Syllabus
S.Y.B.Sc. (IT)
SEMESTER - III, PAPER - II
COMPUTER GRAPHIC

Unit I

Introduction Computer Graphics and Primitive Algorithms: Introduction to Image and Objects, Image Representation, Basic Graphics Pipeline, Bitmap and Vector-Based Graphics, Applications of Computer Graphics, Display Devices, Cathode Ray Tubes, Raster-Scan Display, Random-Scan Display, Flat Panel Display, Input Technology, Coordinate System Overview,

Scan-Conversion of graphics primitives: Scan-Conversion of a Lines (Digital Differential Analyzer Algorithm, Bresenham's Line-Drawing Algorithm, Scan-Conversion of Circle and Ellipse (Bresenham's Method of Circle Drawing, Midpoint Circle Algorithm), Drawing Ellipses and Other Conics.

Unit II

Two Dimensional Transformation: Introduction to transformations, Transformation Matrix, Types of Transformations in Two-Dimensional Graphics: Identity Transformation, Scaling, Reflection, Shear Transformations, Rotation, Translation, Rotation about an Arbitrary Point, Combined Transformation, Homogeneous Coordinates, 2D Transformations using Homogeneous Coordinates

Unit III

Three-dimensional transformations, Objects in Homogeneous Coordinates; Three-Dimensional Transformations: Scaling, Translation, Rotation, Shear Transformations, Reflection, World Coordinates and Viewing Coordinates, Projection, Parallel Projection, Perspective Projection.

Unit IV

Viewing and Solid Area Scan-Conversion : Introduction to viewing and clipping, viewing Transformation in Two Dimensions, Introduction to Clipping, Two-Dimensional Clipping, Point Clipping, Line Clipping, Introduction to a Polygon Clipping, Viewing and Clipping in Three Dimensions, Three-Dimensional Viewing Transformations, Text Clipping

Introduction to Solid Area Scan-Conversion, Inside - Outside Test, Winding Number Method and Coherence Property, Polygon Filling, Seed Fill Algorithm, Scan-Lino Algorithm, Priority Algorithm, Scan Conversion of Character, Aliasing, Anti-Aliasing, Halftoning, Thresholding and Dithering

Unit V

Introduction to curves, Curve Continuity, Conic Curves, Piecewise Curve Design, Parametric Curve Design, Spline Curve Representation, Bezier Curves, B-Spline Curves, Fractals and its applications.

Surface Design : Bilinear Surfaces, Ruled Surfaces, Developable Surfaces, Coons Patch, Sweep Surfaces, Surface of Revolution, Quadric Surfaces, Constructive Solid Geometry, Bezier Surfaces, B-Spline Surfaces, Subdivision Surfaces.

Visible Surfaces : Introduction to visible and hidden surfaces, Coherence for visibility, Extents and Bounding Volumes, Back Face Culling, Painter's Algorithm, Z-Buffer Algorithm, Floating Horizon Algorithm, Roberts Algorithm.

Unit VI

Object Rendering : Introduction Object-Rendering, Light Modeling Techniques, illumination Model, Shading, Flat Shading, Polygon Mesh Shading, Gaurand Shading Model, Phong Shading, Transparency Effect, Shadows, Texture and Object Representation, Ray Tracing, Ray Casting, Radiosity, Color Models.

Introduction to animation, Key-Frame Animation, Construction of an Animation Sequence, Motion Control Methods, Procedural Animation, Key-Frame Animation vs. Procedural Animation, Introduction to Morphing, Three-Dimensional Morphing.

Books :

Computer Graphics, R. K. Maurya, John Wiley.
Mathematical elements of Computer Graphics, David F. Rogers, J. Alan Adams, Tata McGraw-Hill.
Procedural elements of Computer Graphics, David F. Rogers, Tata McGraw-Hill.

Reference:

Computer Graphics, Donald Hearn and M. Pauline Baker, Prentice Hall of India.
Computer Graphics, Steven Harrington, McGraw-Hill.
Computer Graphics Principles and Practice, J.D. Foley, A Van Dam, S. K. Feiner and R. L. Phillips, Addision Wesley.
Principles of Interactive Computer Graphics, William M. Newman, Robert F. Sproull, Tata McGraw-Hill.
Introduction to Computer Graphics, J.D. Foley, A. Van Dam, S. K. Feiner, J.F. Hughes and R.L. Phillips, Addision Wesley.

Practical (Suggested) :

Should contain at least 10 programs developed using C++. Some sample practical are listed below.

1. Write a program with menu option to input the line coordinates from the user to generate a line using Bresenham's method and DDA algorithm. Compare the lines for their values on the line.
2. Develop a program to generate a complete circle based on.

   a) Bresenham's circle algorithm

   b) Midpoint Circle Algorithm
3. Implement the Bresenham's / DDA algorithm for drawing line (programmer is expected to shift the origin to the center of the screen and divide the screen into required quadrants)
4. Write a program to implement a stretch band effect. (A user will click on the screen and drag the mouse / arrow keys over the screen coordinates. The line should be updated like rubber-band and on the right-click gets fixed).
5. Write program to perform the following 2D and 3D transformations on the given input figure

   a) Rotate through θ

   b) Reflection

   c) Scaling

   d) Translation
6. Write a program to demonstrate shear transformation in different directions on a unit square situated at the origin.
7. Develop a program to clip a line using Cohen-Sutherland line clipping algorithm between $(X_1, Y_1)(X_2, Y_2)$ against a window $(X_{\min}, Y_{\min})(X_{\max}, Y_{\max})$.
8. Write a program to implement polygon filling.
9. Write a program to generate a 2D/3D fractal figures (Sierpinski triangle, Cantor set, tree etc).
10. Write a program to draw Bezier and B-Spline Curves with interactive user inputs for control polygon defining the shape of the curve.
11. Write a program to demonstrate 2D animation such as clock simulation or rising sun.
12. Write a program to implement the bouncing ball inside a defined rectangular window.

1

COMPUTER GRAPHICS - FUNDAMENTALS

Unit Structure

1.0OBJECTIVES

The objective of this chapter is

- To understand the basics of computer graphics.
- To be aware of applications of computer graphics.
- To know the elements of computer graphics.

1.1 INTRODUCTION

Computer graphics involves display, manipulation and storage of pictures and experimental data for proper visualization using a computer. It provides methods for producing images and animations (sequence of images). It deals with the hardware as well as software support for generating images.

Basically, there are four major operations that we perform in computer graphics:

- Imaging: refers to the representation of 2D images.
- Modeling: refers to the representation of 3D images.
- Rendering: refers to the generation of 2D images from 3D models.

Animation: refers to the simulation of sequence of images over time.

1.2 INTRODUCTION TO IMAGE AND OBJECTS

An image is basically representation of a real world object on a computer. Itcan be an actual picture display, a stored page in a video memory, or a source code generated by a program. Mathematically, an image is a two - dimensional array of data with intensity or a color value at each element of the array.

Objects are real world entities defined in three – dimensional world coordinates. In computer graphics we deal with both 2D and 3D descriptions of an object. We also study the algorithms and procedures for generation and manipulation of objects and images in computer graphics.

Check your Progress:

1. Define image and object.
2. How an image is represented mathematically?

1.3 IMAGE REPRESENTATION

Image representation is the approximations of the real world displayed in a computer. A picture in computer graphics is represented as a collection of discrete picture elements termed as pixels. A pixel is the smallest element of picture or object that can be represented on the screen of a device like computer.

Check your progress:
1. Define pixel.

1.4 BASIC GRAPHIC PIPELINE

In computer graphics, the graphics pipeline refers to a series of interconnected stages through which data and commands related to a scene go through during rendering process.

It takes us from the mathematical description of an object to its representation on the device.

The figure shown below illustrates a 3D graphic pipeline.

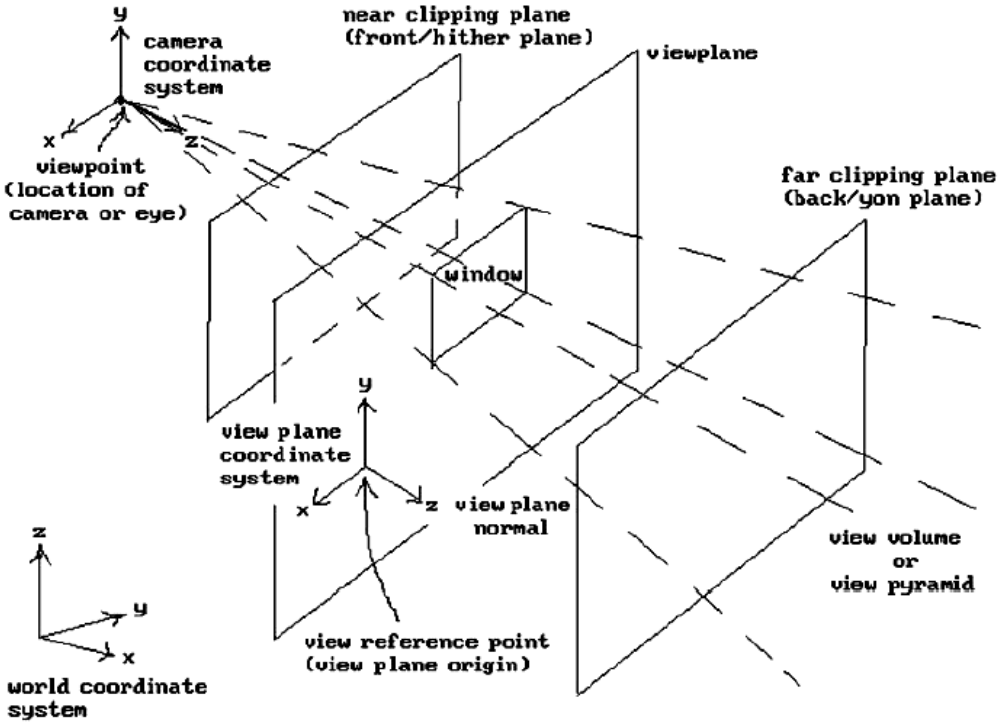


Figure 1.1: A 3D graphic pipeline

The real world objects are represented in *world coordinate system*. It is then projected onto a *view plane*. The projection is done from the *viewpoint* of the position of a camera or eye. There is an associated *camera coordinate system* whose z axis specifies the view direction when viewed from the viewpoint. The infinite volume swept by the rays emerging from the viewpoint and passing through the window is called as *view volume* or *view pyramid*. *Clipping planes* (near and far) are used to limit the output of the object.

The mapping of an object to a graphic device requires the transformation of view plane coordinates to physical device coordinates. There are two steps involved in this process.

(i) The window to a viewport transformation. The viewport is basically a sub – rectangle of a fixed rectangle known as logical screen.
(ii) The transformation of logical screen coordinates to physical device coordinates.

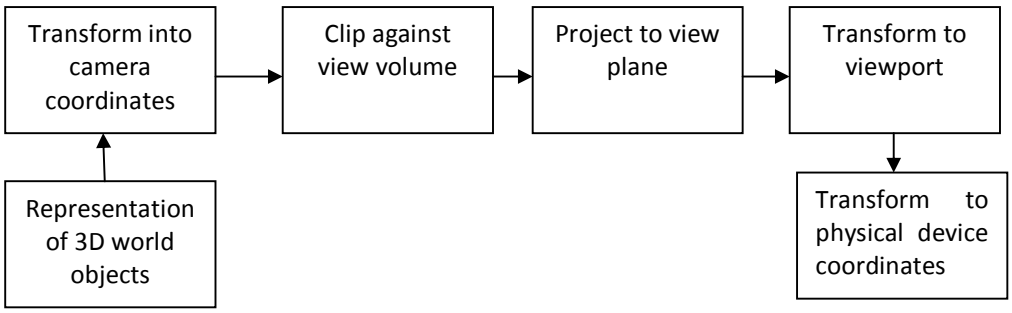


Figure : Sequence of transformation in viewing pipeline

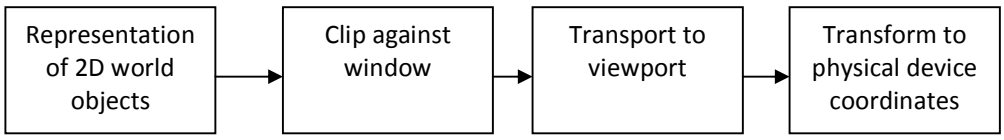


Figure 1.2: 2D coordinate system to physical device coordinates transformation.

The figures above depict the graphic pipeline and the 2D coordinate transformation to physical device coordinates.

Check your Progress:

1. Differentiate between world coordinates system and camera coordinate system.
2. Define view volume.

1.5 BITMAP AND VECTOR – BASED GRAPHICS

Computer graphics can be classified into two categories: Raster or Bitmap graphics and Vector graphics.

Bitmap graphics:

- It is pixel based graphics.
- The position and color information about the image are stored in pixels arranged in grid pattern.
- The Image size is determined on the basis of image resolution.
- These images cannot be scaled easily.
- Bitmap images are used to represent photorealistic images which involve complex color variations.

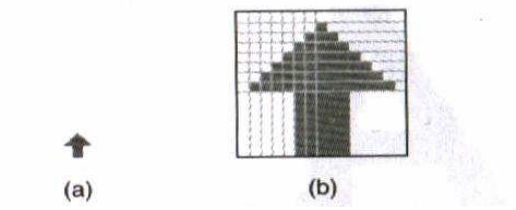


Figure 1.3
(a) An arrow image (b) magnified arrow image with pixel grid

The above figure shows a bitmap arrow image in its actual size and magnified image with pixel grid.

Vector graphics:

- The images in vector graphics are basically mathematically based images.
- Vector based images have smooth edges and therefore used to create curves and shapes.

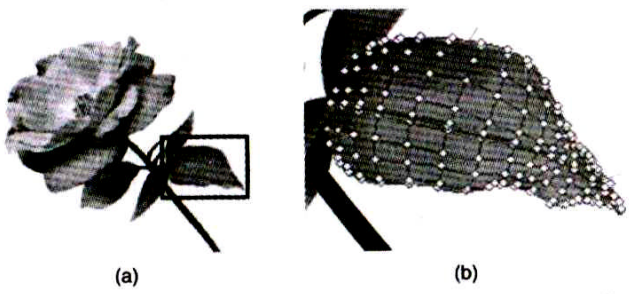


Figure 1.4
(a) A rose image (b) vector description of leaf of rose

- These images are appropriate for precise illustrations but not good for photorealistic images.

- These images are easily scalable due to their mathematical structure. Figure 1.4(a) and (b) shows a rose image and vector description of leaf of rose.

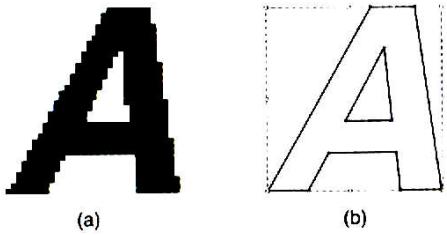


Figure 1.5 (a) A bitmap image (b) a vector image

The above figure shows a bitmap and vector image of the letter A.

Check your Progress:

1. Which graphic system is better for photorealistic images?
2. In which graphic system images are easily scalable?

1.6 APPLICATIONS OF COMPUTER GRAPHICS

Computer graphics finds its application in various areas; some of the important areas are discussed below:

- **Computer-Aided Design**: In engineering and architectural systems, the products are modeled using computer graphics commonly referred as CAD (Computer Aided Design).

  In many design applications like automobiles, aircraft, spacecraft, etc., objects are modeled in a wireframe outline that helps the designer to observe the overall shape and internal features of the objects.

  CAD applications are also used in computer animations. The motion of an object can be simulated using CAD.

- **Presentation graphics**: In applications like summarizing of data of financial, statistical, mathematical, scientific and economic research reports, presentation graphics are used. It increases the understanding using visual tools like bar charts, line graphs, pie charts and other displays.

- **Computer Art**: A variety of computer methods are available for artists for designing and specifying motions of an object. The object can be painted electronically on a graphic tablet using stylus with different brush strokes, brush widths and colors. The artists can also use combination of 3D modeling packages, texture mapping, drawing programs and CAD software to paint and visualize any object.

- **Entertainment**: In making motion pictures, music videos and television shows, computer graphics methods are widely used. Graphics objects can be combined with live actions or can be used with image processing techniques to transform one object to another (morphing).

- **Education and training**: Computer graphics can make us understand the functioning of a system in a better way. In physical systems, biological systems, population trends, etc., models makes it easier to understand.

  In some training systems, graphical models with simulations help a trainee to train in virtual reality environment. For example, practice session or training of ship captains, aircraft pilots, air traffic control personnel.

- **Visualization**: For analyzing scientific, engineering, medical and business data or behavior where we have to deal with large amount of information, it is very tedious and ineffective process to determine trends and relationships among them. But if it is converted into visual form, it becomes easier to understand. This process is termed as visualization.

- **Image processing**: Image processing provides us techniques to modify or interpret existing images. One can improve picture quality through image processing techniques and can also be used for machine perception of visual information in robotics.

  In medical applications, image processing techniques can be applied for image enhancements and is been widely used for CT (Computer X-ray Tomography) and PET (Position Emission Tomography) images.

- **Graphical User Interface**: GUI commonly used these days to make a software package more interactive. There are multiple window system, icons, menus, which allows a computer setup to be utilized more efficiently.

Check your progress:

1. Fill in the blanks

(a) GUI stands for**.....**

(b) **.....** provides us techniques to modify or interpret existing images.

2. Explain how computer graphics are useful in entertainment industry.

1.7 DISPLAY DEVICES

There are various types of displays like CRT, LCD and Plasma. We will discuss each of these three in brief.

- **CRT (Cathode Ray Tube)** is one of the mostly used display technology. In CRT, a beam of electrons emitted by an electron gun strikes on specified positions on phosphor coated screen after passing through focusing and deflecting systems.

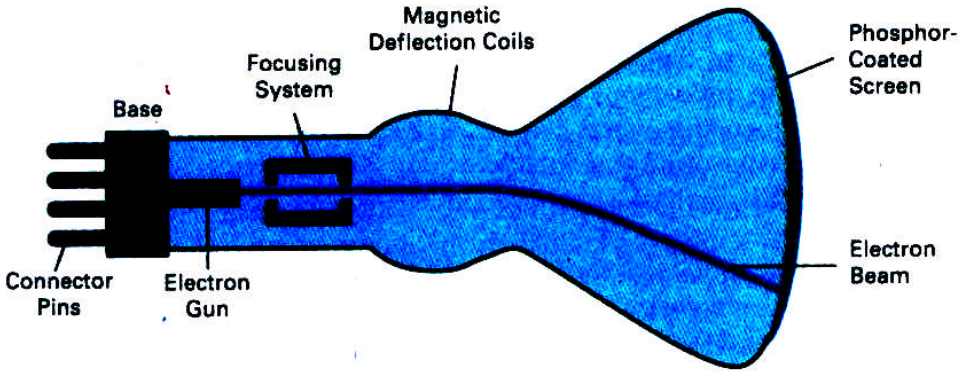


Figure 1.6 : Elements of CRT

- **LCD Display**: LCD stands for Liquid Crystal Display
  - **o** Organic molecules that remain in crystalline structure without external force, but re-aligns themselves like liquid under external force
  - **o** So LCDs realigns themselves to EM field and changes their own polarizations

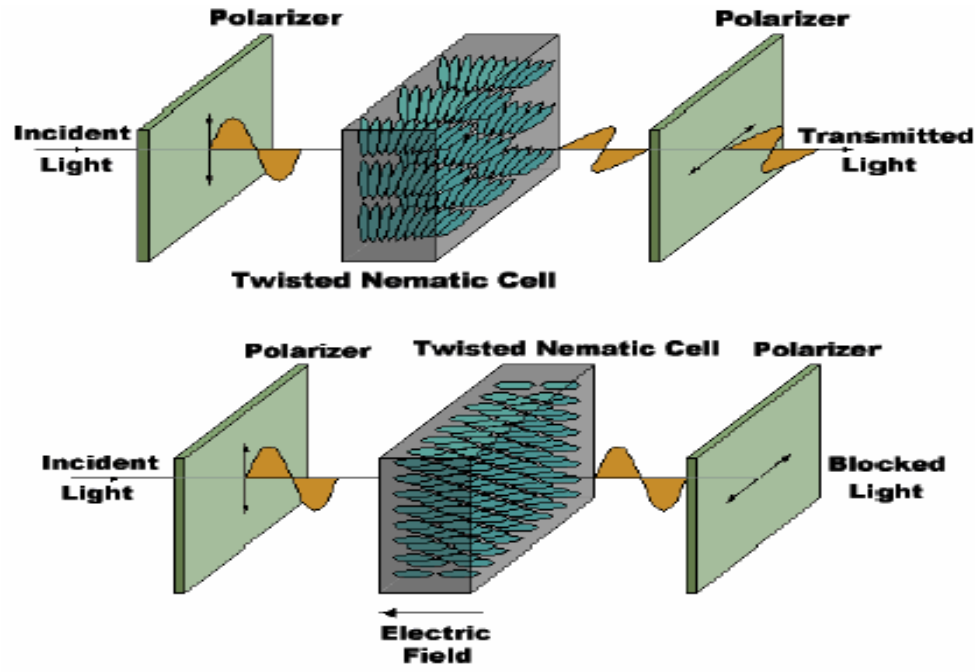


Figure 1.7 : LCD display

- **There are two types of LCD displays:**

- **Active Matrix LCD**:
  - Electric field is retained by a capacitor so that the crystal remains in a constant state.
  - Transistor switches are used to transfer charge into the capacitors during scanning.
  - The capacitors can hold the charge for significantly longer than the refresh period
  - Crisp display with no shadows.
  - More expensive to produce.

- **Passive matrix LCD**:
  - LCD slowly transit between states.
  - In scanned displays, with a large number of pixels, the percentage of the time that LCDs are excited is very small.
  - Crystals spend most of their time in intermediate states, being neither "On" or "Off".
  - These displays are not very sharp and are prone to ghosting.

- **Plasma display**:

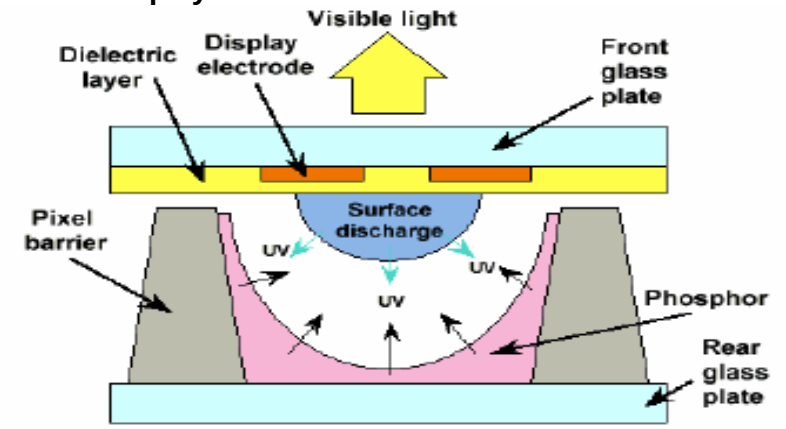


Figure1.8 (showing the basic structure of plasma display)

- These are basically fluorescent tubes.
- High- voltage discharge excites gas mixture (He, Xe), upon relaxation UV light is emitted, UV light excites phosphors.
- Some of its features are
  - Large view angle
  - Large format display
  - Less efficient than CRT, more power
  - Large pixels: 1mm (0.2 mm for CRT)
  - Phosphors depletion

- In CRT monitors there are two techniques of displaying images.

- **Raster scan displays**: A rectangular array of points or dots.In a raster scan system, the electron beam is swept across the screen, one row at a time from top to bottom.As the electron beam moves across each row, the beam intensity is turned on and off to create a pattern of illuminated spots. See the figure below.

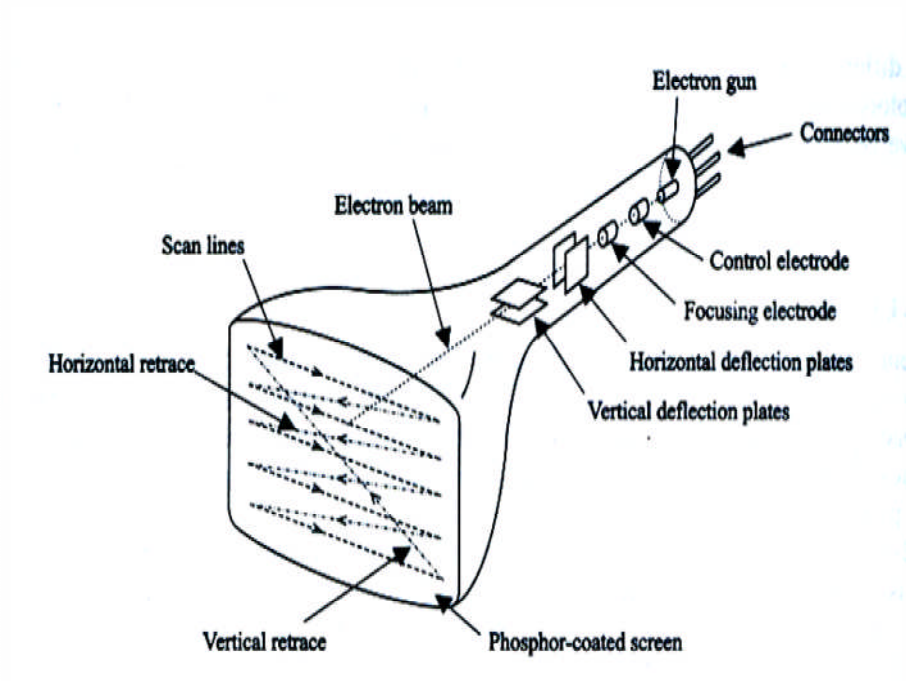


Figure 1.9 : Raster scan display

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

- **Vertical retrace**: At the end of each frame (displayed in 1/80th to 1/60th of a second) the electron beam returns to the top left corner of the screen to begin the next frame.

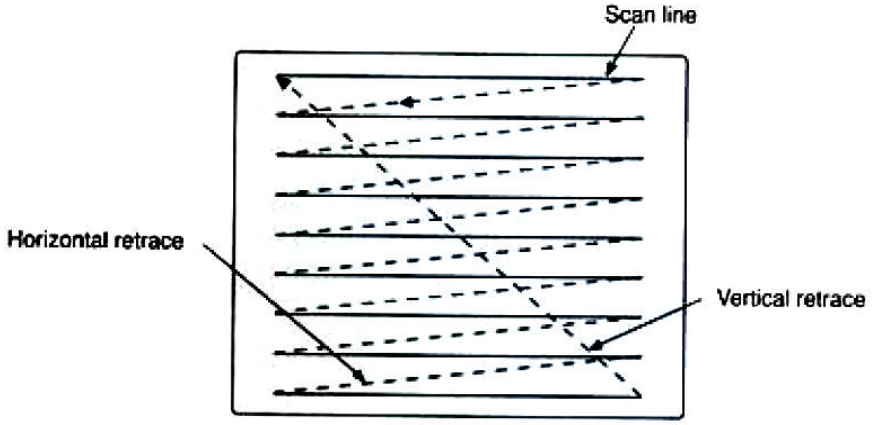


Figure 1.10 (showing horizontal and vertical retrace)

- **Random scan display**: Random scan display is the use of geometrical primitives such as points, lines, curves, and polygons, which are all based upon mathematical equation. In a random scan display, a CRT has the electron beam directed only to the parts of the screen where a picture is to be drawn. Random scan monitors draw a picture one line at a time. See the figure below.

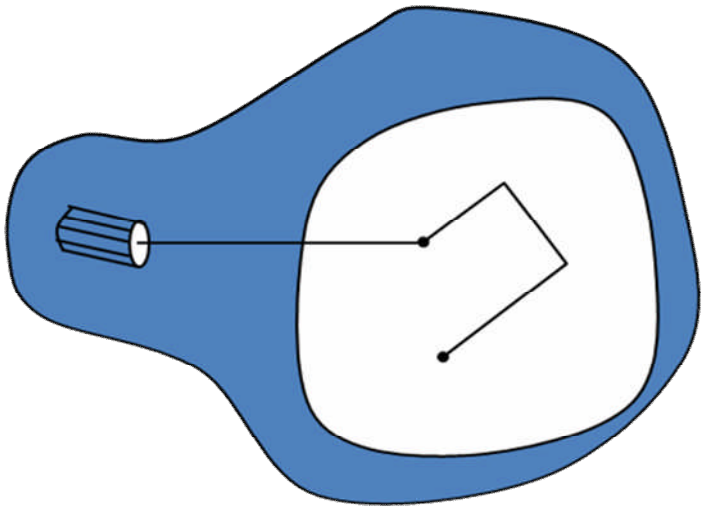


Figure 1.11: Random Scan display

- Refresh rate depends on the number of lines to be displayed.
- Picture definition is now stored as a line-drawing commands an area of memory referred to as refresh display file.
- To display a picture, the system cycle through the set of commands in the display file, drawing each component line in turn.
- Random scan displays are designed to draw all the component lines of a picture 30 to 60 times each second.
- Random scan displays have higher resolution than raster systems.

