplines such as signal processing, computational geometry and data base theory.

The uses of scientific visualisation can be classified into two main categories: *analysis* and *teaching*.

Analysis and Understanding: Very frequently, scientists have large sets of data (often in the form of lists of numbers) that need to be understood and often a theory needs to be formulated that explains their relationship. It would be very difficult to go through these lists manually or otherwise and make any sense out of them, unless some graphical technique is used for the initial approach. If the data set is massive, a short (or long) animation of the data can give the scientists a first idea of how to approach the situation.

Examples of the different uses of animation:

- In the area of space exploration, computer animaton serves most valuable

functions. For example, astronomers use computers to do animations if high speed jets penetrating different gases, to determine why a few galaxies flare dramatically.

- Animation can be used in software engineering, where an algorithm can be animated, in order to understand how it works or to debug it. Spotting errors using animation, becomes much easier.
- Animation allows designers and engineers to visualize complex process and to make better decisions regardig them.

Teaching and Communicating: One of the most difficult aspects of teaching is communicating ideas effectively. When this becomes too difficult using the classical teaching tools (speech, blackboard etc.) animation can be used to convey information. From its nature, an animation sequence contains much more information than a single image or page of text. This, and the fact that an animation can be very "pleasing to the eye", makes animation the perfect tool for learning.

Two examples of the use of animation for learning are:

- Programs that show the planetary system in action in three dimensions make it very easy for kids to understand rather than using tables of sizes, periods and diameters.
- Astrophysicists at the National Center for Supercomputing Applications, work with artists, in order to explain some phenomena which cannot be seen such as the visualisation of the gravitational field of a Schwarzchild black hole. The latter is not visible as it absorbs all light that falls onto it. The only way of experimenting with it is to animate it on a computer.
- **Medical animation** is very useful for marketing, training, and educating because it can get right to the point, clearly and quickly without showing anything you don't want the audience to see. With 3D animation, it is possible to show a patient in proper position and then show the incision site or simply make the skin fade to transparent in a matter of seconds.

15.6 FUTURE OF ANIMATION

Computer Animation is definitely the ultimate tool of the future. Its popularity is increasing in the area of entertainment, but scientific visualisation is the application that will utilise animation to all of its extent. In almost all areas of science, there is a vast amount of information available. Yet the most difficult part of it, is analysing the data and making sense out of it.

Animation by its nature, requires a lot of computer resources. As the speed of computers increases steadily and their prices decrease, animation is becoming available

to everyone. Animation is a very good educational tool, and can be used for almost anything. It is the only way to see and understand things that cannot be seen otherwise. Visualisation is the key to understanding and animation is going to be a huge part of it.

CHECK YOUR PROGRESS

Q.1: What is a Storyboard?

.....

.....

Q.2: Fill in the blanks of the following:

i) ..... is a series of frames that make up an action.

ii) ..... is a sequence of drawings used to visualise the animation and to communicate ideas clearly.

iii) An animation technique where there is transformation of object shapes from one form to another called .....

15.7 LET US SUM UP

- Designing any animation follows the following four sequence - Storyboard layout, Object definitions, Key-frame specifications, Generation of in-between frames.
- Morphing is a technique in 2D or 3D where there is a transformation of object shapes from one form to another.

15.8 FURTHER READING

1) Kuperberg, Marica; Bowman, Martin. *A Guide to Computer Animation for TV, Games, Multimedia and Web*. Focal Press.

2) Van Verth, J. M. & Bishop, L. M. (2008). *Essential Mathematics for Games and Interactive Applications: A programmer's Guide*. CRC Press.

15.9 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: A storyboard is a sequence of drawings that is used to help visualise the animation and to communicate ideas clearly.

Ans. to Q. No. 2: i) cycle; ii) storyboard; iii) morphing

15.10 MODEL QUESTIONS

Q.1: Briefly describe design of animation sequence.

Q.2: What is Morphing?

Q.3: Describe 2D Animation.

Q.4: Describe 3D Animation.

*** ***** ***



REFERENCES

1) Agarwal, Udit. (2011). *Computer Graphics*. Katson Books.
2) Dam, Andries Van; John, F. Hughes; Foley, James D.; Feiner, Steven K. (2002); *Computer Graphics Principles and Practice in C*. Pearson Education.
3) Foley, James D. *Computer graphics: Principles and Practice*.
4) Hearn, Donald; Baker, M. Pauline. (2nd Edition). *Computer Graphics*. Prentice-Hall.
5) Kuperberg, Marica; Bowman, Martin. *A Guide to Computer Animation for TV, Games, Multimedia and Web*. Focal Press.
6) Newman and Sproull. (1979). *Principles of Interactive Computer Graphics*. McGraw-Hill.
7) Parent, Rick; Ebert, David S. *Computer Animation Complete*. Elsevier.
8) Plastock and Kalley. (1986). *Computer Graphics, Schaum's Outlines*. McGraw-Hill.
9) Shirley, Peter; Marschner, Steve. (3rd Edition). *Fundamentals of Computer Graphics*.
10) Van Verth, J. M. & Bishop, L. M. (2008). *Essential Mathematics for Games and Interactive Applications: A programmer's Guide*. CRC Press.