There are some parameters or properties related to graphic displays like CRT:

- **Persistence**: In case of CRT, persistence refers to the property of a phosphor defining its life time, i.e., how long they continue to emit light after the CRT beam is removed.

- **Resolution**: The maximum number of points that can be displayed without overlap on a CRT is referred to as the resolution. In other words, it is the number of points per unit length that can be plotted horizontally and vertically.

- **Aspect ratio**: It is the ratio of the number of vertical points to the number of horizontal points necessary to produce equal-length lines in both directions on the screen.

- **Frame buffer**: Frame buffer also known as refresh buffer is the memory area that holds the set of intensity values for all the screen points.

- **Pixel**: It refers a point on the screen. It is also known as pel and is shortened form of 'picture element'.

- **Bitmap or pixmap**: A frame buffer is said to be bitmap on a black and white system with one bit per pixel. For systems with multiple bits per pixel, the frame buffer is referred to as pixmap.

- **Graphical images** - used to add emphasis, direct attention, illustrate concepts, and provide background content. Two types of graphics:
  - Draw-type graphics or vector graphics – represent an image as a geometric shape
  - Bitmap graphics – represents the image as an array of dots, called pixels

- **Three basic elements for drawing in graphics are:**

  - **Point**: A point marks a position in space. In pure geometric terms, a point is a pair of x, y coordinates. It has no mass at all. Graphically, however, a point takes form as a dot, a visible mark. A point can be an insignificant fleck of matter or a concentrated locus of power. It can penetrate like a bullet, pierce like a nail, or pucker like a kiss. A mass of points becomes texture, shape, or plane. Tiny points of varying size create shades of gray.

  - **Line**: A line is an infinite series of points. Understood geometrically, a line has length, but no breadth. A line is the connection between two points, or it is the path of a moving point. A line can be a positive mark or a negative gap. Lines appear at the edges of objects and where two planes meet. Graphically, lines exist in many weights; the thickness and texture as well as the path of the mark determine its visual presence. Lines are drawn with a pen, pencil, brush, mouse, or digital code. They can be straight or curved, continuous or broken. When a line reaches a certain thickness, it becomes a plane. Lines multiply to describe volumes, planes, and textures.

  - **Plane**: A plane is a flat surface extending in height and width. A plane is the path of a moving line; it is a line with breadth. A line closes to become a shape, a bounded plane. Shapes are

planes with edges. In vector–based software, every shape consists of line and fill. A plane can be parallel to the picture surface, or it can skew and recede into space. Ceilings, walls, floors, and windows are physical planes. A plane can be solid or perforated, opaque or transparent, textured or smooth.

Check your progress:
1. Explain different display technologies.
2. Differentiate between Raster scan and Random scan display.

1.8 INPUT TECHNOLOGY

There are different techniques for information input in graphical system. The input can be in the form of text, graphic or sound. Some of the commonly used input technologies are discussed below.

1.8.1Touch Screens

A touch screen device allows a user to operate a touch sensitive device by simply touching the display screen. The input can be given by a finger or passive objects like stylus. There are three components of a touch screen device: a touch sensor, a controller and a software driver.

A touch sensor is a touch sensitive clear glass panel. A controller is a small PC card which establishes the connection between a touch sensor and the PC. The software driver is a software that allows the touch screen to work together with the PC. The touch screen technology can be implemented in various ways like resistive, surface acoustic, capacitive, infrared, strain gauge, optical imaging, dispersive signal technology, acoustic pulse recognition and frustrated total internal reflection.

1.8.2Light pen

A light pen is pen shaped pointing device which is connected to a visual display unit. It has light sensitive tip which detects the light from the screen when placed against it which enables a computer to locate the position of the pen on the screen. Users can

point to the image displayed on the screen and also can draw any object on the screen similar to touch screen with more accuracy.

1.8.3 Graphic tablets

Graphic tablets allow a user to draw hand draw images and graphics in the similar way as is drawn with a pencil and paper.

It consists of a flat surface upon which the user can draw or trace an image with the help of a provided stylus. The image is generally displayed on the computer monitor instead of appearing on the tablet itself.

Check your Progress:

1. Name different types of touch screen technologies.
2. Differentiate between light pen and graphic tablet.

1.9 COORDINATE SYSTEM OVERVIEW

To define positions of points in space one requires a coordinate system. It is way of determining the position of a point by defining a set of numbers called as coordinates. There are different coordinate systems for representing an object in 2D or 3D.

1.9.1 Cartesian coordinate system

It is also known as rectangular coordinate system and can be of two or three dimensions. A point in Cartesian coordinate system can be defined by specifying two numbers, called as x – coordinate and the y – coordinate of that point.

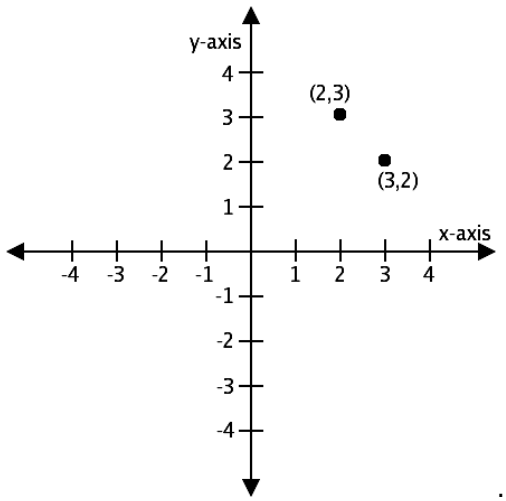


Figure 1.12: Cartesian coordinate system

In the above figure, there are two points (2, 3) and (3, 2) are specified in Cartesian coordinate system

1.9.2 Polar coordinate system

In polar coordinate system, the position of a point is defined by specifying the distance (radius) from a fixed point called as origin and the angle between the line joining the point and the origin and the polar axis (horizontal line passing through the origin).

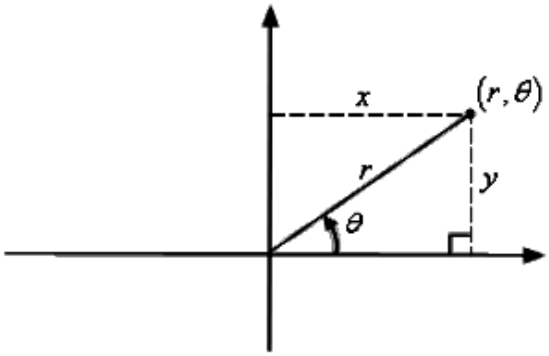


Figure 1.13: Polar coordinate system

The above figure shows a point (r, θ) in polar coordinates.

Check your Progress:

Fill in the blanks

1. The position of a point is defined by specifying the distance (radius) from a fixed point called as …….

2. A point in Cartesian coordinate system can be defined by specifying two numbers, called as …….. and the ……. of that point.

   Answers: 1. Origin
   2. x - coordinate, y – coordinate.

1.10 LET US SUM UP

We learnt about computer graphics, its application in different areas. We studied various display and input technologies. We also studied basic graphic pipeline, bitmap and vector based graphics. Then we learnt the elements of computer graphics in which came to know about the terms like persistence, resolution, aspect ratio, frame buffer, pixel and bitmap. Finally we studied about the coordinate system.

1.11 REFERENCES AND SUGGESTED READING

(1) Computer Graphics, Donald Hearn, M P. Baker, PHI.

(2) Procedural elements of Computer Graphics, David F. Rogers, Tata McGraw Hill.

(3) Computer Graphics, Rajesh K. Maurya, Wiley - India

1.12 EXERCISE

1. What are the major operations that we perform on Computer Graphics?
2. Define some of the applications of Computer Graphics.

3. Define graphic pipeline and the process involved in it.
4. Differentiate between bitmap and vector based graphics.

5. Define the following terms:
    a. Persistence
    b. Aspect ratio
    c. Frame buffer
    d. Resolution
    e. Pixel

6. Define horizontal and vertical retrace.

2

SCAN CONVERSION OF GRAPHICS PRIMITIVES

Unit Structure

2.0 OBJECTIVES

The objective of this chapter is

- To understand the basic idea of scan conversion techniques.
- To understand the algorithms for scan conversion of line, circle and other conics.

2.1 INTRODUCTION

Scan conversion or rasterization is the process of converting the primitives from its geometric definition into a set of pixels that make the primitive in image space. This technique is used to draw shapes like line, circle, ellipse, etc. on the screen. Some of them are discussed below

2.2 SCAN – CONVERSION OF LINES

- A straight line can be represented by a slope intercept equation as
  $y = m.x + b$
  where m represents the slope of the line and b as the y intercept.

- If two endpoints of the line are specified at positions (x_1,y_1) and (x_2,y_2), the values of the slope m and intercept b can be determined as
  $$m = \frac{y_2 - y_1}{x_2 - x_1}$$
  $$b = y_1 - m.x_1$$

- If Δx and Δy are the intervals corresponding to x and y respectively for a line, then for given interval Δx, we can calculate Δy.
  $$\Delta y = m\Delta x$$
  Similarly for given interval Δy, Δx can be calculated as
  $$\Delta x = \frac{\Delta y}{m}$$

- For lines with magnitude $|m| < 1$, Δx can be set proportional to a small horizontal deflection and the corresponding horizontal deflection is et proportional to Δy and can be calculated as $\Delta y = m\Delta x$

- For lines with $|m|>1$, Δy can be set proportional to small vertical deflection and corresponding Δx which is set proportional to horizontal deflection is calculated using $\Delta x = \frac{\Delta y}{m}$

The following shows line drawn between points (x_1, y_1) and (x_2, y_2).

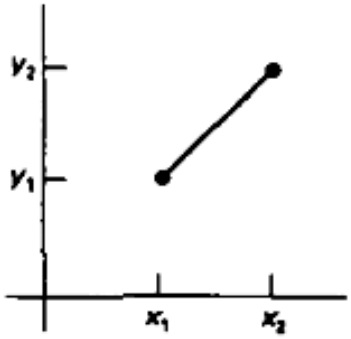


Figure 2.1 : A line representation in Cartesian coordinate system

2.2.1 Digital Differential Analyzer (DDA) Algorithm

- Sampling of the line at unit interval is carried out in one coordinate and corresponding integer value for the other coordinate is calculated.

- If the slope is less than or equal to 1(|m| ≤ 1), the coordinate x is sampled at unit intervals (∆x = 1) and each successive values of y is computed as

  $$y_{k+1} = y_k + m$$

  where k varies from 1 to the end point value taking integer values only. The value of y calculated is rounded off to the nearest integer value.

- For slope greater than 1 (|m| > 1), the roles of y and x are reversed, i.e., y is sampled at unit intervals (∆y = 1) and corresponding x values are calculated as

  $$x_{k+1} = x_k + \frac{1}{m}$$

- For negative value slopes, we follow the same procedure as above, only the sampling unit ∆x and ∆y becomes '-1' and

  $$y_{k+1} = y_k - m \quad for\ |m| \leq 1$$

  $$x_{k+1} = x_k - \frac{1}{m} \quad for\ |m| > 1$$

  Pseudocode for DDA algorithm is as follows

```
LineDDA(Xa, Ya, Xb, Yb)    // to draw a line from (Xa, Ya) to
(Xb, Yb)
    {
Set dx = Xb - Xa, dy = Yb - Ya;
Set steps = dx;
SetX = Xa, Y = Ya;
int c = 0;
Call PutPixel(Xa, ya);
For (i=0; i <steps; i++)
        {
X = X + 1;
            c = c + dy;  // update the fractional part
If (c > dx)
                { // (that is, the fractional part is greater than 1
        now
Y = y +1;   // carry the overflowed integer over
                c = c - dx  // update the fractional part
Call PutPixel(X, Y);
        }
         }
}
```

2.2.2 Bresenham's Line Drawing Algorithm

This line drawing algorithm proposed by Bresenham, is an accurate and efficient raster-line generating algorithm using only incremental integer calculations.

For lines |m| ≤ 1, the Bresenham's line drawing algorithm

I. Read the end points of the line and store left point in (x_0, y_0)

II. Plot (x_0, y_0), the first point.

III. Calculate constants Δx, Δy, $2\Delta y$ and $2\Delta y - 2\Delta x$, and obtain a decision parameter p_0

$$p_0 = 2\Delta y - \Delta x$$

IV. Perform the following test for each x_k, starting at k = 0 if $p_k < 0$, then next plotting point is (x_{k+1}, y_k) and

$$p_{k+1} = p_k + 2\Delta y$$

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

$$p_{k+1} = p_k + 2\Delta y - 2\Delta x$$

V. Repeat step 4 Δx times.

For a line with positive slope more than 1, the roles of the x and y directions are interchanged.

Check your progress:

1. Fill in the blanks

(a) .....of the line at unit interval is carried out in one coordinate and corresponding integer value for the other coordinate is calculated.

(b) Bresenham's line drawing algorithm is an accurate and efficient raster-line generating algorithm using only .....calculations.

2. Compare DDA and Bresenham's line drawing algorithm.

Answers: 1(a) sampling (b) incremental integer

2.3 SCAN – CONVERSION OF CIRCLE AND ELLIPSE

A circle with centre (x_c, y_c) and radius r can be represented in equation form in three ways

- Analytical representation: $r^2 = (x – x_c)^2 + (y – y_c)^2$
- Implicit representation : $(x – x_c)^2 + (y – y_c)^2 – r^2 = 0$
- Parametric representation: $x = x_c + r \cos\theta$

$y = y_c + y\sin\theta$

A circle is symmetrical in nature. Eight – way symmetry can be used by reflecting each point about each 45^o axis. The points obtained in this case are given below with illustration by figure.

$P_1 = (x, y)$ $P_5 = (-x, -y)$
$P_2 = (y, x)$ $P_6 = (-y, -x)$
$P_3 = (-y, x)$ $P_7 = (y, -x)$
$P_4 = (-x, y)$ $P_8 = (x, -y)$

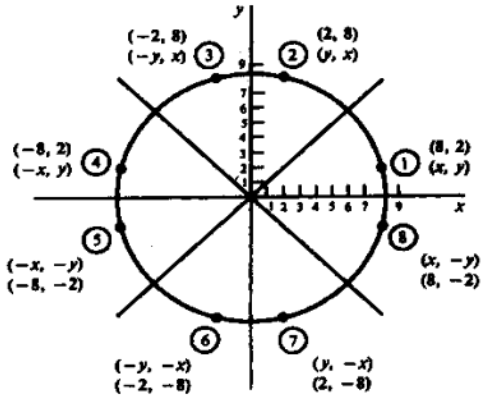


Figure 2.2 : Eight way symmetry of a circle

3.1 Bresenham's circle drawing algorithm

Let us define a procedure Bresenham_Circle (Xc,Yc, R) procedure for Bresenham's circle drawing algorithm for circle of radius R and centre (Xc, Yc)

```
Bresenham_Circle (Xc,Yc, R)
{
        Set X = 0;
        Set Y= R;
        Set D = 3 – 2R;
        While (X < Y)
           {
                Call Draw_Circle (Xc, Yc, X, Y);
                X=X+1;
                If (D < 0)
                { D = D + 4X + 6; }
                Else
                   {
                        Y = Y – 1;
                        D = D + 4(X – Y) + 10;
```

```
                }
            Call Draw_Circle (Xc, Yc, X, Y);
             }
    }
Draw_Circle (Xc, Yc, X, Y)
   {
       Call PutPixel (Xc + X, Yc, +Y);
       Call PutPixel (Xc – X, Yc, +Y);
       Call PutPixel (Xc + X, Yc, – Y);
       Call PutPixel (Xc – X, Yc, – Y);
       Call PutPixel (Xc + Y, Yc, + X);
       Call PutPixel (Xc – Y ,Yc, – X);
       Call PutPixel (Xc + Y, Yc, – X);
       Call PutPixel (Xc – Y, Yc, – X);
   }
```

2.3.2Midpoint circle drawing algorithm

This algorithm uses the implicit function of the circle in the following way

$f(x, y) = (x – x_c)^2 + (y – y_c)^2 – r^2$

here $f(x, y) < 0$ means (x, y) is inside the circle

$f(x, y) = 0$ means (x, y) is on the circle

$f(x, y) > 0$ means (x, y) is outside the circle

The algorithm now follows as

```
        Midpoint_Circle( Xc, Yc, R)
            {
                Set X = 0;
                Set Y = R;
                Set P = 1 – R;
                While (X < Y)
                    {
                        Call Draw_Circle( Xc, Yc, X, Y);
                        X = X + 1;
                        If (P < 0)
                        {P = P + 2X + 6; }
                        Else
                            {
                                Y = Y – 1;
                                P = P + 2 (X – Y) + 1;
                            }
                        Call Draw_Circle( Xc, Yc, X, Y);
                    }
            }
```

2.3.3 Midpoint Ellipse Algorithm

This is a modified form of midpoint circle algorithm for drawing ellipse. The general equation of an ellipse in implicit form is

$f(x, y) = b^2x^2 + a^2y^2 – a^2b^2 = 0$

Now the algorithm for ellipse follows as

```
MidPoint_Ellipse( Xc, Yc, Rx, Ry)
   {
      /* Xc and Yc here denotes the x coordinate and y
      coordinate of the center of the ellipse and Rx and Ry
      are the x-radius and y-radius of the ellipse
      respectively */
      Set Sx = Rx * Rx;
      Set Sy = Ry * Ry;
      Set X = 0;
      Set Y = Ry;
      Set Px = 0;
      Set Py = 2 * Sx * Y;
      Call Draw_Ellipse (Xc, Yc, X, Y);
      Set P = Sy – (Sx * Ry) + (0.25 * Sx);/* First Region*/
      While ( Px<Py)
         {
            X = X + 1;
            Px = Px + 2 * Sy;
            If (P < 0)
               {P = P + Sy + Px;}
            Else
               {
                  Y = Y – 1;
                  Py = Py – 2 * Sx;
                  P = P + Sy + Px – Py;
               }
            Call Draw_Ellipse (Xc, Yc, X, Y);
         }
      P = Sy * (X + 0.5)^2 + Sx * (Y – 1)^2 – Sx * Sy;
/*Second Region*/
      While (Y > 0)
         {
            Y = Y – 1;
            Py = Py – 2 * Sx;
            If (P > 0)
               {P = P + Sx – Py;}
            Else
               {
                  X = X + 1;
                  Px = Px + 2 * Sy;
                  P = P + Sx – Py + Px;
               }
            Call Draw_Ellipse (Xc, Yc, X, Y);
          }
   }

Draw_Ellipse (Xc, Yc, X, Y)
   {
```

```
        Call PutPixel (Xc + X, Yc + Y);
        Call PutPixel (Xc – X, Yc + Y);
        Call PutPixel (Xc + X, Yc – Y);
        Call PutPixel (Xc – X, Yc – Y);
}
```

Check your progress:

1. Give three representations of circle, also give their equations.
2. Fill in the blanks

In midpoint circle drawing algorithm if

f (x, y) < 0 means (x, y) is .....the circle
f (x, y) = 0 means (x, y) is .....the circle
f (x, y) > 0 means (x, y) is .....the circle

Answers: 2. inside, on, outside

2.4 DRAWING ELLIPSES AND OTHER CONICS

The equation of an ellipse with center at the origin is given as

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}=1$$

Using standard parameterization, we can generate points on it as

$$\theta \rightarrow (a\sin\theta, b\sin\theta)$$

Differentiating the standard ellipse equation we get

$$\frac{dy}{dx}=-\frac{x}{y}\frac{b^2}{a^2}$$

Now the DDA algorithm for circle can be applied to draw the ellipse.
Similarly a conic can be defined by the equation

$$ax^2+bxy+cy^2+dx+ey+f=0$$

If starting pixel on the conic is given, the adjacent pixel can be determined similar to the circle drawing algorithm.

Check your Progress:

1. Write down the equation of a standard ellipse.
2. Which scan conversion technique can be applied to draw an ellipse?

2.5 LET US SUM UP

We learnt about the scan conversion technique and how it is used to represent line, circle and ellipse. The DDA and Bresenham's line drawing algorithm were discussed. We then learnt Bresenham's and Midpoint circle drawing algorithm. Midpoint ellipse drawing algorithm was also illustrated. Finally we learnt about drawing ellipse and other conics.

2.6 REFERENCES AND SUGGESTED READING

(1) Computer Graphics, Donald Hearn, M P. Baker, PHI.
(4) Procedural elements of Computer Graphics, David F. Rogers, Tata McGraw Hill.
(5) Computer Graphics, Rajesh K. Maurya, Wiley – India.

2.7 EXERCISE

7. Describe scan conversion?
8. Explain briefly the DDA line drawing algorithm.
9. Explain the Bresenham's line drawing algorithm with example.
10. Discuss scan conversion of circle with Bresenham's and midpoint circle algorithms.
11. Explain how ellipse and other conics can be drawn using scan conversion technique.

3

TWO DIMENSIONAL TRANSFORMATIONS I

Unit Structure

3.0 OBJECTIVES

The objective of this chapter is

- To understand the basics of 2D transformations.
- To understand transformation matrix and types of 2D transformations.
- To understand two dimensional Identity transformations.
- To understand 2D Scaling and Reflection transformations.
- To understand Shear transformations in 2D.

3.1 INTRODUCTION

Transformations are one of the fundamental operations that are performed in computer graphics. It is often required when object is defined in one coordinate system and is needed to observe in some other coordinate system. Transformations are also useful in animation. In the coming sections we will see different types of transformation and their mathematical form.

3.2 INTRODUCTION TO TRANSFORMATIONS

In computer graphics we often require to transform the coordinates of an object (position, orientation and size). One can view object transformation in two complementary ways:

(i) **Geometric transformation**: Object transformation takes place in relatively stationary coordinate system or background.

(ii) **Coordinate transformation**: In this view point, coordinate system is transformed instead of object.

On the basis of preservation, there are three classes of transformation

- **Rigid body**: Preserves distance and angle. Example – translation and rotation
- **Conformal**: Preserves angles. Example- translation, rotation and uniform scaling
- **Affine**: Preserves parallelism, means lines remains lines. Example- translation, rotation, scaling, shear and reflection

In general there are four attributes of an object that may be transformed

(i) Position(translation)
(ii) Size(scaling)
(iii) Orientation(rotation)
(iv) Shapes(shear)

Check your progress:

1. Differentiate between geometrical and coordinate transformation.
2. Fill in the blanks
   (a) Rigid body transformation preserves.....
   (b) Conformal transformation preserves.....
   (c) Affine transformation preserves.....

Answers: 2(a) distance and angle (b) angles (c) parallelism

3.3 TRANSFORMATION MATRIX

Transformation matrix is a basic tool for transformation. A matrix with n×m dimensions is multiplied with the coordinate of objects. Usually 3×3 or 4×4 matrices are used for transformation. For example consider the following matrix for rotation operation

$$R = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}$$

We will be using transformation matrix to demonstrate various translation operations in the subsequent sections.

Check your progress:

1. Write down transfomation matrix for rotation operation at angle $\theta = 90^0$.

2. Obtain transformation matrix for 60^0rotation.

Answer: 1. $\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$

2. $\begin{bmatrix} \frac{1}{2} & -\frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & \frac{1}{2} \end{bmatrix}$

3.4TYPES OF TRANSFORMATION IN TWO – DIMENSIONAL GRAPHICS

In 2D transformations, only planar coordinates are used. For this purpose a 2x2 transformation matrix is utilized. In general, 2D transformation includes following types of transformations:

I. Identity transformation
II. Scaling
III. Reflection
IV. Shear transformation
V. Rotation
VI. Translation

3.5 IDENTITY TRANSFORMATION

In identity transformation, each point is mapped onto itself. There is no change in the source image on applying identity transformation. Suppose T is the transformation matrix for identity transformation which operates on a point P (x, y) which produces point P' (x', y'), then

P'(x', y') =[x' y']
= [P] [T]

$= [x\ y] \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

$=[x\ y]$

We can see that on applying identity transformation we obtain the same points. Here the identity transformation is

$[T] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

The identity transformation matrix is basically anxn matrix with ones on the main diagonal and zeros for other values.

Check your progress:
Fill in the blanks

1. In identity transformation, each point is mapped onto …….
2. The identity transformation matrix is basically anxn matrix with …… on the main diagonal and ……. for other values.

Answers: 1. Itself
2. ones, zeros.

3.6 SCALING

This transforms changes the size of the object. We perform this operation by multiplying scaling factors s_x and s_y to the original coordinate values (x, y) to obtain scaled new coordinates (x', y').

$x' = x.\ s_x$ $\qquad$ $y' = y.\ s_y$

In matrix form it can be represented as

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

For same s_x and s_y, the scaling is called as uniform scaling. For different s_x and s_y , the scaling is called as differential scaling.

Check your Progress:
Fill in the blanks

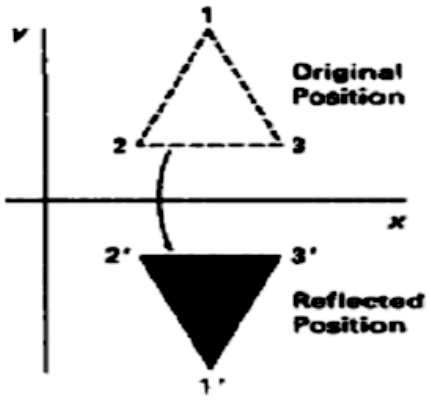
1. Scaling changes the …… of the object.
2. For same s_x and s_y, the scaling is called as ……. Scaling.

Answers: 1. Size.
2. uniform.

3.7REFLECTION

In reflection transformation, the mirror image of an object is formed. In two dimensions, it is achieved through rotating the object by 180 degrees about an axis known as axis of reflection lying in a plane. We can choose any plane of reflection in xy plane or perpendicular to xy plane.

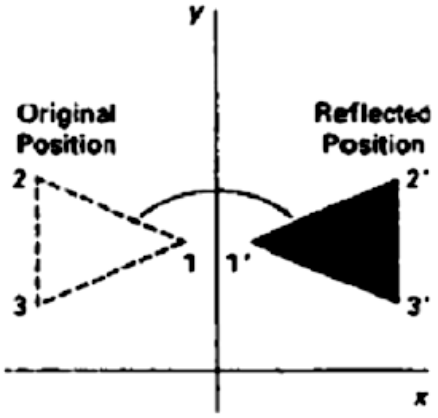
For example, reflection about x axis (y = 0) plane can be done with the transformation matrix



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

Figure 3.1 : Reflection transformation about x axis

A reflection about y axis (x = 0) is shown in the figure below which can be done by following transformation matrix.



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

Figure 3.2 : Reflection about y axis

Check Your Progress:Fill in the blanks

1. In reflection transformation, the …… image of an object is formed.
2. 2D reflection transformation can be achieved through rotating the object by …….degrees.

Answers: 1. Mirror
2. 180.

3.8 SHEAR TRANSFORMATIONS

An object can be considered to be composed of different layers. In shear transformation, the shape of the object is distorted by producing the sliding effect of layers over each other. There are two general shearing transformations, one which shift coordinates of x axis and the other that shifts y coordinate values.

The transformation matrix for producing shear relative to x axis is

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

producing transformations

$x' = x + sh_x.y, \quad y' = y$

wheresh_x is a shear parameter which can take any real number value. The figure below demonstrates this transformation

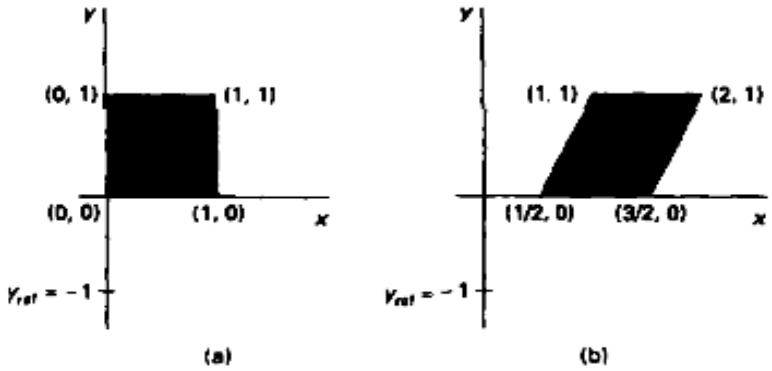


Figure 3.3 : 2D shear transformation

Check your Progress:

Fill in the blanks

1. In shear transformation, the shape of the object is distorted by producing the ……. effect of layers.
2. The shear parameter can take any ……number value.

Answers:

1. sliding
2. real

3.9 LET US SUM UP

We learnt about the basics of two dimensional transformations. We studied about transformation matrix and various types of transformations in 2D. Then we learnt about identity transformations, scaling and reflection transformations in two dimensions. Finally we understood the 2D shear transformation.

3.10 REFERENCES AND SUGGESTED READING

(1) Computer Graphics, Donald Hearn, M P. Baker, PHI.
(6) Procedural elements of Computer Graphics, David F. Rogers, Tata McGraw Hill.
(7) Computer Graphics, Rajesh K. Maurya, Wiley – India.

3.11 EXERCISE

1. Explain transformation and its importance.
2. Describe using transformation matrix, following 2D transformations
(i) Translation (ii) Scaling (iii) Reflection
3. Scale a triangle with respect to the origin, with vertices at original coordinates (10,20), (10,10), (20,10) by s_x=2, s_y=1.5.
4. What is the importance of homogenous coordinates?
5. Explain two dimensional shear transformations.
6. Obtain the transformation matrix for reflection along diagonal line (y = x axis).

Answers: 3. (20,30), (20,15), and (40,15)

4

TWO DIMENSIONAL TRANSFORMATIONS II

Unit Structure

4.0 OBJECTIVES

The objective of this chapter is

- To understand 2D rotation transformation.
- To understand 2D translation transformations.
- To understand two dimensional combined transformations.
- To understand homogenous coordinates and 2D transformation using homogenous coordinates.
- To understand Shear transformations in 2D.

4.1 INTRODUCTION

This chapter is the extension of the previous chapter in which we will discuss the rotation transformation about origin and about any arbitrary point. We will also learn about the translation transformation in which the position of an object changes. The homogenous coordinates and 2D transformation using homogenous coordinates will also be explained.

4.2 ROTATION

In rotation transformation, an object is repositioned along a circular path in the xy plane. The rotation is performed with certain angle θ, known as rotation angle. Rotation can be performed in two ways: about origin or about an arbitrary point called as rotation point or pivot point.

Rotation about origin: The pivot point here is the origin. We can obtain transformation equations for rotating a point (x, y) through an angleθ to obtain final point as (x', y') with the help of figure as

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

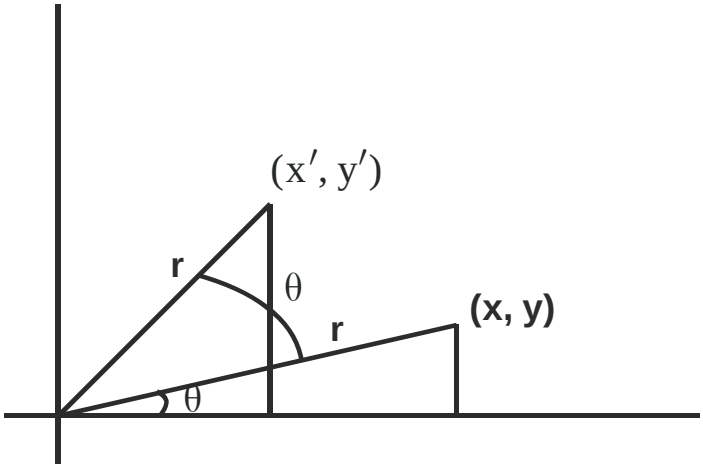


Figure 4.1 : Rotation about origin

The transformation matrix for rotation can be written as

$$R = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}$$

Hence, the rotation transformation in matrix form can be represented as (x_1, y_1)

$$P' = R.P$$

Check your progress:

1. Find the new equation of line in new coordinates (x', y') resulting from rotation of 90^0. [use line equation y = mx + c].

Answer: 1. y' = (-1/m)x – c/m.

4.3 TRANSLATION

The repositioning of the coordinates of an object along a straight line path is called as translation. The translation transformation is done by adding translation distance t_x and t_y to the original coordinate position (x, y) to obtain new position (x', y').

$$x' = x + t_x, \qquad y' = y + t_y$$

The pair (t_x, t_y) is called as translation vector or shift vector.
In the matrix form, it can be written as

$$P' = P + T \quad \text{, where}$$

$$P = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad P' = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad T = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

The figure below shows the translation of an object. Here coordinate points defining the object are translated and then it is reconstructed.

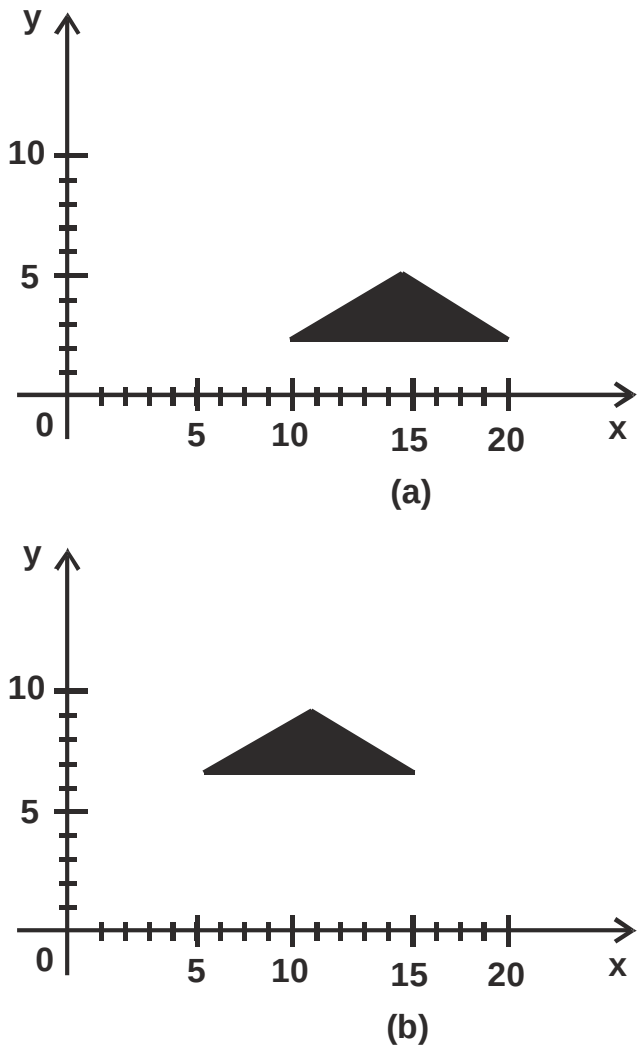


Figure 4.2 : 2D translation transformation

Check your Progress:

1. Translate a triangle with vertices at original coordinates (10, 20), (10,10), (20,10) by t_x=5, t_y=10.

Answer:
1. (15, 30), (15, 20), and (25, 20)

4.4ROTATION ABOUT AN ARBITRARY POINT

It is often required in many applications to rotate an object about an arbitrary point rather than the origin. Rotation about an arbitrary pivot point (x_r, y_r) is shown in the figure below

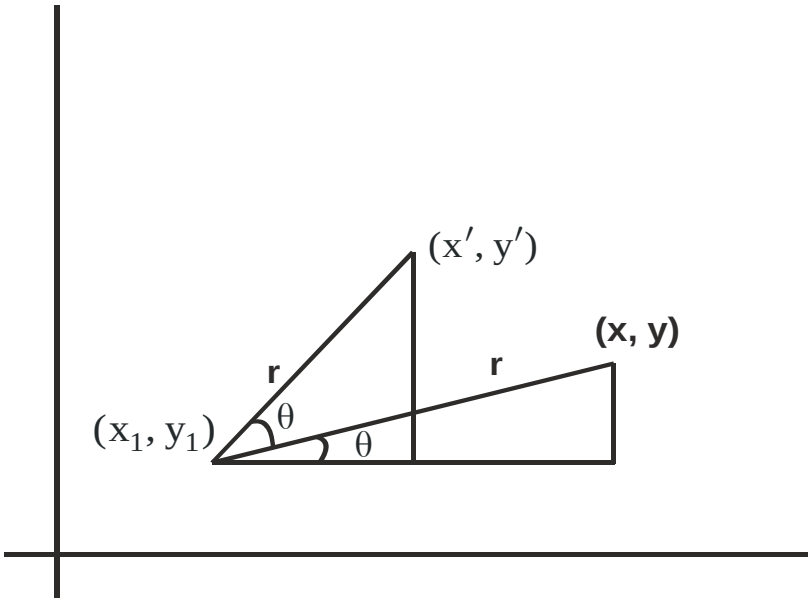


Figure 4.3 : Rotation about an arbitrary point

The corresponding equation obtained will be

$$x' = x_t + (x - x_t)\cos\theta - (y - y_t)\sin\theta$$
$$y' = y_t + (x - x_t)\sin\theta + (y - y_t)\cos\theta$$

We can see the difference between this rotation transformation from the previous one .This one contains the additive terms as well as the multiplicative factors on the coordinate values.

Let us understand the rotation about an arbitrary point through an example. Suppose we have to rotate a triangle ABC by 90 degree about a point (1, -1). The coordinates of the triangle are A (4, 0), B (8, 3) and C (6, 2).

The triangle ABC can be represented in matrix as

$$\begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} 4 & 0 \\ 8 & 3 \\ 6 & 2 \end{bmatrix}$$

The point about which the triangle has to be rotated be P = (-1, 1) and the rotation matrix for 90 degree rotation is

$$R_{90} = \begin{bmatrix} \cos 90 & sin90 \\ -sin90 & cos\ 90 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Now we can perform the rotation operation in three steps. In first step we will have to translate the arbitrary point to the origin.

Let A'B'C' be the new coordinates obtained after translation operation.

$$\begin{bmatrix} A' \\ B' \\ C' \end{bmatrix} = ABC - P = \begin{bmatrix} 4 & 0 \\ 8 & 3 \\ 6 & 2 \end{bmatrix} - \begin{bmatrix} -1 & 1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 5 & -1 \\ 9 & 2 \\ 7 & 1 \end{bmatrix}$$

Now the second step is to rotate the object. Let A''B''C'' be new coordinates after applying rotation operation to A'B'C', then

$$\begin{bmatrix} A'' \\ B'' \\ C'' \end{bmatrix} = A'B'C'.[R_{90}] = \begin{bmatrix} 5 & -1 \\ 9 & 2 \\ 7 & 1 \end{bmatrix}.\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 5 \\ -2 & 9 \\ -1 & 7 \end{bmatrix}$$

In third step we translate back the coordinates

$$\begin{bmatrix} A''' \\ B''' \\ C''' \end{bmatrix} = \begin{bmatrix} A'' \\ B'' \\ C'' \end{bmatrix} + [P] = \begin{bmatrix} 1 & 5 \\ -2 & 9 \\ -1 & 7 \end{bmatrix} + \begin{bmatrix} -1 & 1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 6 \\ -3 & 10 \\ -2 & 8 \end{bmatrix}$$

The coordinates A'''B'''C''' is the required result.

Check your progress:

1. What are the steps involved in rotating an object about an arbitrary point?

2. What is the difference between transformation about origin and rotation about an arbitrary point?

_____

_____

_____

4.5 COMBINED TRANSFORMATION

A sequence of transformation is said to be as composite or combined transformations can be represented by product of matrices. The product is obtained by multiplying the transformation matrices in order from right to left.

For example, two successive translations applied to position P to obtain P' is calculated as

$$P' = \{T(t_{x2}, t_{y2}).\ T(t_{x1}, t_{y1})\}.\ P$$

The expanded form of the multiplication of translation vectors of above equation can be written as

$$\begin{bmatrix} 1 & 0 & t_{x2} \\ 0 & 1 & t_{y2} \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & t_{x1} \\ 0 & 1 & t_{y1} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_{x1} + t_{x2} \\ 0 & 1 & t_{y1} + t_{y2} \\ 0 & 0 & 1 \end{bmatrix}$$

$$T(t_{x2}, t_{y2}).\ T(t_{x1}, t_{y1}) = T(t_{x1} + t_{x2}, t_{y1} + t_{y2})$$

Check your progress:

1. Explain Combined transformation with an example.

4.6HOMOGENOUS COORDINATES

Representing 2D coordinates in terms of vectors with two components turns out to be rather awkward when it comes to carry out manipulations in computer graphics. Homogenous coordinates allow us to treat all transformation in the same way, as matrix multiplications. The consequence is that our 2-vectors become extended to 3-vectors, with a resulting increase in storage and processing.

Homogenous coordinates means that we represent a point (x, y) by the extended triplet (x, y, w). In general w should be non-zero. The normalized homogenous coordinates are given by (x/w, y/w, 1) where (x/w, y/w) are the Cartesian coordinates at the point. Note in homogenous coordinates (x, y, w) is the same as

(x/w, y/w, 1) as is (ax; ay; aw) where a can be any real number. Points with w=0 are called points at infinity, and are not frequently used.

Check your progress:

1. What is the significance of homogenous coordinates?

4.7 2D TRANSFORMATIONS USING HOMOGENOUS COORDINATES

The homogenous coordinates for transformations are as follows:
For translation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

For rotation

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

For scaling

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

Check your progress:

1. Obtain translation matrix for tx=2, ty=3 in homogenous coordinate system.

2. Obtain scaling matrix for sx=sy=2 in homogenous coordinate system.

Answers: 1. $\begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix}$

2. $\begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

4.8 LET US SUM UP

We learnt the basics of transformation and its use. Then we studied two dimensional transformation and its types which were translation, scaling, rotation, reflection and shear. The transformation matrix corresponding to each transformation operation was also studied. Homogenous coordinate system and its importance were also discussed.

4.9 REFERENCES AND SUGGESTED READING

(1) Computer Graphics, Donald Hearn, M P. Baker, PHI.
(8) Procedural elements of Computer Graphics, David F. Rogers, Tata McGraw Hill.
(9) Computer Graphics, Rajesh K. Maurya, Wiley – India.

4.10 EXERCISE

12. Find the new coordinates of the point (2, -4) after the rotation of 30^0.
13. Rotate a triangle about the origin with vertices at original coordinates (10, 20), (10, 10), (20, 10) by 30 degrees.
14. Show that successive rotations in two dimensions are additive.
15. Obtain a matrix for two dimensional rotation transformation by an angle θ in clockwise direction.
16. Obtain the transformation matrix to reflect a point A (x, y) about the line y = mx + c.

Answers: 1. (√3+2, 1-2√3)
2. (-1.34, 22.32), (3.6, 13.66), and (12.32, 18.66)

5

THREE DIMENSIONAL TRANSFORMATIONS I

Unit Structure

5.0 OBJECTIVES

The objective of this chapter is

- To understand the Objects in Homogeneous Coordinates
- To understand the transformation matrix for 3D transformation.
- To understand the basics of three- dimensional transformations
- To understand the 3D scaling transformation.
- To understand 3D translation and rotation transformations.
- To understand 3D shear and reflection transformations.

5.1 INTRODUCTION

In two dimensions there are two perpendicular axes labeled x and y. A coordinate system in three dimensions consists similarly of three perpendicular axes labeled x, y and z. The third axis makes the transformation operations different from two dimensions which will be discussed in this chapter.

5.2 OBJECTS IN HOMOGENOUS COORDINATES

Homogeneous coordinates enables us to perform certain standard operations on points in Euclidean (XYZ) space by means of matrix multiplications. In Cartesian coordinate system, a point is represented by list of*n* points, where *n* is the dimension of the space. The homogeneous coordinates corresponding to the same point require *n+1* coordinates. Thus the two-dimensional point (x, y) becomes (x, y, 1) in homogeneous coordinates, and the three-dimensional point (x, y, z) becomes (x, y, z, 1). The same concept can be applied to higher dimensions. For example, Homogeneous coordinates in a seven-dimensional Euclidean space have eight coordinates.

In combined transformation, a translation matrix can be combined with a translation matrix, a scaling matrix with a scaling matrix and similarly a rotation matrix with a rotation matrix. The scaling matrices and rotation matrices can also be combined as both of them are 3x3 matrices. If we want to combine a translation matrix with a scaling matrix and/or a rotation matrix, we will first have to change the translation matrix in homogenous coordinate system.

Further in three dimensional, for uniform transformation we need to add a component to the vectors and increase the dimension of the transformation matrices. This for components representation is known as homogenous coordinate representation.

5.3THREE DIMENSIONAL TRANSFORMATIONS

The transformations procedure in three dimensions is similar to transformations in two dimensions.

- **3D Affine Transformation**:

A coordinate transformation of the form:

$x' = a_{xx}\, x + a_{xy}\, y + a_{xz}\, z + b_x$,
$y' = a_{yx}\, x + a_{yy}\, y + a_{yz}\, z + b_y$,
$z' = a_{zx}\, x + a_{zy}\, y + a_{zz}\, z + b_z$,

is called a 3D affine transformation. It can also be represented by transformation matrix as given below

$$\begin{pmatrix} x' \\ y' \\ z' \\ w \end{pmatrix} = \begin{pmatrix} a_{xx} & a_{xy} & a_{xz} & b_x \\ a_{yx} & a_{yy} & a_{yz} & b_y \\ a_{zx} & a_{zy} & a_{zz} & b_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

- Translation, scaling, shearing, rotation (or any combinations of them) are examples affine transformations.
- Lines and planes are preserved.
- Parallelism of lines and planes are also preserved, but not angles and length.

- **Object transformation**:
  Objects can be transformed using 2D and 3D transformation techniques

  - **Line**: Lines can be transformed by transforming the end points.
  - **Plane (described by 3-points)**:It can be transformed by transforming the 3-points.
  - **Plane (described by a point and normal)**: Point is transformed as usual. Special treatment is needed for transforming normal.

Check your progress:

1. Explain 3D affine transformation.
2. Explain object transformation.

5.4 SCALING

Scaling transformation changes the dimension of a 3D object determined by scale factors in all three directions.

$x' = x.\ s_x$ $y' = y.\ s_y$ $z' = z.\ s_z$

The transformation matrix and scaling through it

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

Check your Progress:

1. Explain the scaling process in three dimension.
2. Derive scaling matrix with sx=sy=2 and sz=1.

Answer: 2. $\begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

5.5 TRANSLATION

The three dimensional object displaced from its original position can be represented as

$x' = x + t_x, \qquad y' = y + t_y, \qquad z' = z + t_z$

The 3D translation by transformation matrix can be represented as

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

or $P' = T \cdot P$

Check your Progress:

1. Write down the matrix for three dimensional translation transformation.
2. Obtain 3D translation transformation matrix for tx=4, ty=3, tz=2.

Answer: 2. $\begin{bmatrix} 1 & 0 & 0 & 4 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

5.6 ROTATION

Rotations in three dimensions require specification of axis of rotation apart from prescription of the rotation angle. Rotation operation can be performed with respect to any axis. The following transformation matrix corresponds to the rotation about z axis.

$$R_z(\theta) = \begin{bmatrix} cos\theta & -sin\theta & 0 & 0 \\ sin\theta & cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Similarly transformation matrices for rotation about x and y axes are

$$R_Y(\theta) = \begin{bmatrix} cos\theta & 0 & sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -sin\theta & 0 & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_X(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\theta & -sin\theta & 0 \\ 0 & sin\theta & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

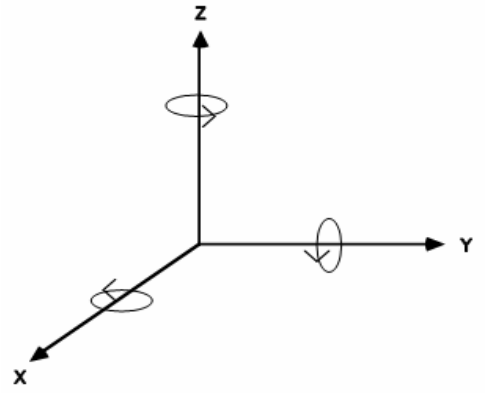


Figure 5.1: The three axes

The above figure illustrates the rotation about the three axes.

Properties of Rotation matrix:

- Determinant of the rotation matrix is 1.
- Columns and rows are mutually orthogonal unit vectors, i.e., orthonormal (inverse of any matrix is equal to transpose of that matrix).
- Product of any pair of rotation (orthonormal) matrices is also orthonormal.

Check your progress:

1. Explain three dimensional rotation transformation.
2. Derive the three dimensional rotation matrix about y axis with rotation angle 90 degrees.

Answer: 2. $\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

5.7 SHEAR TRANSFORMATIONS

Three dimensional shear transformation is similar to the two dimensional shear transformation. It produces the slanting effect to the image in a given direction of x, y or z. The shear transformation in x direction maintains y and z coordinates but produces change in x coordinate. It causes tilt left or right effect depending on the x – shear value. In the similar fashion y – shear and z – shear transformation produces slanting effect in the y and z direction respectively. The matrix for three dimensional shear transform is given by

$$\begin{bmatrix} 1 & a & b & c \\ c & 1 & d & 0 \\ e & f & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Check your Progress:

1. Explain shear transformation in three dimension.
2. Obtain 3D shearing transformation matrix for a=c=2, b=d=3, e=f=1.

Answer: 2. $\begin{bmatrix} 1 & 2 & 3 & 2 \\ 2 & 1 & 3 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

5.8 REFLECTION

The reflection transformation of a three dimensional object is performed with respect to a reflection axis or reflection plane in which the object is basically rotated by 180 degree. The following matrix corresponds to transformation matrix for reflection with respect xy plane

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

Check your Progress:

1. Explain reflection transformation.

5.9 LET US SUM UP

We learnt about the homogenous coordinates and its significance. We studied about three dimensional transformations. Then we learnt three dimensional scaling, rotation and translation transformation. Then we studied about shear and reflection transformation.

5.10 REFERENCES AND SUGGESTED READING

(1) Computer Graphics, Donald Hearn, M P. Baker, PHI.
(10) Procedural elements of Computer Graphics, David F. Rogers, Tata McGraw Hill.
(11) Computer Graphics, Rajesh K. Maurya, Wiley – India.

5.11 EXERCISE

17. Find the translation matrix for tx = 2, ty = 2, tz = 4.
18. Obtain scaling matrix for $s_x = 2$, $s_y = 3$ and $s_z = 1$
19. Obtain rotation matrix for $\theta = 45^0$ along z axis.
20. Find the rotation matrix for $\theta = 30^0$ about x axis.

21. Explain the significance of homogenous coordinates in three dimensional transformation.

Answers: 1. $\begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

2. $\begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

3. $\begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

4. $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{\sqrt{3}}{2} & -\frac{1}{2} & 0 \\ 0 & \frac{1}{2} & \frac{\sqrt{3}}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

❖❖❖❖

6

THREE DIMENSIONAL TRANSFORMATIONS II

Unit Structure

6.0 OBJECTIVES

The objective of this chapter is to understand

- World Coordinates and Viewing Coordinates
- Projection transformation – Parallel projections and perspective projection

6.1 INTRODUCTION

Projections help us to represent a three dimensional object into two dimensional plane. It is basically mapping of a point onto its image in the view plane or projection plane. There are different types of projection techniques. In this chapter we are going to discuss the basic idea of projection.

6.2 WORLD COORDINATES AND VIEWING COORDINATES

Objects in general are said to be specified by the coordinate system known as **world coordinate system (WCS)**. Sometimes it is required to select a portion of the scene in which the objects are placed. This portion is captured by a rectangular area whose edges are parallel to the axes of the WCS and is known as **window**.

- In simple words, a window refers to the area of a picture that is to be viewed.
- The area of the display device to which the window is mapped is known as viewport.
- The mapping of a part of scene specified by WCS to device coordinates is called as viewing transformation.
- The process of conversion of WCS coordinates of an object to normalized device coordinates is referred as window-to-viewport mapping.

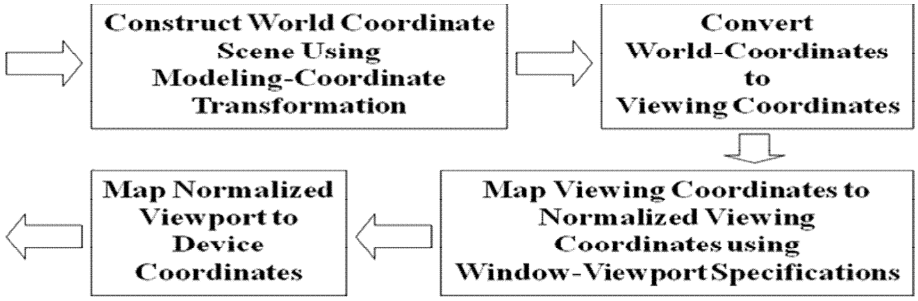


Figure 6.1 : Window to viewport mapping

- **Normalised device coordinates**:
  - **Normalised device coordinates** are the co-ordinates of the device expressed in normalised form.
  - The **normalised device co-ordinates** are thus the coordinates used to express the display space.
  - The co-ordinates are thus expressed in terms of their relative position on the display.
  - Conventionally **(0, 0)** is at the bottom left hand corner of the display and **(1, 1)** is the top right corner of the display.
  - Useful as they are device-independent.

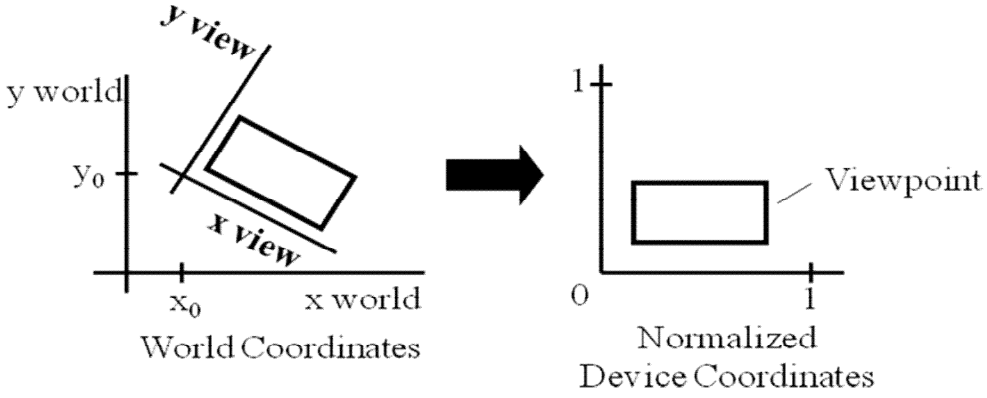


Figure 6.2: World coordinates and normalized device coordinate

Check your progress:

1. A rectangular area whose edges are parallel to the axes of the WCS and is known as.....**.**

2. Define normalised device coordinates.

Answers: 1. window

6.3 PROJECTION

Projection is the process of representing a 3D object onto a 2D screen. It is basically a mapping of any point P (x, y, z) to its image P (x', y', z') onto a plane called as projection plane. The projection transformation can be broadly classified into two categories: Parallel and Perspective projections.

6.4 PARALLEL PROJECTION

In parallel projections the lines of projection are parallel both in reality and in the projection plane. The orthographic projection is one of the most widely used parallel projections.

Orthographic projection: Orthographic projection utilizes perpendicular projectors from the object to a plane of projection to generate a system of drawing views.

- These projections are used to describe the design and features of an object.

- It is one of the parallel projection form, all the projection lines are orthogonal to the projection plane.

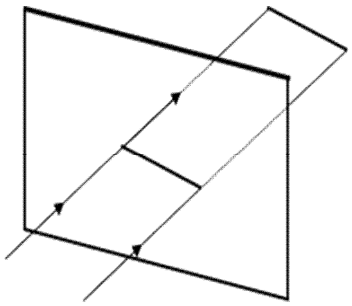


Figure 6.3: Projection plane and projection lines in orthogonal projection

- It is often used to generate the front, top and side views of an object.

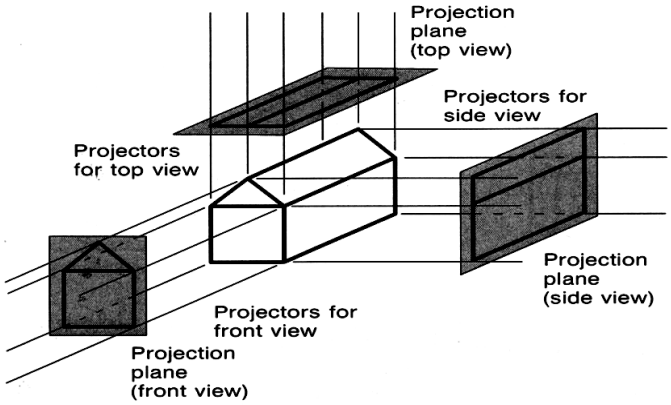


Figure 6.4: Views in orthographic projection

- It is widely used in engineering and architectural drawings.
- Orthographic projection that displays more than one face of an object is known as axonometric orthographic projections.

- **Axonometric projections** use projection planes that are not normal to a principal axis. On the basis of projection plane normal N = (dx, dy, dz) subclasses are
  - **Isometric**: | dx | = | dy | = | dz | i.e. N makes equal angles with all principal axes.

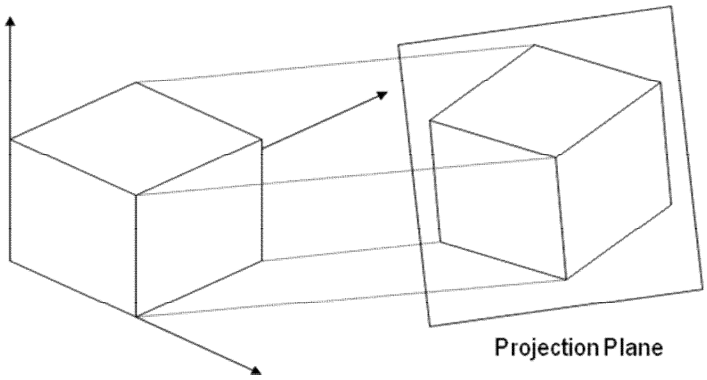


Figure 6.5: Axonometric projection

- **Dimetric** : | dx | = | dy |
- **Trimetric** : | dx | ≠ | dy | ≠ | dz |

Check your Progress:

1. Define axonometric projections.
2. Differentiate between isometric, dimetric and trimetric projections.

6.5 PERSPECTIVE PROJECTION

This projection method borrows idea from the artists who uses the principle of perspective drawing of three dimensional objects and scenes. The center of projection can be said analogous to the eye of the artist and the plane containing the canvas can be considered as view plane. Perspective projection is used to model 3D objects on 2D plane. It is done by projecting 3D points along the lines that pass through the single viewpoint until they strike an image plane.

- **Frustum view volume**: It specifies everything that can be seen with the camera or eye. It is defined by left plane, right plane, top plane, bottom plane, front (near) plane and back (far) plane.

The following figure illustrates perspective projection

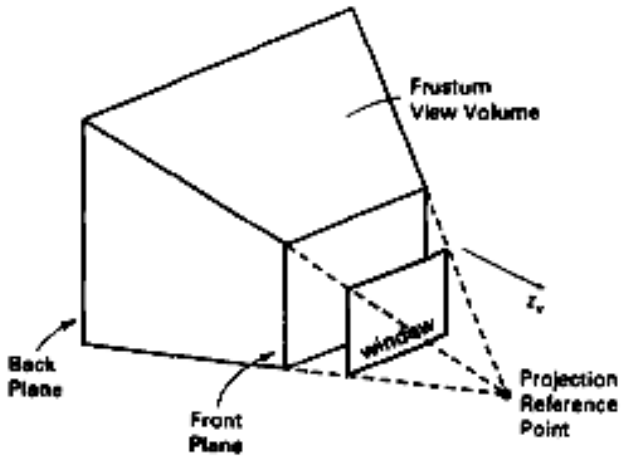


Figure 6.6: Perspective projection

- **Center of projection**: When a 3D scene is projected towards a single point, the point is called as **center of projection**. Vanishing points parallel to one of the principal axis is known as **principal vanishing point**. Projection from 3D to 2D is defined by straight **projection rays (projectors)** emanating from the center of projection, passing through each point of the object, and intersecting the **projection plane** to form a projection.

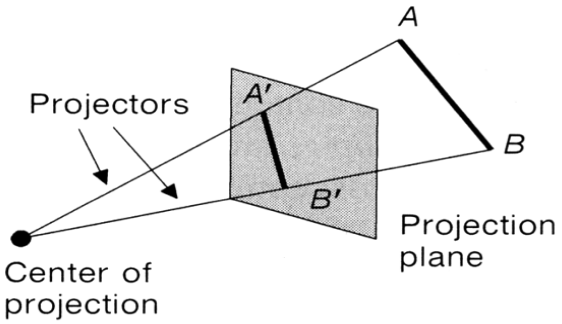


Figure 6.7: Perspective projection illustrating center of projection , projectors and projection plane

- **Perspective foreshortening**: It is the term used for the illusion in which the object or length appears smaller as the distance from the center of projection increases.

- **Vanishing points**: One more feature of perspective drawing is that sometimes a certain set of parallel lines appear to meet at a point. These points are known as vanishing points.

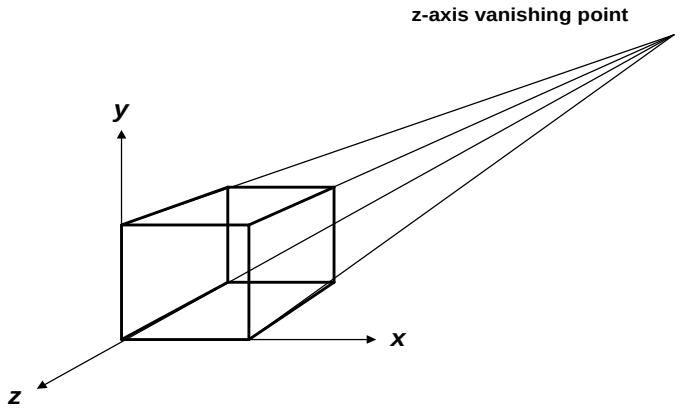


Figure 6.8: Vanishing point

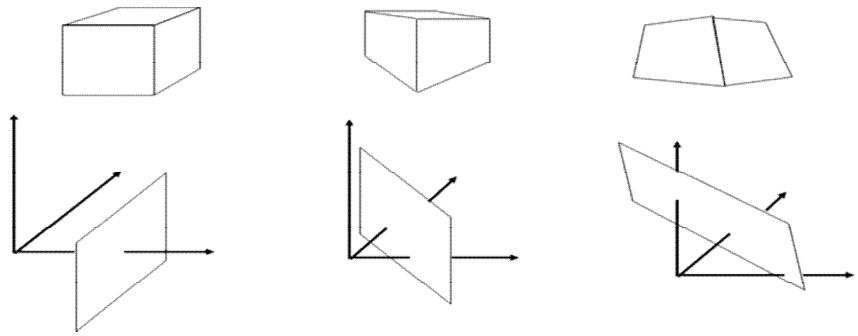
- **Principle vanishing point**: If a set of lines are parallel to one of the three axes, the vanishing point is called an axis vanishing point (Principal Vanishing Point). There are at most 3 such points, corresponding to the number of axes cut by the projection plane

  - **One-point**:
    - One principle axis cut by projection plane
    - One axis vanishing point

  - **Two-point**:
    - Two principle axes cut by projection plane
    - Two axis vanishing points

  - **Three-point**:
    - Three principle axes cut by projection plane
    - Three axis vanishing points

The following figure shows the three types of principle vanishing points



(a) One point (b) Two point (c) Three point

Figure 6.9 : Types of vanishing points

- **View confusion**: Objects behind the center of projection are projected upside down and backward onto the view plane.

- **Topological distortion**: A line segment joining a point which lies in front of the viewer to a point in back of the viewer is projected to a broken line of infinite extent.

Check your progress:

1. Define centre of projection.
2. What is the term used for the illusion in which the object or length appears smaller as the distance from the center of projection increases?

   Answer: 2. Perspective foreshortening

6.6 LET US SUM UP

In this chapter we learnt about world coordinate system and view coordinates. We then learnt the fundamental definition of projection. Orthographic projection with its application was discussed in short. We then learnt perspective projection and terms associated with it.

6.7 REFERENCES AND SUGGESTED READING

(1) Computer Graphics, Donald Hearn, M P. Baker, PHI.
(12) Procedural elements of Computer Graphics, David F. Rogers, Tata McGraw Hill.
(13) Computer Graphics, Rajesh K. Maurya, Wiley – India.

6.8 EXERCISE

1. Explain world coordinate system.
2. Define viewing coordinates.
3. Explain orthographic projection with its applications.
4. What is topological distortion?
5. Describe perspective projection and explain perspective foreshortening and vanishing points.

7

VIEWING AND SOLID AREA SCAN-CONVERSION

Unit Structure:

7.0 OBJECTIVES

The objective of this chapter is

- To understand the basics of concept of viewing transformations.
- To understand point clipping, line clipping and polygon clipping
- To understand the concept of text clipping.

7.1 INTRODUCTION TO VIEWING AND CLIPPING

Windowing and clipping

A “picture” is a “scene” consists of different objects.

The individual objects are represented by coordinates called as “model” or “local” or ”master” coordinates.

The objects are fitted together to create a picture, using co-ordinates called a **word coordinate *(WCS).***

The created “picture” can be displayed on the output device using **“physical device coordinates” *(PDCS).***

The mapping of the pictures elements from ***“WCS”*** to ***“PDCS”*** is called a viewing transformation.

Defination: a finite region selected in world coordinates is called as ***‘window ’*** and a finite region on which the window is mapped, on the output device is called a ***‘view point’.***

Viewing Pipeline

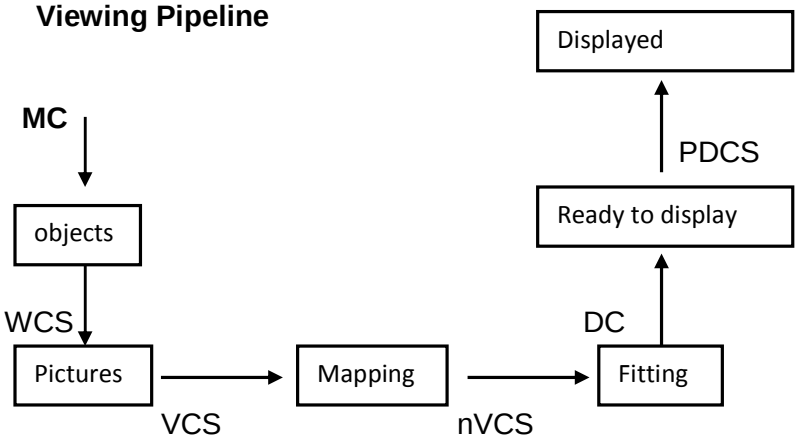


Fig. 7.1 Viewing Pipeline

Check your Progress:

1. What is PDCS?

2. Define window.

7.2 VIEWING TRANSFORMATION IN TWO DIMENSIONS

Viewing Transformation / a complete mapping from window to view point.

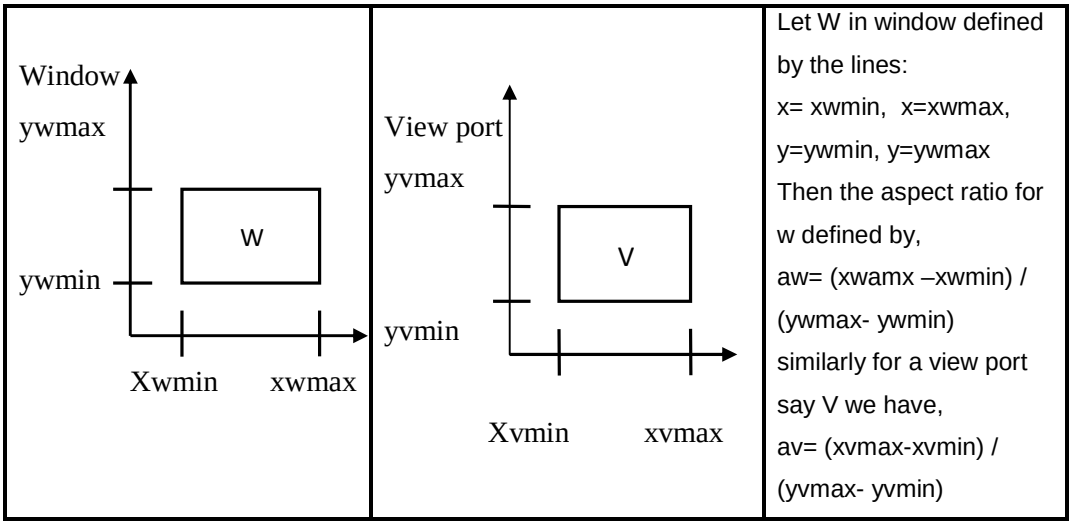


Fig. 7.2 Window and Viewpoint

7.3 INTRODUCTION TO CLIPPING

The process which divides the given picture into two parts : visible and Invisible and allows to discard the invisible part is known as clipping. For clipping we need reference window called as clipping window.

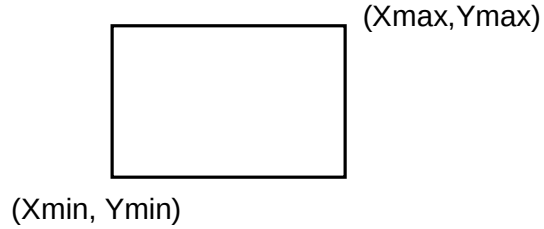


Fig. 7.3 Window

7.3.1 POINT CLIPPING

Discard the points which lie outside the boundary of the clipping window. Where,
Xmin ≤ X ≤ Xmax and
Ymin ≤ Y ≤Ymax

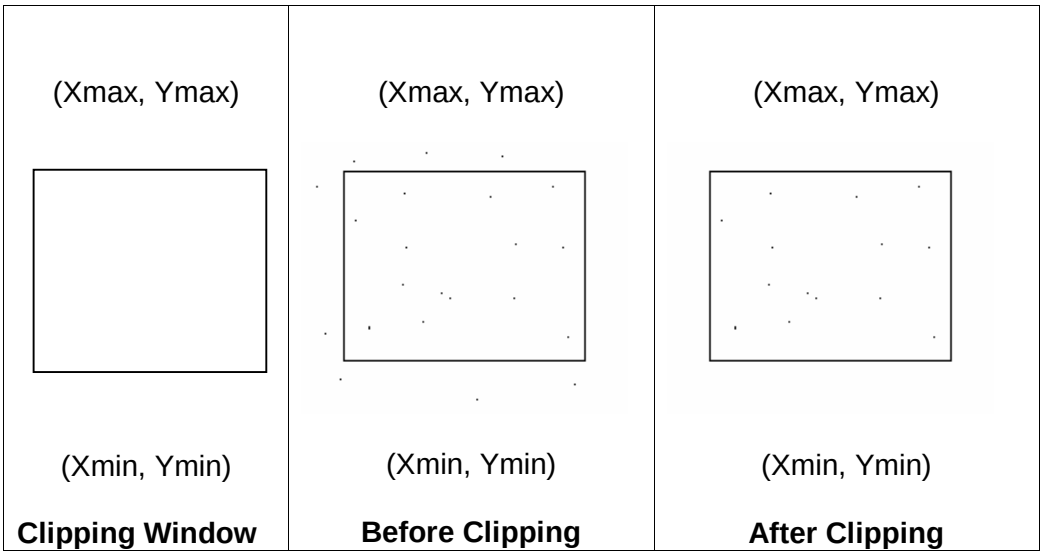


Fig. 7.4

7.3.2 LINE CLIPPING

Discard the part of lines which lie outside the boundary of the window.

We require:

1. To identify the point of intersection of the line and window.
2. The portion in which it is to be clipped.
   The lines are divided into three categories.

a) Invisible
b) Visible
c) Partially Visible [Clipping Candidates]

To clip we give 4- bit code representation defined by

Bit 1	Bit 2	Bit 3	Bit 4
Ymax	Ymin	Xmax	Xmin

Fig. 7.5

Where, Bits take the volume either 0 or 1 and
Here, we divide the area containing the window as follows. Where, the coding is like this, Bit value = 1 if point lies outside the boundary OR

= 0 if point lies inside the boundary.
(Xmin ≤ X ≤ Xmax and Ymin ≤ Y ≤Ymax)

Bit 1 tells you the position of the point related to Y=Ymax

Bit 2 tells you the position of the point related to Y=Ymin

Bit 3 tells you the position of the point related to X=Xmax

Bit 4 tells you the position of the point related to X=Xmin

1001	0001	0101
1000	0000 Clip Window	0100
1010	0010	0110

Fig. 7.6 Bit Code Representation

Rules for the visibility of the line:

1. If both the end points have bit code 0000 the line is visible.
2. If atleast one of the end point in non zero and
   a) The logical “AND”ing is 0000 then the line is Partially Visible
   b) If the logical “AND”ing isnon-zero then line is Not Visible.

Cohen-Sutherland Line Clipping Algorithm

For each line:

1. Assign codes to the endpoints
2. Accept if both codes are 0000, display line
3. Perform bitwise AND of codes
4. Reject if result is not 0000, return
5. Choose an endpoint outside the clipping rectangle
6. Test its code to determine which clip edge was crossed and find the intersection of the line and that clip edge (test the edges in a consistent order)
7. Replace endpoint (selected above) with intersection point
8. Repeat

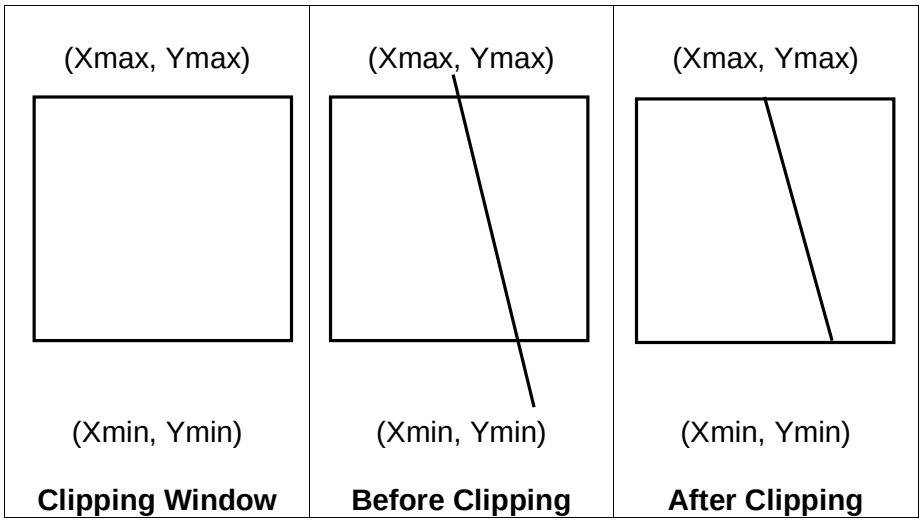


Fig. 7.7

Check your Progress :

Fill in the blanks

1. For clipping we need reference window called as _____window.

2. While clipping lines are divided into three categories invisible, visible and_____ visible.

7.4 INTRODUCTION TO A POLYGON CLIPPING

Polygon Clipping

Sutherland Hodgman Polygon Clipping algorithm

1. The polygon is stored by its vertices and edges, say v1, v2, v3 ,……vn and e1, e2, e3,…. en.
2. Polygon is clipped by a window we need 4 clippers.

   Left clipper , Right Clipper, Bottom Clipper, Top Clipper
3. After clipping we get a different set of vertices say $v1'$, $v2'$, $v3'$,…… vn'
4. Redraw the polygon by joining the vertices $v1'$, $v2'$, $v3'$,…… vn' appropriately.

Algorithm:

1. Read v1, v2, v3 ,……vn coordinates of polygon.
2. Readcliping window. (Xmin, Ymin)(Xmax, Ymax)
3. For every edge do {
4. Compare the vertices of each edge of the polygon with the plane taken as the clipping plane.
5. Save the resulting intersections and vertices in the new list } // according to the possible relationships between the edge and the clipping boundary.
6. Draw the resulting polygon.

The output of the algorithm is a list of polygon vertices all of which are on the visible side of the clipping plane.

Here, the intersection of the polygon with the clipping plane is a line so every edge is individually compare with the clipping plane. This is achieved by considering two vertices of each edge which lies around the clipping boundary or plane. This results in 4 possible relationships between the edge and the clipping plane.

1st possibility:

If the 1st vertex of an edge lies outside the window boundary and the 2nd vertex lies inside the window boundary.

Here, point of intersection of the edge with the window boundaryand the second vertex are added to the putput vertex list (V1, v2)→(V1', v2)

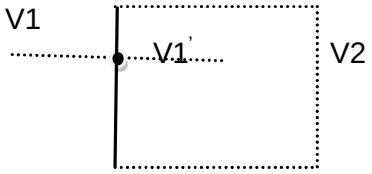


Fig. 7.8

2nd possibility:

If both the vertices of an edge are inside of the window boundary only the second vertex is added to the vertex list

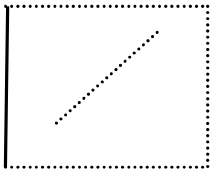


Fig. 7.9

3rd possibility:

If the 1st vertex is inside the window and 2nd vertex is outside only the intersection point is add to the output vertex list.

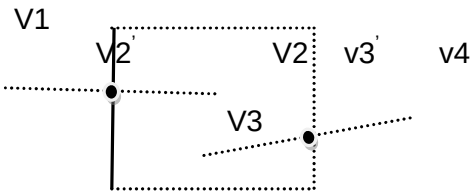


Fig. 7.10

4th possibility:

If both vertices are outside the window nothing is added to the vertex list.

Once all vertices are processed for one clipped boundary then the output list of vertices is clipped against the next window boundary going through above 4 possibilities. We have to consider the following points.

1) The visibility of the point. We apply inside-outside test.
2) Finding intersection of the edge with the clipping plane.

7.5 VIEWING AND CLIPPING IN THREE DIMENSIONS

We extend rays from the viewer's position through the corners of the viewing window; we define a volume that represents all objects seen by the eye. This viewing volume is shown in the left diagram of Figure. Anything outside the volume will not be visible in the window. When we apply the perspective projection, objects further away from the viewer become smaller, and objects in front of the window appear larger. Logically, this is identical to "warping" the viewing pyramid into a viewing rectangular solid in which the sides of the viewing box are parallel. For example, the cube shown in the left viewing volume becomes warped to the non-parallel object shown on the right. Now, the process of clipping becomes much simpler.

Clipping in 3D is similar to clipping in 2D. Everything outside of the canonical window that is not visible to the user is removed prior to display. Objects that are inside are retained, and objects that cross the window boundary need to be modified, or "clipped" to the portion that is visible. This is where the effect of the perspective transformation shown in Figure simplifies the process.

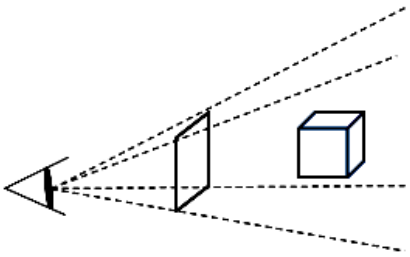


Fig. 7.11

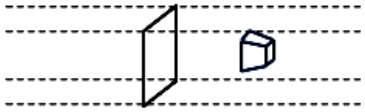


Fig. 7.12

If we were clipping to the sides of the pyramid as shown on the left, the calculations would be substantially more complex than the 2D clipping operations previously described. However, after the perspective transformation, clipping to the edges of the window is identical to clipping to the edges of the 2D window. The same algorithms can be used looking at the **x** and **y** coordinates of the points to clip.

To complicate matters, however, we have the added capability in 3D of defining clipping planes that are parallel to the viewing window, but at different depths from the viewer. These are often referred to as "near" and "far" clipping planes as shown in Figure. The concept is that objects that are too close to the viewer, or too far away, are not visible and should not be considered. In addition, without clipping against the near clipping plane, you would see objects that were behind the camera! If it were a simple matter

of culling objects based on their depths and clipping those that fell between the two planes, it would be no problem. However, the complexity arises when objects cross the boundaries of the near and far planes similar to when objects cross the edges of the windows. The objects need to be “clipped” to the far and near planes as well as to the edges of the window.

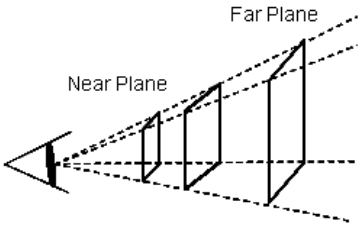


Fig. 7.13

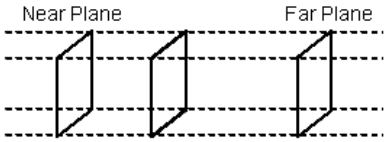


Fig. 7.14

7.6 THREE-DIMENSIONAL VIEWING TRANSFORMATIONS

3D Viewing Transformation :

The basic idea of the 3D viewing transformation is similar to the 2D viewing transformation. That is, a viewing window is defined in world space that specifies how the viewer is viewing the scene. A corresponding view port is defined in screen space, and a mapping is defined to transform points from world space to screen space based on these specifications. The view port portion of the transformation is the same as the 2D case. Specification of the window, however, requires additional information and results in a more complex mapping to be defined. Defining a viewing window in world space coordinates is exactly like it sounds; sufficient information needs to be provided to define a rectangular window at some location and orientation. The usual viewing parameters that are specified are: Eye Point the position of the viewer in world space.

Look Point the point that the eye is looking at View Distance the distance that the window is from the eye Window Size the height and width of the window in world space coordinates Up Vector which direction represents “up” to the viewer, this parameter is sometimes specified as an angle of rotation about the viewing axis These parameters are illustrated in Figure.

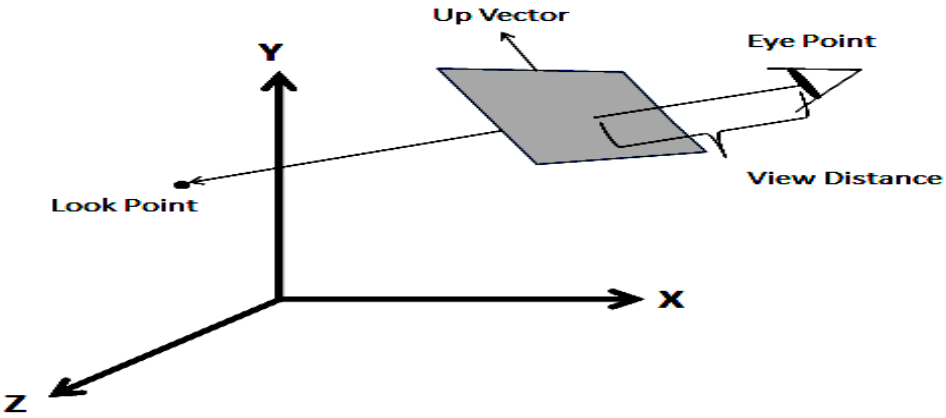


Fig.7.15

The Eye Point to the Look Point forms a viewing vector that is perpendicular to the viewing window. If you want to define a window that is not perpendicular to the viewing axis, additional parameters need to be specified. The Viewing Distance specifies how far the window is from the viewer. Note from the reading on projections, that this distance will affect the perspective calculation. The window size is straightforward. The Up Vector determines the rotation of the window about the viewing vector. From the viewer's point of view, the window is the screen. To draw points at their proper position on the screen, we need to define a transformation that converts points defined in world space to points defined in screen space. This transformation is the same as the transformation that positions the window so that it lies on the XY plane centered about the origin of world space.

The process of transforming the window, using the specified parameters, to the origin, aligned with the XY plane can be broken into the following steps:

1. Compute the center of the window and translate it to the origin
2. Perform two rotations about the X and Y axes to put the window in the XY plane
3. Use the Up Vector to rotate the window about the Z axis and align it with the positive Y axis
4. Use the Window Height and Width to scale the window to the canonical size

These four steps can be combined into a single transformation matrix that can be applied to all points in world space. After the transformation, points are ready for final projection, clipping, and drawing to the screen. The perspective transformation

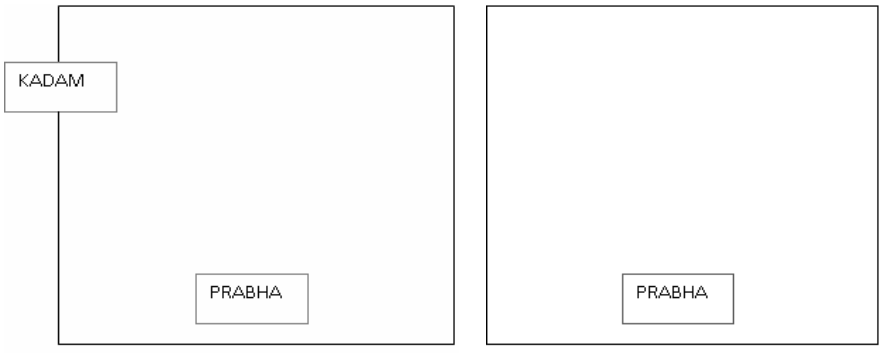
occurs after points have been transformed through the viewing transformation. The perspective and view port transformations will not be repeated here.

7.7 TEXT CLIPPING

- Depends on methods used to generate characters & the requirements of a particular application
- Methods or processing character strings relative to a window boundary,
  - All-or-none string clipping strategy
  - All or none character clipping strategy
  - Clip the components of individual characters

All-or-none string clipping strategy

- Simplest method, fastest text clipping
- All string - inside clip window, keep it, and otherwise discard.
- Bounding rectangle considered around the text pattern
- If bounding position of rectangle overlap with window boundaries, string is rejected.



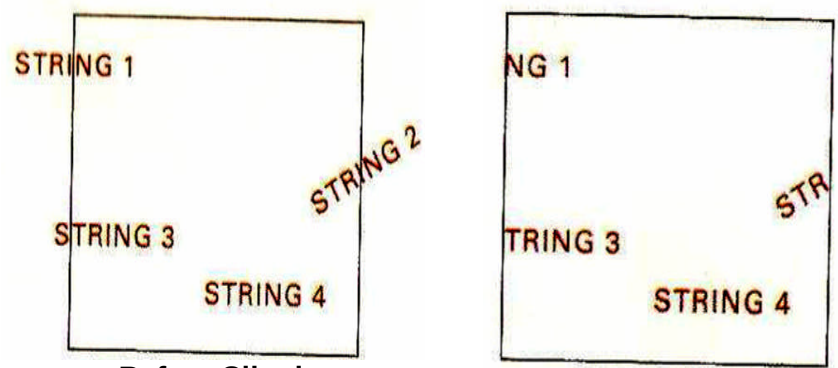
Before Clipping **After Clipping**

Text clipping using a bounding rectangle about the entire string

Fig. 7.16

All or none character clipping strategy :

- Discard or reject an entire character string that overlaps a window boundary i.e, discard those characters that are not completely inside the window.
- Compare boundary limits of individual characters with the window.
- Any character which is outside or overlapping the window boundary are clipped.



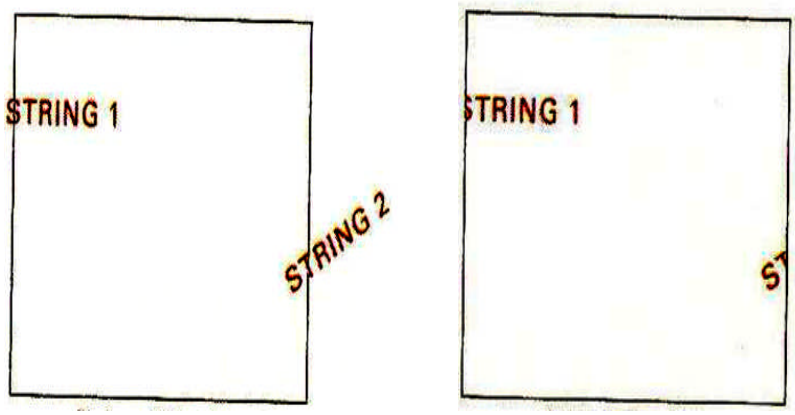
Before Clipping **After Clipping**

Text clipping using a bounding rectangle about individual characters

Fig. 7.17

Clip the components of individual characters :

- Treat characters same as lines
- If individual char overlaps a clip window boundary, clip off the parts of the character that are outside the window



Before Clipping **After Clipping**

Text clipping is performed on the components of individual characters.

Fig. 7.18

Check your Progress:
True or False.

1. The perspective and viewport transformations will not be repeated in3D Viewing Transformation.
2. In Sutherland Hodgman Polygon Clipping algorithm polygon is clipped by a window we need 4 clippers.
3. To check the visibility of the point, we apply inside-outside test.

7.8 LET US SUM UP

- Point clipping, line clipping, polygon clipping and text clipping are types of the clipping.
- Normally window and view points are ‘rectangular’ shaped.
- The viewing transformation is also called as windowing transformation.
- Discarding and removing the invisible region of object from the given window is known as clipping.

7. 9 REFERENCES AND SUGGESTED READING

- Computer Graphics, Donald Hearn, M P. Baker, PHI.
- Procedural elements of Computer Graphics, David F. Rogers, Tata McGraw Hill.
- Computer Graphics, Amarendra Sinha, A. Udai,, Tata McGraw Hill.
- Computer Graphics,A. P. Godase, Technical Publications Pune.

7.10 EXERCISE

1. What is point clipping?
2. Explain Cohen-Sutherland Line clipping algorithm.
3. Explain the polygon clipping algorithm.
4. Write a short note on text clipping.
5. Define: window, View point.

8

INTRODUCTION TO SOLID AREA SCAN-CONVERSION

Unit Structure:

8.0 OBJECTIVE

The objective of this chapter is

- To understand polygon filling techniques and algorithms.
- To understand scan conversion of characters.
- To understand concepts of anti-alising, half toning, thresholding and diathering.

8.1 INTRODUCTION

To perform the scan conversion or to fill the polygon, we need the pixels inside the polygon as well as those on the boundary. The pixels which are inside the polygon can be determined by using the following two test: Inside outside test and Winding number test.

8.2 INSIDE–OUTSIDE TEST

1. Inside outside test (Even- Odd Test)

We assume that the vertex list for the polygon is already stored and proceed as follows.

1. Draw any point outside the range Xmin and Xmax and Ymin and Ymax. Draw a scanline through P upto a point A under study

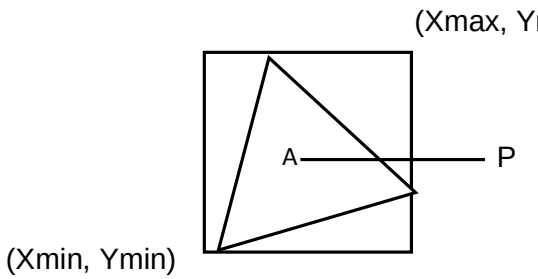


Fig. 8.1

2. If this scan line

i) Does not pass through any of the vertices then its contribution is equal to the number of times it intersects the edges of the polygon. Say C if

a) C is odd then A lies inside the polygon.
b) C is even then it lies outside the polygon.

ii) If it passes through any of the vertices then the contribution of this intersection say V is,

a) Taken as 2 or even. If the other points of the two edges lie on one side of the scan line.
b) Ttaken as 1 if the other end points of the 2 edges lie on the opposite sides of the scan- line.
c) Here will be total contribution is C + V.

Remark : Here, the points on the boundary are taken care of by calling the procedure for polygon generation.

8.2 WINDING NUMBER METHOD AND COHERENCE PROPERTY

Winding number algorithm :

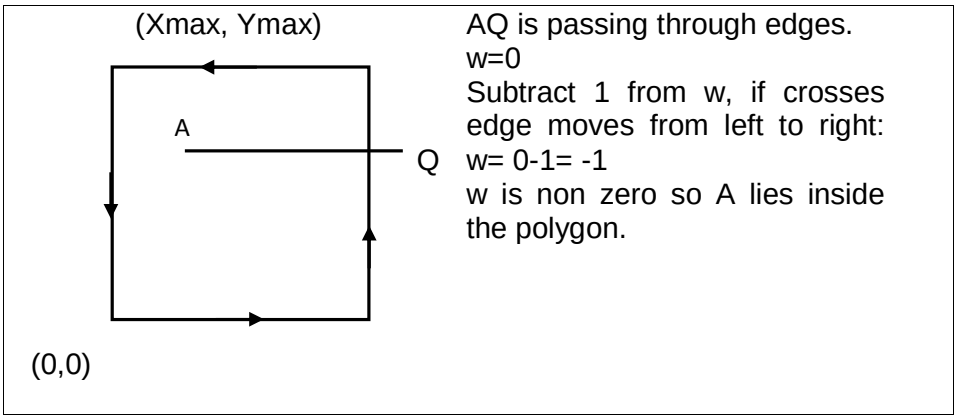
This is used for non- overlapping regions and polygons only.

Steps:

1. Take a point A within the range (0,0) to (Xmax, Ymax) Joint it to any point Q outside this range.

2. Give directions to all the edges in anticlockwise direction.

3. Check whether Q passing through any of the vertices. If so ignored the position of Q. choose a new Q so that AQ does not pass through any of the vertices but passes through only edges.



4. Initialize winding number w=0. Observe the edges intersecting AQ and
1) Add 1 to w if cross edge moves from right to left
2) Subtract 1 from w, if crosses edge moves from left to right.

5. If final count of w is zero
1) A lies outside the polygon
2) Non zero, A lies inside the polygon.
3) Illuminate the interior position till all the pixels in the above set range are painted.

8.3 POLYGON FILLING AND SEED FILL ALGORITHM

Polygon filling algorithm

There are two types of polygon filling algorithm.
1. Scan conversion polygon filling algorithm
2. Seed filling algorithms

Besides these algorithms we can use
a) Boundary fill algorithms and
b) Flood fill algorithm

Seed Fill

To fill a polygon we start with a seed and point the neighboring points till the boundary of the polygon is reached. If boundary pixels are not reaching pixels are illuminated one by one and the process is continuing until the boundary points are reached. Here, at every step we need check the boundary. Hence, this algorithm is called “boundary fill algorithm”.

To find the neighboring pixels we can use either a 4 connected or 8 connected region filling algorithm.

The algorithm is recursive one and can be given as follows:

Here, we specify the parameters, fore-color by F and back-color by B

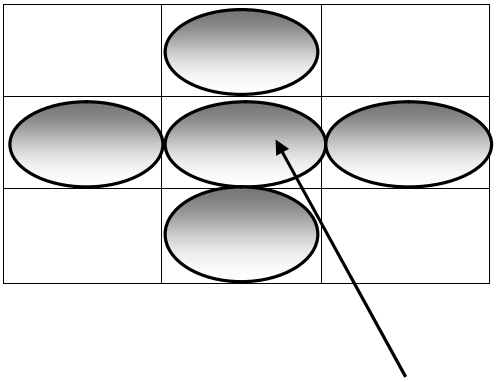


Fig. 8.3 Neighboring Pixels

```
Seed pixel  (x,y)
Seed_Fill ( x, y, F, B)// 4 connected approach
{
//Seed= getpixel (x,y);
If (getpixel(x,y)!= B && getpixel(x,y)!= F)
{
putpixel (x,y,F);
Seed_Fill (x-1, y, F, B);
Seed_Fill (x+1, y, F, B);
Seed_Fill (x, y-1, F, B);
Seed_Fill (x, y+1, F, B);
}
}
```

getpixel (): is a procedure which gives the color of the specified pixel.
putpixel(): is a procedure which draws the pixel with the specified color.
B : is the boundary color.

Drawbacks: in Seed Fill algorithm we have 2 drawbacks.

1. If some inside pixels are already painted with color F then the recursive branch terminates leaving further internal pixels unfilled.
2. The procedure required stacking of neighboring pixels. If the polygon is too large the stack space may became insufficient for all the neighboring pixels.

To remove the above drawbacks we use the second approach. Scan Line Filling algorithm.

Check your Progress:
True or False
1. In winding number test directions to all the edges in anticlockwise direction
2. Seed fill algorithm fails if polygon is large.
3. In inside outside test if C (count) is odd then A lies inside the polygon.

8.4 SCAN-LINE ALGORITHM

Scan Line Algorithm.

In scanline filling algorithm, we take the intersection of each scanline with the edges of the polygon.

Steps :

1. Read n
2. Read (xi,yi) for all i=1,2,3……n
3. Read edges and store it in the array E which will be sorted accordingly to y axies.
4. Xmin=a; xmax=b; ymin=c; ymax=d
5. Take intersection
6. Take the scanline y=c and scan from x=a to x=b
7. Find the intersecting edges of E with y=c by comparing the y coordinate of the end points with y=c
8. Activate those edges
9. Scan through the line y=c and compute the next x position by appling the formulation
   $X_{k+1}= x_k +1/m$
   Check whether the point $(X_{k+1,}, Y_k)$ is inside or outside the polygon, by inside outside procedure. If the point $(X_{k+1,}, Y_k)$ is inside , paint it.
10. Repeat the procedure from Y_c to Y_d i.e. y=c to y=d.

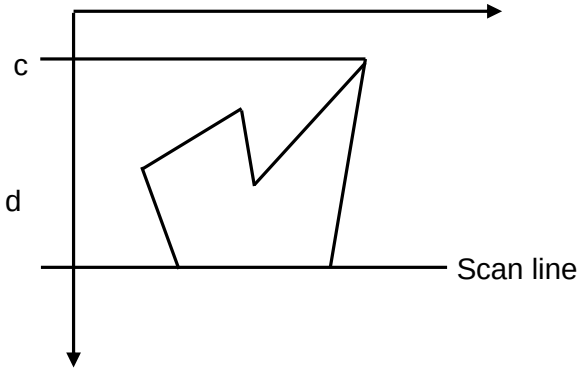


Fig. 8.4

Here, all the pixels inside the polygon can be painted without leaving any of the neighboring pixels. If the point of intersection of an edge and the scanline is a vertex, we shorten one of the edges. So that it will be contributed to the intersection is 1. If the endpoint of the two edges are on one side of the scan line and the contribution will be 2 if the other points are on the opposite side of the scanline.

The scan line filling algorithm can be applied for the curve closed boundary as follows:

1. Determine the pixels position along the curve boundary by using any of the incrementing methods

2. Filling the pixels by scanning through a scanline which spans between the boundary points. If the shape to be filled is regular geometrical figure like circle, ellipses etc. use symmetric property of geometrical figure to reduce boundary point calculations.

8.5 PRIORITY ALGORITHM

In the context of computer graphics, priority fill is a Hidden Line/Surface Removal algorithm which establishes a priority list based upon the depth of parts of an object, so that the parts farthest from the viewer are rendered first.

The algorithm continues in reverse priority, just as an artist would create a painting starting with the background, then elements or objects at an intermediate distance would be added and finally those objects in the foreground. Priority fill is also known as the Painter's algorithm

8.6 SCAN CONVERSION OF CHARACTER

Meanings:

- **Glyph:** In information technology, a glyph (pronounced GLIPH; from a Greek word meaning carving) is a graphic symbol that provides the appearance or form a character. A glyph can be an alphabetic or numeric font or some other symbol that pictures an encoded character.

- **Contour:** A line drawn on a map connecting points of equal height or an outline especially of curving or irregular figure: SHAPE

Character fonts, such as letters and digits, are the building blocks of textural content of an image presented in variety of styles and attributes. Character fonts on raster scanned display devices are usually represented by arrays of bits that are displayed as a matrix of black and white dots. Value for Black - 0 and White - 1.

There are three basic kinds of computer font file data formats:

- Bitmap font consists of a series of dots or pixels, representing the image of each glyph in each face and size.

- Outline fonts use Bezier curves, drawing instructions and mathematical formulas to describe each glyph, which make the character outline scalable to any size.

- Stroke fonts use a series of specified lines and additional informational information to define the profile, size and shape of a line in a specific face and size, which together describe the appearance of the glyph.

A scan conversion is essentially the job of coloring inside the character outlines contained in the font; scan converter is able to maintain the continuity of character bitmaps by performing dropout control. Dropouts occur when the space within the outlines becomes so narrow that pixel centers are missed.

The process of a scan conversion consists of four steps:

1. Measurement: The outline of the character is traversed point by point and contour by contour in order to find the maximum and minimum coordinate values of the outline. In addition, the amount of workspace memory that will be needed to perform steps 2 and 3 is calculated.

2. Rendering: Every contour is broken into lines and splines. Calculations are made to find the point at which each line or spline intersects with scan lines. The intersections for each scanline are scaled from left to right.

3. Filling: Using the sorted intersections, runs of pixels are set for each scan line of the bitmap from top to bottom.

4. Dropout control: If dropout control is enabled, the intersection list is checked again looking for dropouts. If various criteria are met, it is decided which dropout pixel to set, and then it is set. The dropout control requires scanning in the vertical as well as the horizontal directions.

Check your Progress:

Fill in the blanks.
1. Priority fill is also known as the_____ algorithm
2. If the endpoint of the two edges are on one side of the scan line and the contribution will be _____.

8.7 ALIASING, ANTI-ALIASING, HALF TONING

Aliasing:

Aliasing is the distortion of information due to low- frequency sampling. Low- frequency sampling results in highly periodic images being rendered incorrectly. For example, a fence or building might appear as a few broad stripes rather than many individual smaller stripes.

Anti-Aliasing:

Anti-aliasing is the process of blurring sharp edges in pictures to get rid of the jagged edges on lines. After an image is rendered, some applications automatically anti-alias images. The program looks for edges in an image, and then blurs adjacent pixels to produce a smoother edge. In order to anti-alias an image when rendering, the computer has to take samples smaller than a pixel in order to figure out exactly where to blur and where not to.

A large atmark rendered without antialiasing

Antialiasing smoothes out the jaggies.

Fig. 8.5

Half Toning :

Many hardcopy devices are bi-level: they produce just two intensity levels. Then to expand the range of available intensities there is Halftoning or clustered-dot ordered dither

It make the most use of the **spatial integration** that our eyes perform. If we view a very small area from a sufficiently large viewing distance, our eyes average fine detail within the small area and record only the overall intensity of the area.

In halftoning approximation, we have two different cases. First when the image array being shown is smaller than the display device's pixel array. In this case multiple display pixels can be used for one image pixel. And second when the image array has the same size of display device arrays

xample of a 2×2 dither pattern

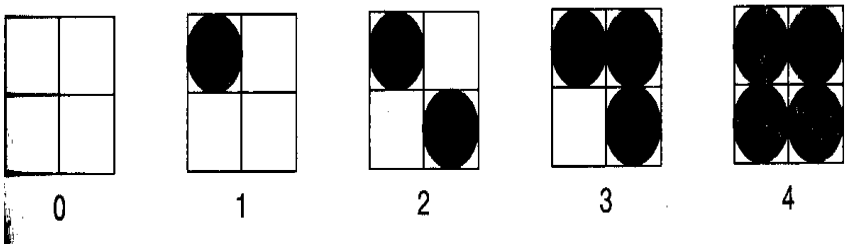


Fig. 8.6

8.8 THRESHOLDING AND DITHERING

Dithering is the process of converting an image with a certain bit depth to one with a lower bit depth. For example: Original image Dithered to 256 colors

When an application dithers an image, it converts colors that it cannot display into patterns of two or more colors that closely resemble the original. You can see that in the B&W image. Patterns of different intensities of black and white pixels are converted represent different shades of gray.

Thresholding is a process where an image gets divided in two different colors i.e. Black and White. This kind of image is also called as binary image, since it is divided in to two colors Black – 0 and White – 1. In this process one or more than one thresholding points get decided and then the gray level values in the given image are get adjusted accordingly.
Example : Thresholding point 4

1	0	7
4	6	2
7	5	3

0	0	7
0	7	0
7	7	0

The Original Image	Threhold Image
3x3 image . 3bit image. L= 2^{bit} size $=2^3 = 8$ Gray values L= 0 to 7 Lmin (Black) - 0 Lmax (White) - 7	If color(Gray Value) <= T ---- 0 (Lmin) Else If color(Gray Value) > T ---- 7(Lmax)

Check your Progress:

1. Define: Thresholding.
2. What is anti- alising?
3. Explain Half Toning.

8.9 LET US SUM UP

- The pixels which are inside the polygon can be determined by using the following two test: Inside outside test and Winding number test.
- The scan line filling algorithm can be applied for the curve closed boundary
- Priority fill is also known as the Painter's algorithm
- Anti-aliasing is the process of blurring sharp edges in pictures to get rid of the jagged edges on lines
- Dithering is the process of converting an image with a certain bit depth to one with a lower bit depth.
- Thresholding is a process where an image gets divided in two different colors

8.10 REFERENCES AND SUGGESTED READING

- Procedural elements of Computer Graphics, David F. Rogers, Tata McGraw Hill.

- Computer Graphics, ISRD Group, Tata McGraw Hill.
- Computer Graphics, Amarendra Sinha, A. Udai,, Tata McGraw Hill.
- Computer Graphics,A. P. Godase, Technical Publications Pune.
- Computer Graphics, Donald Hearn, M P. Baker, PHI.

8.11 EXERCISE

1. What are the polygon filling techniques? Explain any one.
2. Write a short note on Scan conversion of character.
3. Explain Inside- Outside test.
4. Define: Aliasing, Anti- Aliasing.
5. Write a short note on Thresholding and Dithering.

9

INTRODUCTION TO CURVES

Unit Structure:

9.0 OBJECTIVE

The objective of this chapter is

- To understand concept of curve and different types of curves.
- To understand difference between Bezier and B- Spline curves.
- To understand concept of fractals and its branches and different application areas.

9.1 INTRODUCTION

Curves (and surfaces) are specified by the user in terms of points and are constructed in an interactive process. Here, are few points which we have to consider since we are going to learn about curves: Control Points, Multi valued, Axis Independent, Global or Local Curve Control, Diminishing variation and versatility, Order of continuity, Parametric function, Blending functions.

Fractal was coined in 1975, by mathematician Benoit Mandelbrot to describe an intricate looking set of curves, many of which were never seen before the advent of computers, because of its ability to perform quickly massive calculations. Fractals are figures with an infinite amount of detail. When magnified, they don't become more simple, but remain as complex as they were without magnification

9.2 CURVE CONTINUITY

A breakpoint is where two curve segments meet within a piecewise curve. The continuity of a curve at a breakpoint describes how those curves meet at the breakpoint.

There are four possible types of continuity:

No continuity: It means the curves do not meet at all.

C^0continuity : Here, it may be a sharp point where they meet.

C^1continuity: The curves have identical tangents at the breakpoint and the curves join smoothly. **C^2continuity**: The curves have identical curvature at the breakpoint and curvature continuity implies both tangential and positional continuity.

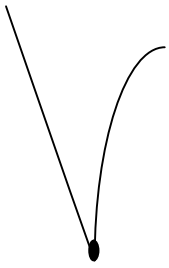


Fig. 9.1

Fig. 9.2

Fig. 9.3

9.3 CONIC CURVES

Both circles and ellipses are special cases of a class of curves known as conics. Conics are distinguished by second-degree discriminating functions of the form:

$$f(x,y) = Ax^2 + Bxy + Cy^2 + Dx + Ey + F$$

The values of the constants, A, B, C, D, E, and F determines the type of curve as follows:

$B^2 - 4AC < 0$ circle if (A = C and B = 0), ellipse otherwise

$B^2 - 4AC = 0$ line if (A = B = C = 0), parabola otherwise

$B^2 - 4AC > 0$ hyperbola

To make things confusing, mathematicians often refer to the term B^2 - 4AC as the conic discriminate. Here we will stick to the computer graphics definition of a discriminant as a function that partitions interior and exterior half-spaces.

Curves of this form arise frequently in physical simulations, such as plotting the path of a projectile shot from a canon under the influence of gravity (a parabola), or the near collision of like-charged particles (hyperbolas).

Conics, like circles posses symmetry, but not nearly to the same extent. A circle is a very special case of conic, it is so special that it is often considered a non-generic conic. Typically a conic will have only one (parabola) or two (ellipse or hyperbola) symmetric axes.

In order to compute the slope at each point we'll need to find derivatives of the discriminating equation:

$$\frac{\partial f(x,y)}{\partial x} = 2Ax + By + D$$

$$\frac{\partial f(x,y)}{\partial x} = Bx + 2Cy + E$$

Using these equations we can compute the instantaneous slope at every point on the conic curve.

9.4 PIECEWISE CURVE DESIGN

The order of the curve determines the minimum number of control points necessary to define the curve. You must have at least order control points to define a curve. To make curves with more than order control points, you can join two or more curve segments into a piecewise curve

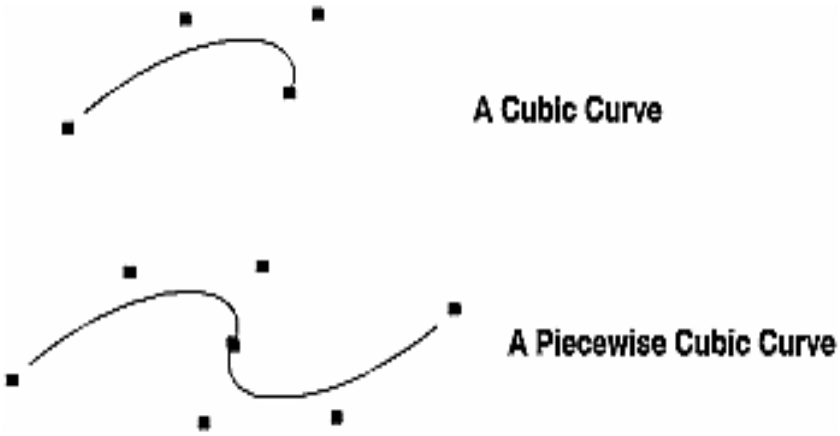


Fig. 9.4

Check your Progress:

Fill in the blanks.

1. _____ and ellipses are special cases of a class of curves known as conics **.**

2. To make curves with more than order control points, you can join two or more curve segments into a _____.

9.5 PARAMETRIC CURVE DESIGN

A parametric curve that lies in a plane is defined by two functions, x(t) and y(t), which use the independent parameter t. x(t) and y(t) are coordinate functions, since their values represent the coordinates of points on the curve. As t varies, the coordinates (x(t), y(t)) sweep out the curve. As an example consider the two functions:

x(t) = sin(t)

y(t) = cos(t)

As t varies from zero to 360, a circle is swept out by (x(t), y(t)).

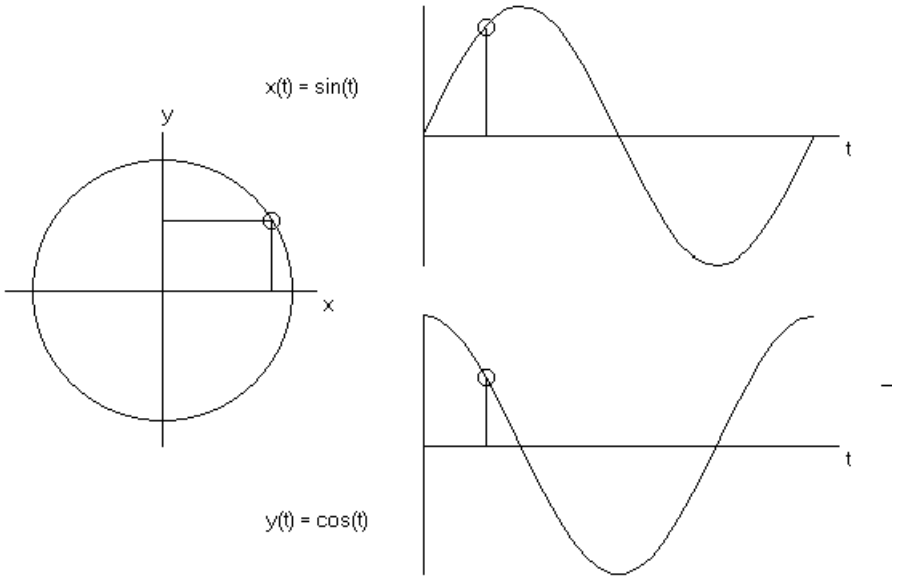


Fig. 9.5 **Fig. 9.6**

9.6 SPLINE CURVE REPRESENTATION

A spline curve is a mathematical representation for which it is easy to build an interface that will allow a user to design and control the shape of complex curves and surfaces. The general approach is that the user enters a sequence of points, and a curve is constructed whose shape closely follows this sequence. The points are called control points. A curve that actually passes through each control point is called an interpolating curve; a curve that passes near to the control points but not necessarily through them is called an approximating curve.

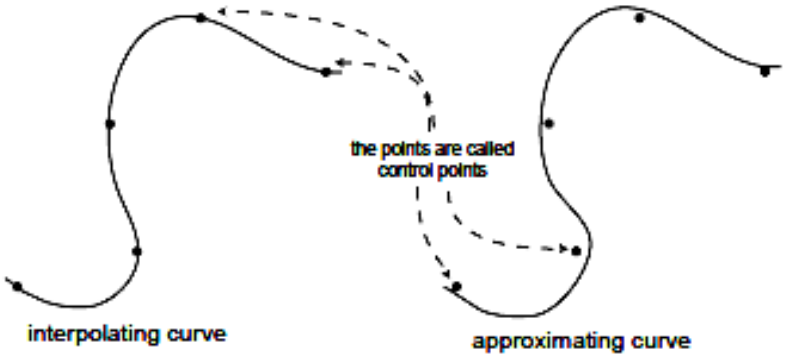


Fig. 9.7

9.7 BEZIER CURVES

Bezier curve

Bezier curve section can be fitted to any number of control points. The number of control points to be approximated and their relative position determine the degree of the Bezier polynomial. A Bezier curve can be specified with boundary conditions, with blending function. Suppose we are given n+1 control point positions: Pk =(Xk,Yk,Zk) with k varing from 0 to n. these coordinate points can be blended to produce the following position vector P(u)_, which describes the path of an approximating Bezier polynomial function between P0 and Pn.

$P(u)= \sum^{n}_{k=0} Pk \; BEZ_{k,n}(u)$ ……. $0 \le u \le 1$.

The Bernstein polynomials:

$BEZ_{k,n}(u) = C(n,k)\, u^{k} (1-u)$

C(n,k)= binomial coefficients.

$C(n,k)=n!/ k! (n-k)!$

$BEZ_{k,n}(u) = (1-u)\, BEZ_{k,n-1}(u) + u\, BEZ_{k-1,n-1}(u)$ ……. $n > k \ge 1$

$X(u)= \sum^{n}_{k=0} X_k \; BEZ_{k,n}(u)$

$Y(u)= \sum^{n}_{k=0} Y_k \; BEZ_{k,n}(u)$

3 points :	Generate a parabola
4 points :	A cubic curve
3 collinear control points :	Generate a straight line segment

Why use?

1. Easy to implement
2. Reasonably powerful in curve design.
3. Efficient methods for determining coordinate positions along a Bezier curve can be set up using recursive calculations.

   $C(n,k)=((n-k+1)/k)\; C(n,k-1)$ ……. $n \ge k$

   .

Properties:

1. Bezier curves are always passes through the first and last control points.
2. The slop at the beginning of the curve is along the line joining the first two control points and the slop at the end of the curve is along the line joining the last two end points.
3. It lies within the convex hull of the control points.

Cubic Bezier Curves.

It gives reasonable design flexibility while avoiding the increased calculations needed with higher order polynomials.

$BEZ_{0,3}(u) = (1-u)^3$
$BEZ_{1,3}(u) = 3u(1-u)^2$
$BEZ_{2,3}(u) = 3u^2(1-u)$
$BEZ_{3,3}(u) = u^3$

At u=0 and u=1 only non zero blending function is $BEZ_{0,3}$ and $BEZ_{3,3}$ respectively. Thus, the cubic curve will always pass through control points P0 and P3

The $BEZ_{1,3}$ and $BEZ_{2,3}$ influence the shape of the curve at intermediate values of parameter u, so that the resulting curve tends toward points p1 and p2. $BEZ_{1,3}$ is max at u=1/3
$BEZ_{2,3}$ is max at u=2/3.

Bezier curves do not allow for local control of the curve shape. If we reposition any one of the control points, the entire curve will be affected.

9.8 B-SPLINE CURVES

B-splines are not used very often in 2D graphics software but are used quite extensively in 3D modeling software. They have an advantage over Bezier curves in that they are smoother and easier to control. B-splines consist entirely of smooth curves, but sharp corners can be introduced by joining two spline curve segments. The continuous curve of a b-spline is defined by control points.

The equation for k-order B-spline with n+1 control points $(P_0, P_1, ... P_n)$ is $P(t) = \sum_i = 0, n\ N_{i,k}(t) P_i, t_{k-1} <= t <= t_{n+1}$.

In a B-spline each control point is associated with a basis function $N_{i,k}$ which is given by the recurrence relations B-spline basis functions as like as Bezier ones are nonnegative $N_{ik} >= 0$ and have "partition of unity" property $\sum i = 0, n\ N_{i,k}(t) = 1, t_{k-1} < t < t_{n+1}$ therefore $0 <= N_{i,k} <= 1$

As since $N_{ik} = 0$ for $t <= t_j$ or $t >= t_{j+k}$ therefore a control point P_i influences the curve only for $t_i < t < t_{i+k}$.

The main properties of B-splines
- composed of (n-k+2) Bezier curves of k-order joined C^{k-2} continuously at knot values (t_0 , t_1 , ... , t_{n+k})
- each point affected by k control points
- each control point affected k segments
- inside convex hull
- affine invarianc

- uniform B-splines don't interpolate deBoor control points ($\mathbf{P}_0$, $\mathbf{P}_1$, ... , $\mathbf{P}_n$)

Check your Progress:
True or False.
1. Bezier curve is easy to implement.
2. The continuous curve of a b-spline is defined by control points.

9.9 DIFFERENCE BETWEEN BEZIER CURVES AND B-SPLINE CURVES

Bezier Curve	B-Spline Curve
Bezier curves do not need knots	To construct B-splines one needs to specify knots
Bezier curve automatically clams its endpoints.	B-Splines do not interpolate any of its control points.
Bezier curve basis functions are easier to compute.	B-Spline curve requires more computations

Table 9.1

9.10 FRACTALS AND ITS APPLICATIONS

Introduction:

Fractal Geometry has found its applications not only animations or film industries, creating beautiful natural objects , but also in Biological study, Medicines, Telecommunications, Fluid Mechanics, Image Compression, Fractal Music etc.

“Art is the creation of mind, an idea. A painting, a piece of music or a sculpture is only the embodiment of that idea. The idea that nature and mathematics are inextricably linked can be very well proved by using fractal geometry in Computer Graphics. Natural objects can be realistically described by fractal geometry methods.

Fractals can be seen as mysterious expressions of beauty representing exquisite preordained shapes that mimic the universe. Art and science will eventually be seen to be as closely connected as arms to the body. Both are vital elements of order and its discovery. But when art is seen as the ability to do, make, apply, or portray in way that withstands the test of time, its connection with science becomes clearer.

Applications of Fractal Geometry:

Nature:

Fractals have become immensely popular for describing and creatin natural objects like mountains, clouds, flames, etc. that cannot be described in terms of mathematical geometry using triangles or squares. In Computer Graphics, modeling techniques generally assume that an object is a collection of lines or polygons or that it can be described by higher order polynomials e.g. Bezier or B-Spline curves. While these techniques efficiently model solid objects like cars, roads, houses etc. they are not well adapted to representation of natural object features like terrains, snow, smoke, etc.

The Fractal Fern
The "fractal fern" is generated completely by fractals. This is not a digital photograph - it is completely computer-generated.

Fig.9.8 The Fractal Fern

Animations & movies:

The application of fractal has mostly been in the field of animations, motion pictures and visualizations. E.g.: Real popular application is in form of Imaginary Landscapes of outer space in STAR TREK, STAR WARS. Here UCLA mathematicians along with Hollywood filmmakers used the 3D fractals beautifully and created landscapes which looked very real; but were non-existent. Fractal images are used as an alternative to costly elaborate sets to produce fantasy landscapes.

Bacteria Cultures:

Some of the most amazing applications of fractals can be found in such distant areas as the shapes of bacteria cultures. A bacteria culture is all bacteria that originated from a single ancestor and are living in the same place. When a culture is growing, it spreads outwards in different directions from the place where the original organism was placed. Just like **plants** the spreading bacteria can branch and form patterns which turn out to be fractal. The spreading of bacteria can be modeled by fractals such as the **diffusion** fractals, because bacteria spread similarly to nonliving materials.

Biological systems:

Fractal and chaos phenomena specific to non-linear systems are widely observed in biological systems. A study has been established an analytical method based on fractals and chaos theory for two patterns: the dendrite pattern of cells during development in the cerebellum and the firing pattern of intercellular potential. Variation in the development of the dendrite stage was evaluated with fractal dimension, enabling the high order seen there to be quantized

Origin of Fractals:

With the aid of computer graphics, Mandelbrot who then worked at IBM's Watson Research Center was able to show how Julia's work is a source of some of the most beautiful fractals known today. By iterating a simple equation and mapping this equation in the complex plane, Mandelbrot discovered the fractal named after his name, Mandelbrot Set. He has been initially responsible for extending the theory and graphics representation of iterated functions as a special class of new geometry as “Fractal Geometry”.

The fact that any small part of the coast will look similar to the whole thing was first noted by Benoit Mandelbrot. He called shapes like this fractals. In nature, you can find them everywhere. Any tree branch, when magnified, looks like the entire tree. Any rock from a mountain looks like the entire mountain. The theory of fractals was first developed to study nature. Now it is used in a variety of other applications. And, of course, beauty is what makes them popular! And now fractal geometry is providing us with a new perspective to view the world, creating real life landscapes, to data compression, music etc.

Advantages of using Fractal Geometry:

Fractal was coined in 1975, by mathematician Benoit Mandelbrot to describe an intricate looking set of curves, many of which were never seen before the advent of computers, because of its ability to perform quickly massive calculations.

Fractals are figures with an infinite amount of detail. When magnified, they don’t become more simple, but remain as complex as they were without magnification.

The modeling and rendering time required is minimal compared with traditional methods. In this respect our new approach is comparable to the old one, though it’s slower for large, complex scenes.

L-system can be considered as a compression of the film with a factor typically bigger than one million. In general, a production system needs not to build up a 3D database to the complete environment. It can directly draw objects during the interpretation of a symbolic environment string.

In the case of fractals, few properties are:

- a fine structure,
- too much irregularity to be described in traditional geometric language, both locally and globally,
- some form of self-similarity, perhaps approximate or statistical,
- a "fractal dimension"(somehow defined) which is greater than its topological dimension, and
- A simple definition, perhaps recursive.

Classification of IFS and complex fractals:

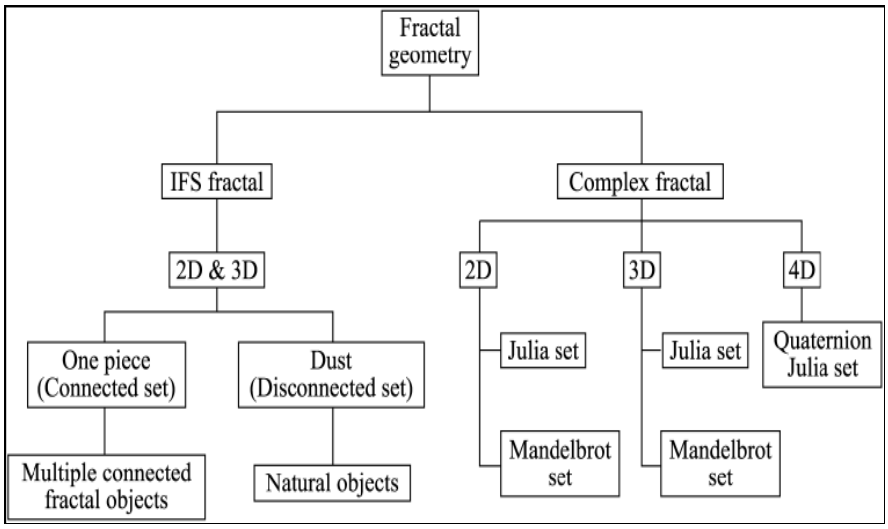


Fig. 9.9

The term “fractal” was introduced for characterizing spatial or temporal phenomena that are continuous but not differentiable (Mandelbrot, 1975).

Fractal is defined as a rough or fragmented geometric shape that can be sub-divided into parts, each of which is (at least approximately) a reduced size copy of the whole. Mathematically, a fractal is defined as a set of points whose fractal dimension exceeds its topological dimension (Mandelbrot, 1983).

In general, the dimension of a fractal is typically a non-integer or a fraction, meaning its dimension is not a whole number and its formation is by an iteration (or recursive) process, and

hence has non integer complexity. When a fractal is magnified, it is infinitely complex. Moreover, upon magnification of a fractal, it can be found that subsets of the fractal resemble the whole fractal, i.e. self-similar.

There are two types of fractal geometries – iterative function system (IFS) fractal and complex fractal. The classification of these two types of fractals is shown in **Classification of IFS and complex fractals**. In general, an IFS fractal is a family of specified mappings that map the whole onto the parts and the iteration of these mapping will result in convergence to an invariant set. There are numerous literatures about IFS and readers can find the details from them (Moran, 1946; Williams, 1971; Hutchinson, 1981; Barnsley and Demko, 1985; Barnsley, 1988).

Fractal geometry and fractal dimension:

Fractal dimension is a measure of how 'complicated' a self-similar figure is. In a rough sense, it measures 'how many points' lie in a given set. A plane is 'larger' than a line. Somehow, though, fractal dimension captures the notion of 'how large a set is'.

Fractal geometry can be considered as an extension of Euclidean geometry.

Conventionally, we consider integer dimensions which are exponents of length, i.e. surface = length^2 or volume = length^3. The exponent is the dimension. Fractal geometry allows for there to be measures which change in a non-integer or fractional way when the unit of measurements changes. The governing exponent *D* is called *fractal dimension*.

Fractal object has a property that more fine structure is revealed as the object is magnified, similarly like morphological complexity means that more fine structure (increased resolution and detail) is revealed with increasing magnification. Fractal dimension measures the rate of addition of structural detail with increasing magnification, scale or resolution. The fractal dimension, therefore, serves as a *quantifier of complexity*.

Ideal points have Euclidean dimension of 0, ideal lines of 1, and perfectly flat planes of 2. However, collection of real points have dimension greater than 0, real lines greater than1, real surfaces greater than 2, etc. At each level, as the dimensions of an object move from one integer to the next, the complexity of the object increases. Euclidean or non-fractal (points, lines, circles, cubes, etc.) may be viewed as fractal objects with the lowest complexity (integer fractal dimensions) within their respective dimension domains (0 to 1, 1 to 2, etc.). Natural objects are often

rough and are not well described by the ideal constructs of Euclidian geometry .One familiar example of naturally occurring fractal curves is coastline. Since all of the curve's features that are smaller than the size of the measuring tool will be missed, whatever is the size of the measuring tool selected, therefore the result obtained depends not only on the coastline itself but also on the length of the measurement tool. The use of fractional power in the measurements compensates for the details smaller than the size of measuring tool – fractal dimension is the unique fractional power that yields consistent estimates of a set's metric properties. Because it provides the correct adjustments factor for all those details smaller than the measuring device, it may also be viewed as a measurement of the shape's roughness. The fractal dimension of an object provides insight into how elaborate the process that generated the object might have been, since the larger the dimension the larger the number of degrees of freedom.

Euclidean Geometry Vs Fractal Geometry:

- In mathematics, Euclidean geometry was known more than 20000 years before sometimes means geometry in the plane which is also called plane geometry. Euclidean geometry in three dimensions is traditionally called solid geometry.

  Fractal Geometry is a modern technique, known just 20 years before.

- Using Euclidean Geometry methods (e.g. B-Spline or Bezier curves) object shapes are described with equations which are adequate or accurate for describing manufactured objects, i.e. those that have smooth surfaces and regular shapes. Natural objects such as mountains, tree, have irregular or fragmented features and Euclidean methods do not realistically model these objects.

  Natural objects can be realistically described with Fractal Geometry methods, where recursive procedures rather than equations are used to model these objects. With only a small amount of code and data you can generate highly detailed, complex scenes.

- Euclidean Geometry methods use simple algebraic formulas to draw various shapes which do not exhibit self-similarity on magnification. E.g. sphere on magnification yields a flat plane; hence it does not exhibit self-similarity. Hence it can be best described by Euclidean Geometry.

Fractals are self-similar and independent of scaling, i.e. various copies of an object can be found in the original object at similar size scales.

Classification of fractals:

Fractals can also be classified according to their self-similarity. There are three types of self-similarity found in fractals:

Exact self-similarity — this is the strongest type of self-similarity; the fractal appears identical at different scales. Fractals defined by iterated function systems often display exact self-similarity. In many fractals, self-similarity is very obvious. For example, it is clearly seen in the picture right. Each of these fractals is composed of smaller versions of itself. When magnified, they turn out to be identical to the entire picture. e.g. Sierpinski triangle, Koch Curve etc.

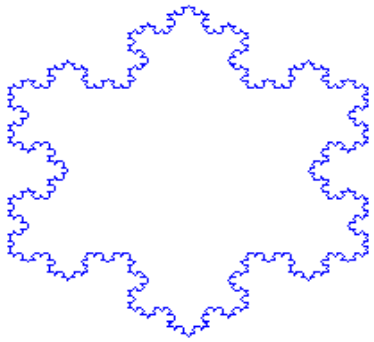


Fig. 9.10 :Koch Curve

- **Quasi-self-similarity** — this is a loose form of self-similarity; the fractal appears approximately (but not exactly) identical at different scales. Quasi-self-similar fractals contain small copies of the entire fractal in distorted and degenerate forms. Fractals defined by recurrence relations are usually quasi-self-similar but not exactly self-similar. For example, the famous Mandelbrot Set doesn't exhibit exactly identical pictures right away. However, on magnification, small versions of it at all levels can be found.

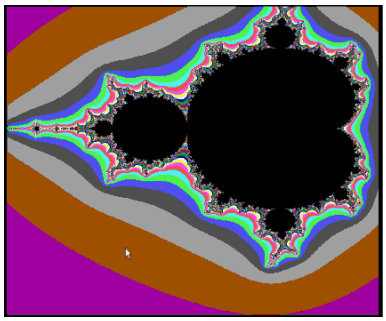


Fig. 9.11 :Mandelbrot Set

- **Statistical self-similarity** — this is the weakest type of self-similarity; the fractal has numerical or statistical measures which are preserved across scales. Most reasonable definitions of "fractal" trivially imply some form of statistical self-similarity. (Fractal dimension itself is a numerical measure which is preserved across scales.) Random fractals are examples of fractals which are statistically self-similar, but neither exactly nor quasi-self-similar. In perfectly self-similar object like Koch Curve, no matter how far it is magnified, we get exactly similar fragmented picture. Compared to a straight line, the Koch Snowflake is obviously better in describing a natural shape such as a coastline or a river. However, there is a major drawback to that, i.e. it is perfectly symmetrical shape. Obviously, normally self-similar fractals are too regular to be realistic. Fractals with statistical self-similarity have parts with different scaling parameters in different coordinate directions. This also may include random variations. Hence to make fractals more realistic, a different type of self-similarity called *statistical self-similarity* is used. E.g. Plot the location of some particle at certain intervals of time; to get a fragmented trajectory with lines randomly located in space.

Now, take one of these lines and plot locations at smaller intervals of time. It is observed that a smaller fragmented line made up of randomly located parts exists. If one of these lines is taken, it is found that it is made up of smaller lines as well. However, this self-similarity is different. Although each line is composed of smaller lines, the lines are random instead of being fixed.

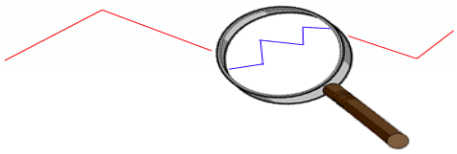


Fig. 9.12 :Statistical Self-Similarity

Statistical self-similarity is found in plasma fractals**.** They are very useful in creatin realistic coastlines and landscapes**.**

- **Mathematical fractals - self-similarity**

Objects considered in Euclidean geometry are sets embedded in Euclidean space and object's dimension is the dimension of the embedding space. One is also accustomed to associate what is called *topological dimension* with Euclidean objects - everybody knows that a point has dimension of 0, a line has dimension of 1, a square is 2 dimensional, and a cube is 3-

dimensional. The topological dimension is preserved when the objects are transformed by a homeomorphism. One cannot use topological dimension for fractals, but instead has to use what is called Hausdorff-Besikovitch dimension, commonly known as *fractal dimension*. In fact, a *formal definition of a fractal* says that it is an object for which the fractal dimension is greater than the topological dimension. But this definition is too restrictive. An alternative definition uses the concept of self-similarity – a fractal is an object made of parts similar to the whole. The notion of self-similarity is the basic property of fractal objects. Taking advantage of self-similarity is one way (called *similarity method*) to calculate fractal dimension.

For example, one can subdivide a line segment into *m* self-similar intervals, each with the same length, and each of which can be magnified by a factor of *n* to yield the original segment. A square or a triangle may be subdivided into n^2 self-similar copies of itself, each of which must be magnified by a factor of *n* to yield the original object. Similarly, a cube can be decomposed into n^2 self-similar copies of itself, each of which must be magnified by a factor of *n* to yield the original cube (Table 1). If one takes the magnification, *n* , and raise it to the power of dimension, *D* , one will get the number of self-similar pieces in the original object, *P*: $P = n^D$

Solving this equation for *D* one easily finds that

$D = \log(P) / \log(n)$

Using this formula one can calculate fractal dimension of some fractals.

Object	**Dimension**	**No. of Copies**
Line	1	$2 = 2^1$
Square	2	$4 = 2^2$
Cube	3	$8 = 2^3$
Any self-similar figure	D	$P = 2^D$
Sierpinski triangle	1.58	$3 = 2^D$

Table 9.2

A mathematical fractal has some infinitely repeating pattern and can be made by the iteration of a certain rule.

Fractal Curves :

Koch curve

A variant of the Koch curve which uses only right-angles.

Variables: F
Constants: + −
Start: F
Rules: (F → F+F−F−F+F)

Here, *F* means "draw forward", + means "turn left 90°", and - means "turn right 90°" (see turtle graphics).

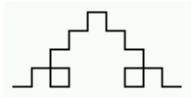
n = 0:

F

n = 1:

F+F-F-F+F

n = 2:



F+F-F-F+F+F+F-F-F+F-F+F-F-F+F-F+F-F-F+F+F+F-F-F+F

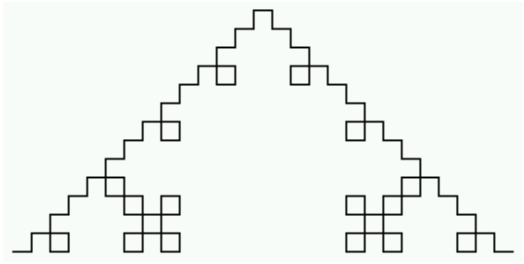


Fig 9.13 *n* = 3:

F+F-F-F+F+F+F-F-F+F-F+F-F-F+F-F+F-F-F+F+F+F-F-F+F+ F+F-F-F+F+F+F-F-F+F-F+F-F-F+F-F+F-F-F+F+F+F-F-F+F- F+F-F-F+F+F+F-F-F+F-F+F-F-F+F-F+F-F-F+F+F+F-F-F+F- F+F-F-F+F+F+F-F-F+F-F+F-F-F+F-F+F-F-F+F+F+F-F-F+F+ F+F-F-F+F+F+F-F-F+F-F+F-F-F+F-F+F-F-F+F+F+F-F-F+F

Sierpinski triangle

The Sierpinski triangle has drawn using an L-system.

Variables: A B
Constants: + −
Start: A
Rules: (A → B−A−B),(B → A+B+A)
Angle: 60º

Here, A and B mean both "draw forward", + means "turn left by angle", and - means "turn right by angle" (see turtle graphics). The angle changes sign at each iteration so that the base of the triangular shapes are always in the bottom (they would be in the top and bottom, alternatively, otherwise).

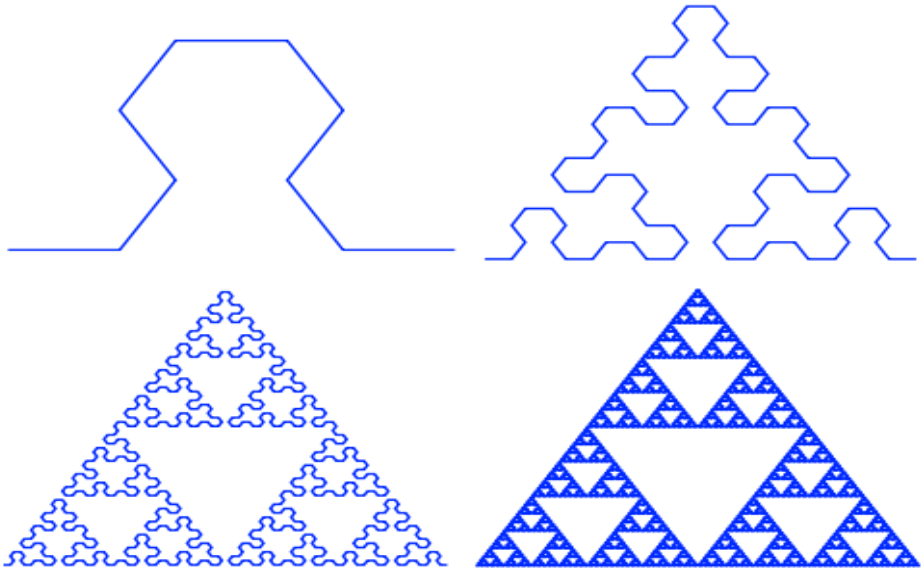


Fig.9.14 Evolution for *n* = 2, *n* = 4, *n* = 6, *n* = 9

Dragon curve

The Dragon curve drawn using an L-system.

Variables: X Y F
Constants: + −
Start: FX
Rules: (X → X+YF+),(Y → -FX-Y)
Angle: 90º

Here, F means "draw forward", - means "turn left 90°", and + means "turn right 90°". X and Y do not correspond to any drawing action and are only used to control the evolution of the curve.

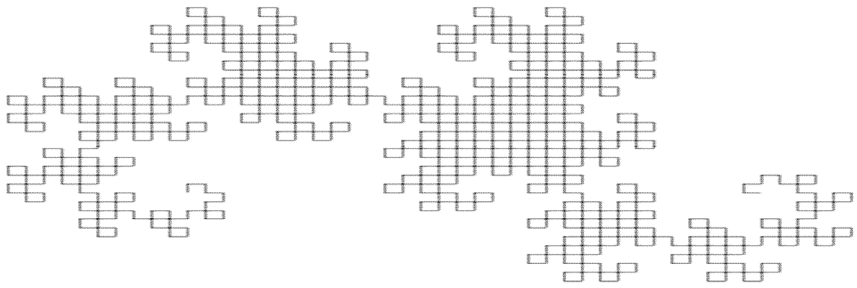


Fig 9.15 Dragon curve for *n* = 10

Fractal plant

Variables: X F
Constants: + −
Start: X
Rules: (X → F-[[X] +X] +F [+FX]-X), (F → FF)
Angle: 25º

Here, F means "draw forward", - means "turn left 25º", and + means "turn right 25º". X does not correspond to any drawing action and is used to control the evolution of the curve. **[**Corresponds to saving the current values for position and angle, which are restored when the corresponding] is executed.
Fractal plant for *n* = 6

Fig .9.16 Fractal plant

Check your Progress:

1. List different types of fractal curves.
2. What are the types of s3lf similarity in fractals?

9.11 LET US SUM UP

- The order of the curve determines the minimum number of control points necessary to define the curve.
- Circles and ellipses are special cases of a class of curves known as conics.
- Bezier Curves are Easy to implement and reasonably powerful in curve design.
- Bezier curves do not allow for local control of the curve shape. If we reposition any one of the control points, the entire curve will be affected.
- Fractals are known for a fine structure, too much irregularity to be described in traditional geometric language, both locally and globally, some form of self-similarity, perhaps approximate or statistical, and a "fractal dimension"(somehow defined) which is greater than its topological dimension, and a simple definition, perhaps recursive.

9.13 REFERENCES AND SUGGESTED READING

- Computer Graphics,A. P. Godase, Technical Publications Pune.
- Mathematical Elements of Computer Graphics, David F. Rogers, Tata McGraw Hill.
- Computer Graphics, Donald Hearn, M P. Baker, PHI.

9.12 EXERCISE

1. Give the properties of B-Spline curve.
2. What is cubic Bezier curve? Explain in detail.
3. Give the applications of fractal geometry.
4. What are fractals?
5. Write a short note on:

a) Parametric curve design
b) Conic curve
c) Cubic

10

SURFACE DESIGN AND VISIBLE SURFACES

Unit Structure:

10.0 OBJECTIVE

The objective of this chapter is

- To understand the different types of surfaces.
- To understand concept of solid geometry.

- To understand concept of visible and hidden surfaces and different algorithms to find and remove the hidden surfaces.

10.1 INTRODUCTION

When objects are to be displayed with color or shaded surface, we apply surface- rendering procedures to the visible surfaces so that the hidden surfaces are obscured. Some visible-surface algorithms establish visibility pixel by pixel across the viewing plane, other determine visibility for object surface as a whole. By removing the hidden lines we also remove information about the shape of the back surfaces of an object.

10.2 TYPES OF SURFACES

- Bilinear Surfaces
- Ruled Surfaces
- Developable surfaces
- Coon patch
- Sweep surfaces
- Surface of revolution
- Quadratic surfaces

10.2.1 BILINEAR SURFACES

A flat polygon is the simplest type of surface. The bilinear surface is the simplest non flat (curved) surface because it is fully defined by means of its four corner points. It is discussed here because its four boundary curves are straight lines and because the coordinates of any point on this surface are derived by linear interpolations. Since this patch is completely defined by its four corner points, it cannot have a very complex shape. Nevertheless it may be highly curved. If the four corners are coplanar, the bilinear patch defined by them is flat. Let the corner points be the four distinct points P00, P01, P10, and P11. The top and bottom boundary curves are straight lines and are easy to calculate.
They are P(u, 0) =(P10 − P00)u + P00 and P(u, 1) =(P11 − P01)u + P01.

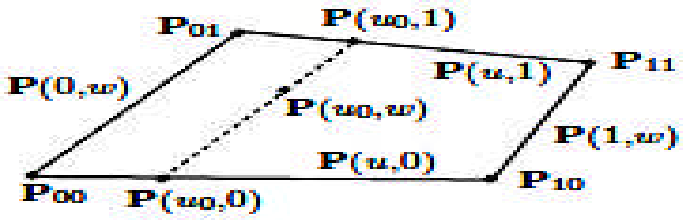


Fig. 10.1

To linearly interpolate between these boundary curves, we first calculate two corresponding points **P**(u0, 0) and **P**(u0, 1), one on each curve, then connect them with a straight line **P**(u0, w). The two points are **P**(u0, 0) = (**P**10 − **P**00)u0 + **P**00 and **P**(u0,1) = (**P**11 − **P**01)u0 + **P**01, and the straight segment connecting them is

P(u0, w) = (**P**(u0, 1) − **P**(u0, 0))w + **P**(u0, 0)
= [(**P**11 − **P**01)u0 + **P**01 −(**P**10 − **P**00)u0 + **P**00]w+ (**P**10 − **P**00)u0 + **P**00.

The expression for the entire surface is obtained when we release the parameter u from its fixed value u0 and let it vary. The result is:

P(u,w) = **P**00(1 − u)(1 − w) + **P**01(1 − u)w + **P**10u(1 − w) + **P**11uw

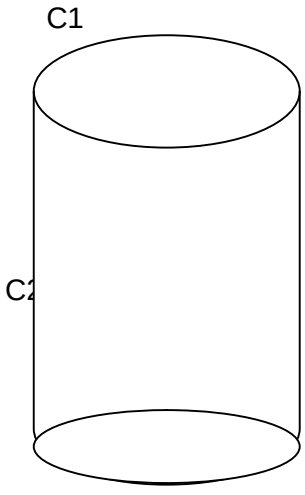
$$= \sum_{i=0}^{1}\sum_{j=0}^{1} B_{11}(u)\,P_{ij}B_{1j}(\omega),$$

$$= \left[B_{10}(u), B_{11}(u)\right]\begin{bmatrix} P_{00} & P_{01} \\ P10 & P_{11}\end{bmatrix}\begin{bmatrix} B_{10}(\omega) \\ B_{11}(\omega)\end{bmatrix}$$

where the functions B1i(t) are the Bernstein polynomials of degree 1. This implies that the bilinear surface is a special case of the rectangular Bezier surface. Mathematically, the bilinear surface is a hyperbolic paraboloid.

10.2.2 RULED SURFACES

Given two curves $\mathbf{C_1}(u)$ and $\mathbf{C_2}(v)$, the ruled surface is the surface generated by connecting line segments between corresponding points, one on each given curve. if t is a value in the domain [0,1] of both curves, a segment between $\mathbf{C_1}(t)$ and $\mathbf{C_2}(t)$ is constructed. This segment is usually referred as a **ruling** at t. As t moves from 0 to 1, the ruling at t sweeps out a surface and this is the ruled surface defined by curves $\mathbf{C_1}(u)$ and $\mathbf{C_2}(v)$. Cylinder is another well-known ruled surface. It is generated from two circles



10.2.3 DEVELOPABLE SURFACES

A developable surface is a surface that can be (locally) unrolled onto a flat plane without tearing or stretching it. If a developable surface lies in three-dimensional Euclidean space, and is complete, then it is necessarily ruled, but the converse is not always true. For instance, the cylinder and cone are developable, but the general hyperboloid of one sheet is not. More generally, any developable surface in three dimensions is part of a complete ruled surface, and so itself must be locally ruled.

10.2.4 COONS PATCH

The Coons patch is constructed from four intersecting curves. Consider a pair of such curves that intersect at a corner **P**ij of the Coons patch. We can employ this pair and the corner to construct a translational surface **P**ij(u,w). Once we construct the four translational surfaces for the four corners of the Coons patch, they can be used to express the entire Coons linear surface patch

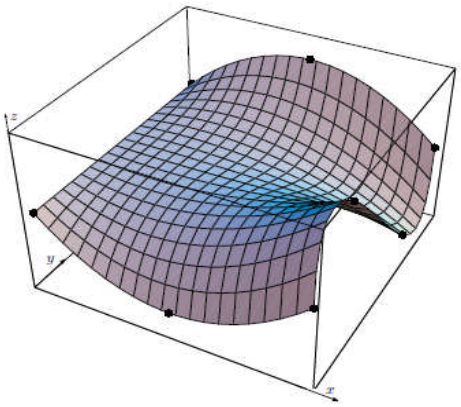


Fig.10.3

10.2.5 SWEEP SURFACES

In sweep representation the 3D objects are created from 2 D shapes. The 2D shape is swept in 2D directions. A) Translation B) Rotational

A) Translational sweep

In translational sweep we replicate a 2D shape and draw the set of connecting lines to produce a 3D object as shown in the fig.

Fig. 10.4

B) Rotational sweep

In rotational sweep 2 D shape or curve is rotated about the axis of rotation to produce 3D object. In general for sweep representation we are allowed to use any path curve and 2D shape. For rotation sweep are allow to rotate 0^0 to 360^0 . Where the translation sweep and rotational sweep is combined to get a 3D object. The corresponding sweep representation is called as general sweep.

10.2.6 SURFACE OF REVOLUTION

A surface of revolution is generated by revolving a given curve about an axis. The given curve is a profile curve while the axis is the axis of revolution. To design a surface of revolution, select Advanced Features followed by Cross Sectional Design. This will bring up the curve system. In the curve system, just design a profile curve based on the condition to be discussed below, and then select Techniques followed by Generate Surface of Revolution. The surface system will display a surface of revolution defined by the given profile curve. Some special restrictions must be followed in order to design a surface of revolution under the curve system. First, the axis of revolution must be the z-axis. Second, the profile curve must be in the xz-plane. However, when brining up the curve system, only the xy-plane is shown. To overcome this problem, one can design a profile curve on the xy-plane and rotate the curve (not the scene) about the x-axis 90 degree (or -90 degree, depending on your need). In this way, the profile curve will be placed on the xz-plane. Many commonly seen and useful surfaces are surfaces of revolution (e.g., spheres, cylinders, cones and tori).

10.2.7 QUADRIC SURFACES

A quadratic surface is a surface in space defined by a quadratic equation:

f(x; y; z) j x2 + y2 = 1g Cylinder
f(x; y; z) j x2 + y2 + z2 = 1g Sphere
f(x; y; z) j x2 + 2xy + y2 + z2 □ 2z = 5g ??

Advantages are as follows:
1. Build 3-dimensional intuition.
2. Techniques useful for contour plots, which you will see more.
3. These surfaces are useful.
4. Will see some of them later in the course.

Basic types of quadratic surfaces
Let’s try some more surfaces.

1. f(x; y; z) j $x^2 + y^2/2 + z^2/3 = 1$ g: An ellipsoid
2. f(x; y; z) j $x^2 + y^2 - z^2 = 1$ g: A hyperboloid of one sheet
3. f(x; y; z) j $x^2 + y^2 - z^2 = -1$ g: A hyperboloid of two sheets
4. f(x; y; z) j $x^2 + y^2 - z^2 = 0$ g: A cone
5. f(x; y; z) j $z = x^2 + y^2$ g: An (elliptic) paraboloid
6. f(x; y; z) j $z = x^2 - y^2$ g: A hyperbolic paraboloid

Check your Progress:

1. What do you mean by developable surface?
2. Define: Translational Sweep and Rotational Sweep.

10.3 CONSTRUCTIVE SOLID GEOMETRY

Constructive solid geometric representation

Here, we combine 3D objects using set operations such as, union, intersection and many more. Here, the 3D object is defined by a volume function and 2 or more volumes are combined to generate a new shape or a new object.

Fig. 10.5

In CSG method we can use objects such as blocks, cylinders, pyramids, cones, spheres, splines.

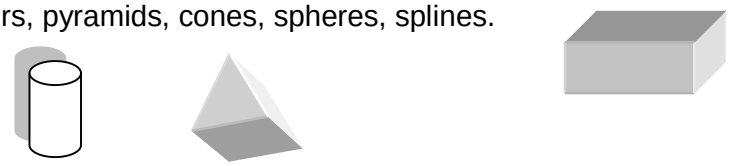


Fig. 10.6

A good graphic package always provides these 3D shapes and the different operations and transformations that can be applied.

10.3.1 BEZIER SURFACES

To create a Bezier surface, we blend a mesh of Bezier curves using the blending function $P(u,v) = \sum_{j-0}^{m} \sum_{k-0}^{n} P_{j,k} BEZ_{j,m}(v) BEZ_{k,n}(u)$ where j and k are points in parametric space and $P_{x,y}$ represents the location of the knots in real space. The Bezier functions specify the weighting of a particular knot. They are the Bernstein coefficients. The definition of

the Bezier functions is $BEZ_{k,n}(u) = C(n,k)\,u^k\,(1-u)^{u-k}$. Where $C(n,k)$ represents the binary coefficients. When u=0, the function is one for k=0 and zero for all other points. When we combine two orthogonal parameters, we find a Bezier curve along each edge of the surface, as defined by the points along that edge. Bezier surfaces are useful for interactive design and were first applied to car body design.

10.3.2 BSPLINE SURFACES

The equation of a B-spline surface follows directly from the equation of a B-spline curve. This relationship is analogous to that between Bezier curves and surfaces. Furthermore, we define the B-spline surface, like the Bzier surface, in terms of a characteristic polyhedron. The shape of the surface approximates the polyhedron. The approximation is weaker the higher the degree.

The tensor product equation of the B-spline surface is $P(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}(u)\,N_{j,q}(v)\,P_{i,j}$ Where $N_{i,p}(u)$ and $N_{j,q}(v)$ are B-spline basis functions of degree p and q, respectively. Note that the **fundamental identities**, one for each direction, must hold: h = m + p + 1 and k = n + q + 1. Therefore, a B-spline surface is another example of tensor product surfaces. As in Bezier surfaces, the set of control points is usually referred to as the **control net** and the range of u and v is 0 and 1. Hence, a B-spline surface maps the unit square to a rectangular surface patch.

10.3.3 SUBDIVISION SURFACES

Subdivision surfaces are defined recursively. The process starts with a given polygonal mesh. A **refinement scheme** is then applied to this mesh. This process takes that mesh and subdivides it, creating new vertices and new faces. The positions of the new vertices in the mesh are computed based on the positions of nearby old vertices. In some refinement schemes, the positions of old vertices might also be altered (possibly based on the positions of new vertices).

This process produces a denser mesh than the original one, containing more polygonal faces. This resulting mesh can be passed through the same refinement scheme again and so on.

The limit subdivision surface is the surface produced from this process being iteratively applied infinitely many times. In practical use however, this algorithm is only applied a limited number of times.

Check your Progress:

Fill in the Blanks.
1. Subdivision surfaces are defined _____.
2. A B-spline surface maps the unit square to a _____surface patch.

10.4 INTRODUCTION TO VISIBLE AND HIDDEN SURFACES

In graphics, a picture /scene consists of objects which are nearer to the “eye“ or block. The other objects are block, some part of the other objects.

The part which cannot be visible needs to be removed to give a realistic effect.

The problem of removing the unwanted surfaces and line is called “hidden surface problem”. There are two methods used to solve hidden surface problem:

- Image- Space Method
- Object –Space Method

Image - Space Method
- Here, we compare every object within the scene with pixel location on the screen
- Here, we apply screen coordinates with above mention conversion.
- If n is the number of objects and N is the number of pixels, then the computational work increases proportional to the product .
- The method dependent on the precession of the screen representation.
- The objects enlarge themselves significantly and do not give correct results
- Eg. Z- Buffer Algorithms.

10.5 COHERENCE FOR VISIBILITY

Coherence is the result of local similarity which can be calculated using the results calculated for one part of the scene or image for other nearby parts as objects have continuous spatial extent, object properties vary smoothly within a small local region in the scene. Calculations can then be made incremental.

Types of coherence:

1. **Object Coherence:** Visibility of an object can often be decided by examining a circumscribing solid (which may be of simple form, eg. A sphere or a polyhedron.)

2. **Face Coherence:** Surface properties computed for one part of a face can be applied to adjacent parts after small incremental modification. (eg. If the face is small, we sometimes can assume if one part of the face is invisible to the viewer, the entire face is also invisible).

3. **Edge Coherence:** The Visibility of an edge changes only when it crosses another edge, so if one segment of an nonintersecting edge is visible, the entire edge is also visible.

4. **Scan line Coherence:** Line or surface segments visible in one scan line are also likely to be visible in adjacent scan lines. Consequently, the image of a scan line is similar to the image of adjacent scan lines.

5. **Area and Span Coherence:** A group of adjacent pixels in an image is often covered by the same visible object. This coherence is based on the assumption that a small enough region of pixels will most likely lie within a single polygon. This reduces computation effort in searching for those polygons which contain a given screen area (region of pixels) as in some subdivision algorithms.

6. **Depth Coherence:** The depths of adjacent parts of the same surface are similar.

7. **Frame Coherence:** Pictures of the same scene at successive points in time are likely to be similar, despite small changes in objects and viewpoint, except near the edges of moving objects.

Most visible surface detection methods make use of one or more of these coherence properties of a scene. To take advantage of regularities in a scene, eg. Constant relationships often can be established between objects and surfaces in a scene.

10.6 EXTENTS AND BOUNDING VOLUMES

Bounding volume for a set of objects is a closed volume that completely contains the union of the objects in the set. Bounding volumes are used to improve the efficiency of geometrical operations by using simple volumes to contain more complex objects. Normally, simpler volumes have simpler ways to test for overlap. A bounding volume for a set of objects is also a bounding

volume for the single object consisting of their union, and the other way around. Therefore it is possible to confine the description to the case of a single object, which is assumed to be non-empty and bounded (finite). Bounding volumes are most often used to accelerate certain kinds of tests. In ray tracing, bounding volumes are used in ray-intersection tests, and in many rendering algorithms, they are used for viewing frustum tests. If the ray or viewing frustum does not intersect the bounding volume, it cannot intersect the object contained in the volume. These intersection tests produce a list of objects that must be displayed. Here, displayed means rendered or rasterized. In collision detection, when two bounding volumes do not intersect, then the contained objects cannot collide, either.

Common types of bounding volume
A **bounding sphere** is a sphere containing the object. In 2-D graphics, this is a circle.

A **bounding ellipsoid** is an ellipsoid containing the object. Ellipsoids usually provide tighter fitting than a sphere.

A **bounding cylinder** is a cylinder containing the object.

A **bounding capsule** is a swept sphere (i.e. the volume that a sphere takes as it moves along a straight line segment) containing the object

A **bounding box** is a cuboid, or in 2-D a rectangle containing the object

Check your Progress:

True or False.
1. In ray tracing, bounding volumes are used in ray-intersection tests.
2. Coherence is the result of global similarity.

10.7 BACK FACECULLING

Back face culling is when you do not render a polygon because its normal is pointing away from the viewer's eye point. A naive way of preforming this check would be to construct a vector from the eye point to a point on the face and compute the dot product of this line of sight vector and the normal vector. If this scalar is positive then the face is pointing away and consequently can be culled. A more sophisticated way of making this check is to transform the face's normal into the projection space, where the line of sight vector is always the negative Z-axis. Then the check for

back face culling is just to see if the Z component of the transformed normal is negative.

Back-face culling directly eliminates polygons not facing the viewer.

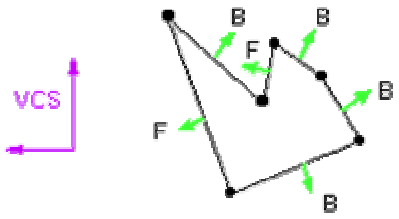


Fig. 10.7

Back-Face Culling in VCS

A first attempt at performing back-face culling might directly use the z-component of the surface normal, as expressed in VCS. This does not always work, however a better strategy is to construct the plane equation for the polygon and to test whether the eye-point falls above or below this plane. Plane(Peye)<0 implies the eyepoint is below the plane containing the polygon and that the polygon should thus be culled

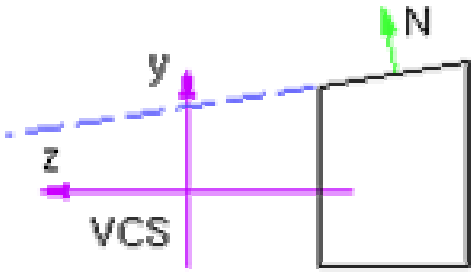


Fig. 10.8

Steps for VCS culling

- Calculate a surface normal, N = (A,B,C). This need not be normalized.
- Compute D in plane equation by substituting any polygon vertex into the plane equation. Plane(P) = Ax + By + Cz + D = 0
- Calculate Plane(eyept) to determine if eye is above or below. This corresponds to checking the sign of D.

Face Culling in NDCS :

In NDCS, the z-component of the surface normal does reflect the true visibility, as desired. If the z-component is positive, the normal points away from the eye and the polygon should thus be culled.

Computing Surface Normals

In order to do the face culling, we need a surface normal.

Method 1

Use the cross-product of two polygon edges. The order in which vertices are stored should be consistent. For example, if polygon vertices are stored in CCW order when viewed from above the `front face', then we could use

N = (P2 - P1) x (P3 - P2)

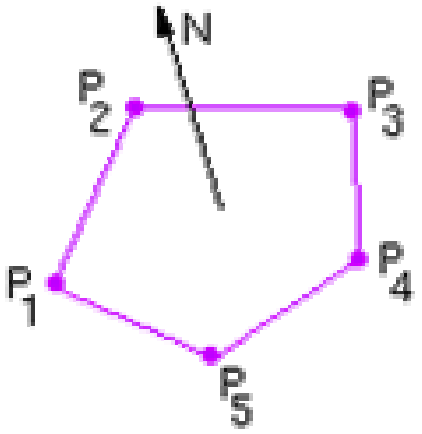


Fig. 10.9

Method 2

A more robust method is to use the projected area onto the yz, xz, and yz planes. To see that areas can be used to calculate a normal, first consider the 2D case.

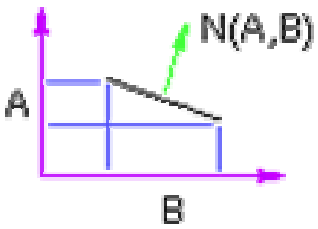


Fig. 10.10

The areas for the required 3D projections (and thus the components of the normal) can be calculated as follows:

$$N_x = \sum_{i=1}^{n} (y_i - y_j)(z_i + z_j)$$
$$N_y = \sum_{i=1}^{n} (z_i - z_j)(x_i + x_j)$$
$$N_z = \sum_{i=1}^{n} (x_i - x_j)(y_i + y_j)$$

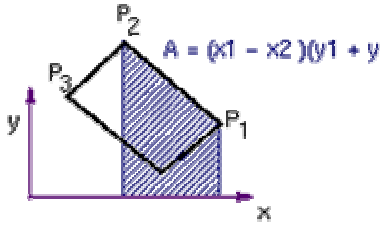


Fig. 10.11

10.8 PAINTER'S ALGORITHM

Painter's Algorithm

• Sort polygons by farthest depth.

• Check if polygon is in front of any other.

• If no, render it.

• If yes, has its order already changed backward?

– If no, render it.

– If yes, break it apart.

Which polygon is in front? Our strategy: apply a series of tests.

– First tests are cheapest

– Each test says poly1 is behind poly2, or maybe.

1. If min z of poly1 > max z poly2 ------1 in back.

2. The plane of the polygon with smaller z is closer to viewer than other polygon.

(a,b,c,)*(x,y,z) >= d.

3. The plane of polygon with larger z is completely behind other polygon.

4. Check whether they overlap in image

a. Use axial rectangle test.

b. Use complete test.

Problem Cases: Cyclic and Intersecting Objects

• Solution: split polygons

Advantages of Painter's Algorithm

– Simple

– Easy transparency

Disadvantages

– Have to sort first

– Need to split polygons to solve cyclic and intersecting objects

10.9 Z-BUFFER ALGORITHM

Z- Buffer Algorithms.

Here, the depth of the surface is given by the coordinates. Algorithm compares the depth of each pixel position. We use the normalized coordinates. So that the range of depth(z) values vary from 0 to 1.

Z=0 denotes the back clipping plane.
Z=1 denotes the front clipping plane.

We use two types of memories.

1) Frame buffer: which stores the intensity values for each pixel position and
2) Z-buffer: which stores the depth of each pixel (x,y) position. Algorithm keeps track of the minimum depth value.

Algorithm:

1) Initialize depth (x,y)=0
   Framebuffer (x,y)=I background for all (x,y)
2) Compute the z-Buffer values by using the equation of the plane. Ax+By+Cz+D=0
   [here, we store information about all the polygonal surface included in the picture.]
   The pixels are scanned by the scaline incremental method.
   Z= -1/C (Ax+By+D) i.e. for any pixel position (Xk,Yk) the depth (Xk,Yk)=Zk
   Zk=-1/C(Axk+Byk+C) The next pixel position is at (Xk+1, Yk) or (Xk,Yk-1)

   Zk+1=-1/C(A(xk+1)+Byk+D)
   Zk+1=-1/C(A(xk+1)+Byk+D) –A/C
   Zk+1=-Zk-A/C calculates the values of depth recursively.

Similarly, the depth values down the edges of intersection of the polygon surface and the scanline are given as calculated as follows:

Let y=mx+c is the example of the left most intersecting edge then from

(Xk,Yk)→(X', Yk-1) along the edge gives,

Yk= mXk+C 1= m(Xk-X')
Yk-1= mX'+C X'= Xk -1/m

The depth value then becomes,

Zk+1=-1/C (A (Xk-1/m) + Byk-1 + D)

Zk+1=-1/C(AXk + Byk - B+D)+ (A/M)/C

Zk+1=-1/C(AXk + Byk +D)+B/C+ (A/M)/C

Zk+1=Zk+ ((A/M)+B)/C

3) If the calculated depth value is Zk+1 and if at (x,y) pixel position Z > calculated depth (x,y) Then,
   Set the depth value as depth(x,y)=z and frame buffer (x,y)= I surface.

4) Repeat steps 2 and 3, till all the polygonal surfaces are processed.

10.10 FLOATING HORIZON ALGORITHM

It is useful for rendering a mathematically defined surface. The algorithm can be implemented in object space or in image space where aliasing may occur. Here given a surface represented in the implicit form F(x,y,z) = 0. The surface is represented by curves drawn on it, say curves of constant height z=c for multiple values of c. It convert the 3D problem to 2D by intersecting the surface with a series of cutting planes at constant values of z.

The function $F(x,y,z)=0$ is reduced to a curve in each of these parallel planes, $y=f(x,z)$ where z is constant for each of the planes.

Pseudocode for floating horizon

- The surface may dip below the closest horizon and the bottom of the surface should be visible
- Keep two horizon arrays, one that floats up and one that floats down
- In image space, aliasing can occur when curves cross previously drawn curves
- Example:
- Let curves at z=0, z=1 and z=2 be defined by

$$y=\begin{cases}20-x \text{ for } 0\le x\le 20,\\ x-20 \text{ for } 20\le x\le 40\end{cases}$$

$$y=\begin{cases}x & \text{for } 0\le x\le 20,\\ -x+4020 \text{ for } 20\le x\le 40\end{cases}$$

y= 10,respectively

Set h[0..40] = 0 and for $z=0,1,2$ compute the value of y saving higher y and drawing the curves

10.11 ROBERTS ALGORITHM

Object space Algorithms:

In object space algorithm we compare every object in scene/ picture with every other object from the same scene. The computational work increases proportionally as the square of number of objects

Object space algorithms are implemented in the physical coordinate system in which objects are described. They required

less storage space. They are particularly useful in engineering applications.

The object space algorithm is Robert's Algorithm.

It gives elegant mathematical solution which is operated in object space because of more efficient image space algorithm. Roberts's algorithm is less appreciated. It requires each scene to be divided into volumes which are either convex or concave. These volumes are further divided to component volumes.

This algorithm is developed in 3 parts.

→ 1st part analysis each part separately to eliminate the hidden plane.

→ 2nd part compares remaining edges of each volume against all other to find which line segments are hidden

➔ 3rd part construct the junction lines for penetrating volume.

Algorithm assumes that a volume consists of plane polygonal faces. The faces consist of edges and the edges consist of individual vertices.

Algorihtm:

1. { start
   Eliminate the hidden plane{
2. For each volume in the scene{
3.
   i) form volume matrix list
   ii) compute the equation of plane for each face polygon of the volume.
   iii) Check the sign of the plane equation.
   iv) find a point which lies inside the volume and find the dot product of the equation of plane with this point. If the dot product is less than zero change the sign of the plane equation.
   v) form the modified volume matrix.
   vi) Multiply by the inverse of the viewing transformation.
   vii) Compute and store the bounding box values as Xmin, Xmax, Ymin, Ymax, Zmin, Zmax for the transformation volume.
1. Identify the hidden plane.{
   i) Take the dot product of test point at infinity and transform volume matrix.
   ii) If the dot product is < 0 then the plane is the hidden plane.
   iii) Eliminate entire polygon forming that plane. This eliminates the necessity for separately identifying hidden lines.
   }
2. Eliminate the line segments for each volume hidden by on other volume in the scene.
   If there is only one volume ---- STOP.
   Else

For a priority list of volumes by performing any efficient sorting algorithm on the z- coordinate

For each volume in the sorted list
DO {

i) Test the self hidden edges against all other volumes.
ii) The volume whose edges are being tested is named as the test object which is tested against the current test volume.
iii) Perform bounding box test. For the test object and test volume. As follows: {

If Xmin(test vol) > Xmax(test obj)
OR Xmax(test vol) < Xmin(test obj)
OR Ymin(test vol) > Ymax(test obj)
OR Ymax(test vol) < Ymin(test obj) Then

The test volume cannot hide any edge of the object
Continue with the next test volume.

Else perform preliminary penetrating text to click if the test object is in sorted in the test volume.

[if the insertion list is empty → No insertion.]
3. Determine the visible junction lines for penetrating volumes.
   If the visibility→ False, Skip through the display routine.
   If no penetrating points have been recorded, Skip through the display routine.
   Form possible junction edges by connections all penetrating points.
   Test all junction edges against the both volumes.
   Test the surviving visible junction edges against all volumes in the scene for visibility. Save the visible segments.
4. Display remaining visible edges
5. } End.

Check your Progress:

1. Which stores the intensity values for each pixel position is known as _____.
2. Which stores the depth of each pixel (x,y) position is known as _____.

10.12 LET US SUM UP

- The bilinear surface is the simplest non flat (curved) surface because it is fully defined by means of its four corner points
- The translation sweep and rotational sweep is combined to get a 3D object. The corresponding sweep representation is called as general sweep.

- The problem of removing the unwanted surfaces and line is called “hidden surface problem”.
- Back-face culling directly eliminates polygons not facing the viewer.

10.13 REFERENCES AND SUGGESTED READING

- Procedural elements of Computer Graphics, David F. Rogers, Tata McGraw Hill.
- Computer Graphics, ISRD Group, Tata McGraw Hill.
- Computer Graphics, Amarendra Sinha, A. Udai,, Tata McGraw Hill.
- Computer Graphics,A. P. Godase, Technical Publications Pune.
- Mathematical Elements of Computer Graphics, David F. Rogers, Tata McGraw Hill.
- Computer Graphics, Donald Hearn, M P. Baker, PHI.
- Computer Graphics: A programming Approach, Steven Harrington, McGraw-Hill.
- Theory and Problems of Computer Graphics, Zhigang Xiang, Roy, plastock, Schaum’s outline series, McGraw-Hill.

10.14 EXERCISE

1. What is hidden surface problem? Discuss in brief.
2. Explain the following models: a) wireframe b) sweep representation
3. What is painter’s algorithm? Explain with an example.
4. Give the z-buffer algorithm and explain it.
5. What is Back-face culling?
6. Explain the steps in Robert’s algorithm.
7. How to construct Bezier Surface?

11

OBJECT RENDERING

Unit Structure:

11.0 OBJECTIVE

The objective of this chapter is

- To understand light modeling techniques and illumination models.
- To understand different types of shading models.
- To understand concepts Ray Tracing and Radiosity.

11.1 INTRODUCTION

We know that the realistic scenes are obtained by generating transformations, perceptive projection of objects and by

applying lighting effects to the visible surface. An illumination model is used to calculate the intensity of light that we are going to see at a given point on the surface of an object. Illumination models are also some time called as lighting or shading models. Here, we consider shading of 3-D objects and its models. In illumination models to calculate the intensity of the color we use the method of the object rendering.

11.2 LIGHT MODELING TECHNIQUES

The object may be illuminated by light which can come from all direction.

A surface that is not exposed directly to a light source will be still visible if nearby objects are illuminated. In basic illumination model we can set a general level of brightness of a scene. The method by which we create such effect usually follows the laws of reflections. The light reflections obtained from various surfaces top produce a uniform illumination is called ambient light or background light. It has no special or directional characteristics the amount of ambient light incident on each object is constant for all the surfaces and for all the direction. When the illumination is uniform from all directions it is called **'diffused illumination'**. Usually diffused illumination is a back ground light which is reflected from walls scenes and curve ceiling, floor.

When we illuminate a shiny surface such as polish metal we observe highlight or bright spot on the shining surface. This phenomenon of reflection of light in concentrated area around the reflection angle is called specular reflection. Due to specular reflection the surface appears to be not in its original color but white, the color of the incident light. The specular reflection angle is always equal to the angle of incident light.

11.3 ILLUMINATION MODEL

An illumination or lighting model is a model or technique for determining the colour of a surface of an object at a given point. A simple illumination model can be based on three components: ambient reflection, diffuse reflection and specular reflection.

11.4 SHADING

A shading model is used in computer graphics to simulate the effects of light shining on a surface. The intensity that we see on a surface is dependent upon two things one the type of light sources and second the surface characteristics (eg. Shining, matte, dull, and opaque or transparent).

11.4.1 FLAT SHADING

Flat shading fills a polygon with a single color computed from one lighting calculation.

Flat shading is appropriate under some conditions:

1. The light source is at infinity so N.L is constant for all points on the polygon.
2. The viewer is at infinity so N.V
3. The object is indeed faceted and not an approximation to a curved object.

11.4.2 POLYGON MESH SHADING

Suppose that we wish to approximate a curved surface by a polygonal mesh. If each polygonal facet in the mesh is shaded individually, it is easily distinguished from neighbors whose orientation is different, producing a "faceted" appearance. This is true if the polygons are rendered using constant shading, interpolated shading, or even per-pixel illumination calculations, because two adjacent polygons of different orientation have different intensities along their borders. The polygon-shading models determine the shade of each polygon individually. Two basic shading models for polygon meshes take advantage of the information provided by adjacent polygons to simulate a smooth surface. In order of increasing complexity (and realistic effect), they are known as Gouraud shading and Phong shading, after the researchers who developed them.

11.4.3 GAURAND SHADING MODEL

Gouraud shading is a form of interpolated shading. The illumination model is applied at vertices and the rest of the polygon's pixels are determined by bi-linear interpolation. If a polygonal object is an approximation of a curved object, then the normal at a vertex is usually the average of the normal of the polygons which meet at that vertex.

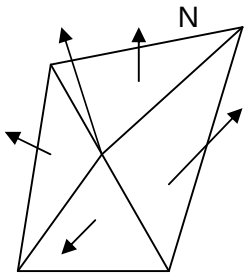


Fig 11.1

11.4.4 PHONG SHADING

Using highlights avoids surfaces that look dull, lifeless, boring, blah. One cool thing about highlights is that, in addition to being all bright an shiny, they change as the object moves. In this way, highlights provide useful visual information about shape and motion.

The simplest model for approximating surface highlights is the Phong model, originally developed by Bui-Tong Phong. In this model, we think of the interaction between light and a surface as having three distinct components:

- Ambient
- Diffuse
- Specular

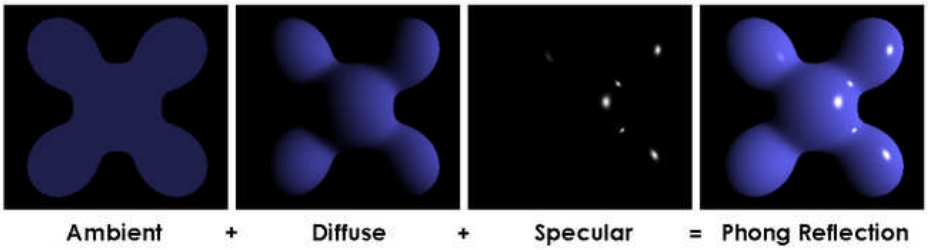


Fig. 11.2

The ambient component is usually given a dim constant value, such as 0.2. It approximates light coming from a surface due to all the non-directional ambient light that is in the environment. In general, you'll want this to be some tinted color, rather than just gray. $[r_a,g_a,b_a]$. For example, a slightly greenish object might have an ambient color of [0.1,0.3,0.1].

The diffuse component is that dot product **n•L** that we discussed in class. It approximates light, originally from light source **L**, reflecting from a surface which is diffuse, or non-glossy. One example of a non-glossy surface is paper. In general, you'll also want this to have a non-gray color value, so this term would in general be a color defined as: $[r_d,g_d,b_d]$(**n•L**).

Finally, the Phong model has a provision for a highlight, or specular, component, which reflects light in a shiny way. This is defined by $[r_s,g_s,b_s]$(**R•L**)^p, where **R** is the mirror reflection direction vector we discussed in class (and also used for ray tracing), and where p is a specular power. The higher the value of p, the shinier the surface.

The complete Phong shading model for a single light source is:

$$[r_a,q_a,b_a]+[r_d,q_d,b_d]\max_0(n.L)+[r_s,q_s,b_s]\max_0(R.L)^p$$

If you have multiple light sources, the effect of each light source $\mathbf{L_i}$ will geometrically depend on the normal, and therefore on the diffuse and specular components, but not on the ambient component. Also, each light might have its own [r,g,b] color. So the complete Phong model for multiple light sources is:

$$[r_a, q_a, b_a] + \sum_i \left([L_r, L_q, L_p] \left([r_d, q_d, b_d] \max_0 (n.L_i) + [r_s, q_s, b_s] \max_0 (n.L_i)^p \right) \right)$$

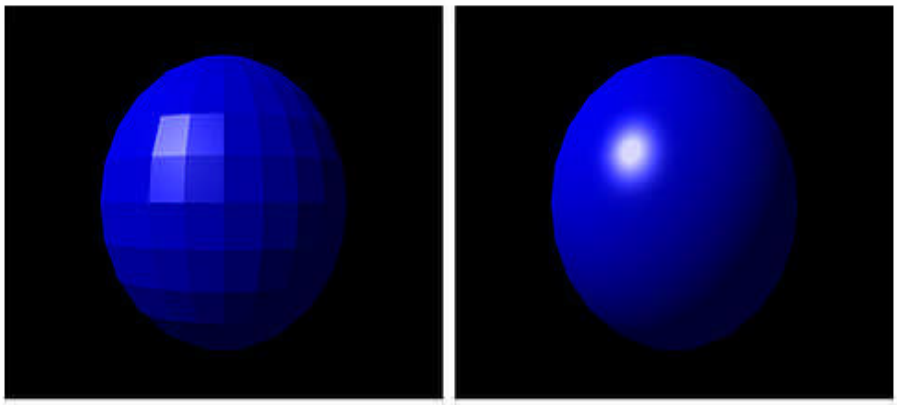


Fig. 11.3

Check your Progress:

True or False

1. Gouraud shading is a form of interpolated shading.
2. Flat shading fills a polygon with a single color computed from more than one lighting calculation.

11.5 TRANSPARENCY EFFECT

The fact that image files are always rectangular can present some limitations in site design. It may be fine for pictures, but it is less desirable for logos, or for images that gradually fade into the background. For relatively simple web pages (such as the one that you are reading now), this restriction is easily worked around: simply match the background of your image to the background of your web page. If you pick the exact same color (easiest if using pure white), the rectangular boundary of your image will be invisible. This simple technique has been utilized for many of the graphics on this page. This technique is less successful if your background is more complex, however. If you use an image as a background, for example, you can't just match one color. And because different web browsers have slight differences in how they display web pages, it's basically impossible to try and match the background of your image to the background of your web page.

11.6 SHADOWS

Shadow can help to create realism. Without it, a cup, eg., on a table may look as if the cup is floating in the air above the table. By applying hidden-surface methods with pretending that the position of a light source is the viewing position, we can find which surface sections cannot be "seen" from the light source => shadow areas.

We usually display shadow areas with ambient-light intensity only.

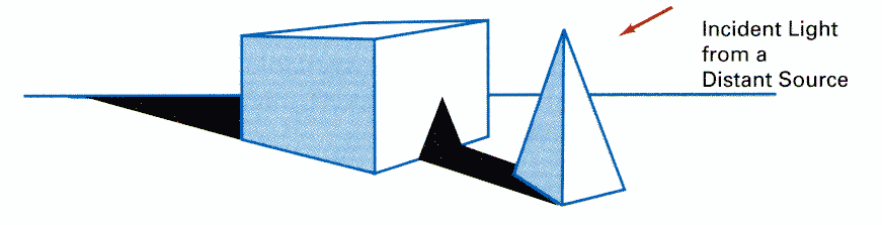


Fig 11.4

11.7 TEXTURE AND OBJECT REPRESENTATION

Since it is still very difficult for the computer to generate realistic textures, a method called texture mapping is developed in which a photograph of real texture is input into the computer and mapped onto the object surface to create the texture for the object. The texture pattern is defined in a M x N array of a texture map indicted by (u, v) coordinates.

For each pixel in the display, map the 4 corners of pixel back to the object surface (for curved surfaces, these 4 points define a surface patch) and then map the surface patch onto the texture map, this mapping computes the source area in the texture map. Due to this the pixel value is modified by weighted sum of the texture's color.

Check your Progress:

1. What is transparency effect?
2. Explain shadow.

11.8 RAY TRACING

Ray tracing follows all rays from the eye of the viewer back to the light sources. This method is very good at simulating specular reflections and transparency, since the rays that are traced through the scenes can be easily bounced at mirrors and refracted by transparent objects.

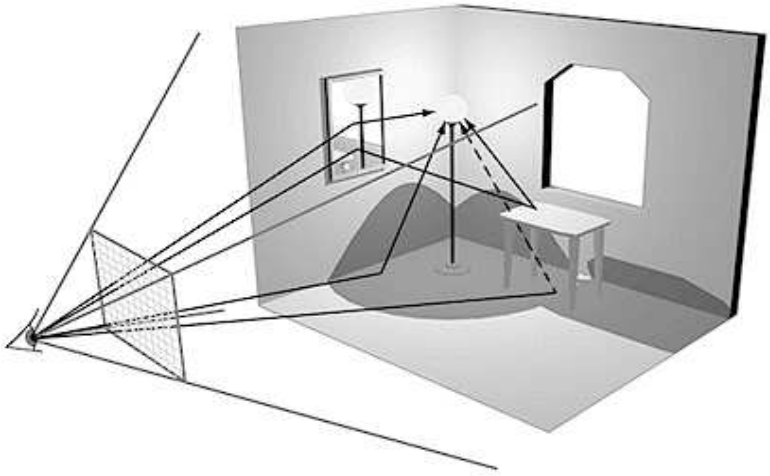


Fig. 11.5

To create an image using ray tracing, the following procedure is performed for each pixel on the computer screen.

1. A ray is traced back from the eye position, through the pixel on the monitor, until it intersects with a surface. When the imaginary line drawn from the eye, through a pixel, into a scene strikes a polygon three things happen.

2. First, the color and brightness values are calculated based on direct illumination from lights directly striking that polygon. We know the reflectivity of the surface from the model description, but we do not yet know the amount of light reaching that surface. To determine the total illumination, we trace a ray from the point of intersection to each light source in the environment (shadow ray). If the ray to a light source is not blocked by another object, the light contribution from that source is used to calculate the color of the surface.

3. Next, the angles of reflection and refraction (the bending of light as it passes through a transparent object such as water) are calculated. Based on the surface's assigned reflectivity and/or transparency, the ray splits and continues off in two new directions. The intersected surface may be shiny or transparent. In this case we also have to determine what is seen in or

through the surface being processed. Steps 1 and 2 are repeated in the reflected (and, in the case of transparency, transmitted) direction until another surface is encountered. The color at the subsequent intersection point is calculated and factored into the original point.

4. If the second surface is yet again a reflective or transparent surface, the ray tracing process repeats once again, and so on, until a maximum number of iterations is reached or until no more surfaces are intersected. Each ray continues bouncing around the scene until it hits a light source, leaves the scene, or reaches some arbitrary number (recursion level). When all the rays have completed their journeys the intensity and color values are combined and the pixel is painted.

11.9 RAY CASTING

The basic goal of ray casting is to allow the best use of the three-dimensional data and not attempt to impose any geometric structure on it. It solves one of the most important limitations of surface extraction techniques, namely the way in which they display a projection of a thin shell in the acquisition space. Surface extraction techniques fail to take into account that, particularly in medical imaging, data may originate from fluid and other materials which may be partially transparent and should be modeled as such. Ray casting doesn't suffer from this limitation.

11.10 RADIOSITY

There is a new method of rendering that was recently developed. It is called radiosity.

It does something all the other rendering methods don't do: it figures out the relationship in the scene of all the objects present. For example, in real life, if you take a bright colored ball and put it into a white room, the walls of the room are going to reflect a little bit of color from the ball, making them look a little reddish for example.

This is not possible in ray tracing, since it does not bounce rays off of matte objects, such as a wall. You can compare the above picture to the one of the ray traced object. All though this is not a very good example, you can see that the checkerboard pattern of the "tri-ball" has a slight effect on the color of the bricks right underneath it. This adds the extra thing to rendered scenes and makes them look extremely realistic. Radiosity produces extremely good results, but unfortunately, there is a tradeoff: rendering time. Before the computer even starts rendering, it has to

solve a certain "radiosity model" which is the relationship of one object on all the other ones in a scene. Then it can start rendering.

Check your progress:
1. Give the procedure of ray tracing.
2. What is radiosity?

11.11 COLOR MODELS

RGB Model:

There are many different ways of specifying color, or color models. The most common is the RGB color model where a color is specified in terms of red, green and blue color components. If we use the RGB color model then the ambient color (reflectivity) of an object is (kaR, kaG, kaB), the diffuse color (reflectivity) of an object kd is (kdR, kdG, kdB) and the color of the light emitted from the point light source as (LdR,LdG,LdB).

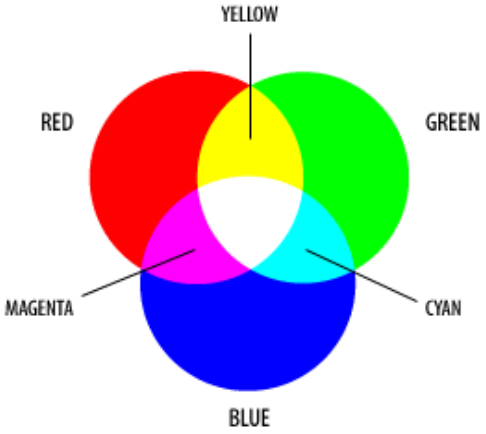


Fig. 11.6

CMY:

It is possible to achieve a large range of colors seen by humans by combining cyan, magenta, and yellow transparent dyes/inks on a white substrate. These are the subtractive primary colors. Often a fourth black is added to improve reproduction of some dark colors. This is called "CMY" or "CMYK" color space. The cyan ink absorbs

red light but transmits green and blue, the magenta ink absorbs green light but transmits red and blue, and the yellow ink absorbs blue light but transmits red and green.

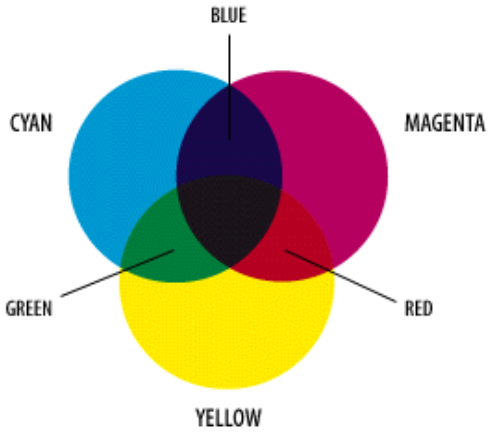


Fig. 11.7

HSV/ HSB Model:

The HSV, or HSB, model describes colors in terms of hue, saturation, and value (brightness). Note that the range of values for each attribute is arbitrarily defined by various tools or standards. Be sure to determine the value ranges before attempting to interpret a value. Hue corresponds directly to the concept of hue in the Color Basics section. The advantages of using hue are the angular relationship between tones around the color circle is easily identified. Shades, tints, and tones can be generated easily without affecting the hue saturation corresponds directly to the concept of tint in the Color Basics section, except that full saturation produces no tint, while zero saturation produces white, a shade of gray, or black.

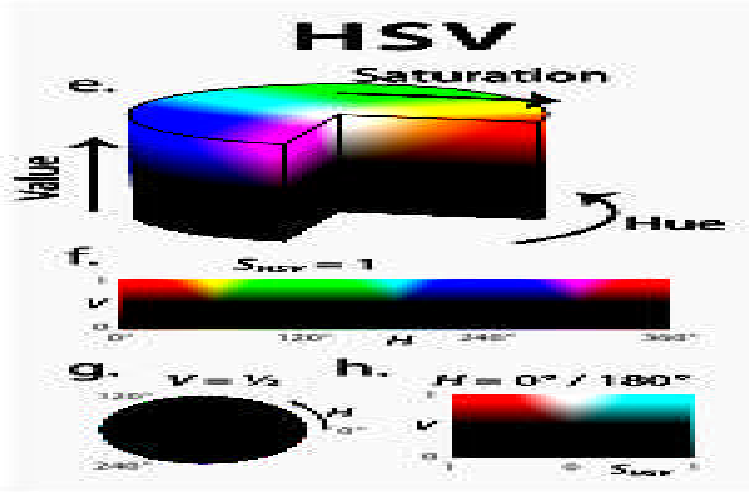


Fig. 11.8

11.12 LET US SUM UP

- When light sources create highlights or bright spots they are called **'specular reflections'**. This effect is usually seen on shining surfaces than on dull surfaces.

- When the illumination is uniform from all directions it is called **'diffused illumination'**.

- Flat shading fills a polygon with a single color

- Gouraud shading is a form of interpolated shading.

11.14 REFERENCES AND SUGGESTED READING

- Procedural elements of Computer Graphics, David F. Rogers, Tata McGraw Hill.
- Computer Graphics, Amarendra Sinha, A. Udai,, Tata McGraw Hill.
- Mathematical Elements of Computer Graphics, David F. Rogers, Tata McGraw Hill.
- Computer Graphics, Donald Hearn, M P. Baker, PHI.

11.13 EXERCISE

1. Write a short note on light modeling techniques.
2. List the different types of shading. Explain any one in details.
3. Write a short note on Phong shading.
4. Write a short note on Gaurand Shading Model

12

INTRODUCTION TO ANIMATION

Unit Structure:

12.0 OBJECTIVE

The objective of this chapter is

- To understand the concepts of computer animation.
- To understand Morphing.

12.1 INTRODUCTION

The main goal of computer animation is to synthesize the desired motion effect which is a mixing of natural phenomena, perception and imagination. The animator designs the object's dynamic behavior with his mental representation of causality

12.2 KEY-FRAME ANIMATION

When someone creates an animation on a computer, they usually don't specify the exact position of any given object on every single frame. They create key frames. Key frames are important frames during which an object changes its size, direction, shape or other properties. The computer then figures out all the in between frames and saves an extreme amount of time for the animator. Two frames are drawn by user In between frames generated by computer.

12.3 CONSTRUCTION OF AN ANIMATION SEQUENCE

Design of animation sequences

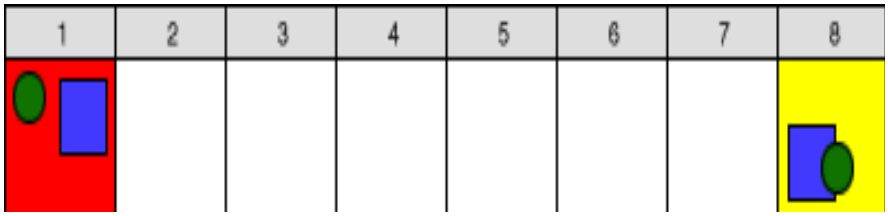
1. **Story board layout**
2. **Object definations**
3. **Key frame specification**
4. **Generation of in between frames**

Visualization applications are generated by the solution of the numerical models for frame- by- frame animation. Each frame of the scene is separately generated and stored. Later, the frames can be recorded on film or they can be consecutively displayed in "real-time playback" mode.

The **story board** is an outline of the action. It defines the motion sequence as a set of basic events that are to take place.

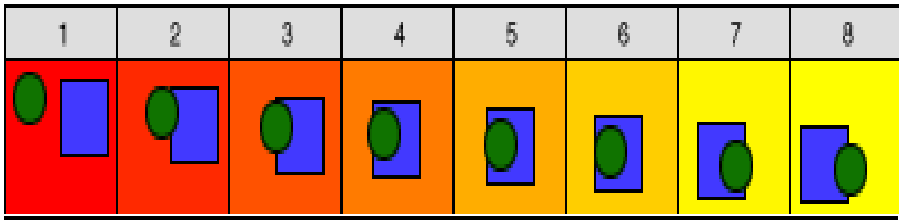
An **Object definition** is given for each participant in the action. Objects can be defined in terms of basic shapes, such as polygon or splines. In addition, the associated movements for each object are specified along with the shape.

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. Some key frames are chosen at extreme positions in the action, other are spaced so that the time interval between key frames is not too great. More key frames are specified for intricate motions than for simple slowly varying motions.

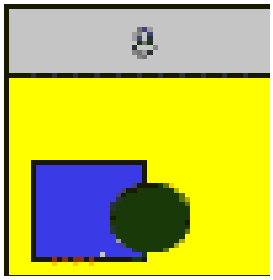


Two Frames drawn by user
Fig. 12.1

In between frames are the intermediate frames between the key frames. The number of in between needed is determined by the media to be used to display the animation. Film requires 24 frames per second, and graphic terminals are refreshed at the rate of 30 to 60 frames per second.



In between frames generated by computer
Fig. 12.2



Final Animation
Fig. 12.3

Check your progress:

Fill in the blanks.

1. _____ frames are the intermediate frames between the key frames.
2. The _____is an outline of the action.
3. _____ is a detailed drawing of the scene at a certain time in the animation sequence.

12.4 MOTION CONTROL METHODS

The key issue of Computer Animation is the way of defining motion, what is commonly known as Motion Control Methods (MCMs). MCMs may be classified based on the nature of the information, which is directly manipulated: geometric, physical, or behavioral

Methods based on Geometric and Kinematics information

These methods are heavily relied upon the animator. **Motion is locally controlled** and defined in terms of coordinates, angles, velocities, or accelerations.

The **Performance Animation** which consists in magnetic or optical measurement and recording of direct actions of a real person for immediate or delayed playback. The technique is

especially used today in production environments for 3D character animation.

Key frame animation is still another popular technique in which the animator explicitly specifies the kinematics by supplying keyframes values whose "in-between" frames are interpolated by the computer.

Inverse kinematics is a technique coming from robotics, where the motion of links of a chain is computed from the end link trajectory.

Image Morphing is a warping-based technique which interpolates the features between two images to obtain a natural in between image. For geometric **deformations**, multi-layered models are particularly useful for modelling 3D characters.

Methods based on Physical information
In these methods, the animator provides physical data and the motion is obtained by solving the dynamic equations. **Motion is globally controlled**. We may distinguish methods based on **parameter adjustment** and **constraint-based methods,** where the animator states in terms of constraints the properties the model is supposed to have, without needing to adjust parameters.

Methods based on Behavioral information
A behavioral motion control method consists of driving the behavior of autonomous creatures by providing high-level directives indicating a specific behavior without any other stimulus.

12.5 PROCEDURAL ANIMATION

Procedural animation corresponds to the creation of a motion by a procedure describing specifically the motion. Procedural animation should be used when the motion can be described by an algorithm or a formula. For example, consider the case of a clock based on the pendulum law. A typical animation sequence may be produced using a program such as:

```
create CLOCK (...);
for FRAME:=1 to NB_FRAMES
TIME:=TIME+1/24;
ANGLE:=A*SIN (OMEGA*TIME+PHI);
MODIFY (CLOCK, ANGLE);
draw CLOCK;
record CLOCK
erase CLOCK
```

Categories of procedural animation

Two large categories of procedural animation are

1. Physics-based modeling/animation
2. Alife (artificial life)

1. Physics-based modeling/animation deals with things that are not alive. Physics-based modeling/animation refers to techniques that include various physical parameters, as well as geometrical information, into models. The behavior of the models is simulated using well-know natural physical laws. Physics-based modeling/animation can be considered as a sub-set of procedural animation and includes particle systems, flexible dynamics, rigid body dynamics, fluid dynamics, and fur/hair dynamics.

Fig. 12.4

Particle systems simulates behaviors of fuzzy objects, such as clouds, smokes, fire, and water.

Flexible dynamics simulates behaviors of flexible objects, such as clothes. A model is built from triangles, with point masses at the triangles' vertices. Triangles are joined at edges with hinges; the hinges open and close in resistance to springs holding the two hinge halves together. Parameters are: point masses, positions, velocities, accelerations, spring constants, wind force, etc..

(**Reference:** D. Haumann and R. Parent, "The behavioral test-bed: obtaining complex behavior from simple rules," Visual Computer, '88.)

Rigid body dynamics simulates dynamic interaction among rigid objects, such as rocks and metals, taking account various physical characteristics, such as elasticity, friction, and mass, to produce rolling, sliding, and collisions. Parameters for "classical" rigid body dynamics are masses, positions, orientations, forces, torques, linear and angular velocities, linear and angular momenta, rotational ineria tensors, etc.

(Reference: J. Hahn, Realistic Animation of Rigid Bodies, Proceedings of SIGGRAPH 88.)

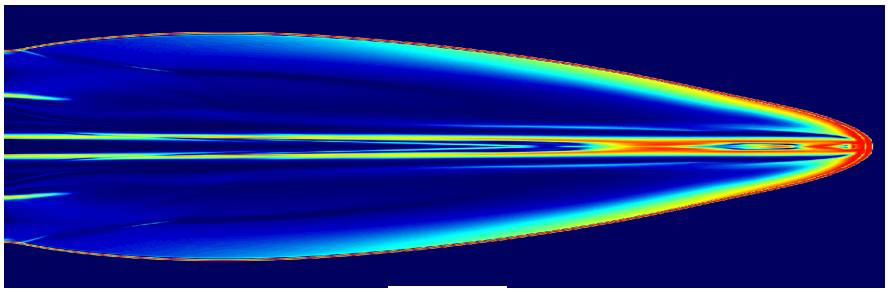


Fig. 12.5

Fluid dynamics simulates flows, waves, and turbulence of water and other liquids.

Fur & hair dynamics generates realistic fur and hair and simulates behaviors of fur and hair. Often it is tied into a rendering method.

2. Alife (artificial life) deals with things are virtually alive.

Fig. 12.6

Behavioral animation simulates interactions of artificial lives. Examples: flocking, predator-prey, virtual human behaviors.

Artificial evolution is the evolution of artificial life forms. The animator plays the role of God. As artificial life forms reproduce and mutate over time, the survival of the fittest is prescribed by the animator's definition of "fittest" (that is artificial 'natural' selection). See Karl Sims's works.

Branching object generation generates plants, trees, and other objects with branching structures and simulate their behaviors. Without a procedural method, building a model of a branching object, such as a tree with a number of branches, requires a lot of time and effort. Branching object generation methods (L-systems &BOGAS) employ user defined rules to generate such objects.

12.6 KEY-FRAME ANIMATION VS. PROCEDURAL ANIMATION

In a procedural animation objects are animated by a procedure -- a set of rules -- not by keyframing. The animator specifies rules and initial conditions and runs simulation. Rules are often based on physical rules of the real world expressed by mathematical equations.

To produce a keyframe animation, the animator creates the behavior of a model manually by using an intuitive “put that there” methodology. The animator has direct control over the positions, shapes, and motions of models at any moment in the animation. On the other hand, to produce a procedural animation the animator provides initial conditions and adjusts rather abstract physical parameters, such as forces and torques, in order to control positions, shapes, and motions of models. The effect of changing a parameter value is often unpredictable in procedural animation. The animator has to run a simulation to see the result.

Check your progress:

1. What are motion control methods?
2. Explain procedural animation.

12.7 INTRODUCTION TO MORPHING

Morphing is a special effect in motion pictures and animations that changes (or morphs) one image into another through a seamless transition. Image morphing means creating a sequence of images which when played in sequence, show one image being slowly changed into another image.

Original Image
Fig. 12.7

Morphed Image
Fig. 12.8

12.8 THREE-DIMENSIONAL MORPHING

Image morphing, the construction of an image sequence depicting a gradual transition between two images, has been extensively investigated. For images generated from 3D models, there is an alternative to morphing the images themselves: 3D morphing generates intermediate 3D models, the morphs, directly from the given models; the morphs are then rendered to produce an image sequence depicting the transformation. 3D morphing overcomes the following shortcomings of 2D morphing as applied to images generated from 3D models:

- In 3D morphing, creating the morphs is independent of the viewing and lighting parameters. Hence, we can create a morph sequence once, and then experiment with various camera angles and lighting conditions during rendering. In 2D morphing, a new morph must be recomputed every time we wish to alter our viewpoint or the illumination of the 3D model.

- 2D techniques, lacking information on the model's spatial configuration, are unable to correctly handle changes in illumination and visibility. Two examples of this type of artifact are: (i) Shadows and highlights fail to match shape changes occuring in the morph. (ii) When a feature of the 3D object is not visible in the original 2D image, this feature cannot be made to appear during the morph; for example, when the singing actor needs to open her mouth during the morph, pulling her lips apart thickens the lips instead of revealing her teeth.

The models subjected to 3D morphing can be described either by geometric primitives or by volumes.

Check your progress:

True or False

1. Morphing is a special effect in motion pictures.
2. In 3D morphing, creating the morphs is dependent of the viewing and lighting parameters.

12.9 LET US SUM UP

- Key frames are important frames during which an object changes its size, direction, shape or other properties.

- Motion Control Methods (MCMs classified based on the nature of the information, which is directly manipulated: geometric, physical, or behavioral.

- Procedural animation should be used when the motion can be described by an algorithm or a formula.

12.10 REFERENCES AND SUGGESTED READING

- Computer Graphics, Amarendra Sinha, A. Udai,, Tata McGraw Hill.

- Computer Graphics, Donald Hearn, M P. Baker, PHI.

- Computer Graphics: A programming Approach, Steven Harrington, McGraw-Hill.

- Theory and Problems of Computer Graphics, Zhigang Xiang, Roy, plastock, Schaum's outline series, McGraw-Hill.

12.11 EXERCISE

1. Explain conventional animation.
2. Write a short note on virtual reality.
3. Write a short note on image morphing.
4. Give the methods of controlling animation.
5. What is frame by frame animation?
6. Differentiate between Key-Frame Animation and Procedural Animation.